QCMTEST Nom et prénom: Introduction à la programmation orientée objet 9/01/2020 Groupe: ..... Cochez les cases en mettant une  $\times$ . Le symbole  $\bigoplus$  indique que la question peut avoir zéro, une ou plusieurs bonnes réponses. Pour ces questions, cocher une bonne réponse apporte des points positifs ; cocher une mauvaise réponse peut apporter des points négatifs. Dans tout le code, les package et les import sont censés être correctement déclarés. Toute classe est supposée être dans le bon package, dans le bon fichier, avec les bons import.Question 1 La classe donnée class ClasseDonnee {} pourrait être sous-classée par extends pourrait être instanciée par new ClasseDonnee ClasseDonnee() La classe donnée Question 2 final class ClasseDonnee { ClasseDonnee() {} pourrait être sous-classée par extends pourrait être instanciée new par ClasseDonnee ClasseDonnee() Question 3  $\bigoplus$ La classe donnée abstract class ClasseDonnee { ClasseDonnee() {} pourrait être sous-classée par extends pourrait être instanciée new ClasseDonnee ClasseDonnee() Question 4 Affiche "Done" Soit le code ci-dessous ; que se passe-t-il?

public class Alpha {

}

public void throwup() throws Exception {

public static void main(String[] args) {

throw new Exception();

Alpha alpha = new Alpha();
alpha.throwup();

System.out.println("Done");

Erreur d'exécution

Erreur de compilation

Erreur de compilation

Erreur d'exécution

exception"

Affiche "Caught exception" et "Finallied

Affiche seulement "Caught exception"

Affiche seulement "Finallied exception"



```
Affiche "Caught exception" et "Finallied
Soit le code ci-dessous ; que se passe-t-il?
                                                             exception"
public class Gamma {
   public void throwup() throws Exception {
                                                             Erreur d'exécution
      throw new Exception();
                                                             Affiche seulement "Finallied exception"
   public static void main(String[] args) {
      Gamma gamma = new Gamma();
                                                             Affiche seulement "Caught exception"
      try {
          gamma.throwup();
                                                             Erreur de compilation
      } catch (Exception e) {
         System.out.println("Caught exception");
      } finally {
          System.out.println("Finallied exception");
}
```

### Question 6 $\bigoplus$

```
Soit le code ci-dessous ; que se passe-t-il?
public class Delta {
   public void throwup() throws Exception {
      throw new Exception();
   public static void main(String[] args) {
       Delta delta = new Delta();
       try {
          delta.throwup();
       } catch (Exception e) {
           System.out.println("Caught exception");
           return;
       } finally {
   System.out.println("Finallied exception");
```

#### Question 7 Soit le code:

```
interface Somethingable {
   void something();
}
interface Whateverable extends Somethingable {
   void whatever();
class TheThing implements Whateverable{
   PPP void something() {
       System.out.println("Somethinging");
   PPP void whatever() {
       System.out.println("Whatevering");
}
```

```
class Doofer {
   void doo(Somethingable sable) {
        sable.XXX();
    void doo(Whateverable sable) {
        sable.YYY();
    void fer(TheThing tt) {
       tt.ZZZ();
   public static void main(String[] args) {
       Doofer doofer = new Doofer();
TheThing the = new TheThing();
        doofer.doo(the);
        doofer.fer(the);
}
```

Par quels niveaux d'accès pourrait-on remplacer PPP?

package-private	public
protected	$oxedsymbol{\Box}$ private

Question 8 $\bigoplus$ placer XXX ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem
something	whatever
Question 9 $\bigoplus$ placer YYY ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem
something	whatever
Question 10 ⊕ remplacer ZZZ ?	Pour le code de la question précédente, par quelles méthodes pourrait-or
something	whatever

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit





Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r\cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r\sin(\theta)$   $\theta = \tan^{-1}\frac{y}{x}$ 

En Java ce dernier est calculé par theta = Math.atan2(y, x)

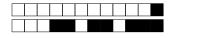
- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

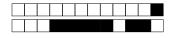
Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.

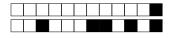




estion 13	Continuez votre réponse à la question précédente ici si nécessaire.	



Question 14	Continuez	votre réponse	à la question	précédente i	ci si nécessaire.	



Question	15	Pour	la	question	précédente,	développez	la	méthode	Polar#equ	als.
										<u></u>

1	
1	

	Une classe peut étendre plusieurs classes
	Une interface peut contenir des méthodes privées
	Une classe peut étendre une classe et implémenter plusieurs interfaces
	Une classe abstraite peut implémenter une interface
	Une interface peut contenir des constructeurs
	Une classe abstraite peut contenir des constructeurs
П	Une classe peut étendre une classe abstraite

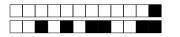


#### Question $17 \oplus$ Soit le code ci-dessous :

```
@FunctionalInterface
interface ComparePerson {
    /**
    * Oreturn negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :

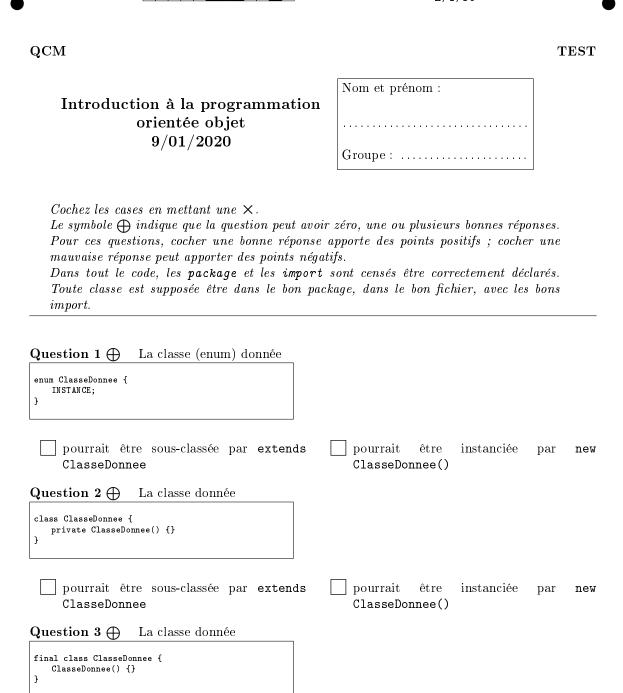


Question 18  $\bigoplus$  Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

Personnel	Faculty
MdC	Person
PEIP	Engineer
Student	Prof



pourrait

ClasseDonnee()

être

instanciée

par

new

pourrait être sous-classée par extends

ClasseDonnee

#### Question 4 $\bigoplus$

Soit le code ci-dessous ; que se passe-t-il ?	Affiche "Caught exception" et "Finallied
<pre>public class Epsilon {     public void throwup() throws RuntimeException {         throw new RuntimeException();     }     public static void main(String[] args) {         Epsilon epsilon = new Epsilon();         try {             epsilon.throwup();         } catch (Exception e) {             System.out.println("Caught exception");         } finally {             System.out.println("Finallied exception");         } }</pre>	exception"  Erreur de compilation  Erreur d'exécution  Affiche seulement "Finallied exception"  Affiche seulement "Caught exception"
}	

#### Question $5 \bigoplus$

Soit le code ci-dessous ; que se passe-t-il ?
public class Beta {
 public void throwup() throws RuntimeException {
 throw new RuntimeException();
 }
 public static void main(String[] args) {
 Beta beta = new Beta();
 beta.throwup();
 System.out.println("Done");
 }
}

# Erreur d'exécution

Erreur de compilation
Affiche "Done"

#### Question 6 $\bigoplus$

Soit le code ci-dessous ; que se passe-t-il ?
public class Gamma {
 public void throwup() throws Exception {
 throw new Exception();
 }
 public static void main(String[] args) {
 Gamma gamma = new Gamma();
 try {
 gamma.throwup();
 } catch (Exception e) {
 System.out.println("Caught exception");
 } finally {
 System.out.println("Finallied exception");
 }
 }
}

Erreur de compilation

Affiche "Caught exception" et "Finallied exception"

Erreur d'exécution

Affiche seulement "Finallied exception"



#### Question 7 Soit le code :

```
interface Somethingable {
   void something();
}

interface Whateverable extends Somethingable {
   void whatever();
}

class TheThing implements Whateverable{
   PPP void something() {
       System.out.println("Somethinging");
   }

   PPP void whatever() {
       System.out.println("Whatevering");
   }
}
```

Par quels niveaux d'accès pourrait-on remplacer PPP?

```
class Doofer {
    void doo(Somethingable sable) {
        sable.XXX();
    }

    void doo(Whateverable sable) {
        sable.YYY();
    }

    void fer(TheThing tt) {
        tt.ZZZ();
    }

    public static void main(String[] args) {
        Doofer doofer = new Doofer();
        TheThing the = new TheThing();
        doofer.doo(the);
        doofer.fer(the);
    }
}
```

 ☐ protected
 ☐ private

 ☐ package-private
 ☐ public

whatever something

**Question 9**  $\bigoplus$  Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer YYY ?

something whatever

**Question 10**  $\bigoplus$  Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer ZZZ ?

whatever something

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit



Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r\cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r\sin(\theta)$   $\theta = \tan^{-1}\frac{y}{x}$ 

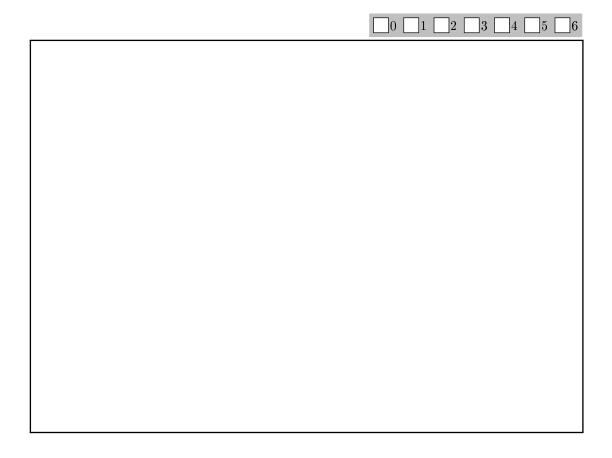
En Java ce dernier est calculé par theta = Math.atan2(y, x)

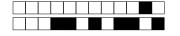
- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.

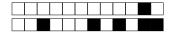




Question 13	Continuez	votre réponse à la question précédente ici si nécessaire.	



Question 14	Continuez	votre réponse	à la question	précédente i	ci si nécessaire.	



 $\hfill \Box$  Une classe abstraite peut contenir des constructeurs

Question	15	Pour	la	question	précédente,	développez	la	méthode	Pola		ls.	
									01		3 [	
$egin{array}{c}  ext{Question} \  ext{} \end{array}$	16 ⊕	Parn	ni l	es affirma	tions ci-desso	ous, cocher c	elles	qui sont	vraies	S :		
Une o	classe pe	eut éte	end	re une cla	sse et implén	nenter plusie	eurs	interface	S			
Une o	classe pe	eut éte	end	re une cla	sse abstraite							
Une i	nterface	peut	coı	ntenir des	constructeur	'S						
Une o	classe pe	eut éte	end	re plusieu	rs classes							
Une o	classe ab	strait	ер	eut implé	menter une i	nterface						
Une i		peut	coı	ntenir des	méthodes pr	rivées						



#### Question $17 \oplus$ Soit le code ci-dessous :

```
@FunctionalInterface
interface ComparePerson {
    /**
    * Oreturn negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

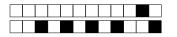
Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :

```
ComparePerson byAge = (p1, p2) -> p1.age - p2.age;

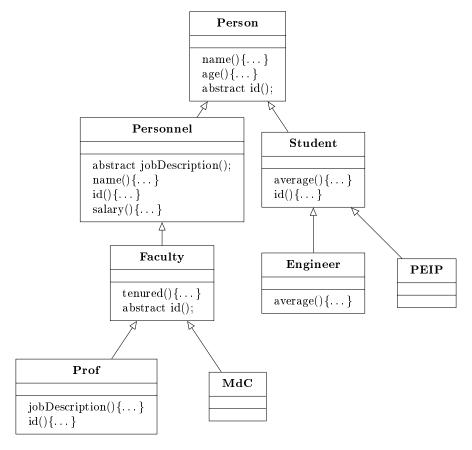
ComparePerson byAge = new ComparePersonByAge();

ComparePerson byAge
= new ComparePersonByAge::compare;
```

```
ComparePerson byAge = new ComparePersonByAge() {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
};
```



Question 18 Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

<pre>Faculty</pre>	Engineer
Person	Student
PEIP	Personnel
Prof	MdC

QCMTEST Nom et prénom: Introduction à la programmation orientée objet 9/01/2020Groupe: ..... Cochez les cases en mettant une  $\times$ . Le symbole  $\bigoplus$  indique que la question peut avoir zéro, une ou plusieurs bonnes réponses. Pour ces questions, cocher une bonne réponse apporte des points positifs ; cocher une mauvaise réponse peut apporter des points négatifs. Dans tout le code, les package et les import sont censés être correctement déclarés. Toute classe est supposée être dans le bon package, dans le bon fichier, avec les bons import.Question 1 

La classe donnée abstract class ClasseDonnee { ClasseDonnee() {} pourrait être sous-classée par extends pourrait être instanciée par new ClasseDonnee() ClasseDonnee Question 2 La classe donnée class ClasseDonnee {} pourrait être sous-classée par extends pourrait être instanciée par new ClasseDonnee ClasseDonnee() Question 3 La classe (enum) donnée enum ClasseDonnee { INSTANCE; } pourrait être sous-classée par extends pourrait être instanciée par new

ClasseDonnee()

ClasseDonnee

Affiche "Caught exception" et "Finallied Soit le code ci-dessous ; que se passe-t-il? exception" public class Delta { public void throwup() throws Exception { Erreur de compilation throw new Exception(); Erreur d'exécution public static void main(String[] args) { Delta delta = new Delta(); Affiche seulement "Caught exception" try { delta.throwup(); Affiche seulement "Finallied exception" } catch (Exception e) { System.out.println("Caught exception"); return; } finally { System.out.println("Finallied exception"); } Question 5

Soit le code ci-dessous ; que se passe-t-il ?

public class Alpha {
 public void throwup() throws Exception {
 throw new Exception();
 }
 public static void main(String[] args) {
 Alpha alpha = new Alpha();
 alpha.throwup();
 System.out.println("Done");
 }
}

#### Question 6

Soit le code ci-dessous ; que se passe-t-il ?

public class Epsilon {
 public void throwup() throws RuntimeException {
 throw new RuntimeException();
 }
 public static void main(String[] args) {
 Epsilon epsilon = new Epsilon();
 try {
 epsilon.throwup();
 } catch (Exception e) {
 System.out.println("Caught exception");
 } finally {
 System.out.println("Finallied exception");
 }
}

Erreur d'exécution

Affiche seulement "Finallied exception"

Affiche "Caught exception" et "Finallied exception"

Affiche seulement "Caught exception"

Erreur de compilation



```
interface Somethingable {
   void something();
}

interface Whateverable extends Somethingable {
   void whatever();
}

class TheThing implements Whateverable{
   PPP void something() {
       System.out.println("Somethinging");
   }

   PPP void whatever() {
       System.out.println("Whatevering");
   }
}
```

```
class Doofer {
    void doo(Somethingable sable) {
        sable.XXX();
    }

    void doo(Whateverable sable) {
        sable.YYY();
    }

    void fer(TheThing tt) {
        tt.ZZZ();
    }

    public static void main(String[] args) {
        Doofer doofer = new Doofer();
        TheThing the = new TheThing();
        doofer.doo(the);
        doofer.fer(the);
    }
}
```

Par quels niveaux d'accès pourrait-on remplacer PPP? protected package-private public private Question 8 Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer XXX? whatever something Question 9 Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer YYY? whatever something Question 10  $\bigoplus$ Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer ZZZ? whatever something

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit





Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r\cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r\sin(\theta)$   $\theta = \tan^{-1}\frac{y}{x}$ 

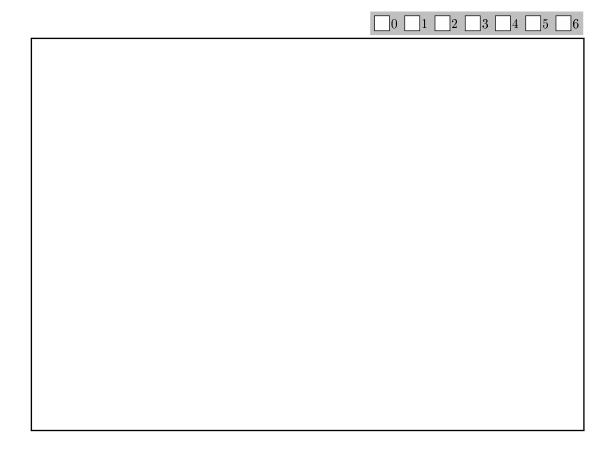
En Java ce dernier est calculé par theta = Math.atan2(y, x)

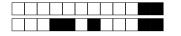
- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.

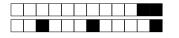




Question 13	Continuez	votre réponse à la question précédente ici si nécessaire.	

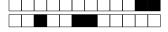


Question 14	Continuez	votre réponse	à la question	précédente	ici si nécessaire.	



Une interface peut contenir des méthodes privées

Question	15	Pour	la	question	précédente,	développez	la	méthode	Polaı	r#equa	ls.
											$3 \square 4$
Question	<b>16</b> $\oplus$	Parn	ni l	es affirma	tions ci-desso	ous, cocher c	elle	s qui sont	vraies	:	
$\square$ Une	classe p	eut éte	end	re une cla	sse et implén	nenter plusie	eurs	interfaces			
Une	classe p	eut éte	end	re plusieu	rs classes						
Une	classe p	eut éte	end	re une cla	sse abstraite						
Une	interfac	e peut	coı	ntenir des	constructeur	'S					
Une	classe a	bstrait	ер	eut implé	menter une i	nterface					
$\square$ Une	classe a	bstrait	ер	eut conte	nir des consti	ructeurs					



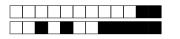
#### Question $17 \oplus$ Soit le code ci-dessous :

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :

```
ComparePerson byAge = new ComparePersonByAge() {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
};

ComparePerson byAge = (p1, p2) -> p1.age - p2.age;

ComparePerson byAge
= new ComparePersonByAge::compare;
```



#### $\textbf{Question 18} \ \bigoplus \quad \text{Soit l'organisation ci-dessous de Polytech'Groland}:$



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

MdC	Prof
PEIP	Personnel
Student	Engineer
Faculty	Person

+4/1/30+ QCMTEST Nom et prénom: Introduction à la programmation orientée objet 9/01/2020 Groupe: ..... Cochez les cases en mettant une  $\times$ . Le symbole  $\bigoplus$  indique que la question peut avoir zéro, une ou plusieurs bonnes réponses. Pour ces questions, cocher une bonne réponse apporte des points positifs ; cocher une mauvaise réponse peut apporter des points négatifs. Dans tout le code, les package et les import sont censés être correctement déclarés. Toute classe est supposée être dans le bon package, dans le bon fichier, avec les bons import.Question 1 La classe donnée class ClasseDonnee { private ClasseDonnee() {} pourrait être sous-classée par extends pourrait être instanciée par new ClasseDonnee() ClasseDonnee Question 2 La classe donnée final class ClasseDonnee { ClasseDonnee() {}

# Question 3 La classe donnée

ClasseDonnee()

être

instanciée

pourrait	${ m \hat{e}tre}$	$instanci\'ee$	par	new	pourrait	être	sous-classée	par	extends
ClasseDor				ClasseDo	onnee				

pourrait être sous-classée par extends

ClasseDonnee

#### Question 4

}

pourrait

par

new

#### Question 5 $\bigoplus$

```
Soit le code ci-dessous ; que se passe-t-il ?
public class Epsilon {
    public void throwup() throws RuntimeException {
        throw new RuntimeException();
    }
    public static void main(String[] args) {
        Epsilon epsilon = new Epsilon();
        try {
            epsilon.throwup();
      } catch (Exception e) {
            System.out.println("Caught exception");
      } finally {
            System.out.println("Finallied exception");
      }
    }
}
```

## Erreur d'exécution

$\mid$ Affiche seul	$\operatorname{lement}$	"Finallied	exception'

#### Erreur de compilation

Affiche	"Caught	exception"	et	"Finallied
exception	$\mathrm{on}^{"}$			

Affiche seulement "Caught	exception'
---------------------------	------------

#### Question 6 $\bigoplus$

```
Soit le code ci-dessous ; que se passe-t-il ?

public class Delta {
    public void throwup() throws Exception {
        throw new Exception();
    }
    public static void main(String[] args) {
        Delta delta = new Delta();
        try {
            delta.throwup();
    } catch (Exception e) {
            System.out.println("Caught exception");
            return;
    } finally {
            System.out.println("Finallied exception");
        }
}
```

# Affiche "Caught exception" et "Finallied exception"

#### Affiche seulement "Caught exception"

Affiche seulement "Finallied exception"

```
Erreur d'exécution
```

#### **Question 7** Soit le code :

```
interface Somethingable {
    void something();
}

interface Whateverable extends Somethingable {
    void whatever();
}

class TheThing implements Whateverable{
    PPP void something() {
        System.out.println("Somethinging");
    }

    PPP void whatever() {
        System.out.println("Whatevering");
    }
}
```

```
class Doofer {
   void doo(Somethingable sable) {
      sable.XXX();
}

void doo(Whateverable sable) {
    sable.YYY();
}

void fer(TheThing tt) {
    tt.ZZZ();
}

public static void main(String[] args) {
    Doofer doofer = new Doofer();
    TheThing the = new TheThing();
    doofer.doo(the);
    doofer.fer(the);
}
```

Par quels niveaux d'accès pourrait-on remplacer PPP?

 ☐ package-private
 ☐ private

 ☐ protected
 ☐ public

Question 8 ⊕ placer XXX ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
whatever	something
Question 9 $\bigoplus$ placer YYY ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
something	whatever
Question 10 $\bigoplus$ remplacer ZZZ ?	Pour le code de la question précédente, par quelles méthodes pourrait-on
something	whatever

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

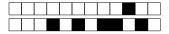
@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit





Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r\cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r\sin(\theta)$   $\theta = \tan^{-1}\frac{y}{x}$ 

En Java ce dernier est calculé par theta = Math.atan2(y, x)

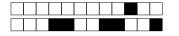
- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.





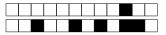
Question 13	Continuez votre réponse à la question précédente ici s	i nécessaire.	



Question 14	Continuez	votre réponse	à la question	précédente	ici si nécessaire.	



Question 15 Pour la question précédente, développez la méthode Polar#equals.  $\boxed{0}$   $\boxed{1}$   $\boxed{2}$   $\boxed{3}$ Une classe peut étendre une classe et implémenter plusieurs interfaces Une classe peut étendre une classe abstraite Une classe peut étendre plusieurs classes Une classe abstraite peut contenir des constructeurs Une classe abstraite peut implémenter une interface Une interface peut contenir des méthodes privées Une interface peut contenir des constructeurs



## Question 17 Soit le code ci-dessous :

```
@FunctionalInterface
interface ComparePerson {
    /**
    * @return negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

```
class Person {
    private int age;
    private String name;

Person(String name, int age) {
        this.age = age;
        this.name = name;
    }
}

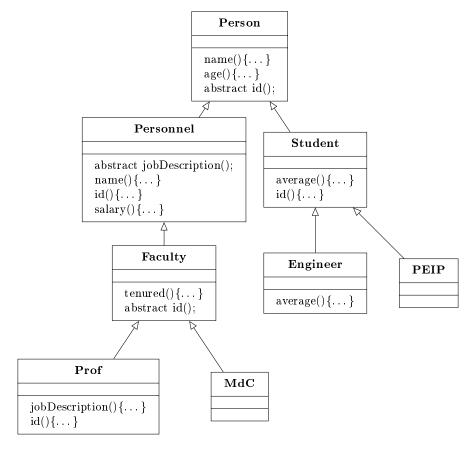
class Main {
    public static void main(String[] args) {
        // code extract here

        System.out.print(byAge.compare(
            new Person("Fred", 23),
            new Person("Not Fred", 24)));
        System.out.print(byAge.compare(
            new Person("Fred", 23),
            new Person("Fred", 23));
        System.out.print(byAge.compare(
            new Person("Not Fred", 23)));
        System.out.print(byAge.compare(
            new Person("Fred", 23),
            new Person("Fred", 23),
            new Person("Not Fred", 22)));
    }
}
```

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :



Question 18  $\bigoplus$  Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

MdC	Faculty
Personnel	Engineer
Prof	Person
PEIP	Student

+5/1/20+ QCMTEST Nom et prénom: Introduction à la programmation orientée objet 9/01/2020 Groupe: ..... Cochez les cases en mettant une  $\times$ . Le symbole  $\bigoplus$  indique que la question peut avoir zéro, une ou plusieurs bonnes réponses. Pour ces questions, cocher une bonne réponse apporte des points positifs ; cocher une mauvaise réponse peut apporter des points négatifs. Dans tout le code, les package et les import sont censés être correctement déclarés. Toute classe est supposée être dans le bon package, dans le bon fichier, avec les bons import.Question 1 La classe donnée class ClasseDonnee {} pourrait être instanciée par new pourrait être sous-classée par extends ClasseDonnee() ClasseDonnee La classe (enum) donnée Question 2 ( enum ClasseDonnee { INSTANCE; pourrait être sous-classée par extends pourrait être instanciée par new ClasseDonnee() ClasseDonnee Question 3 La classe donnée class ClasseDonnee {

# Question $4 \bigoplus$

private ClasseDonnee() {}

ClasseDonnee

Soit le code ci-dessous ; que se passe-t-il ?

public class Gamma {
 public void throwup() throws Exception {
 throw new Exception();
 }
 public static void main(String[] args) {
 Gamma gamma = new Gamma();
 try {
 gamma.throwup();
 } catch (Exception e) {
 System.out.println("Caught exception");
 } finally {
 System.out.println("Finallied exception");
 }
}

pourrait être sous-classée par extends

Erreur de compilation

ClasseDonnee()

être

pourrait

Affiche seulement "Caught exception"

Affiche "Caught exception" et "Finallied exception"

instanciée

new

Affiche seulement "Finallied exception"

Erreur d'exécution



# Question 5 $\bigoplus$

Erreur d'exécution Soit le code ci-dessous ; que se passe-t-il? Affiche "Done" public class Alpha { public void throwup() throws Exception { Erreur de compilation throw new Exception(); public static void main(String[] args) { Alpha alpha = new Alpha(); alpha.throwup(); System.out.println("Done"); Question 6 Affiche "Done" Soit le code ci-dessous ; que se passe-t-il? public class Beta { Erreur d'exécution public void throwup() throws RuntimeException { Erreur de compilation throw new RuntimeException(); public static void main(String[] args) { Beta beta = new Beta(); beta.throwup(); System.out.println("Done"); }

#### **Question 7** Soit le code :

interface Somethingable {
 void something();
}

interface Whateverable extends Somethingable {
 void whatever();
}

class TheThing implements Whateverable{
 PPP void something() {
 System.out.println("Somethinging");
 }

 PPP void whatever() {
 System.out.println("Whatevering");
 }
}

```
class Doofer {
   void doo(Somethingable sable) {
      sable.XXX();
   }

   void doo(Whateverable sable) {
      sable.YYY();
   }

   void fer(TheThing tt) {
      tt.ZZZ();
   }

   public static void main(String[] args) {
      Doofer doofer = new Doofer();
      TheThing the = new TheThing();
      doofer.doo(the);
      doofer.fer(the);
   }
}
```

Par quels niveaux d'accès pourrait-on remplacer PPP?

protected private	<pre>public package-private</pre>
Question 8 $\bigoplus$ placer XXX ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
something	whatever
Question 9 $\bigoplus$ placer YYY ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
whatever	something

+5/3/18+

Question 10 $\bigoplus$ remplacer ZZZ ?	Pour le code de la question précédente, par quelles méthodes pourrait-on
something	whatever

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
    public double re;
    public Complex(double a, double b) {
        re = a;
        im = b;
    }

    public double getRe() {
        return re;
    }

    public void setRe(double re) {
        this.re = re;
    }

    public double getIm() {
        return im;
    }

    public void setIm(double im) {
        this.im = im;
    }
}
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

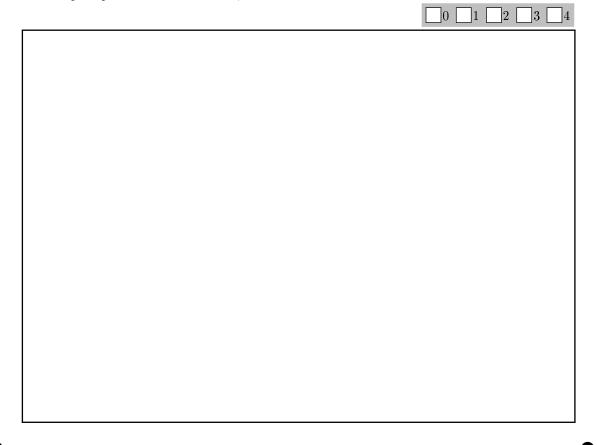
public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit





Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r \cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r \sin(\theta)$   $\theta = \tan^{-1} \frac{y}{x}$ 

En Java ce dernier est calculé par theta = Math.atan2(y, x)

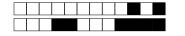
- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

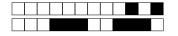
Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.

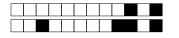




Question 13	Continuez	votre réponse	à la question	précédente i	ci si nécessaire.	0 0

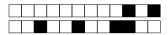


Question 14	Continuez	votre réponse à la question précédente ici si nécessaire.	



Une classe peut étendre plusieurs classes

Question	15	Pour	la	question	précédente,	développez	la	méth	node	Pol	Lar#	equa	ls.	
											]1 [	$\Box 2$ [	3	
Question	•				tions ci-desso menter une i		elle	s qui	$\operatorname{sont}$	vra	ies :			
					sse abstraite	HOCITACE								
_					nir des consti	ructeurs								
Une i	$_{ m nterface}$	peut	cor	ntenir des	méthodes pr	ivées								
Une i	$_{ m nterface}$	peut	cor	ntenir des	constructeur	's								
Une o	classe pe	$\mathrm{eut}$ ét $\epsilon$	end	re une cla	sse et implén	nenter plusie	eurs	inter	faces	3				



## Question $17 \oplus$ Soit le code ci-dessous :

```
@FunctionalInterface
interface ComparePerson {
    /**
    * @return negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

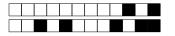
Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :

```
ComparePerson byAge = new ComparePersonByAge();

ComparePerson byAge = new ComparePersonByAge::compare;

ComparePerson byAge = (p1, p2) -> p1.age - p2.age;
```

```
ComparePerson byAge = new ComparePersonByAge() {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
};
```



Question 18  $\bigoplus$  Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

PEIP	Engineer
MdC	Personnel
Student	$oxedsymbol{oxed}$ Faculty
Prof	Person

QCMTEST Nom et prénom: Introduction à la programmation orientée objet 9/01/2020 Groupe: ..... Cochez les cases en mettant une  $\times$ . Le symbole  $\bigoplus$  indique que la question peut avoir zéro, une ou plusieurs bonnes réponses. Pour ces questions, cocher une bonne réponse apporte des points positifs ; cocher une mauvaise réponse peut apporter des points négatifs. Dans tout le code, les package et les import sont censés être correctement déclarés. Toute classe est supposée être dans le bon package, dans le bon fichier, avec les bons import.Question 1 La classe (enum) donnée enum ClasseDonnee { INSTANCE; } pourrait être pourrait être sous-classée par extends instanciée par new ClasseDonnee ClasseDonnee() Question 2 La classe donnée final class ClasseDonnee { ClasseDonnee() {} 7 pourrait être sous-classée par extends pourrait être instanciée par new ClasseDonnee() ClasseDonnee Question 3 ( La classe donnée class ClasseDonnee {} pourrait être sous-classée par extends pourrait être instanciée new ClasseDonnee ClasseDonnee() Question 4 Affiche "Caught exception" et "Finallied Soit le code ci-dessous ; que se passe-t-il? exception" public class Epsilon { public void throwup() throws RuntimeException { Erreur de compilation throw new RuntimeException(); Affiche seulement "Finallied exception"

Erreur d'exécution

Affiche seulement "Caught exception"

public static void main(String[] args) {
 Epsilon epsilon = new Epsilon();

System.out.println("Caught exception");

System.out.println("Finallied exception");

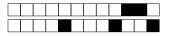
epsilon.throwup();

} catch (Exception e) {

finally {

try {

}



## Question 5 $\bigoplus$

Affiche "Done" Soit le code ci-dessous ; que se passe-t-il? public class Alpha { Erreur d'exécution public void throwup() throws Exception { Erreur de compilation throw new Exception(); public static void main(String[] args) { Alpha alpha = new Alpha(); alpha.throwup(); System.out.println("Done"); Question 6 Soit le code ci-dessous ; que se passe-t-il? Affiche seulement "Finallied exception" public class Gamma { Affiche "Caught exception" et "Finallied public void throwup() throws Exception { exception" throw new Exception(); Erreur d'exécution public static void main(String[] args) { Gamma gamma = new Gamma(); Erreur de compilation try { gamma.throwup(); Affiche seulement "Caught exception" } catch (Exception e) { System.out.println("Caught exception"); } finally { System.out.println("Finallied exception");

#### Question 7 Soit le code :

}

```
interface Somethingable {
    void something();
}

interface Whateverable extends Somethingable {
    void whatever();
}

class TheThing implements Whateverable{
    PPP void something() {
        System.out.println("Somethinging");
    }

    PPP void whatever() {
        System.out.println("Whatevering");
    }
}
```

```
class Doofer {
   void doo(Somethingable sable) {
      sable.XXX();
}

void doo(Whateverable sable) {
   sable.YYY();
}

void fer(TheThing tt) {
   tt.ZZZ();
}

public static void main(String[] args) {
   Doofer doofer = new Doofer();
   TheThing the = new TheThing();
   doofer.doo(the);
   doofer.fer(the);
}
```

Par quels niveaux d'accès pourrait-on remplacer PPP?

public	protected
$\square$ private	package-private
Question 8 $\oplus$ placer XXX ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
whatever	something

Question 9 ⊕ placer YYY ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
whatever	something
Question 10 $\bigoplus$ remplacer ZZZ ?	Pour le code de la question précédente, par quelles méthodes pourrait-on

 $\square$  something

\_\_\_ whatever

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

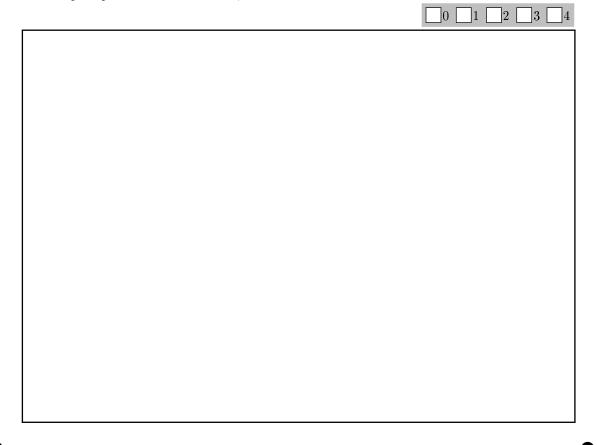
public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

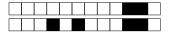
@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit





Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r \cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r \sin(\theta)$   $\theta = \tan^{-1} \frac{y}{x}$ 

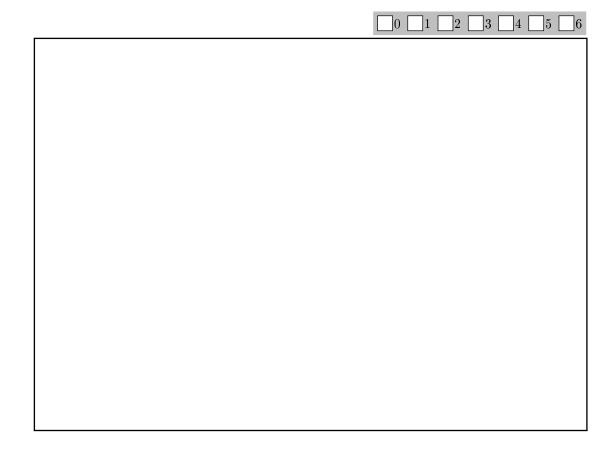
En Java ce dernier est calculé par theta = Math.atan2(y, x)

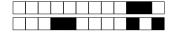
- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
       return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.

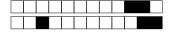




Question 13	Continuez votre réponse à la question précédente ici si nécessaire.	0 0



Question 14	Continuez	votre réponse	à la question	précédente ici	si nécessaire.	



Une classe peut étendre une classe abstraite

Question	15	Pour	la	question	précédente,	dávoloppoz	la	máthada	Dola:	r#ogu	ale	
Question	10	1 Oui	ıa	question	precedence,	developpez	ıa		) \bigcip 1			$\Box_4$
${f Question}$	16 ⊕	Parr	ni l	es affirma	tions ci-desso	ous, cocher c	ell∈	s qui sont	vraie	5:		
Une i	$_{ m nterface}$	e peut	col	ntenir des	constructeur	rs.						
Une o	classe p	eut éte	end	re plusieu	rs classes							
Une o	classe al	bstrait	ер	eut implé	menter une i	$\operatorname{nterface}$						
Une o	classe al	bstrait	ер	eut conte	nir des const	ructeurs						
		_			méthodes pr							
Une o	classe p	eut éte	$\operatorname{end}$	re une cla	sse et implén	${ m nenter}$ plusi $\epsilon$	eurs	interfaces	3			



## Question $17 \oplus$ Soit le code ci-dessous :

```
@FunctionalInterface
interface ComparePerson {
    /**
    * Oreturn negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

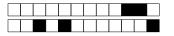
class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :

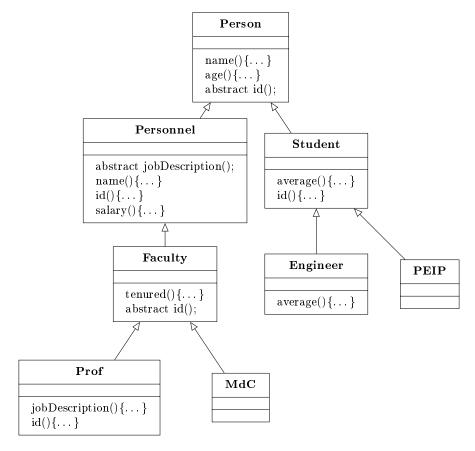
```
ComparePerson byAge = new ComparePersonByAge();

ComparePerson byAge = new ComparePersonByAge() {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
};

ComparePerson byAge = new ComparePersonByAge() {
    ComparePerson byAge = (p1, p2) -> p1.age - p2.age;
}
```



Question 18 Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

Student	Engineer
Person	Prof
MdC	Personnel
Faculty	PEIP

QCMTEST Nom et prénom: Introduction à la programmation orientée objet 9/01/2020 Groupe: ..... Cochez les cases en mettant une  $\times$ . Le symbole  $\bigoplus$  indique que la question peut avoir zéro, une ou plusieurs bonnes réponses. Pour ces questions, cocher une bonne réponse apporte des points positifs ; cocher une mauvaise réponse peut apporter des points négatifs. Dans tout le code, les package et les import sont censés être correctement déclarés. Toute classe est supposée être dans le bon package, dans le bon fichier, avec les bons import.Question 1 La classe donnée class ClasseDonnee { private ClasseDonnee() {} pourrait pourrait être sous-classée par extends être instanciée new ClasseDonnee ClasseDonnee() Question 2 ( La classe donnée abstract class ClasseDonnee { ClasseDonnee() {} pourrait pourrait être sous-classée par extends être instanciée par new ClasseDonnee() ClasseDonnee Question 3 ( La classe donnée class ClasseDonnee {} pourrait être sous-classée par extends pourrait être instanciée par new ClasseDonnee() ClasseDonnee Question 4 Erreur de compilation Soit le code ci-dessous ; que se passe-t-il?

Erreur d'exécution

Affiche "Done"

public class Beta {

}

public void throwup() throws RuntimeException {

throw new RuntimeException();
}
public static void main(String[] args) {
Beta beta = new Beta();
beta.throwup();

System.out.println("Done");

# Question 5 $\bigoplus$

```
Affiche seulement "Caught exception"
Soit le code ci-dessous ; que se passe-t-il?
public class Delta {
                                                            Affiche seulement "Finallied exception"
   public void throwup() throws Exception {
                                                            Erreur de compilation
      throw new Exception();
   public static void main(String[] args) {
                                                            Erreur d'exécution
      Delta delta = new Delta();
      try {
                                                             Affiche "Caught exception" et "Finallied
          delta.throwup();
                                                             exception"
      } catch (Exception e) {
          System.out.println("Caught exception");
          return:
      } finally {
          System.out.println("Finallied exception");
   }
}
Question 6
```

```
Affiche "Caught exception" et "Finallied
Soit le code ci-dessous ; que se passe-t-il?
                                                             exception"
public class Epsilon {
   public void throwup() throws RuntimeException {
                                                             Erreur de compilation
      throw new RuntimeException();
                                                             Erreur d'exécution
   public static void main(String[] args) {
      Epsilon epsilon = new Epsilon();
                                                             Affiche seulement "Finallied exception"
      try {
          epsilon.throwup();
                                                             Affiche seulement "Caught exception"
      } catch (Exception e) {
          System.out.println("Caught exception");
      } finally {
         System.out.println("Finallied exception");
   }
}
```

#### **Question 7** Soit le code :

```
interface Somethingable {
    void something();
}

interface Whateverable extends Somethingable {
    void whatever();
}

class TheThing implements Whateverable{
    PPP void something() {
        System.out.println("Somethinging");
    }

    PPP void whatever() {
        System.out.println("Whatevering");
    }
}
```

```
class Doofer {
  void doo(Somethingable sable) {
     sable.XXX();
  }

  void doo(Whateverable sable) {
     sable.YYY();
  }

  void fer(TheThing tt) {
     tt.ZZZ();
  }

  public static void main(String[] args) {
     Doofer doofer = new Doofer();
     TheThing the = new TheThing();
     doofer.doo(the);
     doofer.fer(the);
  }
}
```

Par quels niveaux d'accès pourrait-on remplacer PPP?

```
        ☐ public
        ☐ protected

        ☐ package-private
        ☐ private
```

Question 8 $\oplus$ placer XXX ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
whatever	something
Question 9 $\bigoplus$ placer YYY ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
whatever	something
Question 10 $\bigoplus$ remplacer ZZZ ?	Pour le code de la question précédente, par quelles méthodes pourrait-on

whatever

something

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

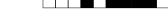
@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit





Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r \cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r \sin(\theta)$   $\theta = \tan^{-1} \frac{y}{x}$ 

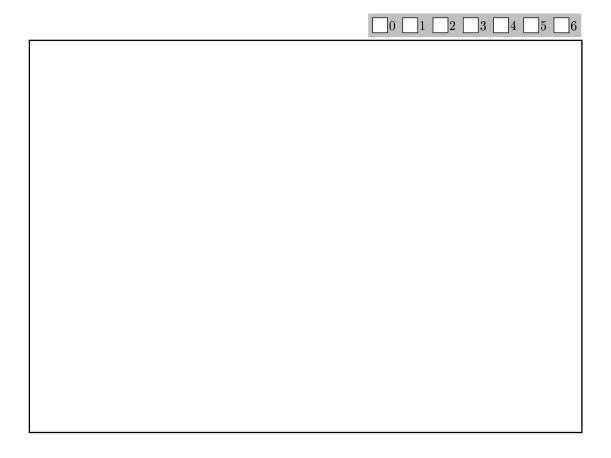
En Java ce dernier est calculé par theta = Math.atan2(y, x)

- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

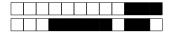
Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.





${f Question}$	13	Continuez	votre	réponse	à la	question	précéde	nte ici s	si nécessair	re. 0 0



Question 14	Continuez	votre réponse à l	la question pré	cédente ici s	si nécessaire.	0 0



Une interface peut contenir des constructeurs

${f Question}$	15	Pour l	a question	précédente,	développez	la 1	méthode		equals.
${f Question}$	<b>16</b> ⊕	Parmi	les affirma	tions ci-dess	ous, cocher c	elles	qui sont	vraies :	
Une o	classe pe	eut éten	dre plusieu	rs classes					
Une i	interface	e peut c	ontenir des	méthodes p	rivées				
	=			sse abstraite					
Une o	classe ab	straite	peut conte	nir des const	ructeurs				
Une o	classe pe	eut éten	dre une cla	sse et implér	menter plusie	eurs	interfaces	3	
Une o	classe ab	$_{ m straite}$	peut implé	menter une i	$_{ m nterface}$				



## Question $17 \oplus$ Soit le code ci-dessous :

```
@FunctionalInterface
interface ComparePerson {
    /**
    * Oreturn negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :

```
ComparePerson byAge = new ComparePersonByAge();

ComparePerson byAge = new ComparePersonByAge() {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
};

ComparePerson byAge = new ComparePersonByAge() {
    ComparePerson byAge = (p1, p2) -> p1.age - p2.age;
}
```



# Question 18 Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

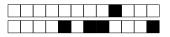
Person	$\Box$ Faculty
Prof	MdC
Personnel	PEIP
Student	Engineer

QCMTEST Nom et prénom: Introduction à la programmation orientée objet 9/01/2020 Groupe: ..... Cochez les cases en mettant une  $\times$ . Le symbole  $\bigoplus$  indique que la question peut avoir zéro, une ou plusieurs bonnes réponses. Pour ces questions, cocher une bonne réponse apporte des points positifs ; cocher une mauvaise réponse peut apporter des points négatifs. Dans tout le code, les package et les import sont censés être correctement déclarés. Toute classe est supposée être dans le bon package, dans le bon fichier, avec les bons import.Question 1 ( La classe (enum) donnée enum ClasseDonnee { INSTANCE; pourrait être sous-classée par extends pourrait être instanciée par new ClasseDonnee ClasseDonnee() Question 2 La classe donnée final class ClasseDonnee { ClasseDonnee() {} } pourrait être sous-classée par extends pourrait être instanciée par new ClasseDonnee ClasseDonnee() Question 3 ( La classe donnée class ClasseDonnee { private ClasseDonnee() {} pourrait être sous-classée par extends pourrait être instanciée par new ClasseDonnee() ClasseDonnee Question 4 Erreur de compilation Soit le code ci-dessous ; que se passe-t-il? public class Alpha { Erreur d'exécution public void throwup() throws Exception { Affiche "Done" throw new Exception(); public static void main(String[] args) {

Alpha alpha = new Alpha();
alpha.throwup();

System.out.println("Done");

}



## Question 5 $\bigoplus$

Affiche seulement "Caught exception" Soit le code ci-dessous ; que se passe-t-il? public class Gamma { Erreur d'exécution public void throwup() throws Exception { Affiche "Caught exception" et "Finallied throw new Exception(); public static void main(String[] args) { Gamma gamma = new Gamma(); Erreur de compilation try { gamma.throwup(); Affiche seulement "Finallied exception" } catch (Exception e) { System.out.println("Caught exception"); } finally { System.out.println("Finallied exception"); } } Question 6  $\bigoplus$ Affiche "Done" Soit le code ci-dessous ; que se passe-t-il? Erreur d'exécution public class Beta { public void throwup() throws RuntimeException {

#### Question 7 Soit le code :

whatever

beta.throwup();

}

throw new RuntimeException();

System.out.println("Done");

public static void main(String[] args) {
 Beta beta = new Beta();

```
interface Somethingable {
    void something();
}

interface Whateverable extends Somethingable {
    void whatever();
}

class TheThing implements Whateverable{
    PPP void something() {
        System.out.println("Somethinging");
    }

    PPP void whatever() {
        System.out.println("Whatevering");
    }
}
```

```
class Doofer {
    void doo(Somethingable sable) {
        sable.XXX();
    }

    void doo(Whateverable sable) {
        sable.YYY();
    }

    void fer(TheThing tt) {
        tt.ZZZ();
    }

    public static void main(String[] args) {
        Doofer doofer = new Doofer();
        TheThing the = new TheThing();
        doofer.doo(the);
        doofer.fer(the);
    }
}
```

Erreur de compilation

Par quels niveaux d'accès pourrait-on remplacer PPP?

package-priv	rate public
protected	private
Question 8 $\bigoplus$ placer XXX ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-

something

+8/3/48	8+
---------	----

Question 9 $\bigoplus$ placer YYY ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
whatever	<pre>something</pre>
Question 10 $\bigoplus$ remplacer ZZZ ?	Pour le code de la question précédente, par quelles méthodes pourrait-on
something	whatever

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit



Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r\cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r\sin(\theta)$   $\theta = \tan^{-1}\frac{y}{x}$ 

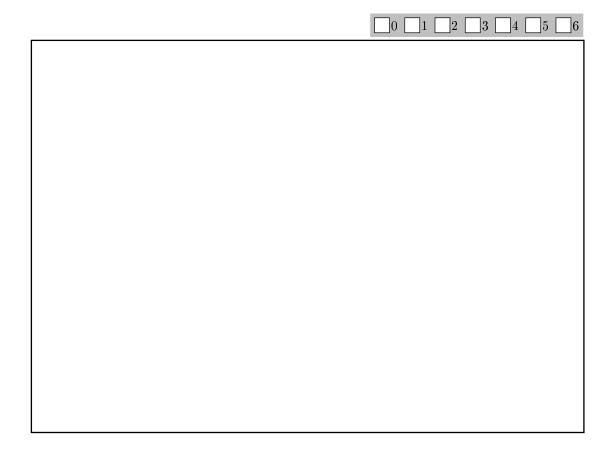
En Java ce dernier est calculé par theta = Math.atan2(y, x)

- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
       return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.





Question	13	Continuez	votre	réponse	à la qu	estion	précédent	e ici s	i nécessaire.	

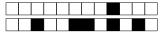


Question 14	Continuez	votre réponse à la question précédente ici si nécessaire.	



Une classe abstraite peut contenir des constructeurs

Question	<b>15</b>	Pour	la	question	précédente,	développez	la	méthode	Pola	r#equ	als.	
										$\square 2$		$\Box 4$
${f Question}$	16 <b>⊕</b>	Parr	ni l	es affirma	tions ci-desso	ous, cocher c	elle	s qui sont	vraies	3:		
Une o	classe p	eut éte	end	re plusieu	rs classes							
Une o	classe p	eut éte	end	re une cla	sse abstraite							
Une i	interfac	e peut	coı	ntenir des	méthodes pr	rivées						
Une i	interfac	e peut	coı	ntenir des	constructeur	`S						
Une o	classe a	bstrait	ер	eut implé	menter une i	$\operatorname{nterface}$						
Une o	classe p	eut éte	end	re une cla	sse et implér	nenter plusie	eurs	interfaces	1			



# Question $17 \oplus$ Soit le code ci-dessous :

```
@FunctionalInterface
interface ComparePerson {
    /**
    * @return negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :

```
ComparePerson byAge = new ComparePersonByAge::compare;

ComparePerson byAge = (p1, p2) -> p1.age - p2.age;

ComparePerson byAge = new ComparePersonByAge();
```

```
ComparePerson byAge = new ComparePersonByAge() {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
};
```



Question 18  $\bigoplus$  Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

MdC	Prof
PEIP	Person
Student	Engineer
Personnel	$\Box$ Faculty

new

QCMTEST Nom et prénom: Introduction à la programmation orientée objet 9/01/2020 Groupe: ..... Cochez les cases en mettant une  $\times$ . Le symbole  $\bigoplus$  indique que la question peut avoir zéro, une ou plusieurs bonnes réponses. Pour ces questions, cocher une bonne réponse apporte des points positifs ; cocher une mauvaise réponse peut apporter des points négatifs. Dans tout le code, les package et les import sont censés être correctement déclarés. Toute classe est supposée être dans le bon package, dans le bon fichier, avec les bons import.Question 1 

La classe donnée abstract class ClasseDonnee { ClasseDonnee() {} pourrait pourrait être sous-classée par extends être instanciée new ClasseDonnee ClasseDonnee() Question 2 La classe (enum) donnée enum ClasseDonnee { INSTANCE; } pourrait être sous-classée par extends pourrait être instanciée par new ClasseDonnee ClasseDonnee() Question 3 ( La classe donnée final class ClasseDonnee { ClasseDonnee() {} pourrait pourrait être sous-classée par extends être instanciée par

ClasseDonnee()

ClasseDonnee

Erreur d'exécution Soit le code ci-dessous ; que se passe-t-il? public class Delta { Affiche seulement "Finallied exception" public void throwup() throws Exception { Affiche "Caught exception" et "Finallied throw new Exception(); public static void main(String[] args) { Delta delta = new Delta(); Affiche seulement "Caught exception" try { delta.throwup(); Erreur de compilation } catch (Exception e) { System.out.println("Caught exception"); return; } finally { System.out.println("Finallied exception"); } Question 5

# Soit le code ci-dessous ; que se passe-t-il? public class Alpha {

public void throwup() throws Exception { throw new Exception(); public static void main(String[] args) { Alpha alpha = new Alpha(); alpha.throwup(); System.out.println("Done");

# Question 6

Soit le code ci-dessous ; que se passe-t-il? public class Epsilon { public void throwup() throws RuntimeException { throw new RuntimeException(); public static void main(String[] args) {
 Epsilon epsilon = new Epsilon(); try { epsilon.throwup(); } catch (Exception e) { System.out.println("Caught exception"); } finally { System.out.println("Finallied exception"); }

Erreur d'exécution

Erreur de compilation

Affiche "Done"

Affiche seulement "Finallied exception"

Erreur d'exécution

Erreur de compilation

Affiche "Caught exception" et "Finallied exception"

Affiche seulement "Caught exception"



## Question 7 Soit le code :

```
interface Somethingable {
   void something();
}

interface Whateverable extends Somethingable {
   void whatever();
}

class TheThing implements Whateverable{
   PPP void something() {
       System.out.println("Somethinging");
   }

   PPP void whatever() {
       System.out.println("Whatevering");
   }
}
```

```
class Doofer {
    void doo(Somethingable sable) {
        sable.XXX();
    }

    void doo(Whateverable sable) {
        sable.YYY();
    }

    void fer(TheThing tt) {
        tt.ZZZ();
    }

    public static void main(String[] args) {
        Doofer doofer = new Doofer();
        TheThing the = new TheThing();
        doofer.doo(the);
        doofer.fer(the);
    }
}
```

Par quels niveaux d'accès pourrait-on remplacer PPP ?

public private

protected package-private

**Question 8**  $\bigoplus$  Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer XXX ?

whatever something

**Question 9**  $\bigoplus$  Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer YYY ?

something whatever

**Question 10**  $\bigoplus$  Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer ZZZ ?

something whatever



Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
}
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

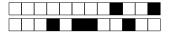
@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit





Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r\cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r\sin(\theta)$   $\theta = \tan^{-1}\frac{y}{x}$ 

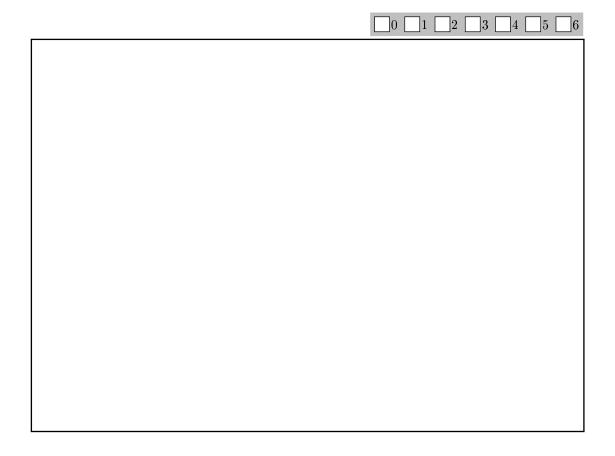
En Java ce dernier est calculé par theta = Math.atan2(y, x)

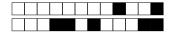
- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

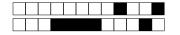
Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.

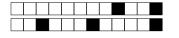




Question 13	Continuez votre réponse à la question précédente ici si nécessaire.	



Question 14	Continuez votre réponse à la question précédente ici si nécessaire.	



Une classe peut étendre une classe abstraite

Question 15 Pour la question précédente, développez la méthode Polar#equals.  $\boxed{0}$   $\boxed{1}$   $\boxed{2}$   $\boxed{3}$ Une classe abstraite peut implémenter une interface Une classe peut étendre plusieurs classes Une classe peut étendre une classe et implémenter plusieurs interfaces Une classe abstraite peut contenir des constructeurs Une interface peut contenir des méthodes privées Une interface peut contenir des constructeurs



# Question $17 \oplus$ Soit le code ci-dessous :

```
@FunctionalInterface
interface ComparePerson {
    /**
    * @return negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

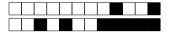
Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :

```
ComparePerson byAge = new ComparePersonByAge() {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
};

ComparePerson byAge = new ComparePersonByAge::compare;

ComparePerson byAge = (p1, p2) -> p1.age - p2.age;

ComparePerson byAge = (p1, p2) -> p1.age - p2.age;
```



Question 18  $\bigoplus$  Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

Person	Personnel
Prof	Student
PEIP	$oxedsymbol{oxed}$ Faculty
Engineer	MdC

QCMTEST Nom et prénom: Introduction à la programmation orientée objet 9/01/2020 Groupe: ..... Cochez les cases en mettant une  $\times$ . Le symbole  $\bigoplus$  indique que la question peut avoir zéro, une ou plusieurs bonnes réponses. Pour ces questions, cocher une bonne réponse apporte des points positifs ; cocher une mauvaise réponse peut apporter des points négatifs. Dans tout le code, les package et les import sont censés être correctement déclarés. Toute classe est supposée être dans le bon package, dans le bon fichier, avec les bons import.Question 1 La classe donnée class ClasseDonnee { private ClasseDonnee() {} pourrait être sous-classée par extends pourrait être instanciée par new ClasseDonnee() ClasseDonnee Question 2 La classe donnée class ClasseDonnee {} pourrait être sous-classée par extends pourrait être instanciée par new ClasseDonnee ClasseDonnee() Question 3  $\bigoplus$ La classe donnée abstract class ClasseDonnee { ClasseDonnee() {} pourrait être sous-classée par extends pourrait être instanciée new ClasseDonnee ClasseDonnee() Question 4

Soit le code ci-dessous ; que se passe-t-il?

throw new RuntimeException();
}
public static void main(String[] args) {

Beta beta = new Beta();
beta.throwup();

System.out.println("Done");

public void throwup() throws RuntimeException {

public class Beta {

}

Affiche "Done"

Erreur de compilation

Erreur d'exécution

# Question 5 $\bigoplus$

```
Soit le code ci-dessous ; que se passe-t-il ?

public class Gamma {
    public void throwup() throws Exception {
        throw new Exception();
    }

    public static void main(String[] args) {
        Gamma gamma = new Gamma();
        try {
            gamma.throwup();
    } catch (Exception e) {
            System.out.println("Caught exception");
    } finally {
            System.out.println("Finallied exception");
        }
    }
}
```

# Erreur de compilation

\_ Erreur d'exécution

Affiche seulement "Caught exception"

Affiche seulement "Finallied exception"

Affiche "Caught exception" et "Finallied exception"

# Question 6 $\bigoplus$

```
Soit le code ci-dessous ; que se passe-t-il ?

public class Delta {
    public void throwup() throws Exception {
        throw new Exception();
    }

    public static void main(String[] args) {
        Delta delta = new Delta();
        try {
            delta.throwup();
    } catch (Exception e) {
            System.out.println("Caught exception");
            return;
    } finally {
            System.out.println("Finallied exception");
        }
}
```

# Affiche seulement "Finallied exception"

Erreur de compilation

Affiche "Caught exception" et "Finallied exception"

Affiche seulement "Caught exception"

Erreur d'exécution

#### **Question 7** Soit le code :

```
interface Somethingable {
    void something();
}

interface Whateverable extends Somethingable {
    void whatever();
}

class TheThing implements Whateverable{
    PPP void something() {
        System.out.println("Somethinging");
    }

    PPP void whatever() {
        System.out.println("Whatevering");
    }
}
```

```
class Doofer {
   void doo(Somethingable sable) {
       sable.XXX();
   }

   void doo(Whateverable sable) {
       sable.YYY();
   }

   void fer(TheThing tt) {
       tt.ZZZ();
   }

   public static void main(String[] args) {
       Doofer doofer = new Doofer();
       TheThing the = new TheThing();
       doofer.doo(the);
       doofer.fer(the);
   }
}
```

Par quels niveaux d'accès pourrait-on remplacer PPP?

□ protected
□ public

□ package-private
□ private

Question 8 $\bigoplus$ placer XXX ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
something	whatever
Question 9 $\bigoplus$ placer YYY ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
something	whatever
Question 10 ⊕ remplacer ZZZ ?	Pour le code de la question précédente, par quelles méthodes pourrait-on
Whatever	something

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

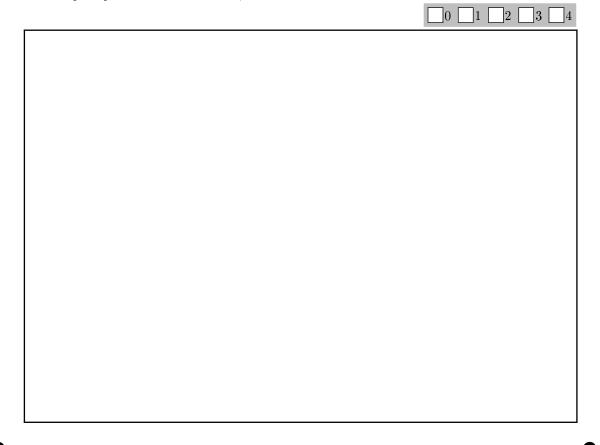
public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit





Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r \cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r \sin(\theta)$   $\theta = \tan^{-1} \frac{y}{x}$ 

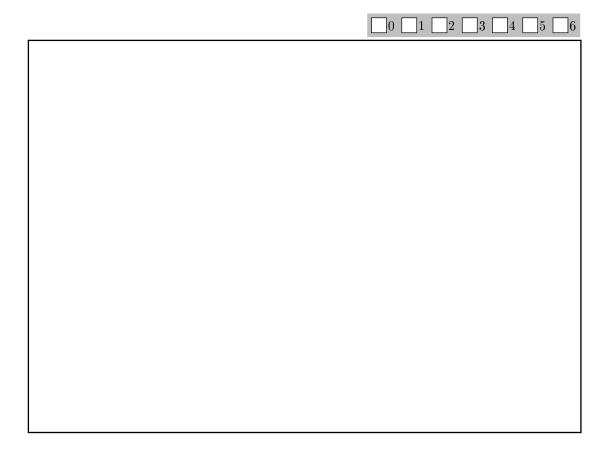
En Java ce dernier est calculé par theta = Math.atan2(y, x)

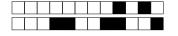
- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.





Question 13	Continuez votre réponse à la question précédente ici si nécessaire.	



Question 14	Continuez	votre réponse	à la question	précédente ici	si nécessaire.	



Une classe peut étendre plusieurs classes

Une classe abstraite peut contenir des constructeurs

Question 15 Pour la question précédente, développez la méthode Polar#equals.  $\boxed{0}$   $\boxed{1}$   $\boxed{2}$   $\boxed{3}$ Une classe peut étendre une classe abstraite Une interface peut contenir des constructeurs Une classe abstraite peut implémenter une interface Une classe peut étendre une classe et implémenter plusieurs interfaces Une interface peut contenir des méthodes privées



# Question $17 \oplus$ Soit le code ci-dessous :

```
@FunctionalInterface
interface ComparePerson {
    /**
    * Oreturn negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :

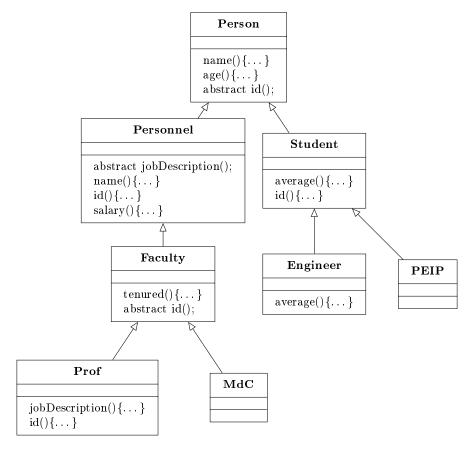
```
ComparePerson byAge = (p1, p2) -> p1.age - p2.age;

ComparePerson byAge = new ComparePersonByAge() {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
};

ComparePerson byAge = new ComparePersonByAge();
```



# Question 18 $\bigoplus$ Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

<pre>Faculty</pre>	PEIP
Personnel	Student
Person	MdC
Engineer	Prof

Affiche seulement "Caught exception"

Affiche "Caught exception" et "Finallied

Affiche seulement "Finallied exception"

Erreur d'exécution

Erreur de compilation

QCMTEST Nom et prénom: Introduction à la programmation orientée objet 9/01/2020 Groupe: ..... Cochez les cases en mettant une  $\times$ . Le symbole  $\bigoplus$  indique que la question peut avoir zéro, une ou plusieurs bonnes réponses. Pour ces questions, cocher une bonne réponse apporte des points positifs ; cocher une mauvaise réponse peut apporter des points négatifs. Dans tout le code, les package et les import sont censés être correctement déclarés. Toute classe est supposée être dans le bon package, dans le bon fichier, avec les bons import.Question 1 La classe (enum) donnée enum ClasseDonnee { INSTANCE; } pourrait être sous-classée par extends pourrait être instanciée par new ClasseDonnee ClasseDonnee() Question 2 La classe donnée class ClasseDonnee { private ClasseDonnee() {} } pourrait être sous-classée par extends pourrait être instanciée par new ClasseDonnee ClasseDonnee() Question 3 ( La classe donnée class ClasseDonnee {} pourrait être sous-classée par extends pourrait être instanciée new ClasseDonnee() ClasseDonnee Question 4

Soit le code ci-dessous ; que se passe-t-il ?
public class Epsilon {
 public void throwup() throws RuntimeException {
 throw new RuntimeException();
 }
 public static void main(String[] args) {
 Epsilon epsilon = new Epsilon();
 try {
 epsilon.throwup();
 } catch (Exception e) {
 System.out.println("Caught exception");
 } finally {
 System.out.println("Finallied exception");
 }
}



Affiche "Done" Soit le code ci-dessous ; que se passe-t-il? public class Alpha { Erreur de compilation public void throwup() throws Exception { Erreur d'exécution throw new Exception(); public static void main(String[] args) { Alpha alpha = new Alpha(); alpha.throwup(); System.out.println("Done"); Question 6 Soit le code ci-dessous ; que se passe-t-il? Affiche "Caught exception" et "Finallied exception" public class Delta { public void throwup() throws Exception { Erreur de compilation throw new Exception(); Affiche seulement "Finallied exception" public static void main(String[] args) { Delta delta = new Delta(); Affiche seulement "Caught exception" try { delta.throwup(); Erreur d'exécution } catch (Exception e) { System.out.println("Caught exception"); return; } finally { System.out.println("Finallied exception"); } }

## Question 7 Soit le code :

```
interface Somethingable {
   void something();
}

interface Whateverable extends Somethingable {
   void whatever();
}

class TheThing implements Whateverable{
   PPP void something() {
       System.out.println("Somethinging");
   }

   PPP void whatever() {
       System.out.println("Whatevering");
   }
}
```

```
class Doofer {
    void doo(Somethingable sable) {
        sable.XXX();
}

void doo(Whateverable sable) {
        sable.YYY();
}

void fer(TheThing tt) {
        tt.ZZZ();
}

public static void main(String[] args) {
        Doofer doofer = new Doofer();
        TheThing the = new TheThing();
        doofer.doo(the);
        doofer.fer(the);
}
```

Par quels niveaux d'accès pourrait-on remplacer PPP ?

public	private
protected	package-private
Question 8 $\bigoplus$ placer XXX ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
whatever	something

Question 9 ⊕ placer YYY ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
something	whatever
$\begin{array}{c} \mathbf{Question}  10 \ \bigoplus \\ \mathbf{remplacer} \ \mathbf{ZZZ} \ ? \end{array}$	Pour le code de la question précédente, par quelles méthodes pourrait-on
something	whatever

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit





Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r\cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r\sin(\theta)$   $\theta = \tan^{-1}\frac{y}{x}$ 

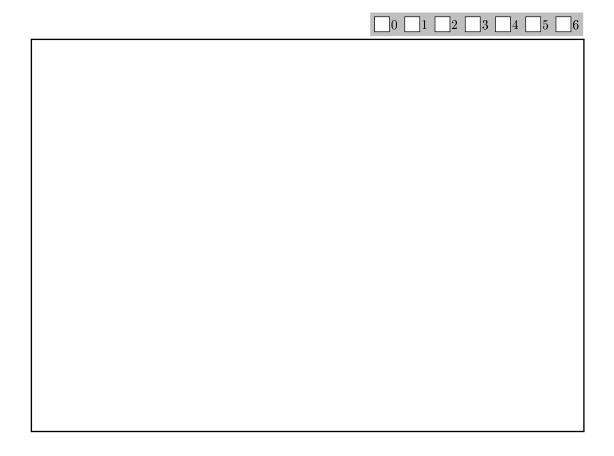
En Java ce dernier est calculé par theta = Math.atan2(y, x)

- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

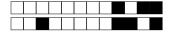
Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.



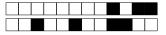
Question 13	Continuez votre réponse à la question précédente ici si nécessaire.	
1		



0



Question	15	Pour	la	question	précédente,	développez	la		Polar#	equals.
Question	•				tions ci-desso		elles	qui sont	vraies :	
	=				sse abstratte sse et implér		eurs	interface	s	
=	_				constructeur	=				
Une o	classe ab	ostrait	ер	eut contei	nir des const	ructeurs				
Une o	classe pe	eut éte	endi	re plusieu	rs classes					
Une i	$\operatorname{nterfac}\epsilon$	e peut	cor	ntenir des	méthodes pr	rivées				
Une o	classe ab	ostrait	ер	eut implé	menter une i	nterface				



# Question $17 \oplus$ Soit le code ci-dessous :

```
@FunctionalInterface
interface ComparePerson {
    /**
    * @return negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :

```
ComparePerson byAge = (p1, p2) -> p1.age - p2.age;

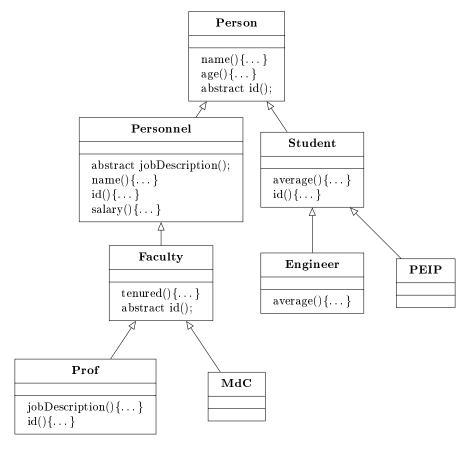
ComparePerson byAge = new ComparePersonByAge();

ComparePerson byAge = new ComparePersonByAge::compare;
```

```
ComparePerson byAge = new ComparePersonByAge() {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
};
```



# Question 18 $\bigoplus$ Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

MdC	Faculty
Student	Person
Personnel	Prof
PEIP	Engineer

QCMTEST Nom et prénom: Introduction à la programmation orientée objet 9/01/2020 Groupe: ..... Cochez les cases en mettant une  $\times$ . Le symbole  $\bigoplus$  indique que la question peut avoir zéro, une ou plusieurs bonnes réponses. Pour ces questions, cocher une bonne réponse apporte des points positifs ; cocher une mauvaise réponse peut apporter des points négatifs. Dans tout le code, les package et les import sont censés être correctement déclarés. Toute classe est supposée être dans le bon package, dans le bon fichier, avec les bons import.Question 1 

La classe donnée final class ClasseDonnee { ClasseDonnee() {} pourrait être sous-classée par extends pourrait être instanciée par new ClasseDonnee() ClasseDonnee Question 2 La classe donnée abstract class ClasseDonnee { ClasseDonnee() {} } pourrait être sous-classée par extends pourrait être instanciée par new ClasseDonnee ClasseDonnee() Question 3 ( La classe donnée class ClasseDonnee { private ClasseDonnee() {} pourrait être sous-classée par extends pourrait être instanciée par new ClasseDonnee() ClasseDonnee Question 4 Erreur d'exécution Soit le code ci-dessous ; que se passe-t-il? public class Beta { Affiche "Done"

Erreur de compilation

public void throwup() throws RuntimeException {

throw new RuntimeException();

System.out.println("Done");

Beta beta = new Beta();
beta.throwup();

}

public static void main(String[] args) {



```
Soit le code ci-dessous ; que se passe-t-il?
                                                            Erreur de compilation
public class Gamma {
                                                            Affiche "Caught exception" et "Finallied
   public void throwup() throws Exception {
                                                            exception"
      throw new Exception();
                                                        Affiche seulement "Finallied exception"
   public static void main(String[] args) {
      Gamma gamma = new Gamma();
                                                            Affiche seulement "Caught exception"
      try {
         gamma.throwup();
                                                            Erreur d'exécution
      } catch (Exception e) {
         System.out.println("Caught exception");
      } finally {
         System.out.println("Finallied exception");
}
Question 6 \bigoplus
                                                            Affiche seulement "Caught exception"
Soit le code ci-dessous ; que se passe-t-il?
```

```
Affiche "Caught exception" et "Finallied
public class Delta {
   public void throwup() throws Exception {
                                                                exception"
      throw new Exception();
                                                               Erreur de compilation
   public static void main(String[] args) {
      Delta delta = new Delta();
                                                                Affiche seulement "Finallied exception"
       try {
          delta.throwup();
       } catch (Exception e) {
          System.out.println("Caught exception");
          return;
      } finally {
   System.out.println("Finallied exception");
```

# Erreur d'exécution

#### Question 7 Soit le code:

```
interface Somethingable {
   void something();
}
interface Whateverable extends Somethingable {
   void whatever();
class TheThing implements Whateverable{
   PPP void something() {
       System.out.println("Somethinging");
   PPP void whatever() {
       System.out.println("Whatevering");
}
```

```
class Doofer {
   void doo(Somethingable sable) {
        sable.XXX();
    void doo(Whateverable sable) {
        sable.YYY();
    void fer(TheThing tt) {
       tt.ZZZ();
   public static void main(String[] args) {
       Doofer doofer = new Doofer();
TheThing the = new TheThing();
        doofer.doo(the);
        doofer.fer(the);
}
```

Par quels niveaux d'accès pourrait-on remplacer PPP?

private	public
package-private	protecte

Question 8 $\bigoplus$ placer XXX ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
something	whatever
Question 9 $\bigoplus$ placer YYY ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
something	whatever
Question 10 $\bigoplus$ remplacer ZZZ ?	Pour le code de la question précédente, par quelles méthodes pourrait-on
something	whatever

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
}
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

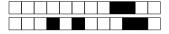
@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit





Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r \cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r \sin(\theta)$   $\theta = \tan^{-1} \frac{y}{x}$ 

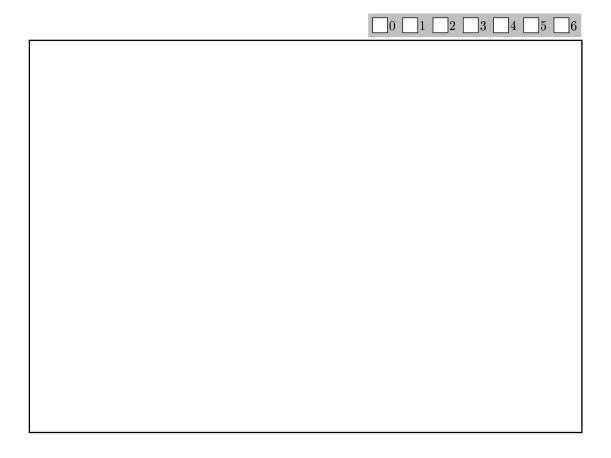
En Java ce dernier est calculé par theta = Math.atan2(y, x)

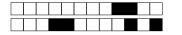
- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

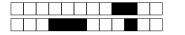
Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.





estion 13	Continuez votre réponse à la question précédente ici si nécessaire.	



uestion 14	Continuez votre réponse à la question précédente ici si nécessaire.	



Question 15 Pour la question précédente, développez la méthode Polar#equals.  $\boxed{0}$   $\boxed{1}$   $\boxed{2}$   $\boxed{3}$ Une classe abstraite peut implémenter une interface Une classe peut étendre plusieurs classes Une interface peut contenir des constructeurs Une interface peut contenir des méthodes privées Une classe abstraite peut contenir des constructeurs Une classe peut étendre une classe abstraite

Une classe peut étendre une classe et implémenter plusieurs interfaces



# Question $17 \oplus$ Soit le code ci-dessous :

```
@FunctionalInterface
interface ComparePerson {
    /**
    * Oreturn negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

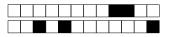
class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :

```
ComparePerson byAge = (p1, p2) -> p1.age - p2.age;

ComparePerson byAge = new ComparePersonByAge();

ComparePerson byAge = new ComparePersonByAge() {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
};
```



Question 18  $\bigoplus$  Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

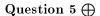
Personnel	$\square$ Faculty
Student	Person
Engineer	Prof
PEIP	MdC

QCMTEST Nom et prénom: Introduction à la programmation orientée objet 9/01/2020 Groupe: ..... Cochez les cases en mettant une  $\times$ . Le symbole  $\bigoplus$  indique que la question peut avoir zéro, une ou plusieurs bonnes réponses. Pour ces questions, cocher une bonne réponse apporte des points positifs ; cocher une mauvaise réponse peut apporter des points négatifs. Dans tout le code, les package et les import sont censés être correctement déclarés. Toute classe est supposée être dans le bon package, dans le bon fichier, avec les bons import.Question 1 La classe donnée class ClasseDonnee {} pourrait être sous-classée par extends pourrait être instanciée par new ClasseDonnee() ClasseDonnee Question 2 La classe (enum) donnée enum ClasseDonnee { INSTANCE; pourrait être sous-classée par extends pourrait être instanciée par new ClasseDonnee ClasseDonnee() Question 3  $\bigoplus$ La classe donnée final class ClasseDonnee { ClasseDonnee() {} pourrait être pourrait être sous-classée par extends instanciée par new ClasseDonnee() ClasseDonnee Question 4 Erreur de compilation Soit le code ci-dessous ; que se passe-t-il? Erreur d'exécution public class Alpha { public void throwup() throws Exception { throw new Exception(); Affiche "Done" public static void main(String[] args) {

Alpha alpha = new Alpha();
alpha.throwup();

System.out.println("Done");

}



Affiche seulement "Caught exception" Soit le code ci-dessous ; que se passe-t-il? public class Epsilon { Erreur de compilation public void throwup() throws RuntimeException { Erreur d'exécution throw new RuntimeException(); public static void main(String[] args) { Affiche seulement "Finallied exception" Epsilon epsilon = new Epsilon(); try { Affiche "Caught exception" et "Finallied epsilon.throwup(); exception" } catch (Exception e) { System.out.println("Caught exception"); } finally { System.out.println("Finallied exception"); } } Question 6  $\bigoplus$ Affiche "Done" Soit le code ci-dessous ; que se passe-t-il? public class Beta { Erreur d'exécution public void throwup() throws RuntimeException { throw new RuntimeException(); Erreur de compilation

#### Question 7 Soit le code :

beta.throwup();

}

placer XXX ?

public static void main(String[] args) {
 Beta beta = new Beta();

System.out.println("Done");

```
interface Somethingable {
    void something();
}

interface Whateverable extends Somethingable {
    void whatever();
}

class TheThing implements Whateverable{
    PPP void something() {
        System.out.println("Somethinging");
    }

    PPP void whatever() {
        System.out.println("Whatevering");
    }
}
```

```
class Doofer {
    void doo(Somethingable sable) {
        sable.XXX();
}

void doo(Whateverable sable) {
        sable.YYY();
}

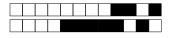
void fer(TheThing tt) {
        tt.ZZZ();
}

public static void main(String[] args) {
        Doofer doofer = new Doofer();
        TheThing the = new TheThing();
        doofer.doo(the);
        doofer.fer(the);
}
```

Par quels niveaux d'accès pourrait-on remplacer PPP?

package-pri	vate public
$\square$ private	protected
Question $8 \oplus$	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-

something whatever



Question 9 $\bigoplus$ placer YYY ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
whatever	something
Question 10 $\bigoplus$ remplacer ZZZ ?	Pour le code de la question précédente, par quelles méthodes pourrait-on
something	whatever

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit



Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r \cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r \sin(\theta)$   $\theta = \tan^{-1} \frac{y}{x}$ 

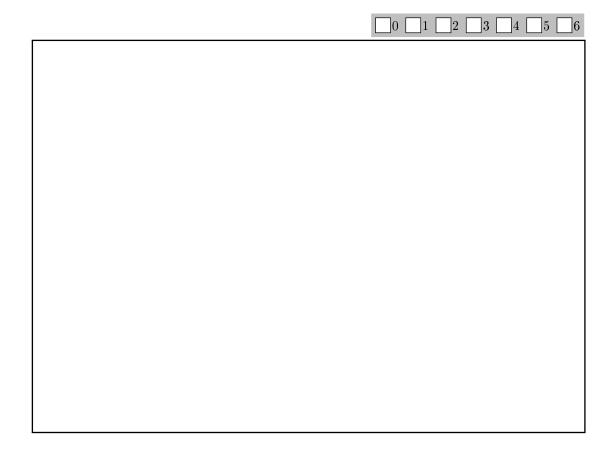
En Java ce dernier est calculé par theta = Math.atan2(y, x)

- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

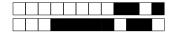
Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.

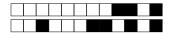




Question 13	Continuez	votre répons	e à la question	précédente ic	i si nécessaire.	0 0



Question 14	Continuez votre réponse à la question précédente ici si nécessaire.	
1		



Une interface peut contenir des constructeurs

Question	15	Pour	la	auestion	précédente,	dávoloppoz	la	máthada	Dola	r#ogu	ale	
Question	10	1 Oui	ıa	question	precedence,	developpez	la			_		$\Box_4$
									, L			
Question	16 <b>⊕</b>	Parn	ni l	es affirma	tions ci-desso	ous, cocher c	ell€	es qui sont	vraie	s:		
Une o	classe p	eut éte	$\operatorname{end}$	re plusieu	rs classes							
Une o	classe a	bstrait	ер	eut contei	nir des const	$\operatorname{ructeurs}$						
Une o	classe p	eut éte	end	re une cla	sse abstraite							
Une o	classe p	eut éte	end	re une cla	sse et implén	nenter plusie	eurs	interfaces	5			
			_	=	menter une i							
Une i	interfac	e peut	coı	ntenir des	méthodes pr	rivées						



# Question $17 \oplus$ Soit le code ci-dessous :

```
@FunctionalInterface
interface ComparePerson {
    /**
    * Oreturn negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :



# Question 18 $\bigoplus$ Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

MdC	Personnel
Faculty	Prof
PEIP	Person
Student	Engineer

QCMTEST Nom et prénom: Introduction à la programmation orientée objet 9/01/2020 Groupe: ..... Cochez les cases en mettant une  $\times$ . Le symbole  $\bigoplus$  indique que la question peut avoir zéro, une ou plusieurs bonnes réponses. Pour ces questions, cocher une bonne réponse apporte des points positifs ; cocher une mauvaise réponse peut apporter des points négatifs. Dans tout le code, les package et les import sont censés être correctement déclarés. Toute classe est supposée être dans le bon package, dans le bon fichier, avec les bons import.Question 1 ( La classe donnée abstract class ClasseDonnee { ClasseDonnee() {} pourrait être sous-classée par extends pourrait être instanciée par new ClasseDonnee ClasseDonnee() Question 2 La classe donnée class ClasseDonnee { private ClasseDonnee() {} } pourrait être sous-classée par extends pourrait être instanciée par new ClasseDonnee ClasseDonnee() Question 3 ( La classe donnée class ClasseDonnee {} pourrait être instanciée pourrait être sous-classée par extends new ClasseDonnee() ClasseDonnee

#### Question 4 (

Soit le code ci-dessous ; que se passe-t-il ?

public class Gamma {
 public void throwup() throws Exception {
 throw new Exception();
 }
 public static void main(String[] args) {
 Gamma gamma = new Gamma();
 try {
 gamma.throwup();
 } catch (Exception e) {
 System.out.println("Caught exception");
 } finally {
 System.out.println("Finallied exception");
 }
}

Erreur de compilation

Affiche seulement "Finallied exception"

Affiche "Caught exception" et "Finallied exception"

Erreur d'exécution

Affiche seulement "Caught exception"



```
Affiche seulement "Caught exception"
Soit le code ci-dessous ; que se passe-t-il?
                                                            Affiche "Caught exception" et "Finallied
public class Delta {
   public void throwup() throws Exception {
                                                            exception"
      throw new Exception();
                                                          Affiche seulement "Finallied exception"
   public static void main(String[] args) {
      Delta delta = new Delta();
                                                            Erreur de compilation
      try {
          delta.throwup();
                                                            Erreur d'exécution
      } catch (Exception e) {
          System.out.println("Caught exception");
          return;
      } finally {
          System.out.println("Finallied exception");
}
Question 6
                                                            Erreur de compilation
Soit le code ci-dessous ; que se passe-t-il?
                                                            Affiche "Done"
public class Beta {
   public void throwup() throws RuntimeException {
                                                            Erreur d'exécution
      throw new RuntimeException();
   public static void main(String[] args) {
```

## Question 7 Soit le code :

Beta beta = new Beta();
beta.throwup();

System.out.println("Done");

```
interface Somethingable {
   void something();
}

interface Whateverable extends Somethingable {
   void whatever();
}

class TheThing implements Whateverable{
   PPP void something() {
       System.out.println("Somethinging");
   }

   PPP void whatever() {
       System.out.println("Whatevering");
   }
}
```

```
class Doofer {
    void doo(Somethingable sable) {
        sable.XXX();
}

void doo(Whateverable sable) {
        sable.YYY();
}

void fer(TheThing tt) {
        tt.ZZZ();
}

public static void main(String[] args) {
        Doofer doofer = new Doofer();
        TheThing the = new TheThing();
        doofer.doo(the);
        doofer.fer(the);
}
```

Par quels niveaux d'accès pourrait-on remplacer PPP ?

protected	private
package-pri	vate public
Question 8 $\bigoplus$ placer XXX ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem
whatever	something

+14/	′3/48+
------	--------

Question 9 $\bigoplus$ placer YYY ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
something	whatever
Question 10 $\bigoplus$ remplacer ZZZ ?	Pour le code de la question précédente, par quelles méthodes pourrait-on
something	whatever

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit



Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r \cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r \sin(\theta)$   $\theta = \tan^{-1} \frac{y}{x}$ 

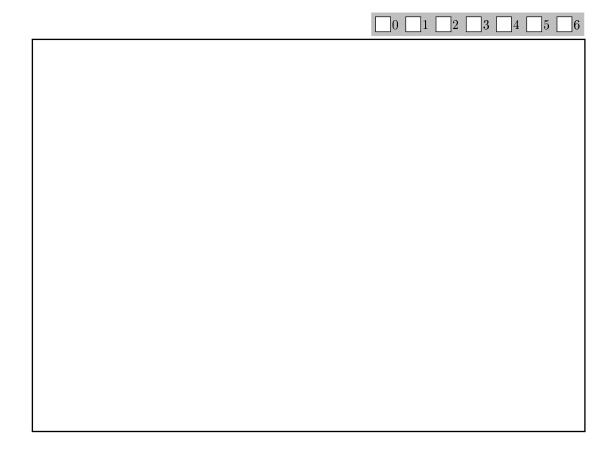
En Java ce dernier est calculé par theta = Math.atan2(y, x)

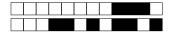
- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

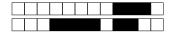
Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.

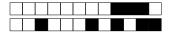




Question 13	Continuez	votre réponse à la question précédente ici si nécessaire.	



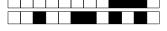
Question 14	Continuez	votre répo	nse à la que	estion précéd	ente ici si n	écessaire.	0 0



Question 15 Pour la question précédente, développez la méthode Polar#equals.  $\boxed{0}$   $\boxed{1}$   $\boxed{2}$   $\boxed{3}$ Une classe peut étendre plusieurs classes Une classe abstraite peut implémenter une interface Une interface peut contenir des méthodes privées Une interface peut contenir des constructeurs Une classe abstraite peut contenir des constructeurs

Une classe peut étendre une classe et implémenter plusieurs interfaces

Une classe peut étendre une classe abstraite



# Question $17 \oplus$ Soit le code ci-dessous :

```
OFunctionalInterface
interface ComparePerson {
    /**
    * Oreturn negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :

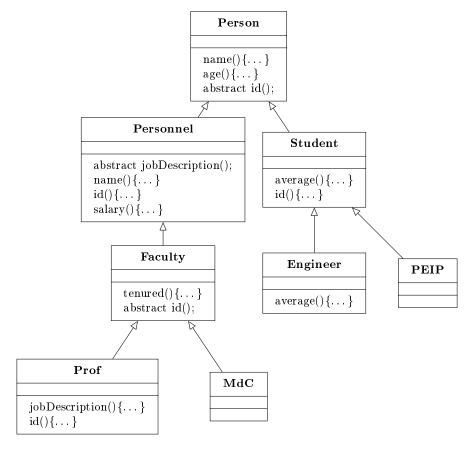
```
ComparePerson byAge = new ComparePersonByAge() {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
};

ComparePerson byAge = (p1, p2) -> p1.age - p2.age;

ComparePerson byAge
= new ComparePersonByAge::compare;
```



# Question 18 $\bigoplus$ Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

MdC	Person
Prof	Student
Engineer	PEIP
Faculty	Personnel

QCMTEST Nom et prénom: Introduction à la programmation orientée objet 9/01/2020 Groupe: ..... Cochez les cases en mettant une  $\times$ . Le symbole  $\bigoplus$  indique que la question peut avoir zéro, une ou plusieurs bonnes réponses. Pour ces questions, cocher une bonne réponse apporte des points positifs ; cocher une mauvaise réponse peut apporter des points négatifs. Dans tout le code, les package et les import sont censés être correctement déclarés. Toute classe est supposée être dans le bon package, dans le bon fichier, avec les bons import.Question 1 ( La classe (enum) donnée enum ClasseDonnee { INSTANCE; pourrait être sous-classée par extends pourrait être instanciée par new ClasseDonnee ClasseDonnee() Question 2 La classe donnée final class ClasseDonnee { ClasseDonnee() {} } pourrait être sous-classée par extends pourrait être instanciée par new ClasseDonnee() ClasseDonnee Question 3 ( La classe donnée abstract class ClasseDonnee { ClasseDonnee() {} pourrait être sous-classée par extends pourrait être instanciée par new ClasseDonnee ClasseDonnee() Question 4 Erreur de compilation Soit le code ci-dessous ; que se passe-t-il? public class Alpha { Affiche "Done" public void throwup() throws Exception { throw new Exception(); Erreur d'exécution public static void main(String[] args) {

Alpha alpha = new Alpha();
alpha.throwup();

System.out.println("Done");

}

# Question 5 $\bigoplus$

Soit le code ci-dessous ; que se passe-t-il? public class Epsilon { public void throwup() throws RuntimeException { throw new RuntimeException(); public static void main(String[] args) { Epsilon epsilon = new Epsilon(); Erreur de compilation try { epsilon.throwup(); } catch (Exception e) { System.out.println("Caught exception"); } finally { System.out.println("Finallied exception"); } }

Erreur d'exécution Affiche "Caught exception" et "Finallied exception" Affiche seulement "Caught exception"

Affiche seulement "Finallied exception"

# Question 6 $\bigoplus$

Soit le code ci-dessous ; que se passe-t-il? public class Gamma { public void throwup() throws Exception { throw new Exception(); public static void main(String[] args) { Gamma gamma = new Gamma(); try { gamma.throwup(); } catch (Exception e) { System.out.println("Caught exception"); } finally { System.out.println("Finallied exception"); } }

Erreur de compilation Affiche "Caught exception" et "Finallied exception" Affiche seulement "Caught exception"

Affiche seulement "Finallied exception"

Erreur d'exécution

#### Question 7 Soit le code:

```
interface Somethingable {
   void something();
interface Whateverable extends Somethingable {
   void whatever();
class TheThing implements Whateverable{
   PPP void something() {
       System.out.println("Somethinging");
   PPP void whatever() {
       System.out.println("Whatevering");
}
```

```
class Doofer {
   void doo(Somethingable sable) {
        sable.XXX();
   void doo(Whateverable sable) {
        sable.YYY();
    void fer(TheThing tt) {
        tt.ZZZ();
   public static void main(String[] args) {
       Doofer doofer = new Doofer();
TheThing the = new TheThing();
        doofer.doo(the);
        doofer.fer(the);
}
```

Par quels niveaux d'accès pourrait-on remplacer PPP?

protected private

public

package-private

Question 8 $\bigoplus$ placer XXX ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
whatever	something
Question 9 $\bigoplus$ placer YYY ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
whatever	<pre>something</pre>
$\begin{array}{c} \mathbf{Question} \ \ 10 \ \bigoplus \\ \mathbf{remplacer} \ \ \mathbf{ZZZ} \ ? \end{array}$	Pour le code de la question précédente, par quelles méthodes pourrait-on
something	whatever

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit



Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r\cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r\sin(\theta)$   $\theta = \tan^{-1}\frac{y}{x}$ 

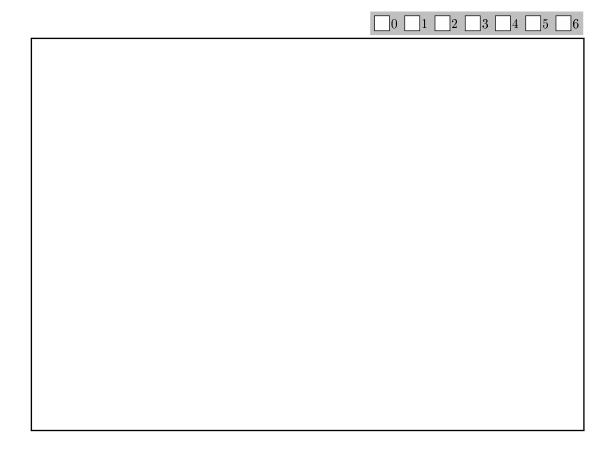
En Java ce dernier est calculé par theta = Math.atan2(y, x)

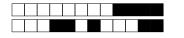
- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

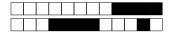
Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.

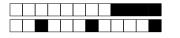




Question	13	Continuez	votre	réponse	à la qu	estion	précédent	e ici s	i nécessaire.	



Question 14	Continuez votre réponse à la question précédente ici si nécessaire.	



Une classe abstraite peut implémenter une interface
 Une classe abstraite peut contenir des constructeurs
 Une interface peut contenir des méthodes privées
 Une interface peut contenir des constructeurs

04:	1 -		D	1_		66-14-	161	1.		_
Question	19		Pour	ıa	question	precedente,	developpez	$^{1a}$	méthode Polar#equal	
										34
Question	16	$\oplus$	Pari	mi l	es affirma	m tions~ci-desso	ous, cocher c	elle	s qui sont vraies:	
Une	class	se pe	eut ét	$_{ m end}$	re plusieu	rs classes				
						sse et implén	nenter plusie	urs	interfaces	
_							nomeor prasie	/ <b>(11</b> 1)	, III.011000b	
∪ne (	CIASS	se pe	eut et	епа	re une cia	sse abstraite				

# Question $17 \oplus$ Soit le code ci-dessous :

```
@FunctionalInterface
interface ComparePerson {
    /**
    * @return negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

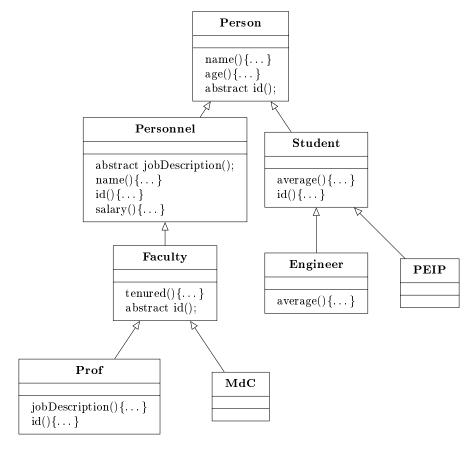
Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :

```
ComparePerson byAge = new ComparePersonByAge();

ComparePerson byAge = (p1, p2) -> p1.age - p2.age;

ComparePerson byAge = new ComparePersonByAge() {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
};
```





Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

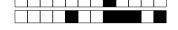
Prof	<pre>Personnel</pre>
Student	Person
PEIP	Engineer
<pre>Faculty</pre>	MdC

Cochez les cases en mettant une  $\times$ .

Le symbole  $\bigoplus$  indique que la question peut avoir zéro, une ou plusieurs bonnes réponses. Pour ces questions, cocher une bonne réponse apporte des points positifs ; cocher une mauvaise réponse peut apporter des points négatifs.

Dans tout le code, les package et les import sont censés être correctement déclarés. Toute classe est supposée être dans le bon package, dans le bon fichier, avec les bons import.

import.	
Question 1	
pourrait être sous-classée par extends ClasseDonnee  Question 2	pourrait être instanciée par new ClasseDonnee()
pourrait être sous-classée par extends ClasseDonnee  Question 3  La classe donnée	pourrait être instanciée par new ClasseDonnee()
class ClasseDonnee {}  pourrait être instanciée par new ClasseDonnee()  Question 4	pourrait être sous-classée par <b>extends</b> ClasseDonnee
<pre>Soit le code ci-dessous ; que se passe-t-il ?  public class Gamma {     public void throwup() throws Exception {         throw new Exception();     }      public static void main(String[] args) {         Gamma gamma = new Gamma();         try {             gamma.throwup();     } catch (Exception e) {             System.out.println("Caught exception");     } finally {             System.out.println("Finallied exception");     } }</pre>	<ul> <li>Erreur d'exécution</li> <li>Erreur de compilation</li> <li>Affiche "Caught exception" et "Finallied exception"</li> <li>Affiche seulement "Finallied exception"</li> <li>Affiche seulement "Caught exception"</li> </ul>



# Question 5 $\bigoplus$

```
Affiche seulement "Caught exception"
Soit le code ci-dessous ; que se passe-t-il?
public class Delta {
                                                            Erreur d'exécution
   public void throwup() throws Exception {
                                                            Affiche seulement "Finallied exception"
      throw new Exception();
   public static void main(String[] args) {
                                                            Erreur de compilation
      Delta delta = new Delta();
      try {
                                                            Affiche "Caught exception" et "Finallied
          delta.throwup();
                                                            exception"
      } catch (Exception e) {
          System.out.println("Caught exception");
          return;
      } finally {
          System.out.println("Finallied exception");
   }
}
Question 6
                                                            Affiche "Done"
Soit le code ci-dessous ; que se passe-t-il?
public class Alpha {
                                                            Erreur d'exécution
   public void throwup() throws Exception {
```

#### **Question 7** Soit le code :

throw new Exception();

Alpha alpha = new Alpha();
alpha.throwup();

System.out.println("Done");

public static void main(String[] args) {

```
interface Somethingable {
    void something();
}

interface Whateverable extends Somethingable {
    void whatever();
}

class TheThing implements Whateverable{
    PPP void something() {
        System.out.println("Somethinging");
    }

    PPP void whatever() {
        System.out.println("Whatevering");
    }
}
```

```
class Doofer {
    void doo(Somethingable sable) {
        sable.XXX();
    }

    void doo(Whateverable sable) {
        sable.YYY();
    }

    void fer(TheThing tt) {
        tt.ZZZ();
    }

    public static void main(String[] args) {
        Doofer doofer = new Doofer();
        TheThing the = new TheThing();
        doofer.doo(the);
        doofer.fer(the);
    }
}
```

Erreur de compilation

Par quels niveaux d'accès pourrait-on remplacer PPP ?

public private	protected package-private
Question 8 $\bigoplus$ placer XXX ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem
something	whatever

+16/3/28+

Question 9 $\bigoplus$ placer YYY ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
whatever	something
Question 10 $\bigoplus$ remplacer ZZZ ?	Pour le code de la question précédente, par quelles méthodes pourrait-or
something	whatever

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

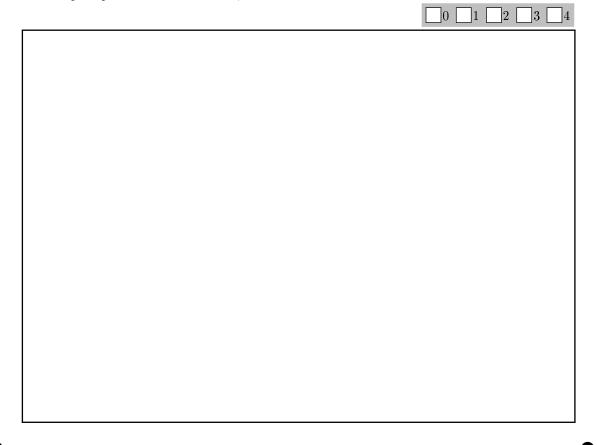
public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

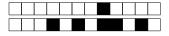
@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit





Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r\cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r\sin(\theta)$   $\theta = \tan^{-1}\frac{y}{x}$ 

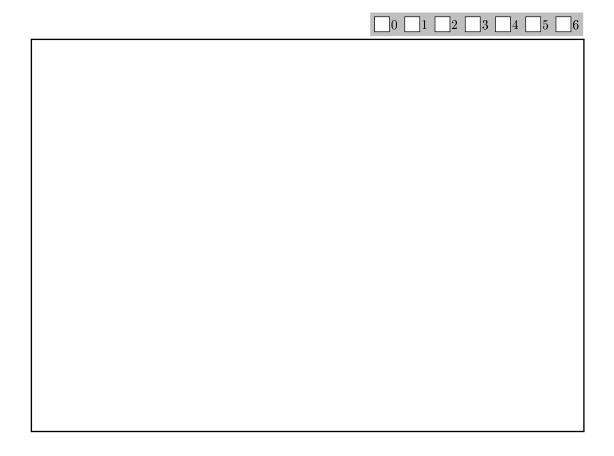
En Java ce dernier est calculé par theta = Math.atan2(y, x)

- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

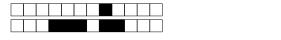
```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

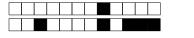
Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.



estion 13	Continuez votre réponse à la question précédente ici si nécessaire.	



Question 14	Continuez	votre réponse	à la question	précédente	ici si nécessaire.	



Une classe abstraite peut contenir des constructeurs

Une classe abstraite peut implémenter une interface

Une interface peut contenir des constructeurs
Une classe peut étendre plusieurs classes



# Question $17 \oplus$ Soit le code ci-dessous :

```
@FunctionalInterface
interface ComparePerson {
    /**
    * @return negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

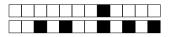
Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :

```
ComparePerson byAge = (p1, p2) -> p1.age - p2.age;

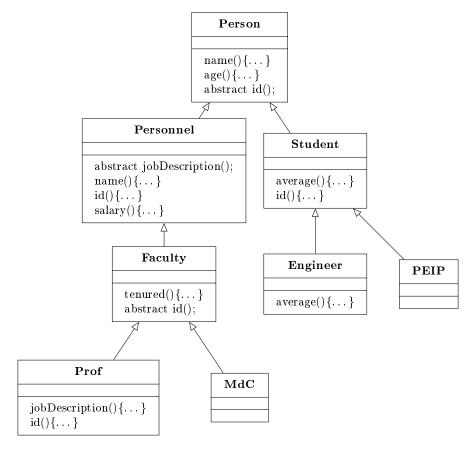
ComparePerson byAge = new ComparePersonByAge();

ComparePerson byAge
= new ComparePersonByAge::compare;
```

```
ComparePerson byAge = new ComparePersonByAge() {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
};
```

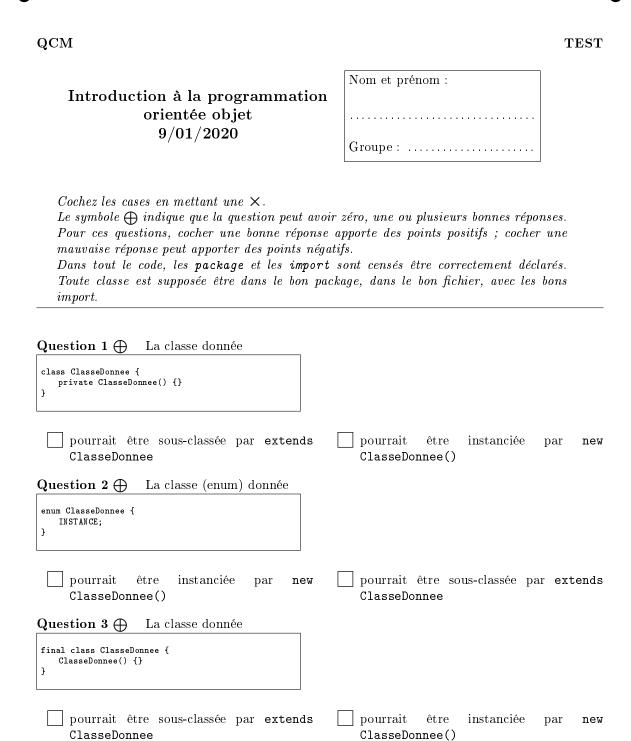


Question 18  $\bigoplus$  Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

Faculty	Prof
Engineer	PEIP
Student	MdC
Personnel	Person



# Question 4 $\bigoplus$

Soit le code ci-dessous ; que se passe-t-il ?
public class Epsilon {
 public void throwup() throws RuntimeException {
 throw new RuntimeException();
 }
 public static void main(String[] args) {
 Epsilon epsilon = new Epsilon();
 try {
 epsilon.throwup();
 } catch (Exception e) {
 System.out.println("Caught exception");
 } finally {
 System.out.println("Finallied exception");
 }
 }
}

# Erreur d'exécution Affiche seulement "Finallied exception" Erreur de compilation Affiche seulement "Caught exception" Affiche "Caught exception" et "Finallied exception"

# Question $5 \bigoplus$

Soit le code ci-dessous ; que se passe-t-il ?
public class Beta {
 public void throwup() throws RuntimeException {
 throw new RuntimeException();
 }
 public static void main(String[] args) {
 Beta beta = new Beta();
 beta.throwup();
 System.out.println("Done");
 }
}

Erreur d'execution
Affiche "Done"
Erreur de compilation

# Question 6 $\bigoplus$

Soit le code ci-dessous ; que se passe-t-il ?

public class Delta {
 public void throwup() throws Exception {
 throw new Exception();
 }
 public static void main(String[] args) {
 Delta delta = new Delta();
 try {
 delta.throwup();
 } catch (Exception e) {
 System.out.println("Caught exception");
 return;
 } finally {
 System.out.println("Finallied exception");
 }
 }
}

Affiche seulement "Finallied exception"
Erreur de compilation
Affiche seulement "Caught exception"
Erreur d'exécution
$\hfill \Box$ Affiche "Caught exception" et "Finallied
exception"



## **Question 7** Soit le code :

```
interface Somethingable {
   void something();
}

interface Whateverable extends Somethingable {
   void whatever();
}

class TheThing implements Whateverable{
   PPP void something() {
       System.out.println("Somethinging");
   }

   PPP void whatever() {
       System.out.println("Whatevering");
   }
}
```

```
class Doofer {
    void doo(Somethingable sable) {
        sable.XXX();
    }

    void doo(Whateverable sable) {
        sable.YYY();
    }

    void fer(TheThing tt) {
        tt.ZZZ();
    }

    public static void main(String[] args) {
        Doofer doofer = new Doofer();
        TheThing the = new TheThing();
        doofer.doo(the);
        doofer.fer(the);
    }
}
```

Par quels niveaux d'accès pourrait-on remplacer PPP? public protected package-private private Question 8 Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer XXX? whatever something Question 9 Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer YYY? something whatever Question 10  $\bigoplus$ Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer ZZZ? something whatever

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit





Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r\cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r\sin(\theta)$   $\theta = \tan^{-1}\frac{y}{x}$ 

En Java ce dernier est calculé par theta = Math.atan2(y, x)

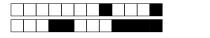
- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

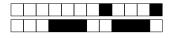
Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.

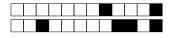




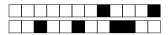
${f Question}$	13	Continuez	votre	réponse	à la	question	précéde	nte ici s	si nécessair	re. 0 0



Question 14	Continuez	votre réponse	à la question	précédente	ici si nécessaire.	



Question 16 $\bigoplus$	Parmi les affirmations ci-dessous, cocher celles qui sont v raies : $ \\$
Une classe ab	estraite peut contenir des constructeurs
Une classe pe	eut étendre une classe et implémenter plusieurs interfaces
Une classe pe	eut étendre plusieurs classes
Une classe ab	straite peut implémenter une interface
Une interface	peut contenir des constructeurs
Une interface	peut contenir des méthodes privées
Une classe pe	eut étendre une classe abstraite



# Question $17 \oplus$ Soit le code ci-dessous :

```
@FunctionalInterface
interface ComparePerson {
    /**
    * @return negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

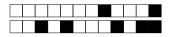
class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :

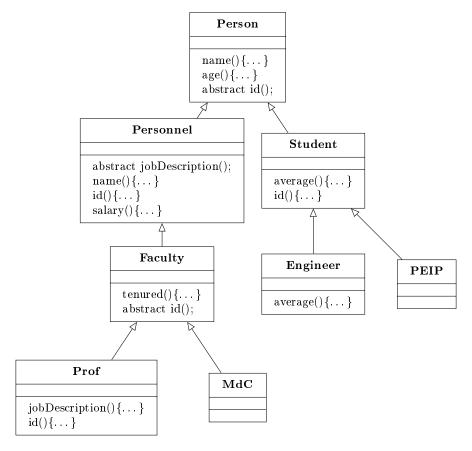
```
ComparePerson byAge = (p1, p2) -> p1.age - p2.age;

ComparePerson byAge = new ComparePersonByAge();

ComparePerson byAge = new ComparePersonByAge() {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
};
```

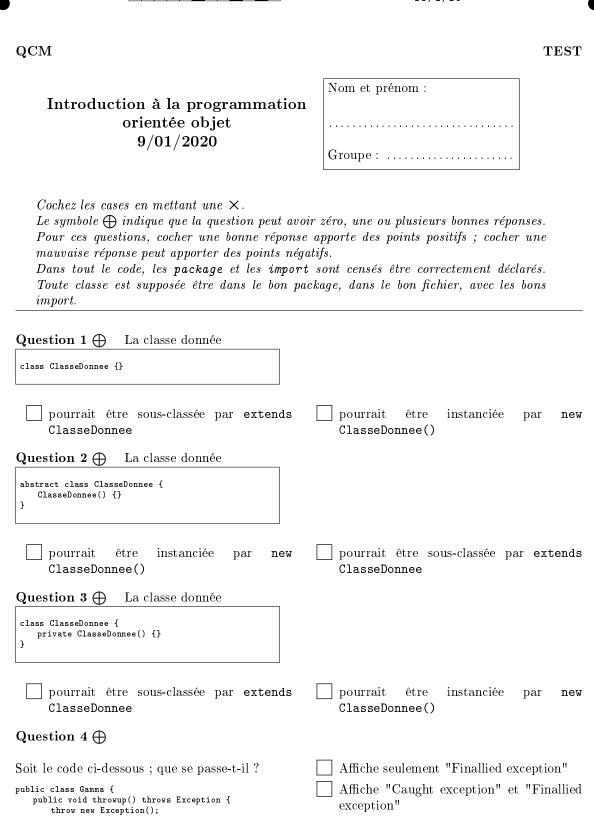


Question 18 Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

Student	MdC
<pre>Faculty</pre>	Person
Engineer	Personnel
Prof	PEIP



System.out.println("Caught exception");
} finally {
 System.out.println("Finallied exception");
}

public static void main(String[] args) {
 Gamma gamma = new Gamma();

gamma.throwup();

} catch (Exception e) {

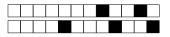
try {

}

Erreur d'exécution

Erreur de compilation

Affiche seulement "Caught exception"



# Question 5 $\bigoplus$

Affiche "Done" Soit le code ci-dessous ; que se passe-t-il? public class Alpha { Erreur de compilation public void throwup() throws Exception { Erreur d'exécution throw new Exception(); public static void main(String[] args) { Alpha alpha = new Alpha(); alpha.throwup(); System.out.println("Done"); Question 6 Soit le code ci-dessous ; que se passe-t-il? Erreur d'exécution public class Epsilon { Affiche seulement "Finallied exception" public void throwup() throws RuntimeException { Affiche seulement "Caught exception" throw new RuntimeException(); public static void main(String[] args) { Affiche "Caught exception" et "Finallied Epsilon epsilon = new Epsilon(); exception" try { epsilon.throwup(); Erreur de compilation } catch (Exception e) { System.out.println("Caught exception"); } finally { System.out.println("Finallied exception");

#### Question 7 Soit le code :

}

```
interface Somethingable {
    void something();
}

interface Whateverable extends Somethingable {
    void whatever();
}

class TheThing implements Whateverable{
    PPP void something() {
        System.out.println("Somethinging");
    }

    PPP void whatever() {
        System.out.println("Whatevering");
    }
}
```

```
class Doofer {
    void doo(Somethingable sable) {
        sable.XXX();
    }

    void doo(Whateverable sable) {
        sable.YYY();
    }

    void fer(TheThing tt) {
        tt.ZZZ();
    }

    public static void main(String[] args) {
        Doofer doofer = new Doofer();
        TheThing the = new TheThing();
        doofer.doo(the);
        doofer.fer(the);
    }
}
```

Par quels niveaux d'accès pourrait-on remplacer PPP?

protected	package-private
$\square$ private	public
Question 8 $\bigoplus$ placer XXX ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem
whatever	something

+18/3/8+

Question 9 $\bigoplus$ placer YYY ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem
something	whatever
Question 10 $\bigoplus$ remplacer ZZZ ?	Pour le code de la question précédente, par quelles méthodes pourrait-or
whatever	something

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

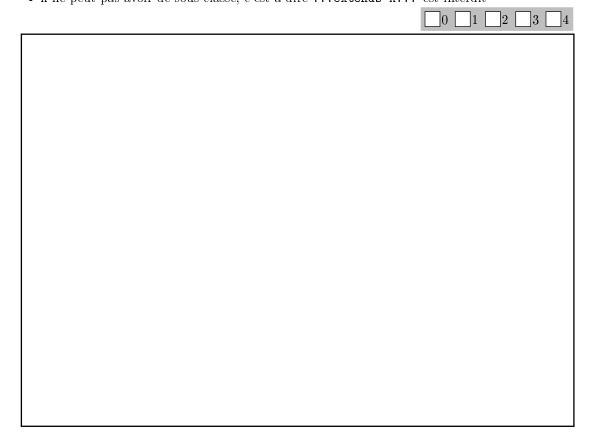
public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

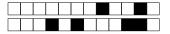
@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit





Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r\cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r\sin(\theta)$   $\theta = \tan^{-1}\frac{y}{x}$ 

En Java ce dernier est calculé par theta = Math.atan2(y, x)

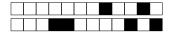
- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.

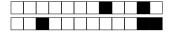




Question 13	Continuez votre réponse à la question précédente ici si nécessaire.	

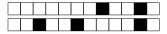


uestion 14	Continuez votre réponse à la question précédente ici si nécessaire.	



Une interface peut contenir des méthodes privées

${f Question}$	15	Pour	la	question	précédente,	développez	la	méthode	Polar	#equal:	s.
										<u></u>	3 4
Question	•				tions ci-desso		elle	s qui sont	vraies	:	
		_			constructeur			intenferes			
				re une cia re plusieu	sse et implén rs classes	nemer prusie	urs	merraces			
	=			=	sse abstraite						
Une classe abstraite peut implémenter une interface											
_					nir des const						

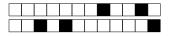


# Question $17 \oplus$ Soit le code ci-dessous :

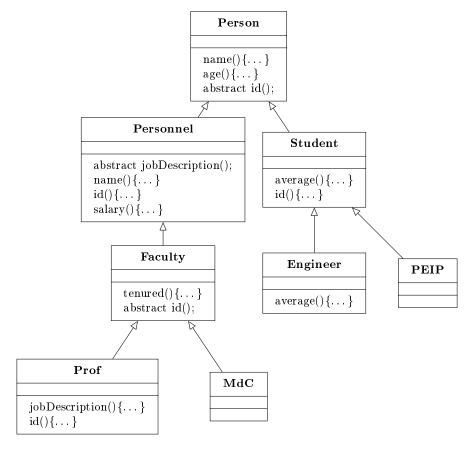
```
@FunctionalInterface
interface ComparePerson {
    /**
    * @return negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :



Question 18  $\bigoplus$  Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

Engineer	Student
MdC	Personnel
Prof	Person
PEIP	Faculty

QCMTEST Nom et prénom: Introduction à la programmation orientée objet 9/01/2020 Groupe: ..... Cochez les cases en mettant une  $\times$ . Le symbole  $\bigoplus$  indique que la question peut avoir zéro, une ou plusieurs bonnes réponses. Pour ces questions, cocher une bonne réponse apporte des points positifs ; cocher une mauvaise réponse peut apporter des points négatifs. Dans tout le code, les package et les import sont censés être correctement déclarés. Toute classe est supposée être dans le bon package, dans le bon fichier, avec les bons import.Question 1 ( La classe (enum) donnée enum ClasseDonnee { INSTANCE; pourrait pourrait être sous-classée par extends être instanciée new ClasseDonnee ClasseDonnee() Question 2 La classe donnée final class ClasseDonnee { ClasseDonnee() {} } pourrait être sous-classée par extends pourrait être instanciée par new ClasseDonnee() ClasseDonnee Question 3 ( La classe donnée class ClasseDonnee { private ClasseDonnee() {} pourrait être sous-classée par extends pourrait être instanciée par new ClasseDonnee() ClasseDonnee Question 4 Erreur d'exécution Soit le code ci-dessous ; que se passe-t-il? Erreur de compilation public class Beta { public void throwup() throws RuntimeException { Affiche "Done" throw new RuntimeException();

public static void main(String[] args) {

Beta beta = new Beta();
beta.throwup();

}

System.out.println("Done");



Affiche "Done" Soit le code ci-dessous ; que se passe-t-il? public class Alpha { Erreur d'exécution public void throwup() throws Exception { Erreur de compilation throw new Exception(); public static void main(String[] args) { Alpha alpha = new Alpha(); alpha.throwup(); System.out.println("Done"); Question 6 Soit le code ci-dessous ; que se passe-t-il? Affiche seulement "Finallied exception" public class Delta { Erreur de compilation public void throwup() throws Exception { Affiche seulement "Caught exception" throw new Exception(); Affiche "Caught exception" et "Finallied public static void main(String[] args) { Delta delta = new Delta(); exception" try { delta.throwup(); Erreur d'exécution } catch (Exception e) { System.out.println("Caught exception"); return; } finally { System.out.println("Finallied exception"); } }

# Question 7 Soit le code :

```
interface Somethingable {
    void something();
}

interface Whateverable extends Somethingable {
    void whatever();
}

class TheThing implements Whateverable{
    PPP void something() {
        System.out.println("Somethinging");
    }

    PPP void whatever() {
        System.out.println("Whatevering");
    }
}
```

```
class Doofer {
    void doo(Somethingable sable) {
        sable.XXX();
}

void doo(Whateverable sable) {
        sable.YYY();
}

void fer(TheThing tt) {
        tt.ZZZ();
}

public static void main(String[] args) {
        Doofer doofer = new Doofer();
        TheThing the = new TheThing();
        doofer.doo(the);
        doofer.fer(the);
}
```

Par quels niveaux d'accès pourrait-on remplacer PPP ?

package-private	public
protected	private

**Question 8**  $\bigoplus$  Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer XXX ?

whatever something

Question 9 $\bigoplus$ placer YYY ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem
whatever	something
Question 10 $\bigoplus$ remplacer ZZZ ?	Pour le code de la question précédente, par quelles méthodes pourrait-or
something	Whatever

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit



Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r\cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r\sin(\theta)$   $\theta = \tan^{-1}\frac{y}{x}$ 

En Java ce dernier est calculé par theta = Math.atan2(y, x)

- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.



Question 13	Continuez votre réponse à la question précédente ici si nécessai	re. 0 0



Question 14	Continuez	votre réponse	à la question	précédente	ici si nécessaire.	



Une classe abstraite peut contenir des constructeurs

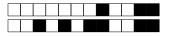
Question 15 Pour la question précédente, développez la méthode Polar#equals.  $\boxed{0}$   $\boxed{1}$   $\boxed{2}$   $\boxed{3}$ Question 16  $\bigoplus$ Parmi les affirmations ci-dessous, cocher celles qui sont vraies : Une interface peut contenir des constructeurs Une interface peut contenir des méthodes privées Une classe peut étendre plusieurs classes Une classe abstraite peut implémenter une interface Une classe peut étendre une classe et implémenter plusieurs interfaces Une classe peut étendre une classe abstraite

# Question 17 Soit le code ci-dessous :

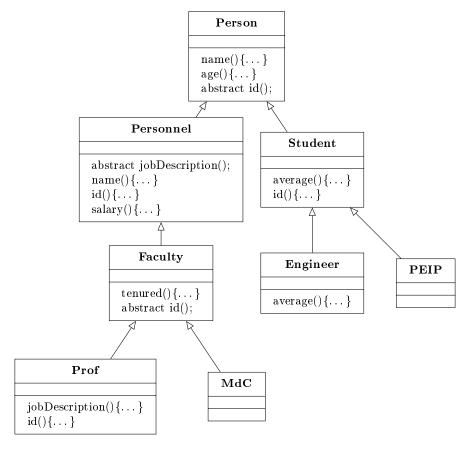
```
@FunctionalInterface
interface ComparePerson {
    /**
    * @return negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :

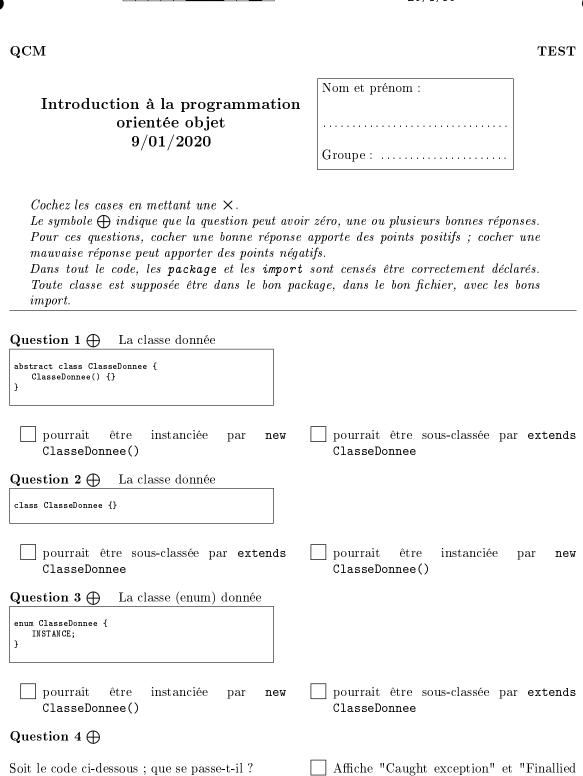


# Question 18 Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

<pre>Engineer</pre>	Personnel
PEIP	Person
MdC	Student
Prof	Faculty



public class Epsilon {
 public void throwup() throws RuntimeException {
 throw new RuntimeException();
 }
 public static void main(String[] args) {
 Epsilon epsilon = new Epsilon();
 try {
 epsilon.throwup();
 } catch (Exception e) {
 System.out.println("Caught exception");
 } finally {
 System.out.println("Finallied exception");
 }
}

exception"

Affiche seulement "Caught exception"

Erreur de compilation

Erreur d'exécution

Affiche seulement "Finallied exception"



# Question 5 $\bigoplus$

Affiche "Caught exception" et "Finallied Soit le code ci-dessous ; que se passe-t-il? exception" public class Gamma { public void throwup() throws Exception { Affiche seulement "Finallied exception" throw new Exception(); Affiche seulement "Caught exception" public static void main(String[] args) { Gamma gamma = new Gamma(); Erreur d'exécution try { gamma.throwup(); Erreur de compilation } catch (Exception e) { System.out.println("Caught exception"); } finally { System.out.println("Finallied exception"); } } Question 6  $\bigoplus$ Soit le code ci-dessous ; que se passe-t-il? Erreur d'exécution public class Beta { Erreur de compilation public void throwup() throws RuntimeException { throw new RuntimeException(); Affiche "Done" public static void main(String[] args) {
 Beta beta = new Beta();

#### Question 7 Soit le code :

System.out.println("Done");

beta.throwup();

}

```
interface Somethingable {
    void something();
}

interface Whateverable extends Somethingable {
    void whatever();
}

class TheThing implements Whateverable{
    PPP void something() {
        System.out.println("Somethinging");
    }

    PPP void whatever() {
        System.out.println("Whatevering");
    }
}
```

```
class Doofer {
    void doo(Somethingable sable) {
        sable.XXX();
    }

    void doo(Whateverable sable) {
        sable.YYY();
    }

    void fer(TheThing tt) {
        tt.ZZZ();
    }

    public static void main(String[] args) {
        Doofer doofer = new Doofer();
        TheThing the = new TheThing();
        doofer.doo(the);
        doofer.fer(the);
    }
}
```

Par quels niveaux d'accès pourrait-on remplacer PPP?

public	private
protected	package-private
Question 8 $\oplus$ placer XXX ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
whatever	something

+20/3/48+	
-----------	--

Question 9 $\bigoplus$ placer YYY ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
whatever	something
Question 10 $\bigoplus$ remplacer ZZZ ?	Pour le code de la question précédente, par quelles méthodes pourrait-on
whatever	something

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
    public double re;
    public Complex(double a, double b) {
        re = a;
        im = b;
    }
    public double getRe() {
        return re;
    }
    public void setRe(double re) {
        this.re = re;
    }
    public double getIm() {
        return im;
    }
    public void setIm(double im) {
        this.im = im;
    }
}
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

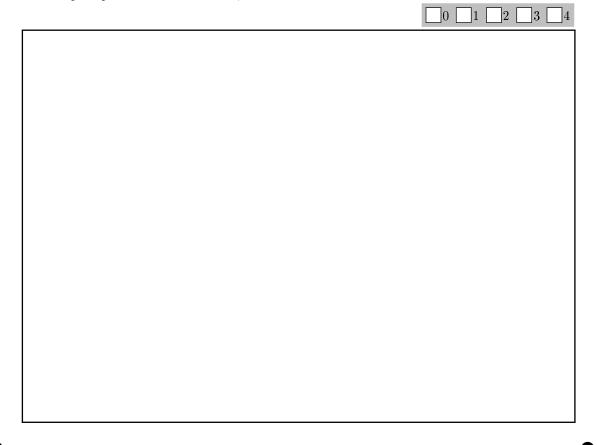
public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit



Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r\cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r\sin(\theta)$   $\theta = \tan^{-1}\frac{y}{x}$ 

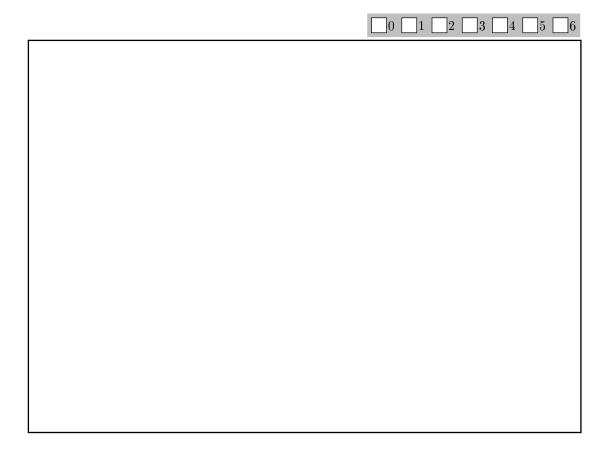
En Java ce dernier est calculé par theta = Math.atan2(y, x)

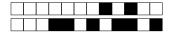
- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

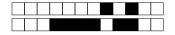
Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.





Question 13	Continuez	votre réponse à la question précédente ici si nécessaire.	



Question 14	Continuez votre réponse à la question précédente ici si nécessaire.	



Question 15 Pour la question précédente, développez la méthode Polar#equals.  $\boxed{0}$   $\boxed{1}$   $\boxed{2}$   $\boxed{3}$ Une classe peut étendre plusieurs classes Une classe peut étendre une classe abstraite Une interface peut contenir des méthodes privées Une classe peut étendre une classe et implémenter plusieurs interfaces Une classe abstraite peut implémenter une interface Une classe abstraite peut contenir des constructeurs

Une interface peut contenir des constructeurs



# Question $17 \oplus$ Soit le code ci-dessous :

```
@FunctionalInterface
interface ComparePerson {
    /**
    * @return negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

```
class Person {
    private int age;
    private String name;

Person(String name, int age) {
        this.age = age;
        this.name = name;
    }
}

class Main {
    public static void main(String[] args) {

        // code extract here

        System.out.print(byAge.compare(
            new Person("Fred", 23),
            new Person("Not Fred", 24)));
        System.out.print(byAge.compare(
            new Person("Not Fred", 23),
            new Person("Not Fred", 23));
        System.out.print(byAge.compare(
            new Person("Fred", 23),
            new Person("Fred", 23),
            new Person("Not Fred", 23),
            new Person("Not Fred", 23);
            new Person("Not Fred", 22)));
}
```

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :

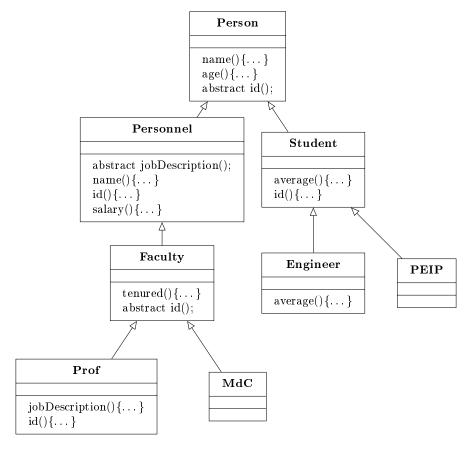
```
ComparePerson byAge = new ComparePersonByAge::compare;

ComparePerson byAge = (p1, p2) -> p1.age - p2.age;

ComparePerson byAge = new ComparePersonByAge() {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
};
```

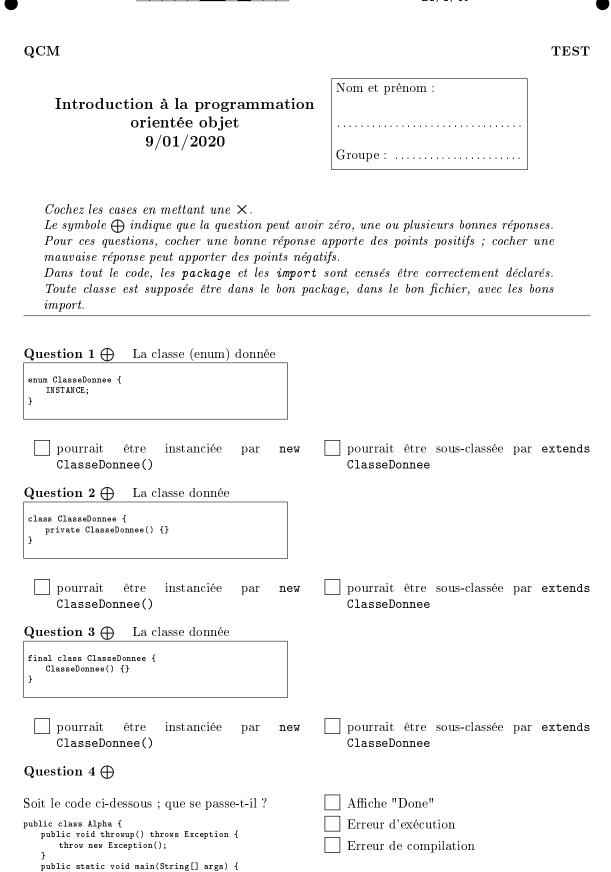


Question 18  $\bigoplus$  Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

MdC	Person
Engineer	Prof
Personnel	Student
Faculty	PEIP



Alpha alpha = new Alpha();
alpha.throwup();

System.out.println("Done");

}

# Question 5 $\bigoplus$

```
Affiche seulement "Finallied exception"
Soit le code ci-dessous ; que se passe-t-il?
public class Delta {
                                                            Affiche "Caught exception" et "Finallied
   public void throwup() throws Exception {
                                                            exception"
      throw new Exception();
                                                        Erreur de compilation
   public static void main(String[] args) {
      Delta delta = new Delta();
                                                            Affiche seulement "Caught exception"
      try {
          delta.throwup();
                                                            Erreur d'exécution
      } catch (Exception e) {
          System.out.println("Caught exception");
          return:
      } finally {
          System.out.println("Finallied exception");
}
Question 6
```

```
Soit le code ci-dessous ; que se passe-t-il ?

public class Gamma {
    public void throwup() throws Exception {
        throw new Exception();
    }
    public static void main(String[] args) {
        Gamma gamma = new Gamma();
        try {
            gamma.throwup();
    } catch (Exception e) {
            System.out.println("Caught exception");
    } finally {
            System.out.println("Finallied exception");
        }
}
```

# Affiche seulement "Caught exception"

Erreur de compilation

 $oxedsymbol{oxed}$  Affiche seulement "Finallied exception"

Affiche "Caught exception" et "Finallied exception"

Erreur d'exécution

#### Question 7 Soit le code :

```
interface Somethingable {
    void something();
}

interface Whateverable extends Somethingable {
    void whatever();
}

class TheThing implements Whateverable{
    PPP void something() {
        System.out.println("Somethinging");
    }

    PPP void whatever() {
        System.out.println("Whatevering");
    }
}
```

```
class Doofer {
    void doo(Somethingable sable) {
        sable.XXX();
    }

    void doo(Whateverable sable) {
        sable.YYY();
    }

    void fer(TheThing tt) {
        tt.ZZZ();
    }

    public static void main(String[] args) {
        Doofer doofer = new Doofer();
        TheThing the = new TheThing();
        doofer.doo(the);
        doofer.fer(the);
    }
}
```

Par quels niveaux d'accès pourrait-on remplacer PPP?

 ☐ private
 ☐ package-private

 ☐ protected
 ☐ public

Question 8 $\bigoplus$ placer XXX ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
whatever	something
Question 9 $\bigoplus$ placer YYY ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
something	whatever
Question 10 ⊕ remplacer ZZZ ?	Pour le code de la question précédente, par quelles méthodes pourrait-on
something	whatever

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
    public double re;
    public Complex(double a, double b) {
        re = a;
        im = b;
    }

    public double getRe() {
        return re;
    }

    public void setRe(double re) {
        this.re = re;
    }

    public double getIm() {
        return im;
    }

    public void setIm(double im) {
        this.im = im;
    }
}
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit





Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r\cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r\sin(\theta)$   $\theta = \tan^{-1}\frac{y}{x}$ 

En Java ce dernier est calculé par theta = Math.atan2(y, x)

- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.

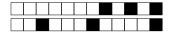




Question 13	Continuez votre réponse à la question précédente ici s	i nécessaire.	



Question 14	Continuez	votre réponse à l	la question pré	cédente ici s	si nécessaire.	0 0



Question 15 Pour la question précédente, développez la méthode Polar#equals.

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □3 □4

□0 □1 □2 □4

□0 □1 □2 □4

□0 □1 □2 □4

□0 □1 □2 □4

□0 □1 □2 □4

□0 □1 □2 □4

□0 □1 □4

□0 □1 □4

□0 □1 □4

□0 □1 □4

□0 □1 □4

□0 □1 □4

□0 □1 □4

□0 □1 □4

□0 □1 □4

□0 □1 □4

□0 □1 □4

□0 □1 □4

□0 □1 □4

□0 □1 □4

□0 □1 □4

□0 □1 □4

□0 □1 □4

□0 □1 □4

□0 □1 □4

□0 □1 □4

□0 □1 □4

□0 □1 □4

□0 □1 □4

□0 □1 □4

□0 □1 □4

□0 □1 □4

□0 □1 □4

□0 □1 □4

□0 □1 □4

□0 □1 □4

□0 □1 □4

□0 □1 □4

□0 □1 □4

□0 □1 □4

□0 □1 □4

□0 □1 □4

□0 □1 □4

□0 □1 □4

□0 □1 □4

□0 □1 □4

□

<b>Question 16</b> ⊕ Parmi les affirmations ci-dessous, cocher celles qui sont vraies
Une classe peut étendre plusieurs classes
Une classe peut étendre une classe abstraite
Une interface peut contenir des méthodes privées
Une classe peut étendre une classe et implémenter plusieurs interfaces
Une classe abstraite peut contenir des constructeurs
Une classe abstraite peut implémenter une interface
Une interface peut contenir des constructeurs



# Question $17 \oplus$ Soit le code ci-dessous :

```
@FunctionalInterface
interface ComparePerson {
    /**
    * Oreturn negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :

```
ComparePerson byAge = new ComparePersonByAge() {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
};

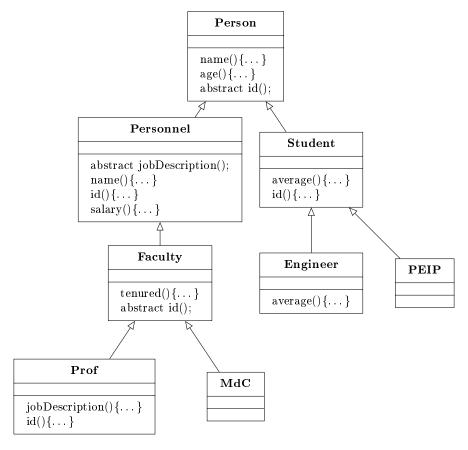
ComparePerson byAge = new ComparePersonByAge();

ComparePerson byAge = (p1, p2) -> p1.age - p2.age;

ComparePerson byAge = (p1, p2) -> p1.age - p2.age;
```



Question 18 Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

Personnel	Prof
PEIP	Engineer
Faculty	Student
Person	MdC

QCMTEST Nom et prénom: Introduction à la programmation orientée objet 9/01/2020 Groupe: ..... Cochez les cases en mettant une  $\times$ . Le symbole  $\bigoplus$  indique que la question peut avoir zéro, une ou plusieurs bonnes réponses. Pour ces questions, cocher une bonne réponse apporte des points positifs ; cocher une mauvaise réponse peut apporter des points négatifs. Dans tout le code, les package et les import sont censés être correctement déclarés. Toute classe est supposée être dans le bon package, dans le bon fichier, avec les bons import.Question 1 

La classe donnée abstract class ClasseDonnee { ClasseDonnee() {} pourrait pourrait être sous-classée par extends être instanciée par new ClasseDonnee ClasseDonnee() Question 2 La classe donnée class ClasseDonnee {} pourrait être sous-classée par extends pourrait être instanciée par new ClasseDonnee() ClasseDonnee Question 3  $\bigoplus$ La classe donnée class ClasseDonnee { private ClasseDonnee() {} pourrait  $\hat{\mathrm{e}}\mathrm{tre}$ pourrait être sous-classée par extends instanciée par new ClasseDonnee() ClasseDonnee Question 4 Erreur d'exécution Soit le code ci-dessous ; que se passe-t-il? Affiche "Done" public class Beta { public void throwup() throws RuntimeException {

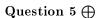
throw new RuntimeException();
}
public static void main(String[] args) {

Beta beta = new Beta();
beta.throwup();

}

System.out.println("Done");

Erreur de compilation



Soit le code ci-dessous ; que se passe-t-il? Erreur de compilation public class Epsilon { Affiche "Caught exception" et "Finallied public void throwup() throws RuntimeException { exception" throw new RuntimeException(); Erreur d'exécution public static void main(String[] args) { Epsilon epsilon = new Epsilon(); Affiche seulement "Finallied exception" try { epsilon.throwup(); Affiche seulement "Caught exception" } catch (Exception e) { System.out.println("Caught exception"); } finally { System.out.println("Finallied exception"); } Question 6  $\bigoplus$ Soit le code ci-dessous ; que se passe-t-il? Erreur de compilation Affiche "Done" public class Alpha { public void throwup() throws Exception { throw new Exception(); Erreur d'exécution public static void main(String[] args) {
 Alpha alpha = new Alpha(); alpha.throwup(); System.out.println("Done"); }

#### Question 7 Soit le code :

```
interface Somethingable {
    void something();
}

interface Whateverable extends Somethingable {
    void whatever();
}

class TheThing implements Whateverable{
    PPP void something() {
        System.out.println("Somethinging");
    }

    PPP void whatever() {
        System.out.println("Whatevering");
    }
}
```

```
class Doofer {
    void doo(Somethingable sable) {
        sable.XXX();
    }

    void doo(Whateverable sable) {
        sable.YYY();
    }

    void fer(TheThing tt) {
        tt.ZZZ();
    }

    public static void main(String[] args) {
        Doofer doofer = new Doofer();
        TheThing the = new TheThing();
        doofer.doo(the);
        doofer.fer(the);
    }
}
```

Par quels niveaux d'accès pourrait-on remplacer PPP?

package-priv	vate protected
public	private
Question 8 $\bigoplus$ placer XXX ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem
something	whatever

$\top$		

Question 9 $\bigoplus$ placer YYY ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
whatever	something
Question 10 $\bigoplus$ remplacer ZZZ ?	Pour le code de la question précédente, par quelles méthodes pourrait-on
something	whatever

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

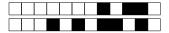
@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit





Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r \cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r \sin(\theta)$   $\theta = \tan^{-1} \frac{y}{x}$ 

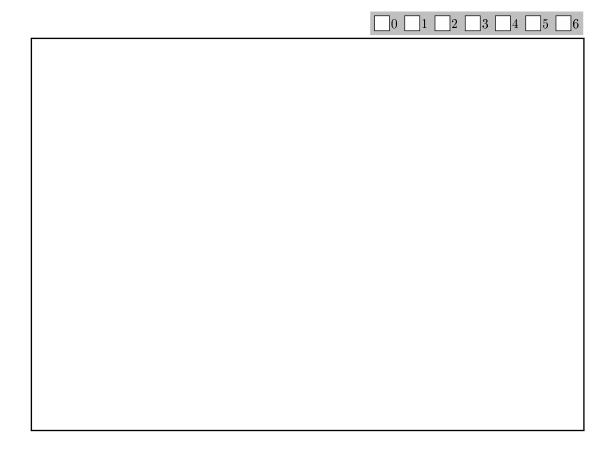
En Java ce dernier est calculé par theta = Math.atan2(y, x)

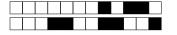
- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

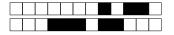
Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.





Question 13	Continuez	votre réponse à la question précédente ici si nécessaire.	



Question 14	Continuez	votre réponse	à la question	précédente i	ci si nécessaire.	



Question 15 Pour la question précédente, développez la méthode Polar#equals.  $\boxed{0}$   $\boxed{1}$   $\boxed{2}$   $\boxed{3}$ Question 16  $\bigoplus$ Parmi les affirmations ci-dessous, cocher celles qui sont vraies : Une interface peut contenir des méthodes privées Une interface peut contenir des constructeurs Une classe abstraite peut implémenter une interface Une classe peut étendre une classe et implémenter plusieurs interfaces Une classe peut étendre une classe abstraite Une classe abstraite peut contenir des constructeurs Une classe peut étendre plusieurs classes

# Question 17 Soit le code ci-dessous :

```
@FunctionalInterface
interface ComparePerson {
    /**
    * @return negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

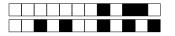
Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :

```
ComparePerson byAge = (p1, p2) -> p1.age - p2.age;

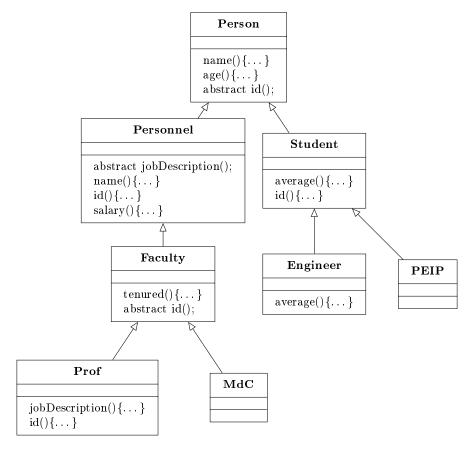
ComparePerson byAge = new ComparePersonByAge();

ComparePerson byAge = new ComparePersonByAge::compare;
```

```
ComparePerson byAge = new ComparePersonByAge() {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
};
```

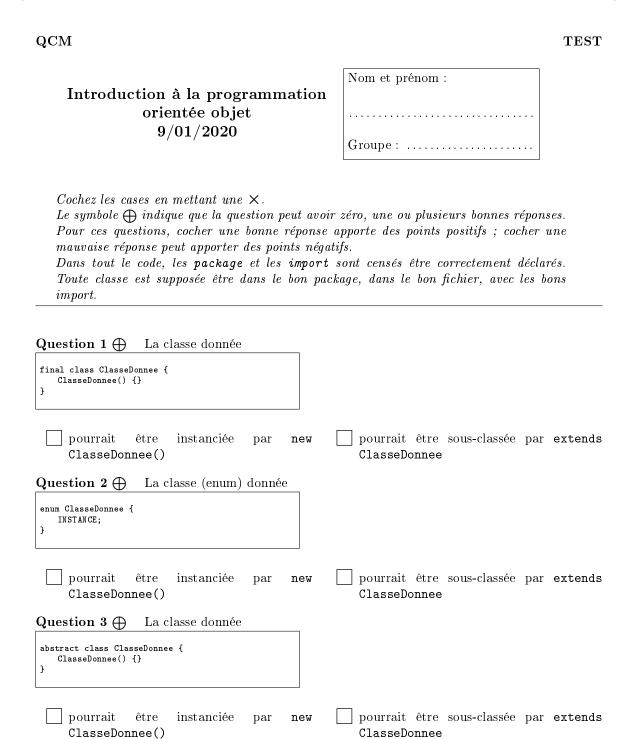


Question 18  $\bigoplus$  Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

<pre>Engineer</pre>	MdC
Person	PEIP
Personnel	$oxedsymbol{oxed}$ Faculty
Prof	Student



```
Erreur de compilation
Soit le code ci-dessous ; que se passe-t-il?
                                                             Affiche seulement "Caught exception"
public class Gamma {
   public void throwup() throws Exception {
                                                             Affiche seulement "Finallied exception"
      throw new Exception();
   public static void main(String[] args) {
                                                             Erreur d'exécution
      Gamma gamma = new Gamma();
       try {
                                                             Affiche "Caught exception" et "Finallied
          gamma.throwup();
                                                             exception"
      } catch (Exception e) {
          System.out.println("Caught exception");
       } finally {
          System.out.println("Finallied exception");
   }
}
Question 5 \bigoplus
```

```
Soit le code ci-dessous ; que se passe-t-il ?

public class Delta {
    public void throwup() throws Exception {
        throw new Exception();
    }
    public static void main(String[] args) {
        Delta delta = new Delta();
        try {
            delta.throwup();
    } catch (Exception e) {
            System.out.println("Caught exception");
            return;
        } finally {
                System.out.println("Finallied exception");
        }
    }
}
```

# Question 6 $\bigoplus$

Soit le code ci-dessous ; que se passe-t-il ?
public class Beta {
 public void throwup() throws RuntimeException {
 throw new RuntimeException();
 }
 public static void main(String[] args) {
 Beta beta = new Beta();
 beta.throwup();
 System.out.println("Done");
 }
}

Affiche seulement	"Finallied	exception"
-------------------	------------	------------

Erreur d'exécution

Affiche "Caught exception" et "Finallied exception"

Affiche seulement "Caught exception"

Erreur de compilation

Erreur de compilation

☐ Affiche "Done"

| Erreur d'exécution



#### Question 7 Soit le code :

```
interface Somethingable {
   void something();
}

interface Whateverable extends Somethingable {
   void whatever();
}

class TheThing implements Whateverable{
   PPP void something() {
       System.out.println("Somethinging");
   }

   PPP void whatever() {
       System.out.println("Whatevering");
   }
}
```

<pre>class Doofer {   void doo(Somethingable sable) {      sable.XXX();   }</pre>	
<pre>void doo(Whateverable sable) {     sable.YYY(); }</pre>	
<pre>void fer(TheThing tt) {     tt.ZZZ(); }</pre>	
<pre>public static void main(String[] args) .    Doofer doofer = new Doofer();    TheThing the = new TheThing();    doofer.doo(the);    doofer.fer(the);</pre>	{
}	

Par quels niveaux d'accès pourrait-on remplacer PPP? public protected package-private private Pour le code de la question précédente, par quelles méthodes pourrait-on rem-Question 8 placer XXX? something whatever Question 9 Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer YYY? whatever something Question 10  $\bigoplus$ Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer ZZZ? whatever something

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit





Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r \cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r \sin(\theta)$   $\theta = \tan^{-1} \frac{y}{x}$ 

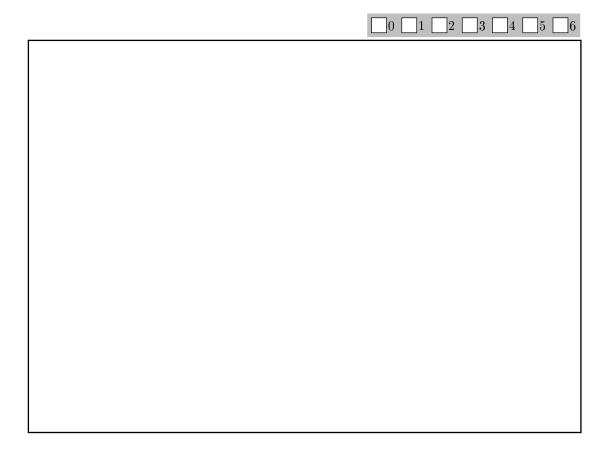
En Java ce dernier est calculé par theta = Math.atan2(y, x)

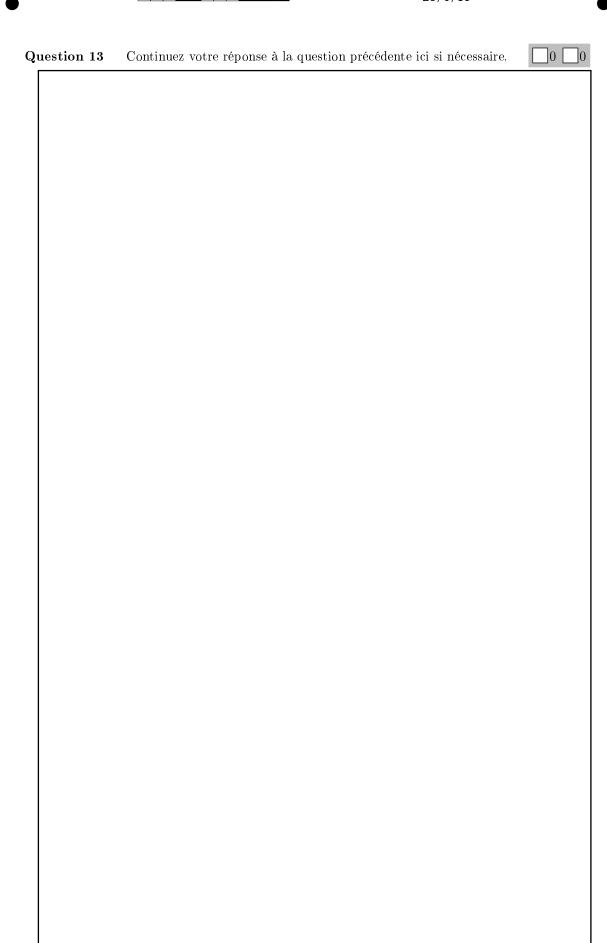
- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

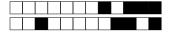
Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.







Question 14	Continuez	votre réponse	à la question	précédente	ici si nécessaire.	



Question	15	Pour	la	question	précédente,	développez	la	Polar#	equals.
Question					tions ci-desso sse et implér				
_					menter une i				
Une o	classe pe	eut ét	$\operatorname{end}$	re plusieu	rs classes				
=					constructeur				
Une o	classe al	ostrait	ер	eut conte	nir des const	ructeurs			
Une i	interface	e peut	COI	ntenir des	méthodes pr	rivées			
Une o	classe pe	eut éte	$\operatorname{end}$	re une cla	sse abstraite				

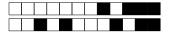


# Question $17 \oplus$ Soit le code ci-dessous :

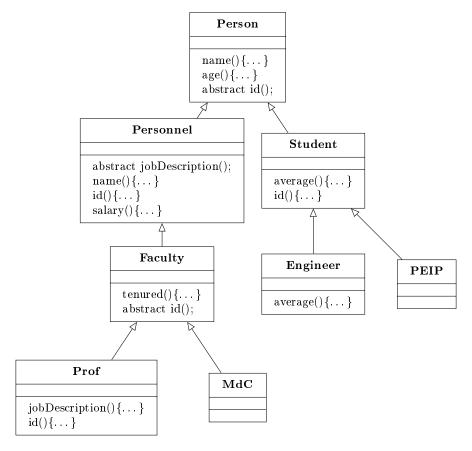
```
@FunctionalInterface
interface ComparePerson {
    /**
    * Oreturn negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :



Question 18  $\bigoplus$  Soit l'organisation ci-dessous de Polytech'Groland :



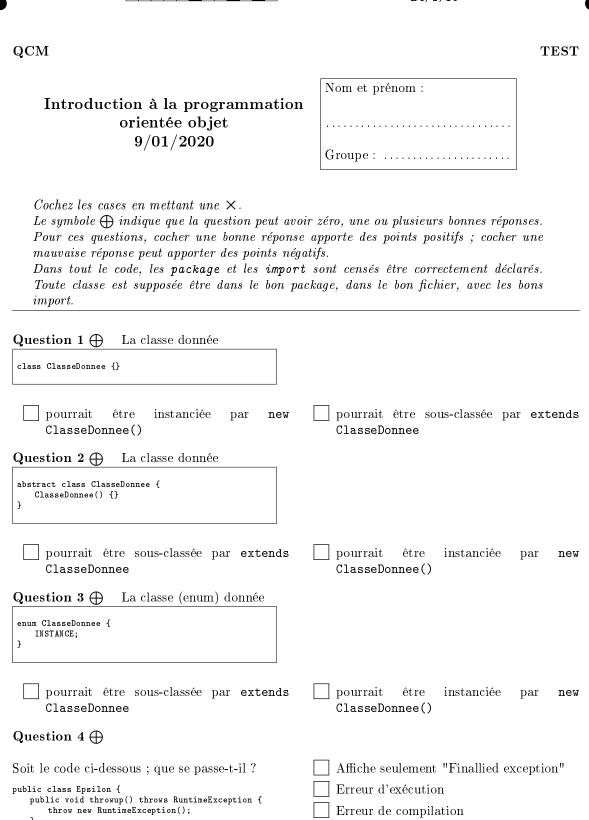
Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

MdC	Student
Personnel	Engineer
Person	$\Box$ Faculty
Prof	PEIP

Affiche seulement "Caught exception"

exception"

Affiche "Caught exception" et "Finallied



public static void main(String[] args) {

System.out.println("Caught exception");

System.out.println("Finallied exception");

Epsilon epsilon = new Epsilon();

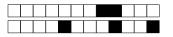
epsilon.throwup();

} catch (Exception e) {

finally {

try {

}



# Question 5 $\bigoplus$

Erreur de compilation Soit le code ci-dessous ; que se passe-t-il? public class Alpha { Erreur d'exécution public void throwup() throws Exception { Affiche "Done" throw new Exception(); public static void main(String[] args) { Alpha alpha = new Alpha(); alpha.throwup(); System.out.println("Done"); Question 6 Soit le code ci-dessous ; que se passe-t-il? Affiche seulement "Caught exception" public class Gamma { Affiche "Caught exception" et "Finallied public void throwup() throws Exception { exception" throw new Exception(); Erreur d'exécution public static void main(String[] args) { Gamma gamma = new Gamma(); Erreur de compilation try { gamma.throwup(); Affiche seulement "Finallied exception" } catch (Exception e) { System.out.println("Caught exception"); } finally { System.out.println("Finallied exception");

#### Question 7 Soit le code :

}

```
interface Somethingable {
    void something();
}

interface Whateverable extends Somethingable {
    void whatever();
}

class TheThing implements Whateverable{
    PPP void something() {
        System.out.println("Somethinging");
    }

    PPP void whatever() {
        System.out.println("Whatevering");
    }
}
```

```
class Doofer {
   void doo(Somethingable sable) {
      sable.XXX();
}

void doo(Whateverable sable) {
   sable.YYY();
}

void fer(TheThing tt) {
   tt.ZZZ();
}

public static void main(String[] args) {
   Doofer doofer = new Doofer();
   TheThing the = new TheThing();
   doofer.doo(the);
   doofer.fer(the);
}
```

Par quels niveaux d'accès pourrait-on remplacer PPP?

private public	<pre>package-private protected</pre>
Question 8 $\bigoplus$	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
something	whatever

+24/3/8+

Question 9 $\bigoplus$ placer YYY ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem
whatever	something
Question 10 $\bigoplus$ remplacer ZZZ ?	Pour le code de la question précédente, par quelles méthodes pourrait-or
whatever	something

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

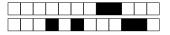
@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit





Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r \cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r \sin(\theta)$   $\theta = \tan^{-1} \frac{y}{x}$ 

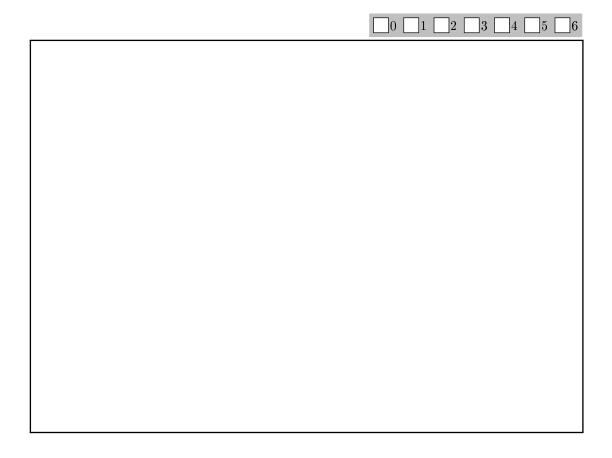
En Java ce dernier est calculé par theta = Math.atan2(y, x)

- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.





Question 13	Continuez votre réponse à la question précédente ici s	i nécessaire.	

Question 14	Continuez	votre réponse	à la question	précédente	ici si nécessaire.	



Question 15 Pour la question précédente, développez la méthode Polar#equals.  $\boxed{0}$   $\boxed{1}$   $\boxed{2}$   $\boxed{3}$ Question 16  $\bigoplus$ Parmi les affirmations ci-dessous, cocher celles qui sont vraies : Une classe peut étendre une classe abstraite Une interface peut contenir des constructeurs Une classe peut étendre une classe et implémenter plusieurs interfaces Une interface peut contenir des méthodes privées Une classe peut étendre plusieurs classes Une classe abstraite peut contenir des constructeurs Une classe abstraite peut implémenter une interface



# Question $17 \oplus$ Soit le code ci-dessous :

```
@FunctionalInterface
interface ComparePerson {
    /**
    * @return negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

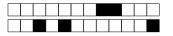
Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :

```
ComparePerson byAge = new ComparePersonByAge();

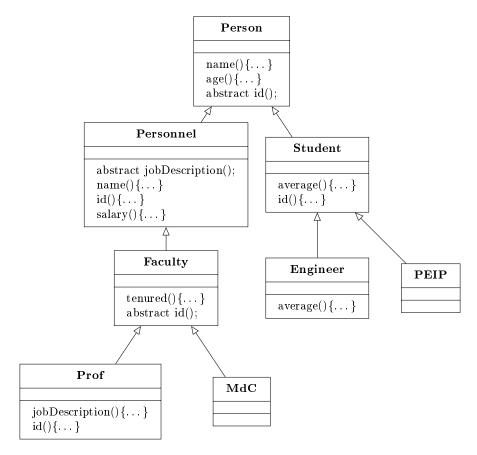
ComparePerson byAge = (p1, p2) -> p1.age - p2.age;

ComparePerson byAge
= new ComparePersonByAge::compare;
```

```
ComparePerson byAge = new ComparePersonByAge() {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
};
```



Question 18  $\bigoplus$  Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

MdC	PEIP
Personnel	Faculty
Student	Person
Engineer	Prof

QCMTEST Nom et prénom : Introduction à la programmation orientée objet 9/01/2020Groupe: ..... Cochez les cases en mettant une  $\times$ . Le symbole  $\bigoplus$  indique que la question peut avoir zéro, une ou plusieurs bonnes réponses. Pour ces questions, cocher une bonne réponse apporte des points positifs ; cocher une mauvaise réponse peut apporter des points négatifs. Dans tout le code, les package et les import sont censés être correctement déclarés. Toute classe est supposée être dans le bon package, dans le bon fichier, avec les bons import.Question 1 

La classe donnée class ClasseDonnee {
 private ClasseDonnee() {} pourrait être sous-classée par extends pourrait être instanciée par new ClasseDonnee() ClasseDonnee Question 2 La classe donnée final class ClasseDonnee { ClasseDonnee() {} pourrait être pourrait être sous-classée par extends instanciée par new ClasseDonnee() ClasseDonnee Question 3 La classe donnée class ClasseDonnee {} pourrait pourrait être sous-classée par extends être instanciée par new

ClasseDonnee()

ClasseDonnee

Soit le code ci-dessous ; que se passe-t-il ?	Erreur d'exécution
<pre>public class Delta {     public void throwup() throws Exception {         throw new Exception();     }     public static void main(String[] args) {         Delta delta = new Delta();         try {             delta.throwup();         } catch (Exception e) {             System.out.println("Caught exception");             return;         } finally {                 System.out.println("Finallied exception");         }     } }</pre>	Affiche seulement "Finallied exception"  Erreur de compilation  Affiche "Caught exception" et "Finallied exception"  Affiche seulement "Caught exception"
Question 5 $\bigoplus$	

Soit le code ci-dessous ; que se passe-t-il ?

public class Beta {
 public void throwup() throws RuntimeException {
 throw new RuntimeException();
}

public static void main(String[] args) {
 Beta beta = new Beta();
 beta.throwup();
 System.out.println("Done");

Affiche "Done"

Erreur d'exécution

Erreur de compilation

#### Question 6 $\bigoplus$

Soit le code ci-dessous ; que se passe-t-il ?
public class Epsilon {
 public void throwup() throws RuntimeException {
 throw new RuntimeException();
 }
 public static void main(String[] args) {
 Epsilon epsilon = new Epsilon();
 try {
 epsilon.throwup();
 } catch (Exception e) {
 System.out.println("Caught exception");
 } finally {
 System.out.println("Finallied exception");
 }
}

Affiche "Caught exception" et "Finallied exception"

Erreur de compilation

Erreur d'exécution

Affiche seulement "Caught exception"

Affiche seulement "Finallied exception"



#### Question 7 Soit le code :

```
interface Somethingable {
    void something();
}

interface Whateverable extends Somethingable {
    void whatever();
}

class TheThing implements Whateverable{
    PPP void something() {
        System.out.println("Somethinging");
    }

    PPP void whatever() {
        System.out.println("Whatevering");
    }
}
```

```
class Doofer {
    void doo(Somethingable sable) {
        sable.XXX();
    }

    void doo(Whateverable sable) {
        sable.YYY();
    }

    void fer(TheThing tt) {
        tt.ZZZ();
    }

    public static void main(String[] args) {
        Doofer doofer = new Doofer();
        TheThing the = new TheThing();
        doofer.doo(the);
        doofer.fer(the);
    }
}
```

Par quels niveaux d'accès pourrait-on remplacer PPP? package-private private public protected Question 8 Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer XXX? whatever something Question 9 Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer YYY? something whatever Question 10  $\bigoplus$ Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer ZZZ? whatever something

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit



Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r \cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r \sin(\theta)$   $\theta = \tan^{-1} \frac{y}{x}$ 

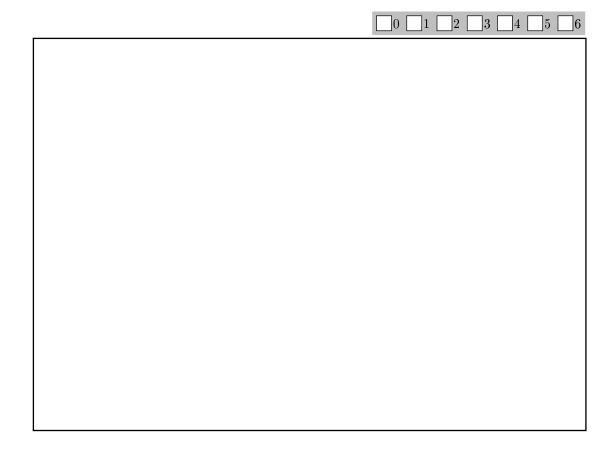
En Java ce dernier est calculé par theta = Math.atan2(y, x)

- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.

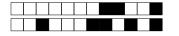




Question 13	Continuez	votre répons	e à la question	précédente ic	i si nécessaire.	0 0



Question 14	Continuez	votre réponse	à la question	précédente	ici si nécessaire.	



Une classe peut étendre une classe abstraite
Une interface peut contenir des méthodes privées

_											
Question	15	Pour	la	question	précédente,	développez	la	méthode	Polar		
											$\square 4$
Question	$16 \bigoplus$	Parn	ni l	es affirma	tions ci-desso	ous, cocher c	elle	s qui sont	vraies	:	
Une o	classe al	bstrait	ер	eut implé	menter une i	$_{ m nterface}$					
Une i	$_{ m nterface}$	e peut	coı	ntenir des	constructeur	:S					
Une o	classe p	eut éte	$\operatorname{end}$	re une cla	sse et implér	nenter plusie	urs	interfaces			
Une o	classe al	bstrait	ер	eut conte	nir des const	ructeurs					
Une o	classe p	eut éte	$\operatorname{end}$	re plusieu	rs classes						



# Question $17 \oplus$ Soit le code ci-dessous :

```
@FunctionalInterface
interface ComparePerson {
    /**
    * Oreturn negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :

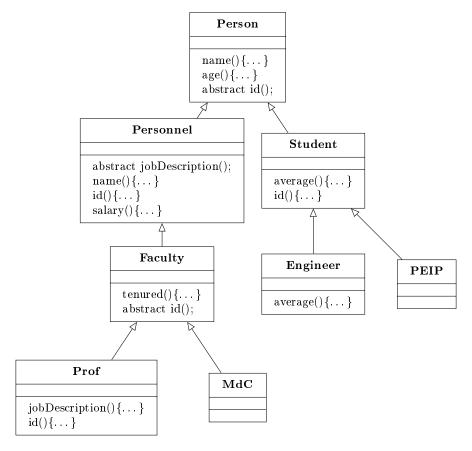
```
ComparePerson byAge = (p1, p2) -> p1.age - p2.age;

ComparePerson byAge = new ComparePersonByAge();

ComparePerson byAge = new ComparePersonByAge() {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
};
```

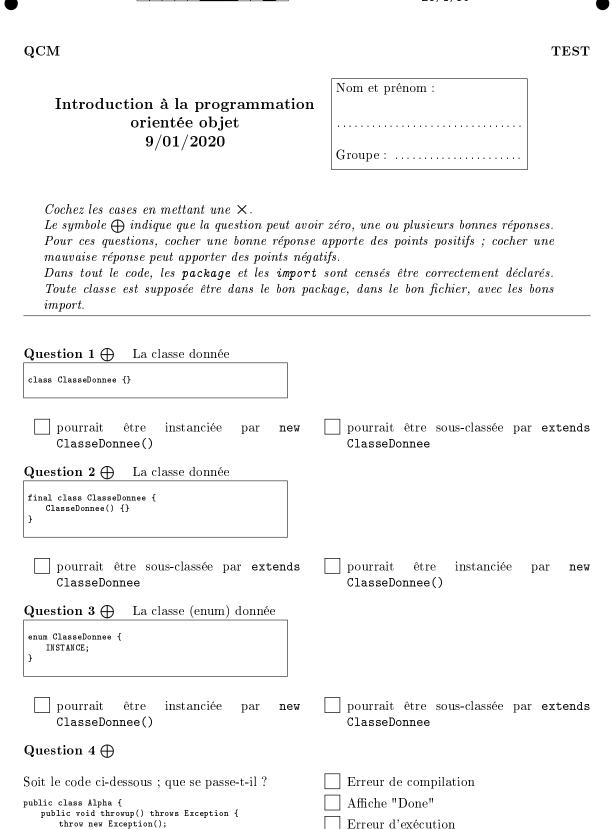


# Question 18 $\bigoplus$ Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

MdC	Student
Faculty	PEIP
Engineer	Prof
Person	Personnel



public static void main(String[] args) {

Alpha alpha = new Alpha();
alpha.throwup();

System.out.println("Done");

}



# Question 5 $\bigoplus$

Affiche seulement "Finallied exception" Soit le code ci-dessous ; que se passe-t-il? Affiche "Caught exception" et "Finallied public class Gamma { public void throwup() throws Exception { exception" throw new Exception(); Affiche seulement "Caught exception" public static void main(String[] args) { Gamma gamma = new Gamma(); Erreur de compilation try { gamma.throwup(); Erreur d'exécution } catch (Exception e) { System.out.println("Caught exception"); } finally { System.out.println("Finallied exception"); } } Question 6  $\bigoplus$ Affiche "Done" Soit le code ci-dessous ; que se passe-t-il? Erreur d'exécution public class Beta { public void throwup() throws RuntimeException { throw new RuntimeException(); Erreur de compilation

#### Question 7 Soit le code :

beta.throwup();

}

public static void main(String[] args) {
 Beta beta = new Beta();

System.out.println("Done");

```
interface Somethingable {
    void something();
}

interface Whateverable extends Somethingable {
    void whatever();
}

class TheThing implements Whateverable{
    PPP void something() {
        System.out.println("Somethinging");
    }

    PPP void whatever() {
        System.out.println("Whatevering");
    }
}
```

```
class Doofer {
    void doo(Somethingable sable) {
        sable.XXX();
    }

    void doo(Whateverable sable) {
        sable.YYY();
    }

    void fer(TheThing tt) {
        tt.ZZZ();
    }

    public static void main(String[] args) {
        Doofer doofer = new Doofer();
        TheThing the = new TheThing();
        doofer.doo(the);
        doofer.fer(the);
    }
}
```

Par quels niveaux d'accès pourrait-on remplacer PPP?

public	protected
package-pri	vate private
Question 8 $\bigoplus$ placer XXX ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem
something	Whatever

Question 9 $\bigoplus$ placer YYY ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
whatever	something
Question 10 ⊕ remplacer ZZZ ?	Pour le code de la question précédente, par quelles méthodes pourrait-on
whatever	something

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit



Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r \cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r \sin(\theta)$   $\theta = \tan^{-1} \frac{y}{x}$ 

En Java ce dernier est calculé par theta = Math.atan2(y, x)

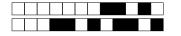
- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
       return "Cartesian + ("
}
```

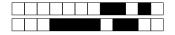
Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.





Question 13	Continuez votre réponse à la question précédente ici s	i nécessaire.	



Question	14	Continuez	votre	réponse	à la qu	estion	précéden	te ici s	i nécessaire	e. 0 0



Une interface peut contenir des méthodes privées

Une classe peut étendre une classe et implémenter plusieurs interfaces

Question 15 Pour la question précédente, développez la méthode Polar#equals.  $\boxed{0}$   $\boxed{1}$   $\boxed{2}$   $\boxed{3}$ Une classe peut étendre plusieurs classes Une classe peut étendre une classe abstraite Une interface peut contenir des constructeurs Une classe abstraite peut contenir des constructeurs Une classe abstraite peut implémenter une interface



# Question $17 \oplus$ Soit le code ci-dessous :

```
@FunctionalInterface
interface ComparePerson {
    /**
    * @return negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :

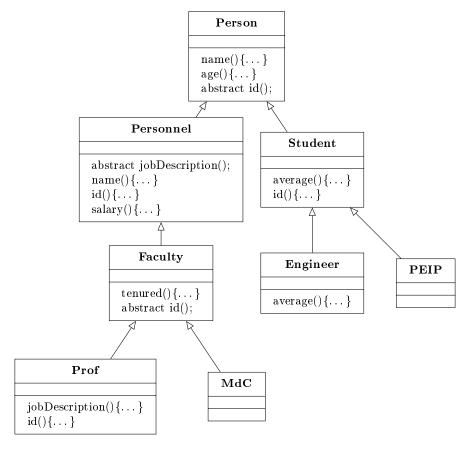
```
ComparePerson byAge = new ComparePersonByAge() {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
};

ComparePerson byAge = (p1, p2) -> p1.age - p2.age;

ComparePerson byAge
= new ComparePersonByAge::compare;
```

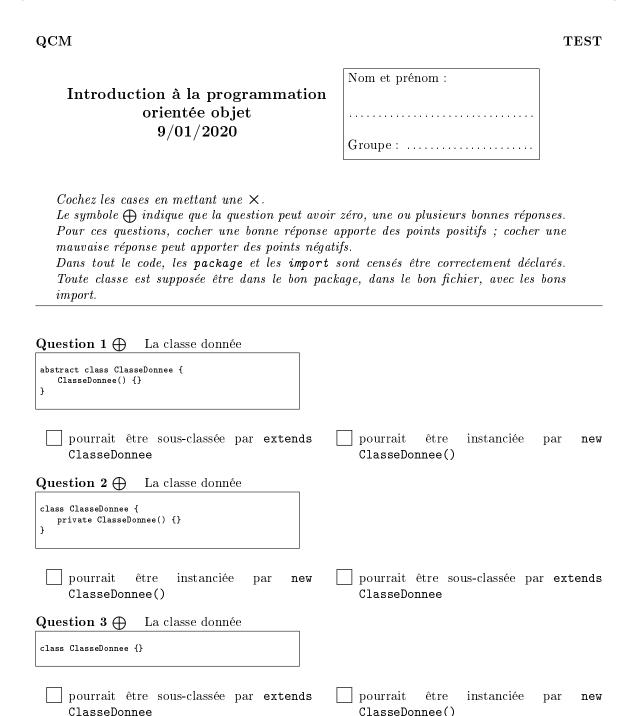


Question 18 Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

PEIP	Person
MdC	Personnel
<pre>Engineer</pre>	Faculty
Student	Prof



ClasseDonnee()

```
Erreur de compilation
Soit le code ci-dessous ; que se passe-t-il?
                                                            Affiche seulement "Finallied exception"
public class Delta {
   public void throwup() throws Exception {
                                                            Affiche "Caught exception" et "Finallied
      throw new Exception();
   public static void main(String[] args) {
      Delta delta = new Delta();
                                                           Erreur d'exécution
      try {
         delta.throwup();
                                                            Affiche seulement "Caught exception"
      } catch (Exception e) {
         System.out.println("Caught exception");
          return;
      } finally {
         System.out.println("Finallied exception");
}
Question 5
                                                           Erreur d'exécution
Soit le code ci-dessous ; que se passe-t-il?
                                                            Affiche seulement "Caught exception"
```

public class Epsilon {
 public void throwup() throws RuntimeException {
 throw new RuntimeException();
 }
 public static void main(String[] args) {
 Epsilon epsilon = new Epsilon();
 try {
 epsilon.throwup();
 } catch (Exception e) {
 System.out.println("Caught exception");
 } finally {
 System.out.println("Finallied exception");
 }
}

### Question 6

Soit le code ci-dessous ; que se passe-t-il ?
public class Alpha {
 public void throwup() throws Exception {
 throw new Exception();
 }
 public static void main(String[] args) {
 Alpha alpha = new Alpha();
 alpha.throwup();
 System.out.println("Done");
 }
}

Erreur de compilation

Affiche "Caught exception" et "Finallied

Affiche seulement "Finallied exception"

Erreur de compilation

Erreur d'exécution

Affiche "Done"

exception"



## Question 7 Soit le code :

something

```
interface Somethingable {
   void something();
}

interface Whateverable extends Somethingable {
   void whatever();
}

class TheThing implements Whateverable{
   PPP void something() {
       System.out.println("Somethinging");
   }

   PPP void whatever() {
       System.out.println("Whatevering");
   }
}
```

Par quels niveaux d'accès pourrait-on remplacer PPP?

<pre>class Doofer {   void doo(Somethingable sable) {      sable.XXX();   }</pre>	
<pre>void doo(Whateverable sable) {     sable.YYY(); }</pre>	
<pre>void fer(TheThing tt) {    tt.ZZZ(); }</pre>	
<pre>public static void main(String[] args) {    Doofer doofer = new Doofer();    TheThing the = new TheThing();    doofer.doo(the);    doofer.fer(the); }</pre>	
}	

protected  $\rfloor$  private public package-private Question 8  $\bigoplus$ Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer XXX? whatever something Question 9 Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer YYY? something whatever Question 10  $\bigoplus$ Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer ZZZ?

whatever

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

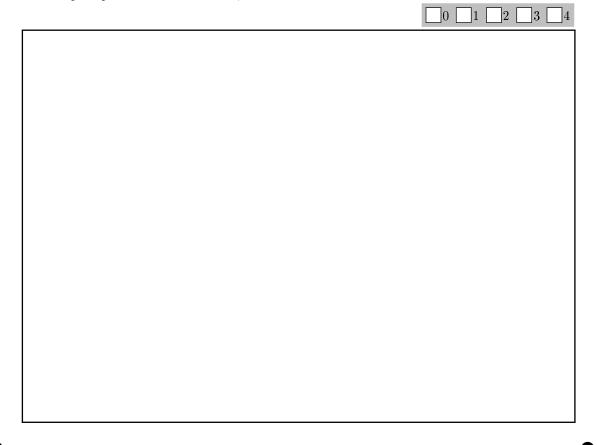
public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit





Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r \cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r \sin(\theta)$   $\theta = \tan^{-1} \frac{y}{x}$ 

En Java ce dernier est calculé par theta = Math.atan2(y, x)

- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.

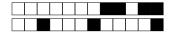




Question 13	Continuez	votre	réponse à	à la question	précédent	e ici si n	iécessaire.	



Question 14	Continuez votre réponse à la question précédente ici si nécessaire.	
1		



Question	15	$\operatorname{Pour}$	la	${\it question}$	précédente,	développez	la	$m\acute{e}thode$	Polar#e	qual	s.	
									)	2	]3 [	4

Question 16 $\oplus$	$^{\prime}$ armi les affirmations ci-dessous, cocher celles qui sont vraies :
Une classe peut	étendre plusieurs classes
Une classe peut	étendre une classe et implémenter plusieurs interfaces
Une classe abst	raite peut implémenter une interface
Une interface p	eut contenir des méthodes privées
Une classe peut	étendre une classe abstraite
Une classe abst	raite peut contenir des constructeurs
Une interface p	eut contenir des constructeurs

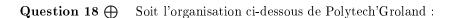


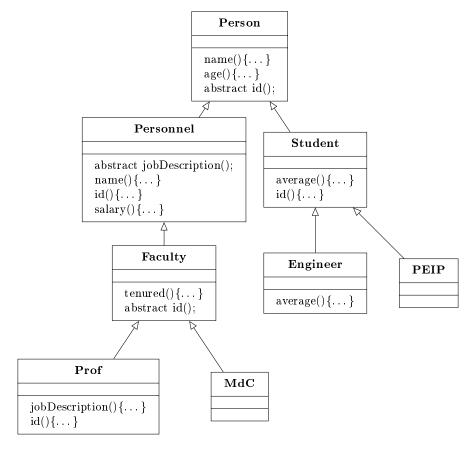
# Question $17 \oplus$ Soit le code ci-dessous :

```
@FunctionalInterface
interface ComparePerson {
    /**
    * @return negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :





Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

Personnel	Prof
Faculty	MdC
Engineer	PEIP
Student	Person

QCMTEST Nom et prénom: Introduction à la programmation orientée objet 9/01/2020 Groupe: ..... Cochez les cases en mettant une  $\times$ . Le symbole  $\bigoplus$  indique que la question peut avoir zéro, une ou plusieurs bonnes réponses. Pour ces questions, cocher une bonne réponse apporte des points positifs ; cocher une mauvaise réponse peut apporter des points négatifs. Dans tout le code, les package et les import sont censés être correctement déclarés. Toute classe est supposée être dans le bon package, dans le bon fichier, avec les bons import.Question 1 

La classe donnée final class ClasseDonnee { ClasseDonnee() {} pourrait être sous-classée par extends pourrait être instanciée par new ClasseDonnee ClasseDonnee() Question 2 La classe (enum) donnée enum ClasseDonnee { INSTANCE; } pourrait être sous-classée par extends pourrait être instanciée par new ClasseDonnee ClasseDonnee() Question 3 ( La classe donnée abstract class ClasseDonnee { ClasseDonnee() {} pourrait être sous-classée par extends pourrait être instanciée par new ClasseDonnee ClasseDonnee() Question 4 Erreur de compilation Soit le code ci-dessous ; que se passe-t-il? public class Beta { Erreur d'exécution public void throwup() throws RuntimeException { Affiche "Done" throw new RuntimeException();

public static void main(String[] args) {

Beta beta = new Beta();
beta.throwup();

}

System.out.println("Done");

# Question 5 $\bigoplus$

```
Soit le code ci-dessous ; que se passe-t-il ?

public class Gamma {
    public void throwup() throws Exception {
        throw new Exception();
    }
    public static void main(String[] args) {
        Gamma gamma = new Gamma();
        try {
            gamma.throwup();
    } catch (Exception e) {
            System.out.println("Caught exception");
    } finally {
        System.out.println("Finallied exception");
    }
}
```

# Affiche "Caught exception" et "Finallied exception" Erreur d'exécution

Affiche seulement "Caught exception"

Affiche seulement "Finallied exception"

Erreur de compilation

# Question 6 $\bigoplus$

```
Soit le code ci-dessous ; que se passe-t-il ?
public class Delta {
    public void throwup() throws Exception {
        throw new Exception();
    }
    public static void main(String[] args) {
        Delta delta = new Delta();
        try {
            delta.throwup();
    } catch (Exception e) {
            System.out.println("Caught exception");
            return;
    } finally {
            System.out.println("Finallied exception");
        }
}
```

# Erreur d'exécution Erreur de compilation

Affiche seulement "Finallied exception"

Affiche seulement "Caught exception"

Affiche "Caught exception" et "Finallied exception"

#### **Question 7** Soit le code :

```
interface Somethingable {
    void something();
}

interface Whateverable extends Somethingable {
    void whatever();
}

class TheThing implements Whateverable{
    PPP void something() {
        System.out.println("Somethinging");
    }

    PPP void whatever() {
        System.out.println("Whatevering");
    }
}
```

```
class Doofer {
   void doo(Somethingable sable) {
       sable.XXX();
   }

   void doo(Whateverable sable) {
       sable.YYY();
   }

   void fer(TheThing tt) {
       tt.ZZZ();
   }

   public static void main(String[] args) {
       Doofer doofer = new Doofer();
       TheThing the = new TheThing();
       doofer.doo(the);
       doofer.fer(the);
   }
}
```

Par quels niveaux d'accès pourrait-on remplacer PPP?

 ☐ protected
 ☐ private

 ☐ public
 ☐ package-private

Question 8 $\bigoplus$ placer XXX ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem
something	whatever
Question 9 $\bigoplus$ placer YYY ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem
whatever	something
Question 10 ⊕ remplacer ZZZ ?	Pour le code de la question précédente, par quelles méthodes pourrait-or
something	whatever

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit





Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r\cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r\sin(\theta)$   $\theta = \tan^{-1}\frac{y}{x}$ 

En Java ce dernier est calculé par theta = Math.atan2(y, x)

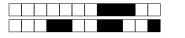
- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
       return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.





Question 13	Continuez votre réponse à la question précédente ici s	i nécessaire.	



Question 14	Continuez votre réponse à la question précédente ici si nécessaire.	
1		



<b>Q</b> uestion	15	Pour	la	question	précédente,	développez	la	$m\acute{e}thod\epsilon$	Pol	ar#equ	als.	
									0	1 2		$\Box 4$
\	1.C. (T)	D		r	.: :: .d		-11	:	4			
Question					tions ci-desso		enes	s qui son	t vra.	ies :		
=			_		nir des const							
_					sse et implér		eurs	interface	es			
					menter une i	nterface						
_				re plusieu								
=	_				sse abstraite							
=					constructeur							
Une !	interfact	e peut	cor	ntenir des	méthodes pr	rivées						

# Question $17 \oplus$ Soit le code ci-dessous :

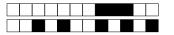
```
@FunctionalInterface
interface ComparePerson {
    /**
    * @return negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :

```
ComparePerson byAge = new ComparePersonByAge::compare;

ComparePerson byAge = new ComparePersonByAge() {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
};
```

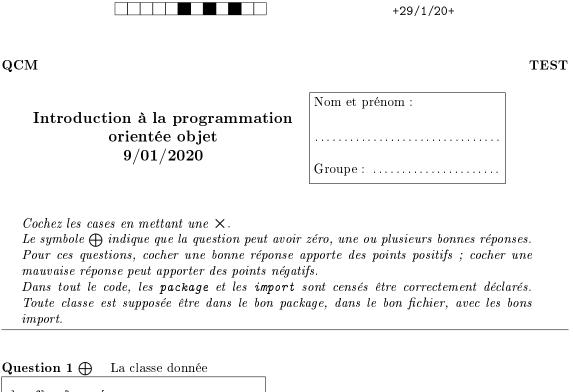


Question 18  $\bigoplus$  Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

Student	PEIP
MdC	Prof
Engineer	Person
Faculty	Personnel



Question 1 class ClasseDonnee { private ClasseDonnee() {} pourrait pourrait être sous-classée par extends être instanciée new ClasseDonnee ClasseDonnee() Question 2 La classe (enum) donnée enum ClasseDonnee { INSTANCE; } pourrait être pourrait être sous-classée par extends instanciée par new ClasseDonnee() ClasseDonnee Question 3 

La classe donnée abstract class ClasseDonnee { ClasseDonnee() {} pourrait être sous-classée par extends pourrait être instanciée par new ClasseDonnee() ClasseDonnee

Soit le code ci-dessous ; que se passe-t-il ?	Affiche seulement "Finallied exception"
<pre>public class Epsilon {     public void throwup() throws RuntimeException {         throw new RuntimeException();     }     public static void main(String[] args) {         Epsilon epsilon = new Epsilon();         try {             epsilon.throwup();       } catch (Exception e) {             System.out.println("Caught exception");       } finally {             System.out.println("Finallied exception");       }     } }</pre>	Erreur de compilation Erreur d'exécution Affiche seulement "Caught exception" Affiche "Caught exception" et "Finalli exception"
Question 5 $\bigoplus$	

```
Soit le code ci-dessous ; que se passe-t-il?
public class Delta {
   public void throwup() throws Exception {
       throw new Exception();
   public static void main(String[] args) {
   Delta delta = new Delta();
       try {
           delta.throwup();
        } catch (Exception e) {
           System.out.println("Caught exception");
           return;
       } finally {
   System.out.println("Finallied exception");
```

# Question 6 $\bigoplus$

Soit le code ci-dessous ; que se passe-t-il? public class Gamma { public void throwup() throws Exception { throw new Exception(); public static void main(String[] args) { Gamma gamma = new Gamma(); try { gamma.throwup(); } catch (Exception e) { System.out.println("Caught exception"); } finally { System.out.println("Finallied exception"); } }

	Erreur	de	compilation
--	--------	----	-------------

Affiche "Caught exception" et "Finallied exception"

et "Finallied

Erreur d'exécution

Affiche seulement "Caught exception"

Affiche seulement "Finallied exception"

Affiche "Caught exception" et "Finallied exception"

Affiche seulement "Caught exception"

Affiche seulement "Finallied exception"

Erreur d'exécution

Erreur de compilation



### Question 7 Soit le code :

```
interface Somethingable {
   void something();
}

interface Whateverable extends Somethingable {
   void whatever();
}

class TheThing implements Whateverable{
   PPP void something() {
       System.out.println("Somethinging");
   }

   PPP void whatever() {
       System.out.println("Whatevering");
   }
}
```

```
class Doofer {
  void doo(Somethingable sable) {
    sable.XXX();
  }
  void doo(Whateverable sable) {
    sable.YYY();
  }
  void fer(TheThing tt) {
    tt.ZZZ();
  }
  public static void main(String[] args) {
    Doofer doofer = new Doofer();
    TheThing the = new TheThing();
    doofer.doo(the);
    doofer.fer(the);
  }
}
```

Par quels niveaux d'accès pourrait-on remplacer PPP? package-private private public protected Question 8 Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer XXX? whatever something Question 9 Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer YYY? something whatever Question 10  $\bigoplus$ Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer ZZZ? whatever something

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit





Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r\cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r\sin(\theta)$   $\theta = \tan^{-1}\frac{y}{x}$ 

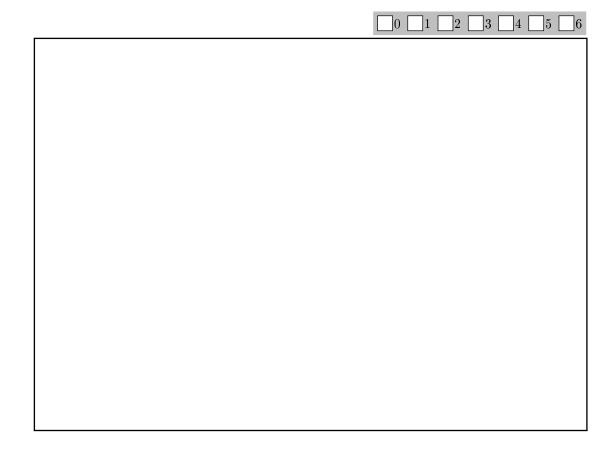
En Java ce dernier est calculé par theta = Math.atan2(y, x)

- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

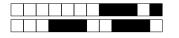
Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.

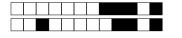




Question 13	Continuez	votre réponse	à la question	précédente i	ci si nécessaire.	0 0



Question 14	Continuez	votre réponse	à la question	précédente	ici si nécessaire.	



Une classe peut étendre une classe abstraite

Question	15	Pour	la	question	précédente,	développez	la	méthode	Polar	.#eana	ls	
& destion	10	1041	100	question	precedence,	developpez	100				3	$\Box$ 4
${f Question}$	<b>16</b> $\oplus$	Parr	ni l	es affirma	tions ci-desso	ous, cocher c	elle	s qui sont	vraies	:		
Une o	classe p	eut éte	end	re plusieu	rs classes							
Une o	classe a	bstrait	ер	eut contei	nir des const	ructeurs						
Une o	classe a	bstrait	ер	eut implé	menter une i	nterface						
					sse et implén		eurs	interfaces				
		=			méthodes pr							
Une i	interfact	e peut	coı	ntenir des	constructeur	.s						



# Question $17 \oplus$ Soit le code ci-dessous :

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :

```
ComparePerson byAge = new ComparePersonByAge() {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
};

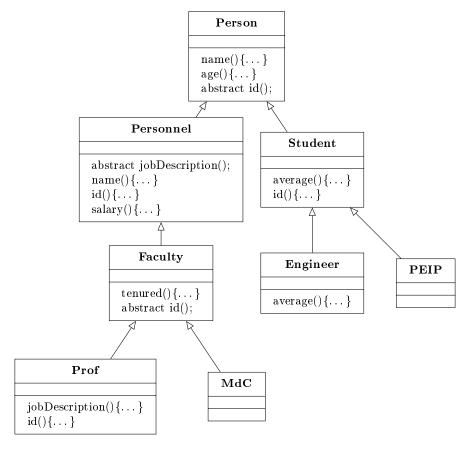
ComparePerson byAge
    = new ComparePersonByAge::compare;

ComparePerson byAge = new ComparePersonByAge();

ComparePerson byAge = new ComparePersonByAge();
```



## Question 18 $\bigoplus$ Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

Prof	Student
Faculty	Engineer
Person	PEIP
MdC	Personnel

QCMTEST Nom et prénom: Introduction à la programmation orientée objet 9/01/2020 Groupe: ..... Cochez les cases en mettant une  $\times$ . Le symbole  $\bigoplus$  indique que la question peut avoir zéro, une ou plusieurs bonnes réponses. Pour ces questions, cocher une bonne réponse apporte des points positifs ; cocher une mauvaise réponse peut apporter des points négatifs. Dans tout le code, les package et les import sont censés être correctement déclarés. Toute classe est supposée être dans le bon package, dans le bon fichier, avec les bons import.Question 1 La classe donnée class ClasseDonnee {} pourrait être sous-classée par extends pourrait être instanciée new par ClasseDonnee ClasseDonnee() Question 2 La classe donnée final class ClasseDonnee { ClasseDonnee() {} pourrait être sous-classée par extends pourrait être instanciée new par ClasseDonnee ClasseDonnee() Question 3  $\bigoplus$ La classe donnée class ClasseDonnee { private ClasseDonnee() {} pourrait  $\hat{\mathrm{e}}\mathrm{tre}$ pourrait être sous-classée par extends instanciée par new ClasseDonnee() ClasseDonnee Question 4 Erreur d'exécution Soit le code ci-dessous ; que se passe-t-il?

public class Beta {

}

public void throwup() throws RuntimeException {

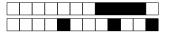
throw new RuntimeException();
}
public static void main(String[] args) {

Beta beta = new Beta();
beta.throwup();

System.out.println("Done");

Erreur de compilation

Affiche "Done"



## Question 5 $\bigoplus$

Erreur d'exécution Soit le code ci-dessous ; que se passe-t-il? Affiche "Done" public class Alpha { public void throwup() throws Exception { Erreur de compilation throw new Exception(); public static void main(String[] args) { Alpha alpha = new Alpha(); alpha.throwup(); System.out.println("Done"); Question 6 Soit le code ci-dessous ; que se passe-t-il? Affiche "Caught exception" et "Finallied exception" public class Epsilon { public void throwup() throws RuntimeException { Erreur d'exécution throw new RuntimeException(); Affiche seulement "Caught exception" public static void main(String[] args) { Epsilon epsilon = new Epsilon(); Erreur de compilation try { epsilon.throwup(); Affiche seulement "Finallied exception" } catch (Exception e) { System.out.println("Caught exception"); } finally { System.out.println("Finallied exception");

#### Question 7 Soit le code :

}

```
interface Somethingable {
    void something();
}

interface Whateverable extends Somethingable {
    void whatever();
}

class TheThing implements Whateverable{
    PPP void something() {
        System.out.println("Somethinging");
    }

    PPP void whatever() {
        System.out.println("Whatevering");
    }
}
```

```
class Doofer {
    void doo(Somethingable sable) {
        sable.XXX();
    }

    void doo(Whateverable sable) {
        sable.YYY();
    }

    void fer(TheThing tt) {
        tt.ZZZ();
    }

    public static void main(String[] args) {
        Doofer doofer = new Doofer();
        TheThing the = new TheThing();
        doofer.doo(the);
        doofer.fer(the);
    }
}
```

Par quels niveaux d'accès pourrait-on remplacer PPP?

private	package-private
public	protected
Question 8 $\oplus$ placer XXX ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
whatever	something

Question 9 ⊕ placer YYY ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
whatever	something
$\begin{array}{c} \mathbf{Question}  10  \bigoplus \\ \mathbf{remplacer}  \mathbf{ZZZ} \ ? \end{array}$	Pour le code de la question précédente, par quelles méthodes pourrait-on
whatever	something

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

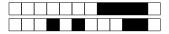
@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit





Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r \cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r \sin(\theta)$   $\theta = \tan^{-1} \frac{y}{x}$ 

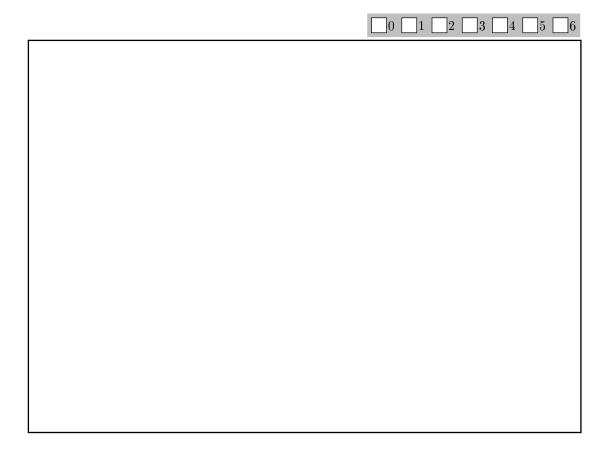
En Java ce dernier est calculé par theta = Math.atan2(y, x)

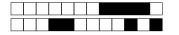
- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
       return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

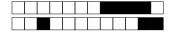
Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.





Question 13	Continuez	votre réponse	à la question	précédente i	ci si nécessaire.	0 0





Une interface peut contenir des méthodes privées

Une classe peut étendre plusieurs classes

Question 15 Pour la question précédente, développez la méthode Polar#equals.  $\boxed{0}$   $\boxed{1}$   $\boxed{2}$   $\boxed{3}$ Une classe abstraite peut implémenter une interface Une classe peut étendre une classe abstraite Une interface peut contenir des constructeurs Une classe peut étendre une classe et implémenter plusieurs interfaces Une classe abstraite peut contenir des constructeurs



## Question $17 \oplus$ Soit le code ci-dessous :

```
@FunctionalInterface
interface ComparePerson {
    /**
    * Oreturn negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :

```
ComparePerson byAge = (p1, p2) -> p1.age - p2.age;

ComparePerson byAge = new ComparePersonByAge::compare;

ComparePerson byAge = new ComparePersonByAge();
```

```
ComparePerson byAge = new ComparePersonByAge() {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
};
```



## Question 18 $\bigoplus$ Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

Prof	Person
MdC	Student
Personnel	PEIP
Engineer	Faculty

QCM

Introduction à la programmation
orientée objet
9/01/2020

Nom et	pı	ré	n	O	n	1	•								
					٠	٠									
Groupe	:														

Cochez les cases en mettant une  $\times$ .

} }

Le symbole  $\bigoplus$  indique que la question peut avoir zéro, une ou plusieurs bonnes réponses. Pour ces questions, cocher une bonne réponse apporte des points positifs ; cocher une mauvaise réponse peut apporter des points négatifs.

Dans tout le code, les package et les import sont censés être correctement déclarés. Toute classe est supposée être dans le bon package, dans le bon fichier, avec les bons import.

Question 1   La classe donnée							
<pre>class ClasseDonnee {    private ClasseDonnee() {} }</pre>							
pourrait être sous-classée par extends ClasseDonnee	pourrait être instanciée par new ClasseDonnee()						
Question 2   La classe donnée							
class ClasseDonnee {}							
pourrait être sous-classée par <b>extends</b> ClasseDonnee	pourrait être instanciée par <b>new</b> ClasseDonnee()						
Question 3   La classe (enum) donnée  enum ClasseDonnee {    INSTANCE; }							
pourrait être sous-classée par extends ClasseDonnee	pourrait être instanciée par <b>new</b> ClasseDonnee()						
Question $4 \oplus$							
Soit le code ci-dessous ; que se passe-t-il ?	Erreur d'exécution						
<pre>public class Gamma {    public void throwup() throws Exception {       throw new Exception();</pre>	Affiche "Caught exception" et "Finallied exception"						
} public static void main(String[] args) {	Affiche seulement "Finallied exception"						
Gamma gamma = new Gamma(); try {	Erreur de compilation						
<pre>gamma.throwup(); } catch (Exception e) {</pre>	Affiche seulement "Caught exception"						
<pre>System.out.println("Caught exception"); } finally {</pre>							
System.out.println("Finallied exception");							



Affiche "Done" Soit le code ci-dessous ; que se passe-t-il? public class Beta { Erreur d'exécution public void throwup() throws RuntimeException { Erreur de compilation throw new RuntimeException(); public static void main(String[] args) { Beta beta = new Beta(); beta.throwup(); System.out.println("Done"); } Question 6 Soit le code ci-dessous ; que se passe-t-il? Erreur de compilation public class Delta { Erreur d'exécution public void throwup() throws Exception { Affiche "Caught exception" et "Finallied throw new Exception(); public static void main(String[] args) { Delta delta = new Delta(); Affiche seulement "Caught exception" try { delta.throwup(); Affiche seulement "Finallied exception" } catch (Exception e) { System.out.println("Caught exception"); return; } finally { System.out.println("Finallied exception");

### Question 7 Soit le code :

}

```
interface Somethingable {
    void something();
}

interface Whateverable extends Somethingable {
    void whatever();
}

class TheThing implements Whateverable{
    PPP void something() {
        System.out.println("Somethinging");
    }

    PPP void whatever() {
        System.out.println("Whatevering");
    }
}
```

```
class Doofer {
    void doo(Somethingable sable) {
        sable.XXX();
}

void doo(Whateverable sable) {
        sable.YYY();
}

void fer(TheThing tt) {
        tt.ZZZ();
}

public static void main(String[] args) {
        Doofer doofer = new Doofer();
        TheThing the = new TheThing();
        doofer.doo(the);
        doofer.fer(the);
}
```

Par quels niveaux d'accès pourrait-on remplacer PPP ?

public protected	package-private private
Question 8 $\bigoplus$ placer XXX ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem
something	whatever

Question 9 $\bigoplus$ placer YYY ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem
something	whatever
Question 10 $\bigoplus$ remplacer ZZZ ?	Pour le code de la question précédente, par quelles méthodes pourrait-or
whatever	something

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit



Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r \cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r \sin(\theta)$   $\theta = \tan^{-1} \frac{y}{x}$ 

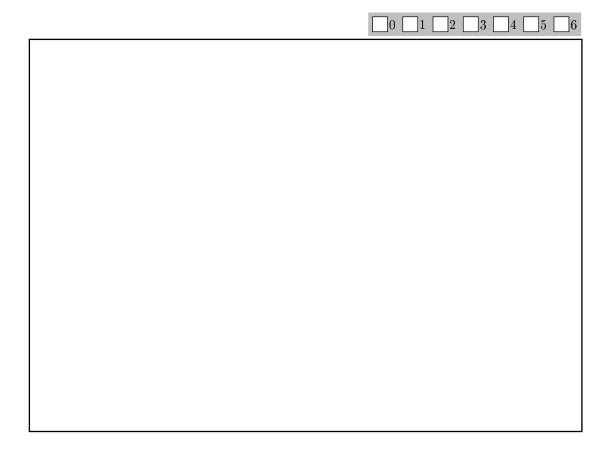
En Java ce dernier est calculé par theta = Math.atan2(y, x)

- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
       return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.





Question 13	Continuez	votre réponse à la question précédente ici si nécessaire.	



Question 14	Continuez	votre réponse à l	la question pré	cédente ici s	si nécessaire.	0 0



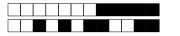
Question	15	Pour	la	question	précédente,	développez	la	méthode	equals. $2  \boxed{3}$	4
Une o	nterface classe ab classe pe	e peut ostrait eut éte eut éte	cor e p endi	ntenir des eut implé re une cla re une cla	tions ci-desse constructeur menter une i sse et implér sse abstraite	rs nterface nenter plusie				
Une i	$_{ m nterface}$	e peut	cor		nir des const méthodes pr rs classes					

# Question $17 \oplus$ Soit le code ci-dessous :

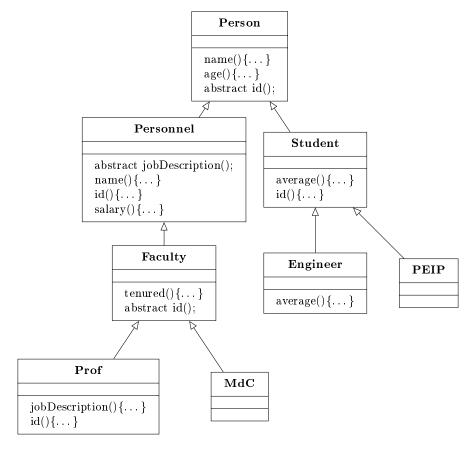
```
@FunctionalInterface
interface ComparePerson {
    /**
    * Oreturn negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :

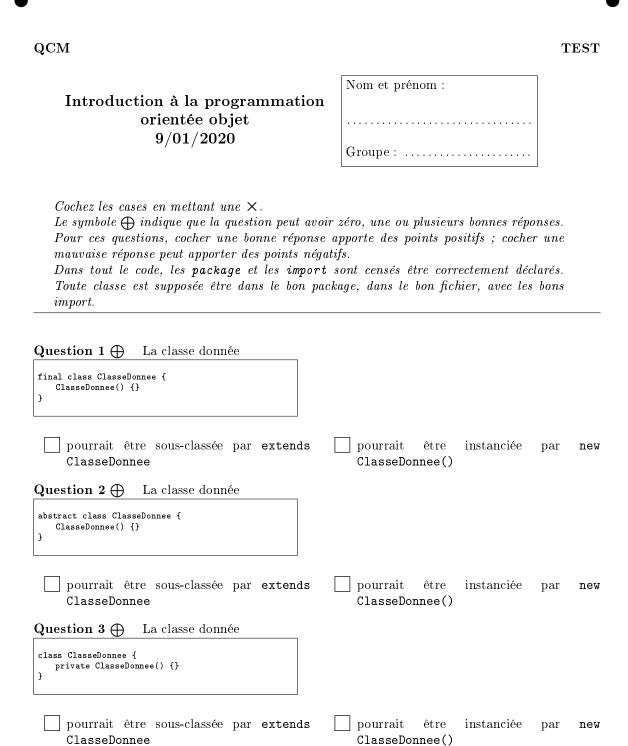


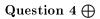
## Question 18 $\bigoplus$ Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

Engineer	Person
Personnel	PEIP
MdC	$\square$ Faculty
Student	Prof





<pre>Soit le code ci-dessous ; que se passe-t-il ?  public class Epsilon {     public void throwup() throws RuntimeException {         throw new RuntimeException();     }      public static void main(String[] args) {         Epsilon epsilon = new Epsilon();         try {             epsilon.throwup();       } catch (Exception e) {             System.out.println("Caught exception");       } finally {             System.out.println("Finallied exception");       } }</pre>	Affiche seulement "Caught exception"  Affiche seulement "Finallied exception"  Erreur d'exécution  Affiche "Caught exception" et "Finallied exception"  Erreur de compilation
Question 5	
<pre>Soit le code ci-dessous ; que se passe-t-il ?  public class Alpha {     public void throwup() throws Exception {         throw new Exception();     }     public static void main(String[] args) {         Alpha alpha = new Alpha();         alpha.throwup();         System.out.println("Done");     } }</pre>	☐ Erreur de compilation ☐ Erreur d'exécution ☐ Affiche "Done"
Question 6	
<pre>Soit le code ci-dessous ; que se passe-t-il ?  public class Beta {     public void throwup() throws RuntimeException {         throw new RuntimeException();     }     public static void main(String[] args) {         Beta beta = new Beta();         beta.throwup();         System.out.println("Done");     } }</pre>	☐ Affiche "Done" ☐ Erreur d'exécution ☐ Erreur de compilation



### Question 7 Soit le code :

```
interface Somethingable {
   void something();
}

interface Whateverable extends Somethingable {
   void whatever();
}

class TheThing implements Whateverable{
   PPP void something() {
       System.out.println("Somethinging");
   }

   PPP void whatever() {
       System.out.println("Whatevering");
   }
}
```

<pre>class Doofer {   void doo(Somethingable sable) {      sable.XXX();   }</pre>	
<pre>void doo(Whateverable sable) {     sable.YYY(); }</pre>	
<pre>void fer(TheThing tt) {    tt.ZZZ(); }</pre>	
<pre>public static void main(String[] args) {    Doofer doofer = new Doofer();    TheThing the = new TheThing();    doofer.doo(the);    doofer.fer(the); }</pre>	
}	

Par quels niveaux d'accès pourrait-on remplacer PPP? public protected private package-private Question 8  $\bigoplus$ Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer XXX? something whatever Question 9 Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer YYY? something whatever Question 10  $\bigoplus$ Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer ZZZ? something whatever

•

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

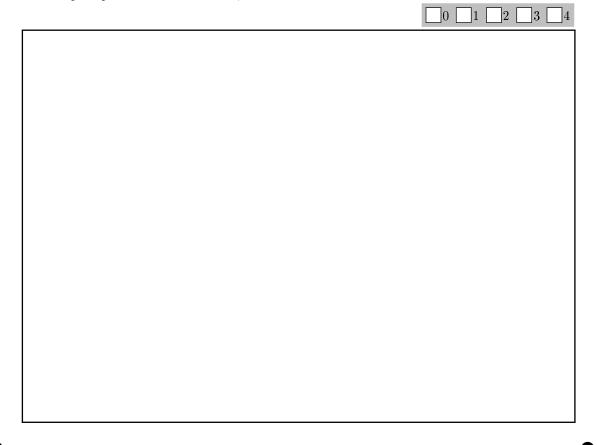
public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit





Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r \cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r \sin(\theta)$   $\theta = \tan^{-1} \frac{y}{x}$ 

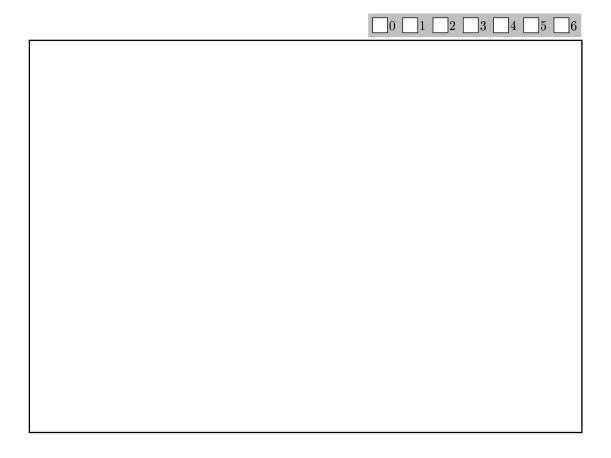
En Java ce dernier est calculé par theta = Math.atan2(y, x)

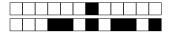
- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.

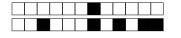




Question 13	Continuez votre réponse à la question précédente ici si nécessaire.	



Question 14	Continuez	votre réponse à l	la question pré	cédente ici s	si nécessaire.	0 0



0 "		D	,			1/ 1	1	4.1 1			-	
Question	15	Pour	la	question	précédente,	developpez	la					
									1		3	4
Question	$16 \bigoplus$	Parr	ni l	es affirma	tions ci-desso	ous, cocher c	elle	s qui sont	vraies	:		
Une o	classe al	bstrait	ер	eut conte	nir des const	ructeurs						
Une i	interface	e peut	col	ntenir des	méthodes pr	rivées						
Une o	classe p	eut éte	$\operatorname{end}$	re une cla	sse abstraite							
Une o	classe al	bstrait	ер	eut implé	menter une i	$_{ m nterface}$						
Une i	interface	e peut	col	ntenir des	constructeur	`S						
Une o	classe p	eut éte	$\operatorname{end}$	re plusieu	rs classes							

Une classe peut étendre une classe et implémenter plusieurs interfaces



## Question $17 \oplus$ Soit le code ci-dessous :

```
@FunctionalInterface
interface ComparePerson {
    /**
    * @return negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :

```
ComparePerson byAge = new ComparePersonByAge();

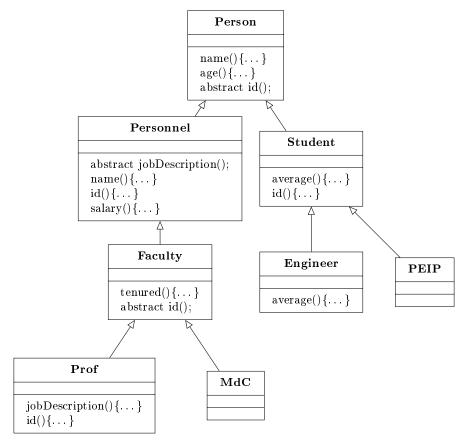
ComparePerson byAge = (p1, p2) -> p1.age - p2.age;

ComparePerson byAge
= new ComparePersonByAge::compare;
```

```
ComparePerson byAge = new ComparePersonByAge() {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
};
```



Question 18  $\bigoplus$  Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

PEIP	Person
Engineer	MdC
Personnel	Student
Faculty	Prof



pourrait être sous-classée par extends

pourrait être sous-classée par extends

ClasseDonnee

ClasseDonnee

Question 2

class ClasseDonnee {}

pourrait

Question 3

pourrait

ClasseDonnee()

ClasseDonnee()

final class ClasseDonnee {
 ClasseDonnee() {}

être

être

La classe donnée

instanciée

instanciée

La classe donnée

par

par

new

new

## Question 4 $\bigoplus$

Soit le code ci-dessous ; que se passe-t-il ?	Erreur de compilation
<pre>public class Delta {     public void throwup() throws Exception {         throw new Exception();     }     public static void main(String[] args) {         Delta delta = new Delta();         try {             delta.throwup();       } catch (Exception e) {             System.out.println("Caught exception");             return;       } finally {             System.out.println("Finallied exception");       }     } }</pre>	Affiche seulement "Finallied exception"  Affiche "Caught exception" et "Finallied exception"  Affiche seulement "Caught exception"  Erreur d'exécution
Question 5 $\bigoplus$	

```
Erreur d'exécution
Soit le code ci-dessous ; que se passe-t-il?
                                                              Affiche seulement "Finallied exception"
public class Gamma {
   public void throwup() throws Exception {
                                                              Affiche "Caught exception" et "Finallied
      throw new Exception();
                                                              exception"
   public static void main(String[] args) {
      Gamma gamma = new Gamma();
                                                              Affiche seulement "Caught exception"
      try {
          gamma.throwup();
                                                              Erreur de compilation
      } catch (Exception e) {
          System.out.println("Caught exception");
      } finally {
   System.out.println("Finallied exception");
   }
}
```

## Question 6 $\bigoplus$

Soit le code ci-dessous ; que se passe-t-il ?

public class Epsilon {
 public void throwup() throws RuntimeException {
 throw new RuntimeException();
 }
 public static void main(String[] args) {
 Epsilon epsilon = new Epsilon();
 try {
 epsilon.throwup();
 } catch (Exception e) {
 System.out.println("Caught exception");
 } finally {
 System.out.println("Finallied exception");
 }
}

Affiche seulement "Finallied exception"

Erreur de compilation

Affiche seulement "Caught exception"

Affiche "Caught exception" et "Finallied

exception"

Erreur d'exécution



### **Question 7** Soit le code :

```
interface Somethingable {
   void something();
}

interface Whateverable extends Somethingable {
   void whatever();
}

class TheThing implements Whateverable{
   PPP void something() {
       System.out.println("Somethinging");
   }

   PPP void whatever() {
       System.out.println("Whatevering");
   }
}
```

```
class Doofer {
  void doo(Somethingable sable) {
    sable.XXX();
  }
  void doo(Whateverable sable) {
    sable.YYY();
  }
  void fer(TheThing tt) {
    tt.ZZZ();
  }
  public static void main(String[] args) {
    Doofer doofer = new Doofer();
    TheThing the = new TheThing();
    doofer.doo(the);
    doofer.fer(the);
  }
}
```

Par quels niveaux d'accès pourrait-on remplacer PPP? package-private private public protected Question 8 Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer XXX? whatever something Question 9 Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer YYY? whatever something Question 10  $\bigoplus$ Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer ZZZ? something whatever



Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit





Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r\cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r\sin(\theta)$   $\theta = \tan^{-1}\frac{y}{x}$ 

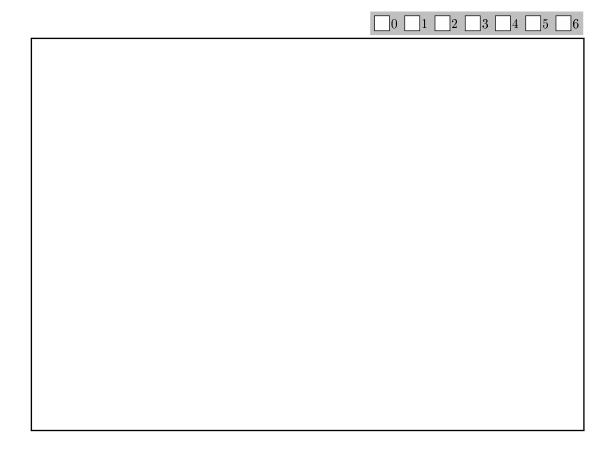
En Java ce dernier est calculé par theta = Math.atan2(y, x)

- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.

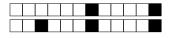




Question 13	Continuez	votre réponse	à la question	précédente i	ci si nécessaire.	0 0



Question 14	Continuez	votre réponse	à la question	précédente	ici si nécessaire.	



Une classe abstraite peut contenir des constructeurs

Une classe peut étendre plusieurs classes

Question	15	Pour	la	question	précédente,	développez	la	méthode Polar#equals.
•				1	,	11		
Question	<b>16</b> $\bigoplus$	Parm	ıi l∈	es affirma	tions ci-desse	ous, cocher c	elle	s qui sont vraies :
Une i	$_{ m nterface}$	e peut (	con	tenir des	constructeur	rs		
Une i	$_{ m nterface}$	e peut (	con	tenir des	méthodes pr	rivées		
						nenter plusie	eurs	interfaces
					menter une i			
Une o	classe pe	eut éte	ndr	e une cla	sse abstraite			



# Question $17 \oplus$ Soit le code ci-dessous :

```
@FunctionalInterface
interface ComparePerson {
    /**
    * @return negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

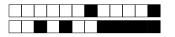
```
class Person {
    private int age;
    private String name;

Person(String name, int age) {
        this.age = age;
        this.name = name;
    }
}

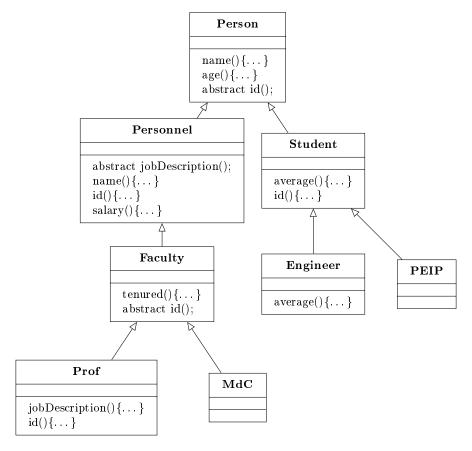
class Main {
    public static void main(String[] args) {
        // code extract here

        System.out.print(byAge.compare(
            new Person("Fred", 23),
            new Person("Not Fred", 24)));
        System.out.print(byAge.compare(
            new Person("Fred", 23),
            new Person("Fred", 23));
        System.out.print(byAge.compare(
            new Person("Not Fred", 23)));
        System.out.print(byAge.compare(
            new Person("Fred", 23),
            new Person("Fred", 23),
            new Person("Not Fred", 22)));
    }
}
```

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :

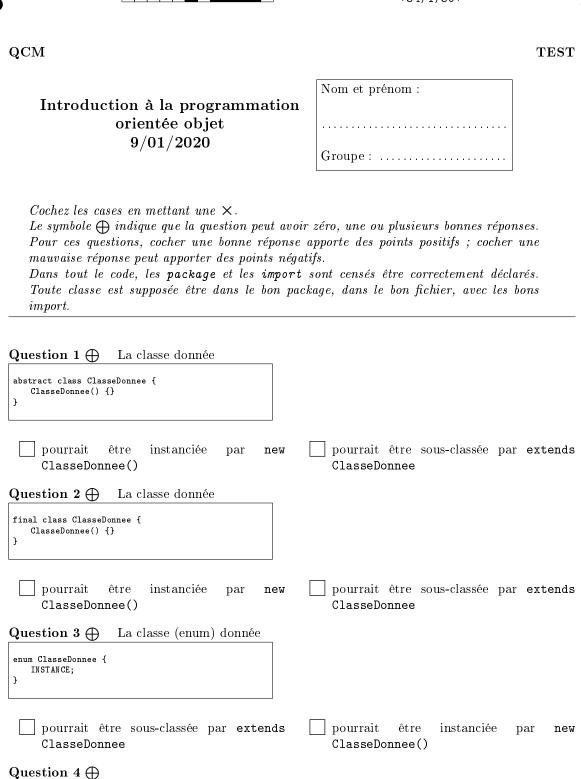


Question 18  $\bigoplus$  Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

MdC	PEIP
Engineer	Person
Prof	Personnel
Student	Faculty



Soit le code ci-dessous ; que se passe-t-il?

public void throwup() throws Exception {

public static void main(String[] args) {

throw new Exception();

Alpha alpha = new Alpha();
alpha.throwup();

System.out.println("Done");

public class Alpha {

}

Erreur de compilation

Affiche "Done"

Erreur d'exécution



```
Erreur d'exécution
Soit le code ci-dessous ; que se passe-t-il?
public class Delta {
                                                            Affiche "Caught exception" et "Finallied
   public void throwup() throws Exception {
                                                            exception"
      throw new Exception();
                                                        Erreur de compilation
   public static void main(String[] args) {
      Delta delta = new Delta();
                                                            Affiche seulement "Caught exception"
      try {
          delta.throwup();
                                                            Affiche seulement "Finallied exception"
      } catch (Exception e) {
          System.out.println("Caught exception");
          return:
      } finally {
          System.out.println("Finallied exception");
}
Question 6
                                                            Erreur de compilation
Soit le code ci-dessous ; que se passe-t-il?
public class Epsilon {
                                                            Affiche "Caught exception" et "Finallied
   public void throwup() throws RuntimeException {
                                                            exception"
      throw new RuntimeException();
```

#### **Question 7** Soit le code :

try {

}

} finally {

public static void main(String[] args) {
 Epsilon epsilon = new Epsilon();

System.out.println("Caught exception");

System.out.println("Finallied exception");

epsilon.throwup();

} catch (Exception e) {

```
interface Somethingable {
   void something();
}

interface Whateverable extends Somethingable {
   void whatever();
}

class TheThing implements Whateverable{
   PPP void something() {
       System.out.println("Somethinging");
   }

   PPP void whatever() {
       System.out.println("Whatevering");
   }
}
```

```
class Doofer {
   void doo(Somethingable sable) {
      sable.XXX();
}

void doo(Whateverable sable) {
    sable.YYY();
}

void fer(TheThing tt) {
    tt.ZZZ();
}

public static void main(String[] args) {
    Doofer doofer = new Doofer();
    TheThing the = new TheThing();
    doofer.doo(the);
    doofer.fer(the);
}
```

Affiche seulement "Finallied exception"

Affiche seulement "Caught exception"

Erreur d'exécution

Par quels niveaux d'accès pourrait-on remplacer PPP?

package-private	$oxedsymbol{\Box}$ $\operatorname{\texttt{private}}$
protected	public

Question 8 $\bigoplus$ placer XXX ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem
whatever	something
Question 9 $\bigoplus$ placer YYY ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem
something	whatever
Question 10 $\bigoplus$ remplacer ZZZ ?	Pour le code de la question précédente, par quelles méthodes pourrait-or
whatever	something

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

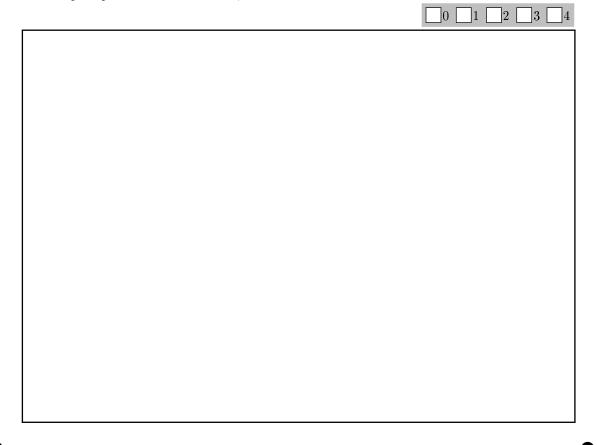
public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit





Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r\cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r\sin(\theta)$   $\theta = \tan^{-1}\frac{y}{x}$ 

En Java ce dernier est calculé par theta = Math.atan2(y, x)

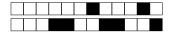
- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.

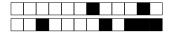




Question 13	Continuez	votre réponse à la question précédente ici si nécessaire.	



Question 14	Continuez	votre réponse à l	la question pré	cédente ici s	i nécessaire.	0 0



Une interface peut contenir des méthodes privées
 Une classe peut étendre une classe abstraite
 Une classe peut étendre plusieurs classes

Question 15 Pour la question précédente, développez la méthode Polar#equals.  $\boxed{0}$   $\boxed{1}$   $\boxed{2}$   $\boxed{3}$ Parmi les affirmations ci-dessous, cocher celles qui sont vraies : Question 16  $\bigoplus$ Une classe peut étendre une classe et implémenter plusieurs interfaces Une classe abstraite peut contenir des constructeurs Une interface peut contenir des constructeurs Une classe abstraite peut implémenter une interface



## Question $17 \oplus$ Soit le code ci-dessous :

```
@FunctionalInterface
interface ComparePerson {
    /**
    * Oreturn negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

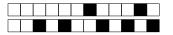
Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :

```
ComparePerson byAge = new ComparePersonByAge();

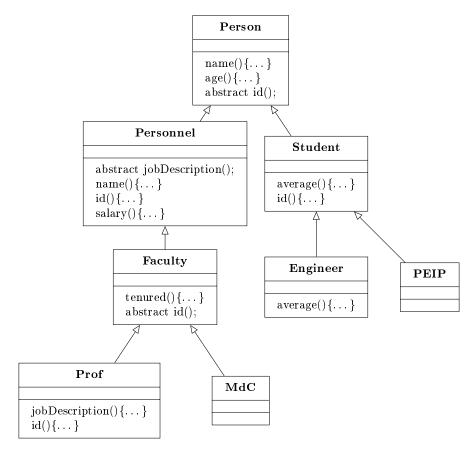
ComparePerson byAge = new ComparePersonByAge::compare;

ComparePerson byAge = (p1, p2) -> p1.age - p2.age;
```

```
ComparePerson byAge = new ComparePersonByAge() {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
};
```



# Question 18 $\bigoplus$ Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

PEIP	Personnel
Person	$oxedsymbol{\Box}$ Faculty
Prof	MdC
Engineer	Student

+35/1/20+ QCMTEST Nom et prénom: Introduction à la programmation orientée objet 9/01/2020 Groupe: ..... Cochez les cases en mettant une  $\times$ . Le symbole  $\bigoplus$  indique que la question peut avoir zéro, une ou plusieurs bonnes réponses. Pour ces questions, cocher une bonne réponse apporte des points positifs ; cocher une mauvaise réponse peut apporter des points négatifs. Dans tout le code, les package et les import sont censés être correctement déclarés. Toute classe est supposée être dans le bon package, dans le bon fichier, avec les bons import.Question 1 La classe donnée class ClasseDonnee {} pourrait être sous-classée par extends pourrait  $\hat{\mathrm{e}}\mathrm{tre}$ instanciée par new ClasseDonnee() ClasseDonnee La classe donnée Question 2 ( class ClasseDonnee { private ClasseDonnee() {}

### Question 3 La classe donnée

ClasseDonnee

abstract class ClasseDonnee {
 ClasseDonnee() {}
}

Dourrait être instanciée par new ClasseDonnee()

pourrait être sous-classée par extends

\_\_\_\_ pourrait être sous-classée par extends ClasseDonnee

instanciée

# Question 4

Soit le code ci-dessous ; que se passe-t-il ?

public class Gamma {
 public void throwup() throws Exception {
 throw new Exception();
 }
 public static void main(String[] args) {
 Gamma gamma = new Gamma();
 try {
 gamma.throwup();
 } catch (Exception e) {
 System.out.println("Caught exception");
 } finally {
 System.out.println("Finallied exception");
 }
 }
}

Affiche "Caught exception" et "Finallied exception"

Affiche seulement "Finallied exception"

Affiche seulement "Caught exception"

Erreur de compilation

Erreur d'exécution

pourrait

ClasseDonnee()

être



# Question 5 $\bigoplus$

Affiche "Done" Soit le code ci-dessous ; que se passe-t-il? public class Beta { Erreur de compilation public void throwup() throws RuntimeException { Erreur d'exécution throw new RuntimeException(); public static void main(String[] args) { Beta beta = new Beta(); beta.throwup(); System.out.println("Done"); } Question 6 Affiche "Done" Soit le code ci-dessous ; que se passe-t-il? public class Alpha { Erreur d'exécution public void throwup() throws Exception { Erreur de compilation throw new Exception(); public static void main(String[] args) { Alpha alpha = new Alpha(); alpha.throwup(); System.out.println("Done");

#### **Question 7** Soit le code :

```
interface Somethingable {
    void something();
}

interface Whateverable extends Somethingable {
    void whatever();
}

class TheThing implements Whateverable{
    PPP void something() {
        System.out.println("Somethinging");
    }

    PPP void whatever() {
        System.out.println("Whatevering");
    }
}
```

```
class Doofer {
    void doo(Somethingable sable) {
        sable.XXX();
    }

    void doo(Whateverable sable) {
        sable.YYY();
    }

    void fer(TheThing tt) {
        tt.ZZZ();
    }

    public static void main(String[] args) {
        Doofer doofer = new Doofer();
        TheThing the = new TheThing();
        doofer.doo(the);
        doofer.fer(the);
    }
}
```

Par quels niveaux d'accès pourrait-on remplacer PPP?

package-priv	vate public protected
Question 8 $\oplus$ placer XXX ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
something	whatever
Question 9 $\bigoplus$ placer YYY ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
something	☐ whatever

+35/3/18+

Question 10 $\bigoplus$ remplacer ZZZ ?	Pour le code de la question précédente, par quelles méthodes pourrait-on
something	☐ whatever

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit





Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r\cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r\sin(\theta)$   $\theta = \tan^{-1}\frac{y}{x}$ 

En Java ce dernier est calculé par theta = Math.atan2(y, x)

- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.





Question 13	Continuez	votre réponse à la question précédente ici si nécessaire.	



Question 14	Continuez	votre réponse	à la question	précédente	ici si nécessaire.	



Une classe abstraite peut contenir des constructeurs

Question	<b>15</b>	Pour	la	question	précédente,	développez	la	méthode Polar#equals.
Question	<b>16</b> $\oplus$	Parn	ni le	es affirma	tions ci-desso	ous, cocher c	elle	s qui sont vraies :
Une o	classe a	abstrait	ер	eut implé	menter une i	nterface		
Une i	interface	ce peut	cor	ntenir des	méthodes pi	rivées		
Une o	classe j	peut éte	endi	re une cla	sse et implér	nenter plusie	eurs	interfaces
Une o	classe j	peut éte	endi	re plusieu	rs classes			
Une	classe 1	peut éte	endi	re une cla	sse abstraite			



## Question $17 \oplus$ Soit le code ci-dessous :

```
@FunctionalInterface
interface ComparePerson {
    /**
    * @return negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

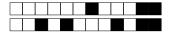
Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :

```
ComparePerson byAge = new ComparePersonByAge();

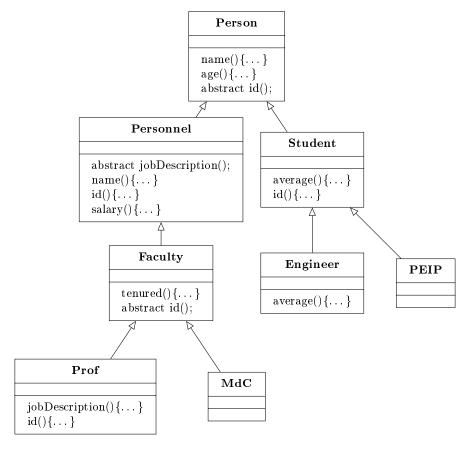
ComparePerson byAge = new ComparePersonByAge::compare;

ComparePerson byAge = (p1, p2) -> p1.age - p2.age;
```

```
ComparePerson byAge = new ComparePersonByAge() {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
};
```

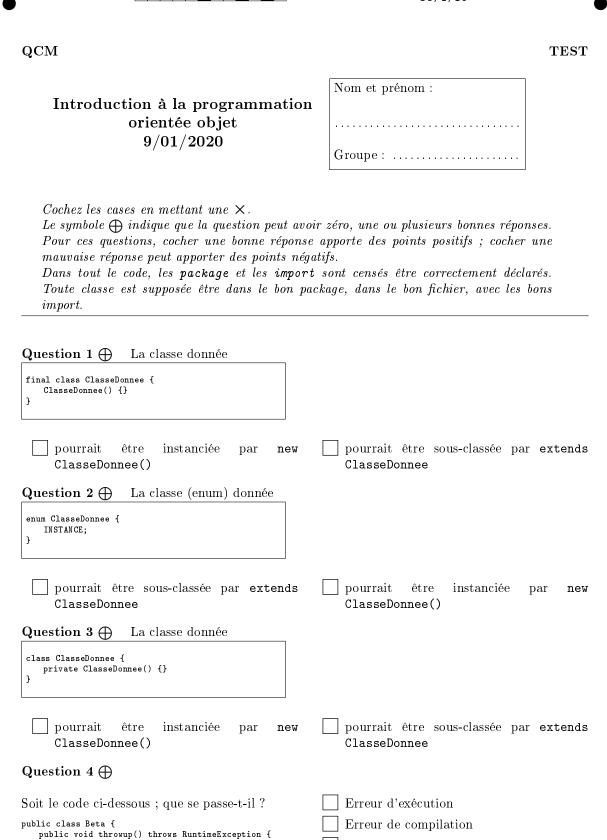


# Question 18 Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

Person	Engineer
Personnel	Faculty
PEIP	MdC
Prof	Student



throw new RuntimeException();
}
public static void main(String[] args) {

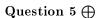
System.out.println("Done");

Beta beta = new Beta();
beta.throwup();

}

Affiche "Done"

Erreur de compilation



Soit le code ci-dessous ; que se passe-t-il?

```
public class Delta {
                                                             Erreur d'exécution
   public void throwup() throws Exception {
                                                             Affiche seulement "Caught exception"
      throw new Exception();
   public static void main(String[] args) {
                                                             Affiche "Caught exception" et "Finallied
      Delta delta = new Delta();
                                                             exception"
      try {
          delta.throwup();
                                                             Affiche seulement "Finallied exception"
      } catch (Exception e) {
          System.out.println("Caught exception");
          return:
      } finally {
          System.out.println("Finallied exception");
   }
}
Question 6
```

```
Affiche seulement "Finallied exception"
Soit le code ci-dessous ; que se passe-t-il?
public class Epsilon {
                                                             Erreur de compilation
   public void throwup() throws RuntimeException {
      throw new RuntimeException();
                                                             Erreur d'exécution
   public static void main(String[] args) {
                                                             Affiche "Caught exception" et "Finallied
      Epsilon epsilon = new Epsilon();
                                                             exception"
      try {
          epsilon.throwup();
                                                             Affiche seulement "Caught exception"
      } catch (Exception e) {
          System.out.println("Caught exception");
      } finally {
         System.out.println("Finallied exception");
   }
}
```

#### **Question 7** Soit le code :

```
interface Somethingable {
    void something();
}

interface Whateverable extends Somethingable {
    void whatever();
}

class TheThing implements Whateverable{
    PPP void something() {
        System.out.println("Somethinging");
    }

    PPP void whatever() {
        System.out.println("Whatevering");
    }
}
```

```
class Doofer {
   void doo(Somethingable sable) {
      sable.XXX();
}

void doo(Whateverable sable) {
    sable.YYY();
}

void fer(TheThing tt) {
    tt.ZZZ();
}

public static void main(String[] args) {
    Doofer doofer = new Doofer();
    TheThing the = new TheThing();
    doofer.doo(the);
    doofer.fer(the);
}
```

Par quels niveaux d'accès pourrait-on remplacer PPP?

private	package-private
protected	public

Question 8 $\bigoplus$ placer XXX ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
something	whatever
Question 9 $\bigoplus$ placer YYY ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
something	whatever
Question 10 $\bigoplus$ remplacer ZZZ ?	Pour le code de la question précédente, par quelles méthodes pourrait-on
Whatever	something

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

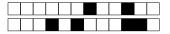
@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit





Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r\cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r\sin(\theta)$   $\theta = \tan^{-1}\frac{y}{x}$ 

En Java ce dernier est calculé par theta = Math.atan2(y, x)

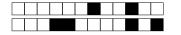
- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.





Question 13	Continuez votre réponse à la question précédente ici si nécessaire.	



Question 14	Continuez votre réponse à la question précédente ici si nécessaire.	



Question 15 Pour la question précédente, développez la méthode Polar#equals.  $\boxed{0}$   $\boxed{1}$   $\boxed{2}$   $\boxed{3}$ Question 16  $\bigoplus$ Parmi les affirmations ci-dessous, cocher celles qui sont vraies : Une classe peut étendre plusieurs classes Une classe abstraite peut contenir des constructeurs Une classe abstraite peut implémenter une interface Une interface peut contenir des méthodes privées Une classe peut étendre une classe abstraite Une interface peut contenir des constructeurs

Une classe peut étendre une classe et implémenter plusieurs interfaces

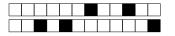


## Question 17 Soit le code ci-dessous :

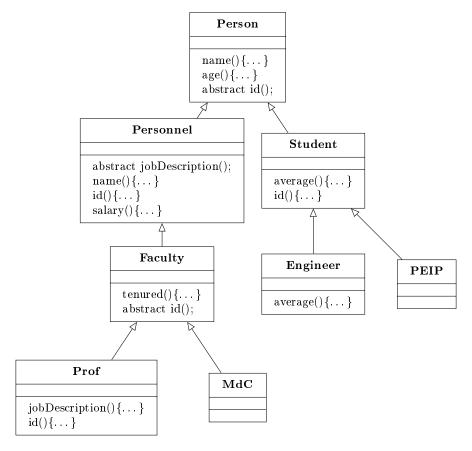
```
OFunctionalInterface
interface ComparePerson {
    /**
    * Oreturn negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :



Question 18  $\bigoplus$  Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

Personnel	Student
Person	MdC
Faculty	PEIP
Prof	Engineer

QCMTEST Nom et prénom: Introduction à la programmation orientée objet 9/01/2020 Groupe: ..... Cochez les cases en mettant une  $\times$ . Le symbole  $\bigoplus$  indique que la question peut avoir zéro, une ou plusieurs bonnes réponses. Pour ces questions, cocher une bonne réponse apporte des points positifs ; cocher une mauvaise réponse peut apporter des points négatifs. Dans tout le code, les package et les import sont censés être correctement déclarés. Toute classe est supposée être dans le bon package, dans le bon fichier, avec les bons import.Question 1 La classe donnée class ClasseDonnee {} pourrait être sous-classée par extends pourrait être instanciée par new ClasseDonnee ClasseDonnee() Question 2 La classe donnée abstract class ClasseDonnee { ClasseDonnee() {} pourrait être sous-classée par extends pourrait être instanciée par new ClasseDonnee ClasseDonnee() Question 3  $\bigoplus$ La classe donnée final class ClasseDonnee { ClasseDonnee() {} \_ pourrait être sous-classée par extends pourrait être instanciée new ClasseDonnee ClasseDonnee() Question 4 Erreur de compilation Soit le code ci-dessous ; que se passe-t-il? Affiche "Done" public class Alpha { public void throwup() throws Exception { throw new Exception(); Erreur d'exécution

public static void main(String[] args) {

Alpha alpha = new Alpha();
alpha.throwup();

System.out.println("Done");

}



```
Soit le code ci-dessous ; que se passe-t-il?
public class Gamma {
   public void throwup() throws Exception {
       throw new Exception();
   public static void main(String[] args) {
       Gamma gamma = new Gamma();
                                                                exception"
       try {
          gamma.throwup();
       } catch (Exception e) {
          System.out.println("Caught exception");
       } finally {
          System.out.println("Finallied exception");
   }
}
```

# Erreur d'exécution

	Amche	seulement	"Caugh	t	exception"
٦		_			_

- Affiche seulement "Finallied exception"
- Affiche "Caught exception" et "Finallied
- Erreur de compilation

## Question 6 $\bigoplus$

```
Soit le code ci-dessous ; que se passe-t-il?
public class Epsilon {
   public void throwup() throws RuntimeException {
       throw new RuntimeException();
   public static void main(String[] args) {
    Epsilon epsilon = new Epsilon();
       try {
           epsilon.throwup();
       } catch (Exception e) {
           System.out.println("Caught exception");
       } finally {
           System.out.println("Finallied exception");
   }
}
```

# Affiche seulement "Finallied exception"

Erreur de compilation

Affiche seulement "Caught exception"

- Affiche "Caught exception" et "Finallied exception"
- Erreur d'exécution

#### Question 7 Soit le code:

```
interface Somethingable {
   void something();
interface Whateverable extends Somethingable {
   void whatever();
class TheThing implements Whateverable{
   PPP void something() {
       System.out.println("Somethinging");
   PPP void whatever() {
       System.out.println("Whatevering");
}
```

```
class Doofer {
   void doo(Somethingable sable) {
        sable.XXX();
   void doo(Whateverable sable) {
        sable.YYY();
    void fer(TheThing tt) {
        tt.ZZZ();
   public static void main(String[] args) {
       Doofer doofer = new Doofer();
TheThing the = new TheThing();
        doofer.doo(the);
        doofer.fer(the);
}
```

Par quels niveaux d'accès pourrait-on remplacer PPP?

private public

package-private

protected



Question 8 $\bigoplus$ placer XXX ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
whatever	$\square$ something
Question 9 $\bigoplus$ placer YYY ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
whatever	$\square$ something
Question 10 $\bigoplus$ remplacer ZZZ ?	Pour le code de la question précédente, par quelles méthodes pourrait-on
whatever	<pre>something</pre>

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

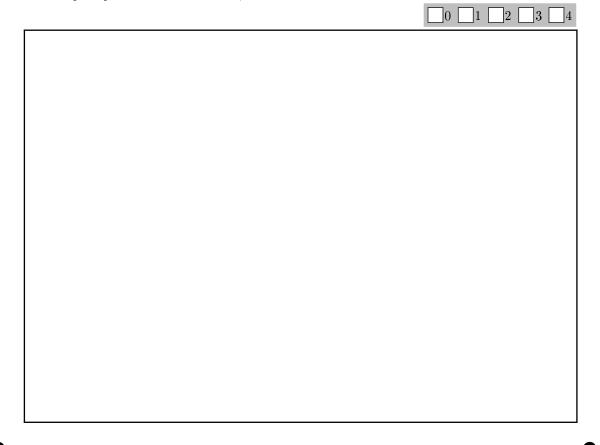
public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit





Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r \cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r \sin(\theta)$   $\theta = \tan^{-1} \frac{y}{x}$ 

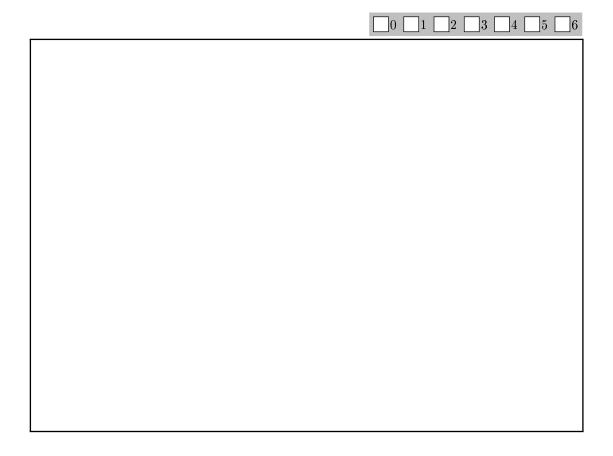
En Java ce dernier est calculé par theta = Math.atan2(y, x)

- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

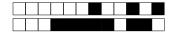
Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.





Question 13	Continuez	votre réponse	à la question	précédente i	ci si nécessaire.	0 0



uestion 14	Continuez votre réponse à la question précédente ici si nécessaire.	



Une classe abstraite peut contenir des constructeurs

Une interface peut contenir des constructeurs

Question 15 Pour la question précédente, développez la méthode Polar#equals.  $\boxed{0}$   $\boxed{1}$   $\boxed{2}$   $\boxed{3}$ Une classe abstraite peut implémenter une interface Une interface peut contenir des méthodes privées Une classe peut étendre plusieurs classes Une classe peut étendre une classe et implémenter plusieurs interfaces Une classe peut étendre une classe abstraite



## Question $17 \oplus$ Soit le code ci-dessous :

```
@FunctionalInterface
interface ComparePerson {
    /**
    * @return negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

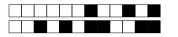
Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :

```
ComparePerson byAge = new ComparePersonByAge::compare;

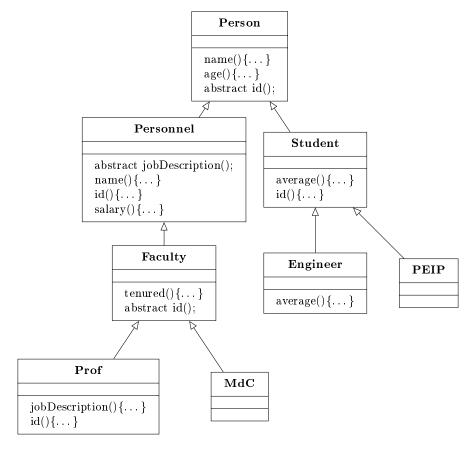
ComparePerson byAge = new ComparePersonByAge();

ComparePerson byAge = (p1, p2) -> p1.age - p2.age;
```

```
ComparePerson byAge = new ComparePersonByAge() {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
};
```



## Question 18 $\bigoplus$ Soit l'organisation ci-dessous de Polytech'Groland :

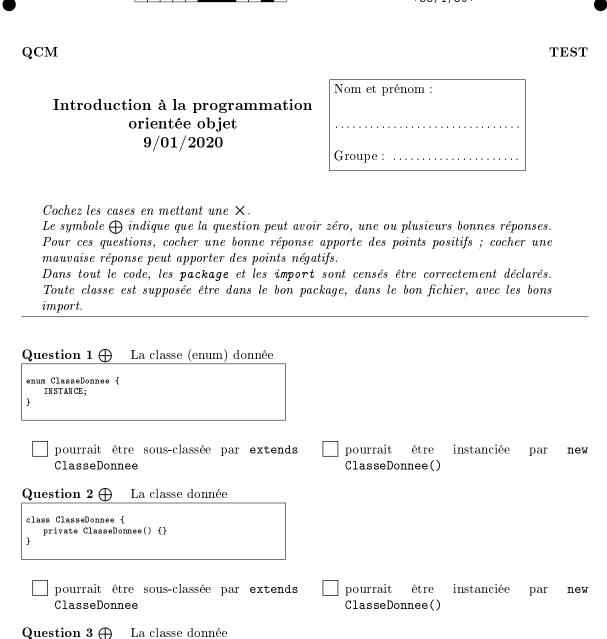


Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

Faculty	Person
MdC	Prof
Engineer	Personnel
PEIP	Student

pourrait être sous-classée par extends

ClasseDonnee



class ClasseDonnee {}

pourrait

ClasseDonnee()

être

instanciée

par

new

Erreur d'exécution Soit le code ci-dessous ; que se passe-t-il? Affiche seulement "Caught exception" public class Delta { public void throwup() throws Exception { Affiche "Caught exception" et "Finallied throw new Exception(); public static void main(String[] args) { Delta delta = new Delta(); Erreur de compilation try { delta.throwup(); Affiche seulement "Finallied exception" } catch (Exception e) { System.out.println("Caught exception"); return; } finally { System.out.println("Finallied exception"); }

## Question 5 $\bigoplus$

Soit le code ci-dessous ; que se passe-t-il? public class Beta { public void throwup() throws RuntimeException { throw new RuntimeException(); public static void main(String[] args) { Beta beta = new Beta(); beta.throwup(); System.out.println("Done");

# Question 6

Soit le code ci-dessous ; que se passe-t-il? public class Alpha { public void throwup() throws Exception { throw new Exception(); public static void main(String[] args) { Alpha alpha = new Alpha(); alpha.throwup(); System.out.println("Done");

Erreur d'exécution

Erreur de compilation

Affiche "Done"

Erreur de compilation

Affiche "Done"

Erreur d'exécution



### Question 7 Soit le code :

```
interface Somethingable {
   void something();
}

interface Whateverable extends Somethingable {
   void whatever();
}

class TheThing implements Whateverable{
   PPP void something() {
       System.out.println("Somethinging");
   }

   PPP void whatever() {
       System.out.println("Whatevering");
   }
}
```

```
class Doofer {
    void doo(Somethingable sable) {
        sable.XXX();
    }

    void doo(Whateverable sable) {
        sable.YYY();
    }

    void fer(TheThing tt) {
        tt.ZZZ();
    }

    public static void main(String[] args) {
        Doofer doofer = new Doofer();
        TheThing the = new TheThing();
        doofer.doo(the);
        doofer.fer(the);
    }
}
```

Par quels niveaux d'accès pourrait-on remplacer PPP? public protected package-private private Question 8 Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer XXX? whatever something Question 9 Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer YYY? something whatever Question 10  $\bigoplus$ Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer ZZZ? whatever something

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit



Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r\cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r\sin(\theta)$   $\theta = \tan^{-1}\frac{y}{x}$ 

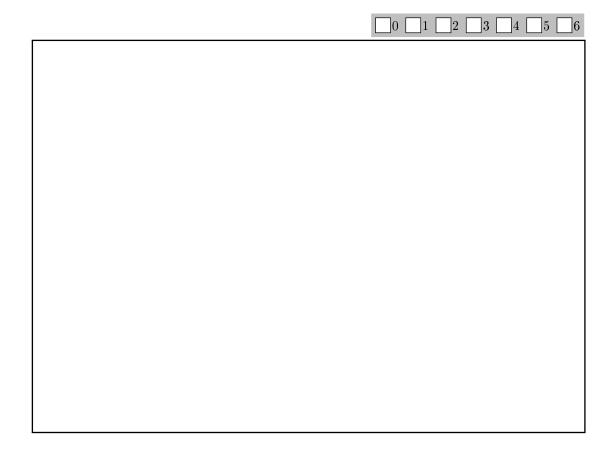
En Java ce dernier est calculé par theta = Math.atan2(y, x)

- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.





Question 13	Continuez votre réponse à la question précédente ici si nécessaire.	



Question 14	Continuez votre réponse à la question précédente ici si nécessaire.	



Une classe peut étendre plusieurs classes

Question 15 Pour la question précédente, développez la méthode Polar#equals.  $\boxed{0}$   $\boxed{1}$   $\boxed{2}$   $\boxed{3}$ Une classe abstraite peut implémenter une interface Une interface peut contenir des méthodes privées Une classe peut étendre une classe abstraite Une classe peut étendre une classe et implémenter plusieurs interfaces Une classe abstraite peut contenir des constructeurs Une interface peut contenir des constructeurs



## Question $17 \oplus$ Soit le code ci-dessous :

```
@FunctionalInterface
interface ComparePerson {
    /**
    * @return negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :

```
ComparePerson byAge = new ComparePersonByAge::compare;

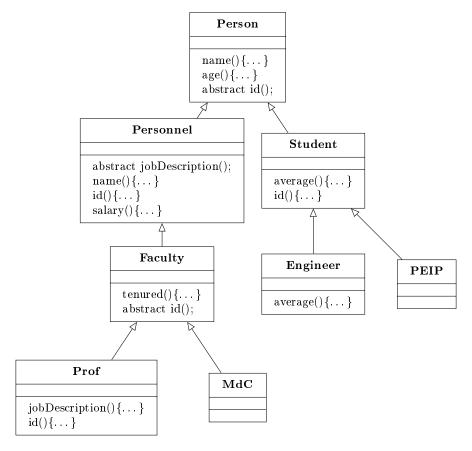
ComparePerson byAge = new ComparePersonByAge();

ComparePerson byAge = (p1, p2) -> p1.age - p2.age;
```

```
ComparePerson byAge = new ComparePersonByAge() {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
};
```



## Question 18 $\bigoplus$ Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

Student	Faculty
Personnel	Prof
MdC	PEIP
Engineer	Person

QCMTEST Nom et prénom: Introduction à la programmation orientée objet 9/01/2020 Groupe: ..... Cochez les cases en mettant une  $\times$ . Le symbole  $\bigoplus$  indique que la question peut avoir zéro, une ou plusieurs bonnes réponses. Pour ces questions, cocher une bonne réponse apporte des points positifs ; cocher une mauvaise réponse peut apporter des points négatifs. Dans tout le code, les package et les import sont censés être correctement déclarés. Toute classe est supposée être dans le bon package, dans le bon fichier, avec les bons import.Question 1 La classe donnée abstract class ClasseDonnee { ClasseDonnee() {} pourrait être pourrait être sous-classée par extends instanciée par new ClasseDonnee ClasseDonnee() Question 2 La classe donnée class ClasseDonnee {} pourrait être sous-classée par extends pourrait être instanciée par new ClasseDonnee() ClasseDonnee Question 3 \( \bigcap \) La classe donnée final class ClasseDonnee { ClasseDonnee() {} pourrait être sous-classée par extends pourrait être instanciée new ClasseDonnee ClasseDonnee()

Question 4 Affiche seulement "Caught exception" Soit le code ci-dessous ; que se passe-t-il ? public class Gamma { Erreur d'exécution public void throwup() throws Exception { Affiche "Caught exception" et "Finallied throw new Exception(); exception" public static void main(String[] args) { Gamma gamma = new Gamma(); Erreur de compilation try { gamma.throwup(); Affiche seulement "Finallied exception" } catch (Exception e) { System.out.println("Caught exception"); finally { System.out.println("Finallied exception"); } }



Erreur d'exécution Soit le code ci-dessous ; que se passe-t-il? Affiche "Done" public class Alpha { public void throwup() throws Exception { Erreur de compilation throw new Exception(); public static void main(String[] args) { Alpha alpha = new Alpha(); alpha.throwup(); System.out.println("Done"); Question 6 Soit le code ci-dessous ; que se passe-t-il? Erreur d'exécution public class Epsilon { Affiche seulement "Finallied exception" public void throwup() throws RuntimeException { Erreur de compilation throw new RuntimeException(); public static void main(String[] args) { Affiche seulement "Caught exception" Epsilon epsilon = new Epsilon(); try { Affiche "Caught exception" et "Finallied epsilon.throwup(); exception" } catch (Exception e) { System.out.println("Caught exception"); } finally { System.out.println("Finallied exception"); }

#### Question 7 Soit le code :

```
interface Somethingable {
    void something();
}

interface Whateverable extends Somethingable {
    void whatever();
}

class TheThing implements Whateverable{
    PPP void something() {
        System.out.println("Somethinging");
    }

    PPP void whatever() {
        System.out.println("Whatevering");
    }
}
```

```
class Doofer {
    void doo(Somethingable sable) {
        sable.XXX();
    }

    void doo(Whateverable sable) {
        sable.YYY();
    }

    void fer(TheThing tt) {
        tt.ZZZ();
    }

    public static void main(String[] args) {
        Doofer doofer = new Doofer();
        TheThing the = new TheThing();
        doofer.doo(the);
        doofer.fer(the);
    }
}
```

Par quels niveaux d'accès pourrait-on remplacer PPP?

$oxedsymbol{\Box}$ $ exttt{private}$	protected
package-priv	vate public
Question 8 $\bigoplus$ placer XXX ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
something	whatever

Question 9 $\bigoplus$ placer YYY ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem
whatever	something
Question 10 $\bigoplus$ remplacer ZZZ ?	Pour le code de la question précédente, par quelles méthodes pourrait-or
something	whatever

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

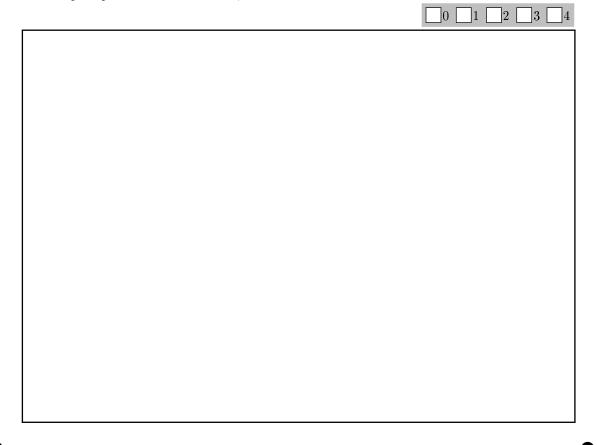
public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit



Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r\cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r\sin(\theta)$   $\theta = \tan^{-1}\frac{y}{x}$ 

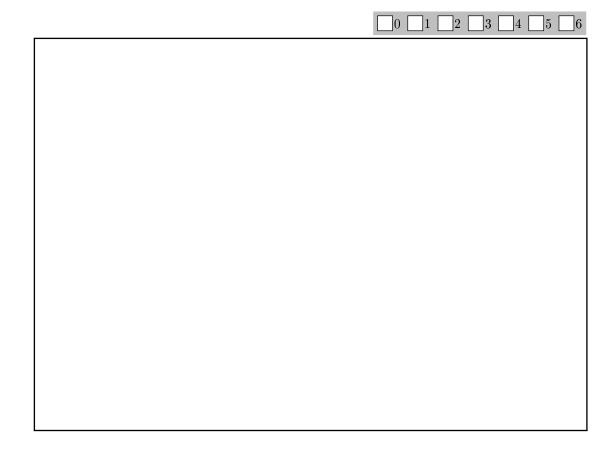
En Java ce dernier est calculé par theta = Math.atan2(y, x)

- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.





Question 13	Continuez	votre réponse à	la question p	récédente i	ci si nécessaire.	



uestion 14	Continuez votre réponse à la question précédente ici si nécessaire.	



Une classe abstraite peut contenir des constructeurs

Une classe peut étendre une classe et implémenter plusieurs interfaces

Question 15 Pour la question précédente, développez la méthode Polar#equals.  $\boxed{0}$   $\boxed{1}$   $\boxed{2}$   $\boxed{3}$ Parmi les affirmations ci-dessous, cocher celles qui sont vraies : Question 16  $\bigoplus$ Une interface peut contenir des méthodes privées Une classe abstraite peut implémenter une interface Une classe peut étendre une classe abstraite Une classe peut étendre plusieurs classes Une interface peut contenir des constructeurs



## Question 17 Soit le code ci-dessous :

```
@FunctionalInterface
interface ComparePerson {
    /**
    * @return negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

```
class Person {
    private int age;
    private String name;

Person(String name, int age) {
        this.age = age;
        this.name = name;
    }
}

class Main {
    public static void main(String[] args) {

        // code extract here

        System.out.print(byAge.compare(
            new Person("Fred", 23),
            new Person("Not Fred", 24)));
        System.out.print(byAge.compare(
            new Person("Not Fred", 23),
            new Person("Not Fred", 23));
        System.out.print(byAge.compare(
            new Person("Fred", 23),
            new Person("Fred", 23),
            new Person("Not Fred", 23),
            new Person("Not Fred", 23);
            new Person("Not Fred", 22)));
}
```

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :

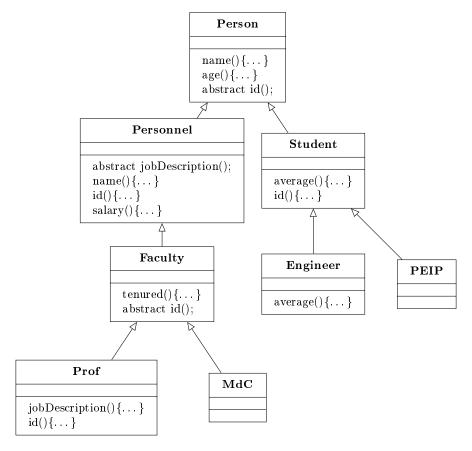
```
ComparePerson byAge = (p1, p2) -> p1.age - p2.age;

ComparePerson byAge = new ComparePersonByAge() {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
};

ComparePerson byAge = new ComparePersonByAge();
```



Question 18  $\bigoplus$  Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

<pre>Faculty</pre>	Engineer
Prof	MdC
Personnel	Student
PEIP	Person

QCMTEST Nom et prénom: Introduction à la programmation orientée objet 9/01/2020 Groupe: ..... Cochez les cases en mettant une  $\times$ . Le symbole  $\bigoplus$  indique que la question peut avoir zéro, une ou plusieurs bonnes réponses. Pour ces questions, cocher une bonne réponse apporte des points positifs ; cocher une mauvaise réponse peut apporter des points négatifs. Dans tout le code, les package et les import sont censés être correctement déclarés. Toute classe est supposée être dans le bon package, dans le bon fichier, avec les bons import.Question 1 La classe donnée class ClasseDonnee { private ClasseDonnee() {} pourrait être sous-classée par extends pourrait être instanciée par ClasseDonnee() ClasseDonnee La classe (enum) donnée Question 2 ( enum ClasseDonnee { INSTANCE; } pourrait être sous-classée par extends pourrait être instanciée par new ClasseDonnee ClasseDonnee() Question 3 ( La classe donnée abstract class ClasseDonnee { ClasseDonnee() {} pourrait être sous-classée par extends pourrait être instanciée par new ClasseDonnee() ClasseDonnee Question 4 Erreur de compilation Soit le code ci-dessous ; que se passe-t-il? public class Beta { Affiche "Done"

Erreur d'exécution

public void throwup() throws RuntimeException {

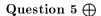
throw new RuntimeException();

System.out.println("Done");

Beta beta = new Beta();
beta.throwup();

}

public static void main(String[] args) {



```
Affiche seulement "Finallied exception"
Soit le code ci-dessous ; que se passe-t-il?
                                                            Affiche "Caught exception" et "Finallied
public class Delta {
   public void throwup() throws Exception {
                                                            exception"
      throw new Exception();
                                                          Erreur d'exécution
   public static void main(String[] args) {
      Delta delta = new Delta();
                                                            Affiche seulement "Caught exception"
      try {
          delta.throwup();
                                                            Erreur de compilation
      } catch (Exception e) {
          System.out.println("Caught exception");
          return:
      } finally {
          System.out.println("Finallied exception");
   }
}
Question 6
                                                          Affiche "Caught exception" et "Finallied
Soit le code ci-dessous ; que se passe-t-il?
                                                            exception"
public class Gamma {
   public void throwup() throws Exception {
                                                            Affiche seulement "Caught exception"
      throw new Exception();
                                                            Affiche seulement "Finallied exception"
   public static void main(String[] args) {
      Gamma gamma = new Gamma();
                                                            Erreur de compilation
      try {
```

#### **Question 7** Soit le code :

gamma.throwup();

} catch (Exception e) {

} finally {

}

System.out.println("Caught exception");

System.out.println("Finallied exception");

```
interface Somethingable {
    void something();
}

interface Whateverable extends Somethingable {
    void whatever();
}

class TheThing implements Whateverable{
    PPP void something() {
        System.out.println("Somethinging");
    }

    PPP void whatever() {
        System.out.println("Whatevering");
    }
}
```

```
class Doofer {
   void doo(Somethingable sable) {
      sable.XXX();
}

void doo(Whateverable sable) {
    sable.YYY();
}

void fer(TheThing tt) {
    tt.ZZZ();
}

public static void main(String[] args) {
    Doofer doofer = new Doofer();
    TheThing the = new TheThing();
    doofer.doo(the);
    doofer.fer(the);
}
```

Erreur d'exécution

Par quels niveaux d'accès pourrait-on remplacer PPP?

package-private	$oxedsymbol{\Box}$ $\operatorname{ t private}$
protected	public

Question 8 $\bigoplus$ placer XXX ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
something	whatever
Question 9 $\bigoplus$ placer YYY ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
something	whatever
Question 10 ⊕ remplacer ZZZ ?	Pour le code de la question précédente, par quelles méthodes pourrait-on
whatever	something

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit





Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r \cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r \sin(\theta)$   $\theta = \tan^{-1} \frac{y}{x}$ 

En Java ce dernier est calculé par theta = Math.atan2(y, x)

- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.





Question 13	Continuez votre réponse à la question précédente ici si nécessaire.	



Question 14	Continuez votre réponse à la question précédente ici si nécessaire.	
1		



Question 15 Pour la question précédente, développez la méthode Polar#equals.  $\boxed{0}$   $\boxed{1}$   $\boxed{2}$   $\boxed{3}$ Une classe peut étendre une classe abstraite Une classe abstraite peut contenir des constructeurs Une interface peut contenir des constructeurs Une classe abstraite peut implémenter une interface Une classe peut étendre plusieurs classes

Une classe peut étendre une classe et implémenter plusieurs interfaces

Une interface peut contenir des méthodes privées



# Question $17 \oplus$ Soit le code ci-dessous :

```
@FunctionalInterface
interface ComparePerson {
    /**
    * @return negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :



Question 18  $\bigoplus$  Soit l'organisation ci-dessous de Polytech'Groland :

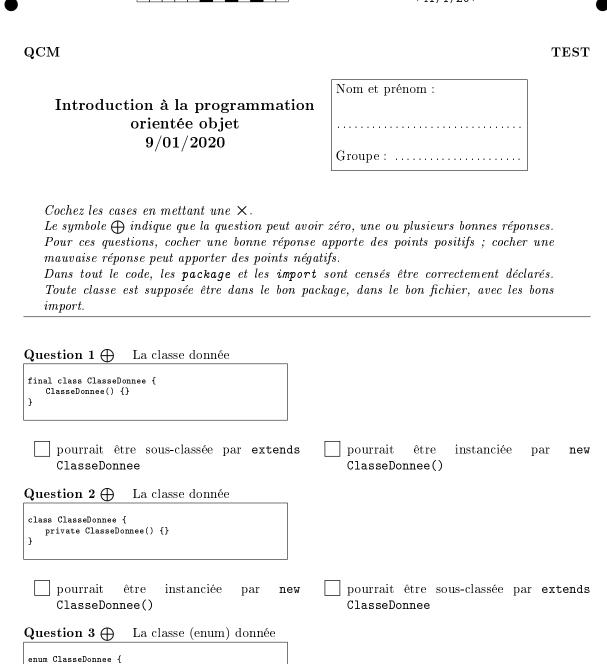


Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

Person	Student
Personnel	PEIP
Prof	Faculty
Engineer	MdC

pourrait être sous-classée par extends

ClasseDonnee



INSTANCE;

pourrait

ClasseDonnee()

être

instanciée

par

new

Soit le code ci-dessous ; que se passe-t-il? Erreur de compilation Affiche seulement "Finallied exception" public class Delta { public void throwup() throws Exception { Erreur d'exécution throw new Exception(); Affiche seulement "Caught exception" public static void main(String[] args) { Delta delta = new Delta(); try { Affiche "Caught exception" et "Finallied delta.throwup(); exception" } catch (Exception e) { System.out.println("Caught exception"); return; } finally { System.out.println("Finallied exception"); } Question 5

Soit le code ci-dessous ; que se passe-t-il ?
public class Beta {
 public void throwup() throws RuntimeException {
 throw new RuntimeException();
 }
 public static void main(String[] args) {
 Beta beta = new Beta();
 beta.throwup();
 System.out.println("Done");
 }
}

# Question 6 $\bigoplus$

Soit le code ci-dessous ; que se passe-t-il ?

public class Gamma {
 public void throwup() throws Exception {
 throw new Exception();
 }
 public static void main(String[] args) {
 Gamma gamma = new Gamma();
 try {
 gamma.throwup();
 } catch (Exception e) {
 System.out.println("Caught exception");
 } finally {
 System.out.println("Finallied exception");
 }
}

Erreur	de	com	pilation

Erreur d'exécution

 $\square$  Affiche "Done"

Affiche "Caught exception" et "Finallied exception"

Erreur de compilation

Erreur d'exécution

Affiche seulement "Caught exception"

Affiche seulement "Finallied exception"



#### **Question 7** Soit le code :

```
interface Somethingable {
   void something();
}

interface Whateverable extends Somethingable {
   void whatever();
}

class TheThing implements Whateverable{
   PPP void something() {
       System.out.println("Somethinging");
   }

   PPP void whatever() {
       System.out.println("Whatevering");
   }
}
```

```
class Doofer {
    void doo(Somethingable sable) {
        sable.XXX();
    }

    void doo(Whateverable sable) {
        sable.YYY();
    }

    void fer(TheThing tt) {
        tt.ZZZ();
    }

    public static void main(String[] args) {
        Doofer doofer = new Doofer();
        TheThing the = new TheThing();
        doofer.doo(the);
        doofer.fer(the);
    }
}
```

Par quels niveaux d'accès pourrait-on remplacer PPP? protected package-private public private Question 8 Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer XXX? whatever something Question 9 Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer YYY? something whatever Question 10  $\bigoplus$ Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer ZZZ? whatever something

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit





Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r \cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r \sin(\theta)$   $\theta = \tan^{-1} \frac{y}{x}$ 

En Java ce dernier est calculé par theta = Math.atan2(y, x)

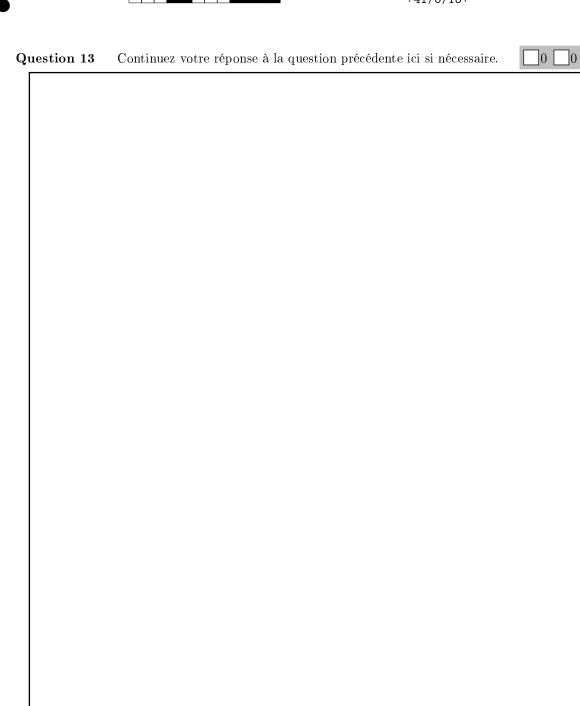
- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

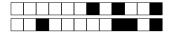
Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.







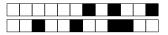
Question 14	Continuez votre réponse à la question précédente ici si nécessaire.	0 0



Une classe peut étendre plusieurs classes

Une interface peut contenir des méthodes privées

Question 15 Pour la question précédente, développez la méthode Polar#equals.  $\boxed{0}$   $\boxed{1}$   $\boxed{2}$   $\boxed{3}$ Question 16  $\bigoplus$ Parmi les affirmations ci-dessous, cocher celles qui sont vraies : Une classe peut étendre une classe abstraite Une classe peut étendre une classe et implémenter plusieurs interfaces Une interface peut contenir des constructeurs Une classe abstraite peut implémenter une interface Une classe abstraite peut contenir des constructeurs



# Question $17 \oplus$ Soit le code ci-dessous :

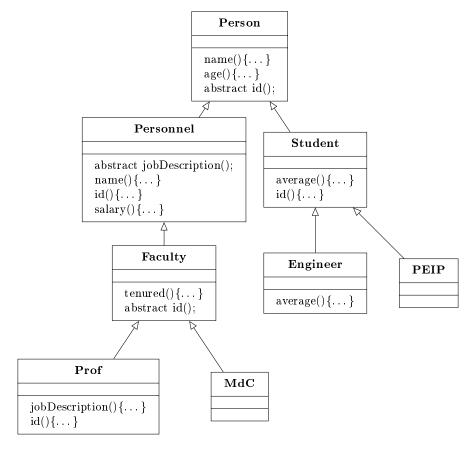
```
@FunctionalInterface
interface ComparePerson {
    /**
    * Oreturn negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :

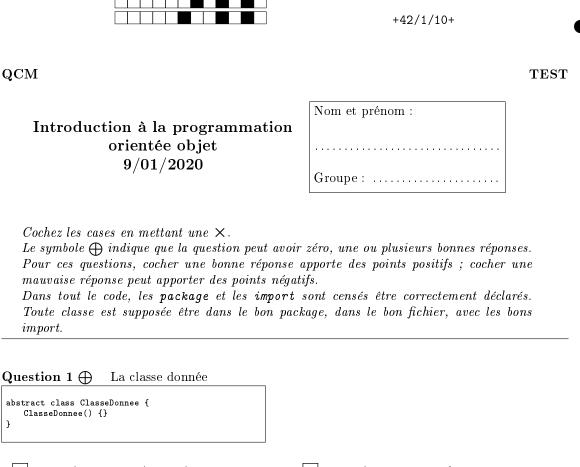


# Question 18 $\bigoplus$ Soit l'organisation ci-dessous de Polytech'Groland :

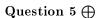


Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

MdC	Faculty
PEIP	Student
Engineer	Person
Personnel	Prof



new pourrait être sous-classée par extends pourrait être instanciée par ClasseDonnee() ClasseDonnee Question 2 La classe donnée class ClasseDonnee {} pourrait être sous-classée par extends pourrait être instanciée par new ClasseDonnee ClasseDonnee() Question 3 La classe (enum) donnée enum ClasseDonnee { INSTANCE; } pourrait être sous-classée par extends pourrait être instanciée new ClasseDonnee ClasseDonnee() Question 4 Erreur d'exécution Soit le code ci-dessous ; que se passe-t-il? Affiche "Done" public class Alpha { public void throwup() throws Exception { throw new Exception(); Erreur de compilation public static void main(String[] args) { Alpha alpha = new Alpha(); alpha.throwup(); System.out.println("Done");



```
Soit le code ci-dessous ; que se passe-t-il?
public class Epsilon {
   public void throwup() throws RuntimeException {
       throw new RuntimeException();
   public static void main(String[] args) {
       Epsilon epsilon = new Epsilon();
       try {
          epsilon.throwup();
                                                                 exception"
       } catch (Exception e) {
          System.out.println("Caught exception");
       } finally {
          System.out.println("Finallied exception");
   }
}
```

Erreur d'exécution	
Affiche seulement "Fin	allied exception"
Erreur de compilation	

Affiche seulement "Caught exception"

Affiche "Caught exception" et "Finallied

# Question 6 $\bigoplus$

```
Soit le code ci-dessous ; que se passe-t-il?
public class Delta {
   public void throwup() throws Exception {
       throw new Exception();
   public static void main(String[] args) {
       Delta delta = new Delta();
       try {
          delta.throwup();
       } catch (Exception e) {
           System.out.println("Caught exception");
           return;
       } finally {
   System.out.println("Finallied exception");
```

# Affiche seulement "Caught exception" Affiche seulement "Finallied exception" Erreur de compilation Erreur d'exécution Affiche "Caught exception" et "Finallied

exception"

#### Question 7 Soit le code:

```
interface Somethingable {
   void something();
}
interface Whateverable extends Somethingable {
   void whatever();
class TheThing implements Whateverable{
   PPP void something() {
       System.out.println("Somethinging");
   PPP void whatever() {
       System.out.println("Whatevering");
}
```

```
class Doofer {
   void doo(Somethingable sable) {
        sable.XXX();
    void doo(Whateverable sable) {
        sable.YYY();
    void fer(TheThing tt) {
       tt.ZZZ();
   public static void main(String[] args) {
       Doofer doofer = new Doofer();
TheThing the = new TheThing();
        doofer.doo(the);
        doofer.fer(the);
}
```

Par quels niveaux d'accès pourrait-on remplacer PPP?

```
protected
                                              package-private
public
                                              private
```



Question 8 $\bigoplus$ placer XXX ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
something	whatever
Question 9 $\bigoplus$ placer YYY ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
whatever	<pre>something</pre>
Question 10 $\bigoplus$ remplacer ZZZ ?	Pour le code de la question précédente, par quelles méthodes pourrait-on
whatever	something

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
    public double re;
    public Complex(double a, double b) {
        re = a;
        im = b;
    }

    public double getRe() {
        return re;
    }

    public void setRe(double re) {
        this.re = re;
    }

    public double getIm() {
        return im;
    }

    public void setIm(double im) {
        this.im = im;
    }
}
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

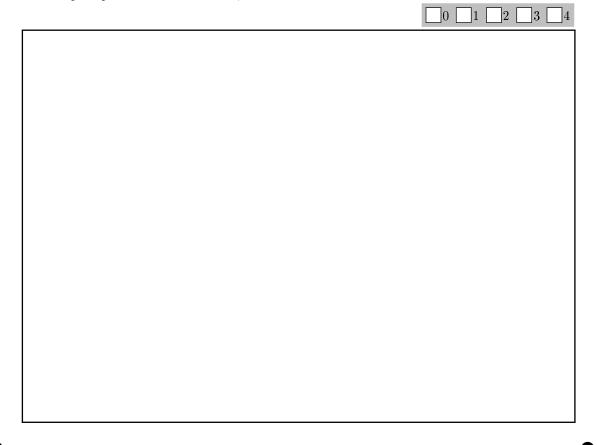
public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit





Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r \cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r \sin(\theta)$   $\theta = \tan^{-1} \frac{y}{x}$ 

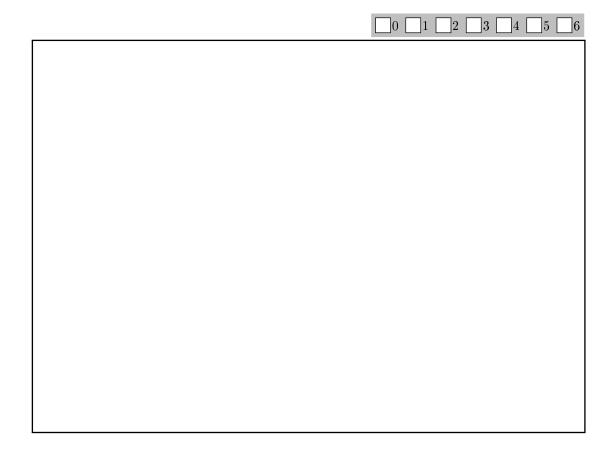
En Java ce dernier est calculé par theta = Math.atan2(y, x)

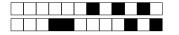
- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.

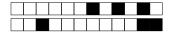




Question 13	Continuez	votre réponse à la question précédente ici si nécessaire.	



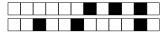
Question 14	Continuez	votre réponse à l	la question pré	cédente ici s	i nécessaire.	0 0



Une classe abstraite peut contenir des constructeurs

Une classe peut étendre une classe abstraite

Question	15	Pour	la.	question	précédente,	développez	la.	méthode	Po <sup>-</sup>	lar#	egna	ls.	
440001011		1041	100	question	presedence,	developpez			) [	]1 [			4
Question	<b>16</b> $\bigoplus$	Parn	ni le	es affirma	tions ci-desso	ous, cocher c	elle	s qui sont	vra	ies :			
_					méthodes pr								
					$\operatorname{sse}$ et implén		eurs	interface	5				
				eut imple: re plusieu	menter une i rs classes	nterface							
_					constructeur	`S							

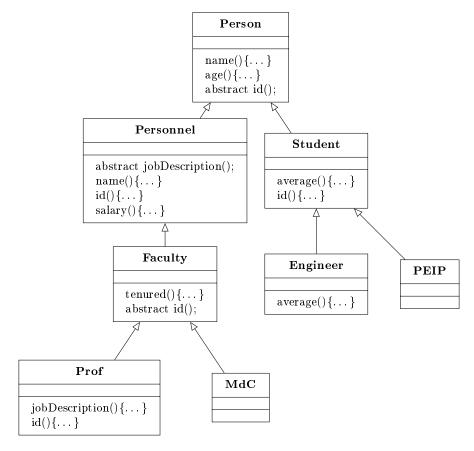


# Question $17 \oplus$ Soit le code ci-dessous :

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :



# Question 18 $\bigoplus$ Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

MdC	Prof
Person	Student
Personnel	Faculty
Engineer	PEIP

QCMTEST Nom et prénom: Introduction à la programmation orientée objet 9/01/2020 Groupe: ..... Cochez les cases en mettant une  $\times$ . Le symbole  $\bigoplus$  indique que la question peut avoir zéro, une ou plusieurs bonnes réponses. Pour ces questions, cocher une bonne réponse apporte des points positifs ; cocher une mauvaise réponse peut apporter des points négatifs. Dans tout le code, les package et les import sont censés être correctement déclarés. Toute classe est supposée être dans le bon package, dans le bon fichier, avec les bons import.Question 1 

La classe donnée final class ClasseDonnee { ClasseDonnee() {} pourrait être sous-classée par extends pourrait être instanciée par new ClasseDonnee ClasseDonnee() Question 2 La classe donnée class ClasseDonnee { private ClasseDonnee() {} } être pourrait être sous-classée par extends pourrait instanciée par new ClasseDonnee() ClasseDonnee Question 3 La classe donnée abstract class ClasseDonnee { ClasseDonnee() {} pourrait être sous-classée par extends pourrait être instanciée par new ClasseDonnee ClasseDonnee() Question 4 Affiche "Done" Soit le code ci-dessous ; que se passe-t-il? public class Beta { Erreur de compilation public void throwup() throws RuntimeException { throw new RuntimeException(); Erreur d'exécution public static void main(String[] args) {

Beta beta = new Beta();
beta.throwup();

}

System.out.println("Done");



<pre>Soit le code ci-dessous ; que se passe-t-il ?  public class Gamma {     public void throwup() throws Exception {         throw new Exception();     }      public static void main(String[] args) {         Gamma gamma = new Gamma();         try {             gamma.throwup();     } catch (Exception e) {             System.out.println("Caught exception");     } finally {         System.out.println("Finallied exception");     } }</pre>	Affiche "Caught exception" et "Finallied exception"  Affiche seulement "Caught exception"  Erreur de compilation  Affiche seulement "Finallied exception"  Erreur d'exécution
Question 6 $\bigoplus$	
<pre>Soit le code ci-dessous ; que se passe-t-il ?  public class Alpha {     public void throwup() throws Exception {         throw new Exception();     }     public static void main(String[] args) {         Alpha alpha = new Alpha();         alpha.throwup();         System.out.println("Done");     } }</pre>	<ul><li>□ Erreur de compilation</li><li>□ Erreur d'exécution</li><li>□ Affiche "Done"</li></ul>

### **Question 7** Soit le code :

```
interface Somethingable {
    void something();
}

interface Whateverable extends Somethingable {
    void whatever();
}

class TheThing implements Whateverable{
    PPP void something() {
        System.out.println("Somethinging");
    }

    PPP void whatever() {
        System.out.println("Whatevering");
    }
}
```

```
class Doofer {
    void doo(Somethingable sable) {
        sable.XXX();
    }

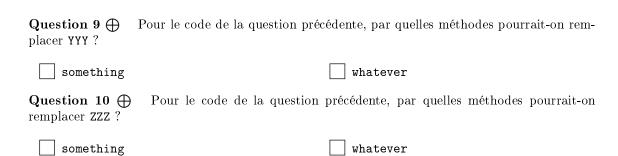
    void doo(Whateverable sable) {
        sable.YYY();
    }

    void fer(TheThing tt) {
        tt.ZZZ();
    }

    public static void main(String[] args) {
        Doofer doofer = new Doofer();
        TheThing the = new TheThing();
        doofer.doo(the);
        doofer.fer(the);
    }
}
```

Par quels niveaux d'accès pourrait-on remplacer PPP ?

package-priv	vate public
protected	private
Question 8 $\bigoplus$ placer XXX ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
something	whatever



Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
    public double re;
    public Complex(double a, double b) {
        re = a;
        im = b;
    }
    public double getRe() {
        return re;
    }
    public void setRe(double re) {
        this.re = re;
    }
    public double getIm() {
        return im;
    }
    public void setIm(double im) {
        this.im = im;
    }
}
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit



Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r \cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r \sin(\theta)$   $\theta = \tan^{-1} \frac{y}{x}$ 

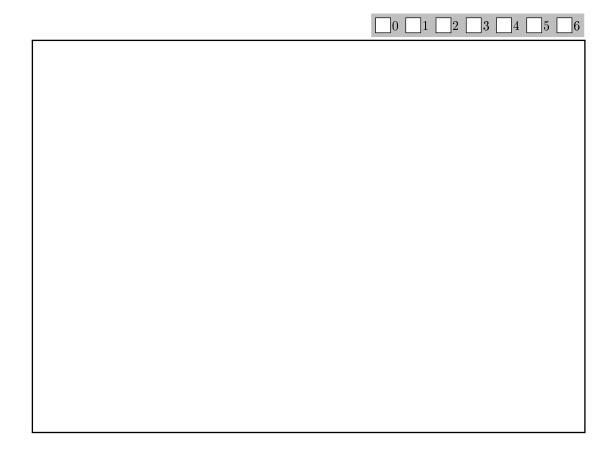
En Java ce dernier est calculé par theta = Math.atan2(y, x)

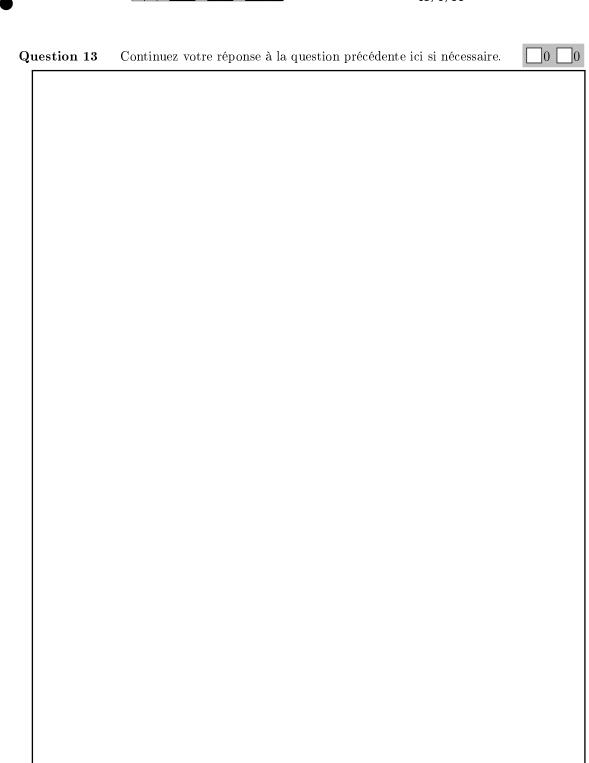
- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.







Question 14	Continuez	votre réponse	à la question	précédente	ici si nécessaire.	



Question 15 Pour la question précédente, développez la méthode Polar#equals.  $\boxed{0}$   $\boxed{1}$   $\boxed{2}$   $\boxed{3}$ Question 16  $\bigoplus$ Parmi les affirmations ci-dessous, cocher celles qui sont vraies : Une classe abstraite peut contenir des constructeurs Une interface peut contenir des méthodes privées Une classe peut étendre plusieurs classes Une classe abstraite peut implémenter une interface Une classe peut étendre une classe abstraite Une interface peut contenir des constructeurs

Une classe peut étendre une classe et implémenter plusieurs interfaces



# Question $17 \oplus$ Soit le code ci-dessous :

```
@FunctionalInterface
interface ComparePerson {
    /**
    * @return negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :

```
ComparePerson byAge = new ComparePersonByAge();

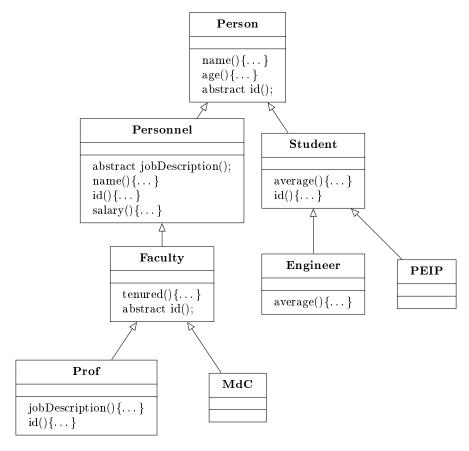
ComparePerson byAge = new ComparePersonByAge::compare;

ComparePerson byAge = (p1, p2) -> p1.age - p2.age;
```

```
ComparePerson byAge = new ComparePersonByAge() {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
};
```



Question 18  $\bigoplus$  Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

Faculty	Engineer
Person	Personnel
Prof	MdC
Student	PEIP

QCMTEST Nom et prénom: Introduction à la programmation orientée objet 9/01/2020 Groupe: ..... Cochez les cases en mettant une  $\times$ . Le symbole  $\bigoplus$  indique que la question peut avoir zéro, une ou plusieurs bonnes réponses. Pour ces questions, cocher une bonne réponse apporte des points positifs ; cocher une mauvaise réponse peut apporter des points négatifs. Dans tout le code, les package et les import sont censés être correctement déclarés. Toute classe est supposée être dans le bon package, dans le bon fichier, avec les bons import.Question 1 La classe donnée class ClasseDonnee {} pourrait être sous-classée par extends pourrait être instanciée new ClasseDonnee ClasseDonnee() La classe donnée Question 2 ( class ClasseDonnee { private ClasseDonnee() {} pourrait être sous-classée par extends pourrait être instanciée ClasseDonnee ClasseDonnee() Question 3  $\bigoplus$  La classe donnée abstract class ClasseDonnee { ClasseDonnee() {} pourrait être instanciée pourrait être sous-classée par extends par new ClasseDonnee() ClasseDonnee Question 4 Soit le code ci-dessous ; que se passe-t-il ? Affiche "Caught exception" et "Finallied

exception" public class Epsilon {  $\verb"public void throwup"() throws RuntimeException \{$ Erreur d'exécution throw new RuntimeException(); Affiche seulement "Finallied exception" public static void main(String[] args) { Epsilon epsilon = new Epsilon(); Affiche seulement "Caught exception" try { epsilon.throwup(); Erreur de compilation } catch (Exception e) { System.out.println("Caught exception"); finally { System.out.println("Finallied exception"); } }



# Question 5 $\bigoplus$

Erreur d'exécution Soit le code ci-dessous ; que se passe-t-il? public class Alpha { Erreur de compilation public void throwup() throws Exception { Affiche "Done" throw new Exception(); public static void main(String[] args) { Alpha alpha = new Alpha(); alpha.throwup(); System.out.println("Done"); Question 6 Soit le code ci-dessous ; que se passe-t-il? Affiche seulement "Finallied exception" public class Gamma { Erreur d'exécution public void throwup() throws Exception { Affiche seulement "Caught exception" throw new Exception(); Affiche "Caught exception" et "Finallied public static void main(String[] args) { Gamma gamma = new Gamma(); exception" try { gamma.throwup(); Erreur de compilation } catch (Exception e) { System.out.println("Caught exception"); } finally { System.out.println("Finallied exception");

#### Question 7 Soit le code :

}

```
interface Somethingable {
    void something();
}

interface Whateverable extends Somethingable {
    void whatever();
}

class TheThing implements Whateverable{
    PPP void something() {
        System.out.println("Somethinging");
    }

    PPP void whatever() {
        System.out.println("Whatevering");
    }
}
```

```
class Doofer {
    void doo(Somethingable sable) {
        sable.XXX();
    }

    void doo(Whateverable sable) {
        sable.YYY();
    }

    void fer(TheThing tt) {
        tt.ZZZ();
    }

    public static void main(String[] args) {
        Doofer doofer = new Doofer();
        TheThing the = new TheThing();
        doofer.doo(the);
        doofer.fer(the);
    }
}
```

Par quels niveaux d'accès pourrait-on remplacer PPP?

$\square$ private		public
package-pri	vate	protected
Question 8 $\bigoplus$ placer XXX ?	Pour le code de la question précé	édente, par quelles méthodes pourrait-on rem
something		whatever

+44/3/48+
-----------

Question 9 $\bigoplus$ placer YYY ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
whatever	something
Question 10 $\bigoplus$ remplacer ZZZ ?	Pour le code de la question précédente, par quelles méthodes pourrait-on
whatever	something

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit



Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r\cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r\sin(\theta)$   $\theta = \tan^{-1}\frac{y}{x}$ 

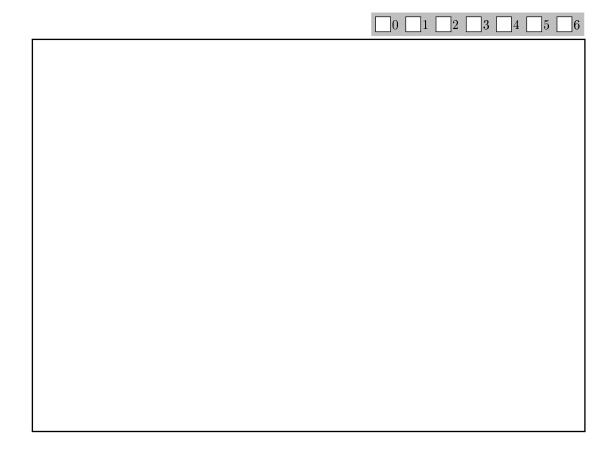
En Java ce dernier est calculé par theta = Math.atan2(y, x)

- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.





Question 13	Continuez votre réponse à la question précédente ici s	i nécessaire.	



Question 14	Continuez	votre réponse	à la question	précédente	ici si nécessaire.	



Question 15 Pour la question précédente, développez la méthode Polar#equals.  $\boxed{0}$   $\boxed{1}$   $\boxed{2}$   $\boxed{3}$ Question 16  $\bigoplus$ Parmi les affirmations ci-dessous, cocher celles qui sont vraies : Une interface peut contenir des méthodes privées Une classe abstraite peut contenir des constructeurs Une classe peut étendre plusieurs classes Une classe peut étendre une classe et implémenter plusieurs interfaces Une interface peut contenir des constructeurs Une classe peut étendre une classe abstraite Une classe abstraite peut implémenter une interface



### Question $17 \oplus$ Soit le code ci-dessous :

```
@FunctionalInterface
interface ComparePerson {
    /**
    * @return negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :

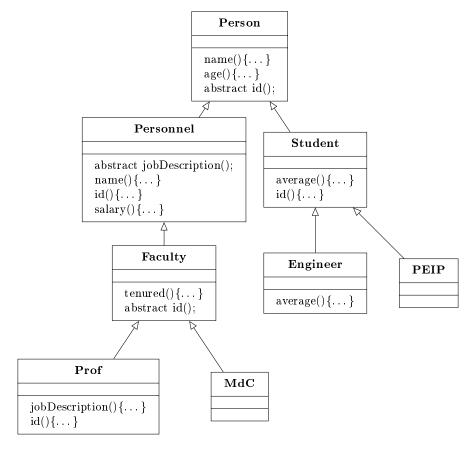
```
ComparePerson byAge = new ComparePersonByAge::compare;

ComparePerson byAge = new ComparePersonByAge();

ComparePerson byAge = new ComparePersonByAge() {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
};
```



Question 18  $\bigoplus$  Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

Prof	MdC
Engineer	$\square$ Faculty
PEIP	Personnel
Person	Student

QCMTEST Nom et prénom: Introduction à la programmation orientée objet 9/01/2020 Groupe: ..... Cochez les cases en mettant une  $\times$ . Le symbole  $\bigoplus$  indique que la question peut avoir zéro, une ou plusieurs bonnes réponses. Pour ces questions, cocher une bonne réponse apporte des points positifs ; cocher une mauvaise réponse peut apporter des points négatifs. Dans tout le code, les package et les import sont censés être correctement déclarés. Toute classe est supposée être dans le bon package, dans le bon fichier, avec les bons import.Question 1 

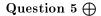
La classe donnée final class ClasseDonnee { ClasseDonnee() {} pourrait être sous-classée par extends pourrait être instanciée par new ClasseDonnee() ClasseDonnee Question 2 La classe (enum) donnée enum ClasseDonnee { INSTANCE; } pourrait pourrait être sous-classée par extends être instanciée par new ClasseDonnee ClasseDonnee() Question 3 ( La classe donnée class ClasseDonnee {} pourrait être sous-classée par extends pourrait être instanciée par new ClasseDonnee() ClasseDonnee Question 4 Affiche "Done" Soit le code ci-dessous ; que se passe-t-il? public class Beta { Erreur de compilation public void throwup() throws RuntimeException {

throw new RuntimeException();
}
public static void main(String[] args) {
Beta beta = new Beta();
beta.throwup();

System.out.println("Done");

}

Erreur d'exécution



```
Soit le code ci-dessous ; que se passe-t-il?
                                                            Erreur d'exécution
public class Delta {
                                                            Affiche seulement "Finallied exception"
   public void throwup() throws Exception {
                                                            Affiche seulement "Caught exception"
      throw new Exception();
   public static void main(String[] args) {
                                                            Erreur de compilation
      Delta delta = new Delta();
      try {
                                                            Affiche "Caught exception" et "Finallied
          delta.throwup();
                                                            exception"
      } catch (Exception e) {
          System.out.println("Caught exception");
          return:
      } finally {
          System.out.println("Finallied exception");
   }
}
Question 6
                                                            Erreur de compilation
Soit le code ci-dessous ; que se passe-t-il?
```

public void throwup() throws RuntimeException {
 throw new RuntimeException();
}
public static void main(String[] args) {
 Epsilon epsilon = new Epsilon();
 try {
 epsilon.throwup();
} catch (Exception e) {
 System.out.println("Caught exception");
} finally {

System.out.println("Finallied exception");
}
}

**Question 7** Soit le code :

public class Epsilon {

```
interface Somethingable {
    void something();
}

interface Whateverable extends Somethingable {
    void whatever();
}

class TheThing implements Whateverable{
    PPP void something() {
        System.out.println("Somethinging");
    }

    PPP void whatever() {
        System.out.println("Whatevering");
    }
}
```

```
class Doofer {
   void doo(Somethingable sable) {
      sable.XXX();
   }

   void doo(Whateverable sable) {
      sable.YYY();
   }

   void fer(TheThing tt) {
      tt.ZZZ();
   }

   public static void main(String[] args) {
      Doofer doofer = new Doofer();
      TheThing the = new TheThing();
      doofer.doo(the);
      doofer.fer(the);
   }
}
```

Affiche seulement "Finallied exception"

Affiche seulement "Caught exception"

Affiche "Caught exception" et "Finallied

Erreur d'exécution

exception"

Par quels niveaux d'accès pourrait-on remplacer PPP?

 ☐ package-private
 ☐ private

 ☐ public
 ☐ protected

Question 8 $\bigoplus$ placer XXX ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
something	whatever
Question 9 $\bigoplus$ placer YYY ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
something	whatever
Question 10 $\bigoplus$ remplacer ZZZ ?	Pour le code de la question précédente, par quelles méthodes pourrait-on
something	whatever

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit





Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r\cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r\sin(\theta)$   $\theta = \tan^{-1}\frac{y}{x}$ 

En Java ce dernier est calculé par theta = Math.atan2(y, x)

- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.





Question 13	Continuez votre réponse à la question précédente ici s	i nécessaire.	



Question 14	Continuez	votre réponse	à la question	précédente	ici si nécessaire.	



Question 15 Pour la question précédente, développez la méthode Polar#equals.  $\boxed{0}$   $\boxed{1}$   $\boxed{2}$   $\boxed{3}$ Une classe peut étendre plusieurs classes Une classe peut étendre une classe abstraite Une interface peut contenir des méthodes privées Une interface peut contenir des constructeurs

Une classe peut étendre une classe et implémenter plusieurs interfaces

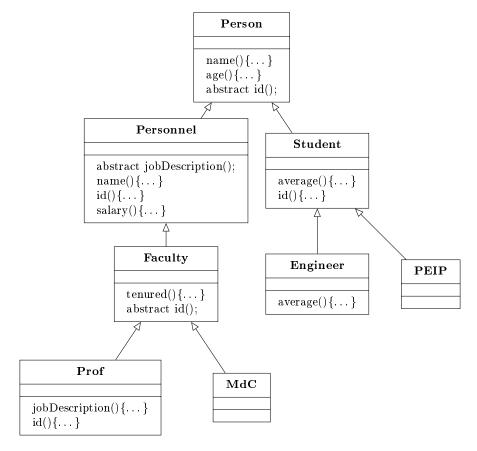
Une classe abstraite peut implémenter une interface
Une classe abstraite peut contenir des constructeurs



### Question $17 \oplus$ Soit le code ci-dessous :

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :





Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

Prof	Personnel
Person	MdC
Student	Faculty
Engineer	PEIP

QCMTEST Nom et prénom: Introduction à la programmation orientée objet 9/01/2020Groupe: ..... Cochez les cases en mettant une  $\times$ . Le symbole  $\bigoplus$  indique que la question peut avoir zéro, une ou plusieurs bonnes réponses. Pour ces questions, cocher une bonne réponse apporte des points positifs ; cocher une mauvaise réponse peut apporter des points négatifs. Dans tout le code, les package et les import sont censés être correctement déclarés. Toute classe est supposée être dans le bon package, dans le bon fichier, avec les bons import.Question 1 La classe (enum) donnée enum ClasseDonnee { INSTANCE; pourrait être sous-classée par extends pourrait être instanciée par new ClasseDonnee() ClasseDonnee Question 2 La classe donnée class ClasseDonnee { private ClasseDonnee() {} pourrait être pourrait être sous-classée par extends instanciée par new ClasseDonnee() ClasseDonnee Question 3 ( La classe donnée final class ClasseDonnee { ClasseDonnee() {} pourrait être sous-classée par extends pourrait être instanciée par new ClasseDonnee() ClasseDonnee

```
Affiche "Caught exception" et "Finallied
Soit le code ci-dessous ; que se passe-t-il?
                                                            exception"
public class Delta {
   public void throwup() throws Exception {
                                                            Erreur d'exécution
      throw new Exception();
                                                            Erreur de compilation
   public static void main(String[] args) {
      Delta delta = new Delta();
                                                            Affiche seulement "Caught exception"
      try {
          delta.throwup();
                                                            Affiche seulement "Finallied exception"
      } catch (Exception e) {
          System.out.println("Caught exception");
          return;
      } finally {
          System.out.println("Finallied exception");
}
Question 5
```

### Question 6 $\bigoplus$

Soit le code ci-dessous ; que se passe-t-il ?

public class Gamma {
 public void throwup() throws Exception {
 throw new Exception();
 }
 public static void main(String[] args) {
 Gamma gamma = new Gamma();
 try {
 gamma.throwup();
 } catch (Exception e) {
 System.out.println("Caught exception");
 } finally {
 System.out.println("Finallied exception");
 }
}

Affiche	"Caught	exception"	$\operatorname{et}$	"Finallied
exception	on"			

Affiche seulement "Finallied exception"

Affiche seulement "Caught exception"

Affiche seulement "Caught exception

Erreur de compilation

Erreur d'exécution



### **Question 7** Soit le code :

```
interface Somethingable {
   void something();
}

interface Whateverable extends Somethingable {
   void whatever();
}

class TheThing implements Whateverable{
   PPP void something() {
       System.out.println("Somethinging");
   }

   PPP void whatever() {
       System.out.println("Whatevering");
   }
}
```

```
class Doofer {
    void doo(Somethingable sable) {
        sable.XXX();
    }

    void doo(Whateverable sable) {
        sable.YYY();
    }

    void fer(TheThing tt) {
        tt.ZZZ();
    }

    public static void main(String[] args) {
        Doofer doofer = new Doofer();
        TheThing the = new TheThing();
        doofer.doo(the);
        doofer.fer(the);
    }
}
```

Par quels niveaux d'accès pourrait-on remplacer PPP ?

protected package-priv	vate [	private public		
Question 8 $\bigoplus$ placer XXX ?	Pour le code de la question précé	edente, par quell	les méthodes pourrai	t-on rem-
something	I	whatever		
Question 9 $\bigoplus$ placer YYY ?	Pour le code de la question précé	edente, par quell	les méthodes pourrai	t-on rem-
something	I	whatever		
Question 10 $\bigoplus$ remplacer ZZZ ?	Pour le code de la question p	orécédente, par	quelles méthodes po	ourrait-on
whatever		something		

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit





Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r\cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r\sin(\theta)$   $\theta = \tan^{-1}\frac{y}{x}$ 

En Java ce dernier est calculé par theta = Math.atan2(y, x)

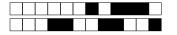
- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.





Question 13	Continuez votre réponse à la question précédente ici si nécessaire.	



Question 14	Continuez	votre réponse à l	la question pré	cédente ici s	si nécessaire.	0 0



Une classe abstraite peut implémenter une interface

Question 15 Pour la question précédente, développez la méthode Polar#equals.  $\boxed{0}$   $\boxed{1}$   $\boxed{2}$   $\boxed{3}$ Question 16  $\bigoplus$ Parmi les affirmations ci-dessous, cocher celles qui sont vraies : Une classe abstraite peut contenir des constructeurs Une interface peut contenir des méthodes privées Une classe peut étendre plusieurs classes Une interface peut contenir des constructeurs Une classe peut étendre une classe et implémenter plusieurs interfaces Une classe peut étendre une classe abstraite



### Question $17 \oplus$ Soit le code ci-dessous :

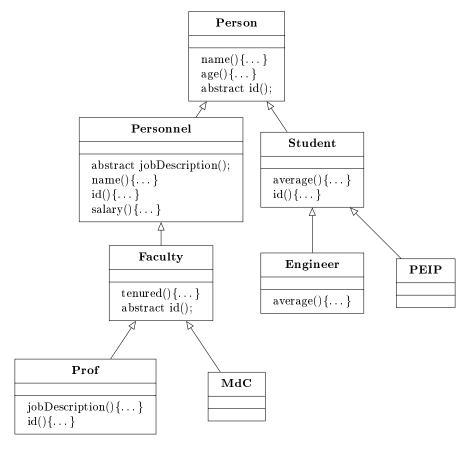
```
@FunctionalInterface
interface ComparePerson {
    /**
    * @return negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :



# Question 18 $\bigoplus$ Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

Engineer	Faculty
Person	MdC
Student	Prof
PEIP	Personnel

 $Groupe: \ \dots \dots \dots \dots$ 

Cochez les cases en mettant une  $\times$ .

Le symbole  $\bigoplus$  indique que la question peut avoir zéro, une ou plusieurs bonnes réponses. Pour ces questions, cocher une bonne réponse apporte des points positifs ; cocher une mauvaise réponse peut apporter des points négatifs.

Dans tout le code, les package et les import sont censés être correctement déclarés. Toute classe est supposée être dans le bon package, dans le bon fichier, avec les bons import.

import.	Package, aans le von picmer, avec les vons
Question 1   La classe donnée	
class ClasseDonnee {}	
pourrait être instanciée par new ClasseDonnee()	pourrait être sous-classée par extends ClasseDonnee
Question 2   La classe donnée	
abstract class ClasseDonnee {     ClasseDonnee() {} }	
pourrait être instanciée par <b>new</b> ClasseDonnee()	pourrait être sous-classée par extends ClasseDonnee
Question 3   La classe donnée	
<pre>class ClasseDonnee {    private ClasseDonnee() {} }</pre>	
pourrait être sous-classée par extends ClasseDonnee	pourrait être instanciée par new ClasseDonnee()
Question $4 \bigoplus$	
Soit le code ci-dessous ; que se passe-t-il ?	Affiche "Caught exception" et "Finallied
<pre>public class Epsilon {    public void throwup() throws RuntimeException {</pre>	exception"
<pre>throw new RuntimeException(); }</pre>	Affiche seulement "Finallied exception"  Erreur d'exécution
<pre>public static void main(String[] args) {     Epsilon epsilon = new Epsilon();</pre>	Erreur de compilation
<pre>try {     epsilon.throwup();</pre>	Affiche seulement "Caught exception"
<pre>} catch (Exception e) {     System.out.println("Caught exception"); } finally file.</pre>	Imene sediement Caught exception
<pre>} finally {     System.out.println("Finallied exception");</pre>	
}	



Erreur d'exécution Soit le code ci-dessous ; que se passe-t-il? Affiche "Done" public class Alpha { public void throwup() throws Exception { Erreur de compilation throw new Exception(); public static void main(String[] args) { Alpha alpha = new Alpha(); alpha.throwup(); System.out.println("Done"); Question 6 Soit le code ci-dessous ; que se passe-t-il? Erreur d'exécution public class Beta { Affiche "Done" public void throwup() throws RuntimeException { Erreur de compilation throw new RuntimeException(); public static void main(String[] args) { Beta beta = new Beta(); beta.throwup(); System.out.println("Done"); }

#### Question 7 Soit le code :

```
interface Somethingable {
    void something();
}

interface Whateverable extends Somethingable {
    void whatever();
}

class TheThing implements Whateverable{
    PPP void something() {
        System.out.println("Somethinging");
    }

    PPP void whatever() {
        System.out.println("Whatevering");
    }
}
```

```
class Doofer {
    void doo(Somethingable sable) {
        sable.XXX();
    }

    void doo(Whateverable sable) {
        sable.YYY();
    }

    void fer(TheThing tt) {
        tt.ZZZ();
    }

    public static void main(String[] args) {
        Doofer doofer = new Doofer();
        TheThing the = new TheThing();
        doofer.doo(the);
        doofer.fer(the);
    }
}
```

Par quels niveaux d'accès pourrait-on remplacer PPP?

private	package-private
public	protected
Question 8 $\bigoplus$ placer XXX ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
something	whatever
Question 9 $\bigoplus$ placer YYY ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
something	whatever

Question 10 $\bigoplus$ remplacer ZZZ ?	Pour le code de la question précédente, par quelles méthodes pourrait-on
something	whatever

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit





Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r\cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r\sin(\theta)$   $\theta = \tan^{-1}\frac{y}{x}$ 

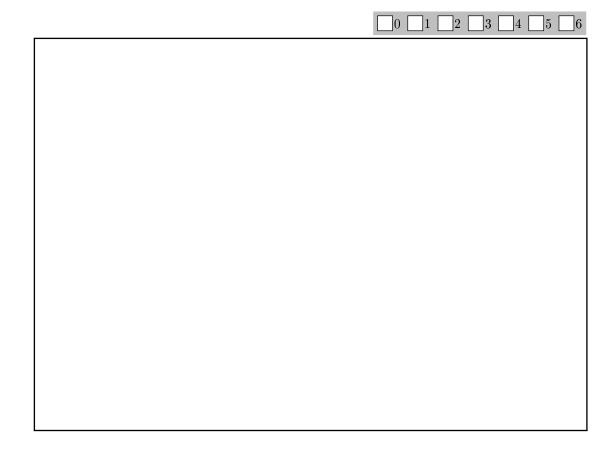
En Java ce dernier est calculé par theta = Math.atan2(y, x)

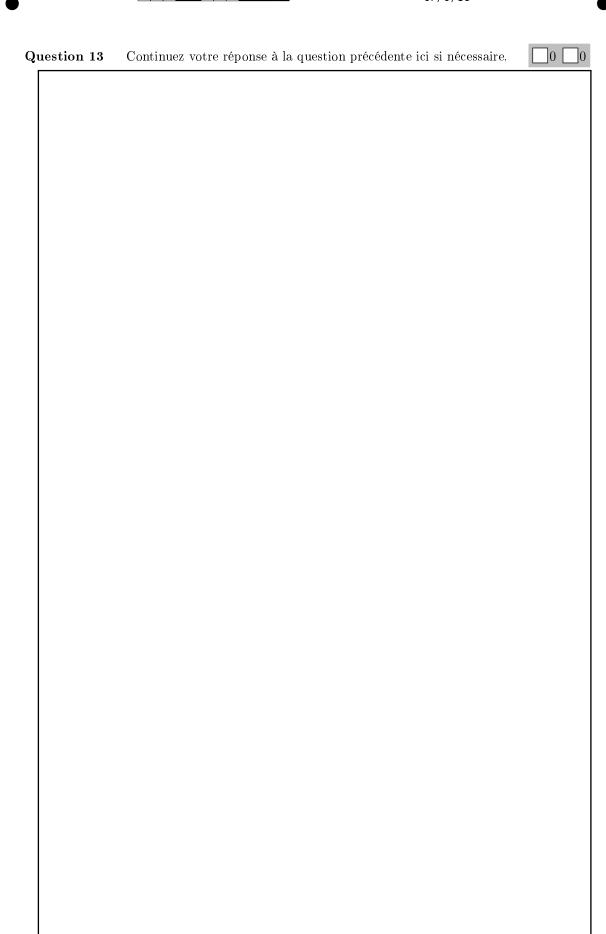
- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.





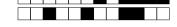


Question 14	Continuez	votre réponse	e à la question	précédente	ici si nécessaire.	



Une classe peut étendre une classe abstraite
 Une interface peut contenir des méthodes privées

Question	15	Pour	la	question	précédente,	développez	la	méthode	Pola		
Question	<b>16</b> $\oplus$	Parn	ni l	es affirma	tions ci-desso	ous, cocher c	elle	s qui sont	vraies	s :	
Une o	classe a	bstrait	ер	eut conte	nir des const	ructeurs					
Une o	classe p	eut éte	$\operatorname{end}$	re plusieu	rs classes						
Une o	classe a	bstrait	ер	eut implé	menter une i	$\operatorname{nterface}$					
					constructeur						
Une o	classe p	eut éte	$\operatorname{end}$	re une cla	sse et implér	nenter plusie	eurs	interfaces			



#### Question $17 \oplus$ Soit le code ci-dessous :

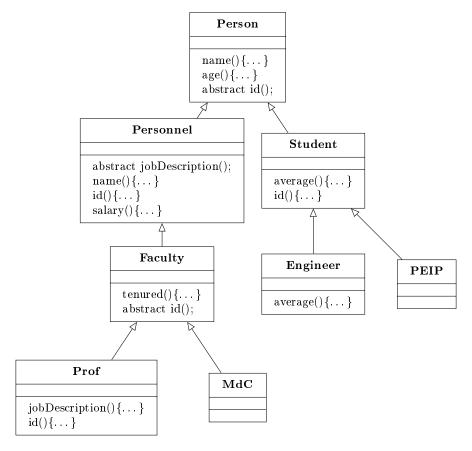
```
@FunctionalInterface
interface ComparePerson {
    /**
    * @return negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :



Question 18  $\bigoplus$  Soit l'organisation ci-dessous de Polytech'Groland :



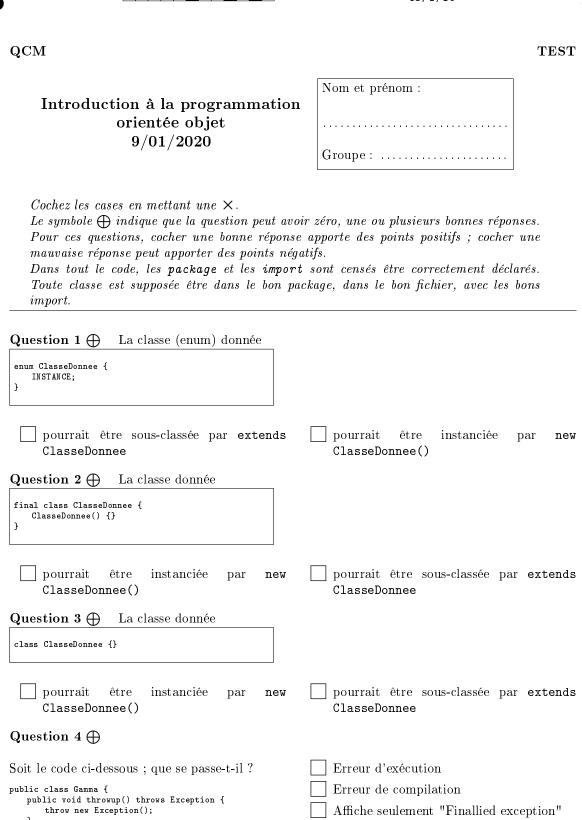
Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

Person	PEIP
Student	Prof
Engineer	Faculty
Personnel	MdC

Affiche seulement "Caught exception"

exception"

Affiche "Caught exception" et "Finallied



public static void main(String[] args) {
 Gamma gamma = new Gamma();

System.out.println("Caught exception");

System.out.println("Finallied exception");

gamma.throwup();

} catch (Exception e) {

try {

}

finally {



## Question 5 $\bigoplus$

```
Affiche seulement "Finallied exception"
Soit le code ci-dessous ; que se passe-t-il?
public class Delta {
                                                            Affiche "Caught exception" et "Finallied
   public void throwup() throws Exception {
                                                            exception"
      throw new Exception();
                                                        Erreur de compilation
   public static void main(String[] args) {
      Delta delta = new Delta();
                                                            Affiche seulement "Caught exception"
      try {
          delta.throwup();
                                                            Erreur d'exécution
      } catch (Exception e) {
          System.out.println("Caught exception");
          return:
      } finally {
          System.out.println("Finallied exception");
   }
}
Question 6
```

```
Soit le code ci-dessous ; que se passe-t-il?
public class Epsilon {
   public void throwup() throws RuntimeException {
       throw new RuntimeException();
   public static void main(String[] args) {
       Epsilon epsilon = new Epsilon();
       try {
          epsilon.throwup();
       } catch (Exception e) {
          System.out.println("Caught exception");
       } finally {
          System.out.println("Finallied exception");
   }
}
```

Affiche "Caught exception" et "Finallied exception"

Affiche seulement "Finallied exception"

Erreur d'exécution

Affiche seulement "Caught exception"

Erreur de compilation

#### Question 7 Soit le code:

```
interface Somethingable {
    void something();
}
interface Whateverable extends Somethingable {
   void whatever();
{\tt class\ The Thing\ implements\ Whateverable} \{
   PPP void something() {
       System.out.println("Somethinging");
   PPP void whatever() {
       System.out.println("Whatevering");
}
```

```
class Doofer {
    void doo(Somethingable sable) {
        sable.XXX();
    void doo(Whateverable sable) {
        sable.YYY();
    void fer(TheThing tt) {
       tt.ZZZ();
   public static void main(String[] args) {
       Doofer doofer = new Doofer();
TheThing the = new TheThing();
        doofer.doo(the);
        doofer.fer(the);
}
```

Par quels niveaux d'accès pourrait-on remplacer PPP?

package-private private public protected

Question 8 $\oplus$ placer XXX ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
something	whatever
Question 9 $\bigoplus$ placer YYY ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
something	whatever
Question 10 ⊕ remplacer ZZZ ?	Pour le code de la question précédente, par quelles méthodes pourrait-on

something

\_\_\_ whatever

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit





Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r \cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r \sin(\theta)$   $\theta = \tan^{-1} \frac{y}{x}$ 

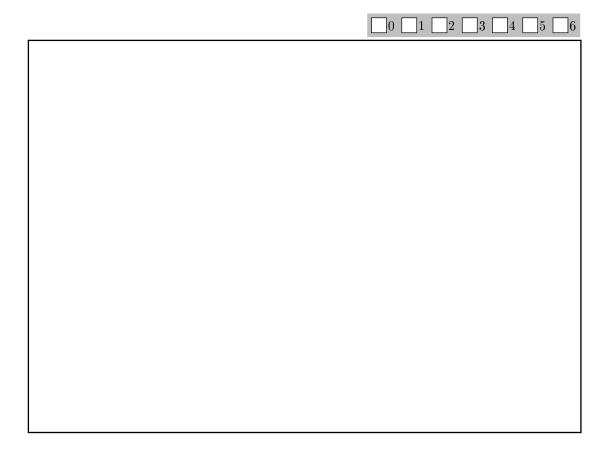
En Java ce dernier est calculé par theta = Math.atan2(y, x)

- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
       return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.

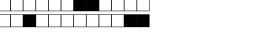




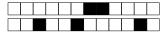
Question 13	Continuez	votre réponse à la question précédente ici si nécessaire.	



Question 14	Continuez	votre réponse	à la question	précédente	ici si nécessaire	e. 0 0



Question 15 Pour la question précédente, développez la méthode Polar#equals.  $\boxed{0}$   $\boxed{1}$   $\boxed{2}$   $\boxed{3}$ Une classe abstraite peut contenir des constructeurs Une classe peut étendre une classe abstraite Une classe abstraite peut implémenter une interface Une interface peut contenir des constructeurs Une interface peut contenir des méthodes privées Une classe peut étendre une classe et implémenter plusieurs interfaces Une classe peut étendre plusieurs classes



#### Question $17 \oplus$ Soit le code ci-dessous :

```
OFunctionalInterface
interface ComparePerson {
    /**
    * Oreturn negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

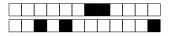
Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :

```
ComparePerson byAge = new ComparePersonByAge();

ComparePerson byAge = (p1, p2) -> p1.age - p2.age;

ComparePerson byAge
= new ComparePersonByAge::compare;
```

```
ComparePerson byAge = new ComparePersonByAge() {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
};
```



 $\textbf{Question 18} \ \bigoplus \quad \text{Soit l'organisation ci-dessous de Polytech'Groland}:$ 



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

PEIP	Engineer
Personnel	Person
Prof	Student
MdC	Faculty

QCMTEST Nom et prénom: Introduction à la programmation orientée objet 9/01/2020 Groupe: ..... Cochez les cases en mettant une  $\times$ . Le symbole  $\bigoplus$  indique que la question peut avoir zéro, une ou plusieurs bonnes réponses. Pour ces questions, cocher une bonne réponse apporte des points positifs ; cocher une mauvaise réponse peut apporter des points négatifs. Dans tout le code, les package et les import sont censés être correctement déclarés. Toute classe est supposée être dans le bon package, dans le bon fichier, avec les bons import.Question 1 

La classe donnée abstract class ClasseDonnee { ClasseDonnee() {} pourrait être sous-classée par extends pourrait être instanciée par new ClasseDonnee ClasseDonnee() Question 2 La classe (enum) donnée enum ClasseDonnee { INSTANCE; } pourrait être sous-classée par extends pourrait être instanciée par new ClasseDonnee ClasseDonnee() Question 3 ( La classe donnée final class ClasseDonnee { ClasseDonnee() {} pourrait être sous-classée par extends pourrait être instanciée par new ClasseDonnee() ClasseDonnee Question 4 Erreur d'exécution Soit le code ci-dessous ; que se passe-t-il? Erreur de compilation public class Alpha { public void throwup() throws Exception { Affiche "Done" throw new Exception();

public static void main(String[] args) {

Alpha alpha = new Alpha();
alpha.throwup();

System.out.println("Done");

}



Affiche "Done" Soit le code ci-dessous ; que se passe-t-il? public class Beta { Erreur d'exécution public void throwup() throws RuntimeException { Erreur de compilation throw new RuntimeException(); public static void main(String[] args) { Beta beta = new Beta(); beta.throwup(); System.out.println("Done"); } Question 6 Soit le code ci-dessous ; que se passe-t-il? Affiche seulement "Caught exception" public class Delta { Erreur de compilation

public class Delta {
 public void throwup() throws Exception {
 throw new Exception();
 }
 public static void main(String[] args) {
 Delta delta = new Delta();
 try {
 delta.throwup();
 } catch (Exception e) {
 System.out.println("Caught exception");
 return;
 } finally {
 System.out.println("Finallied exception");
 }
 }
}

# ☐ Affiche "Caught exception" et "Finallied exception" ☐ Affiche seulement "Finallied exception"

Erreur d'exécution

#### Question 7 Soit le code :

```
interface Somethingable {
    void something();
}

interface Whateverable extends Somethingable {
    void whatever();
}

class TheThing implements Whateverable{
    PPP void something() {
        System.out.println("Somethinging");
    }

    PPP void whatever() {
        System.out.println("Whatevering");
    }
}
```

```
class Doofer {
   void doo(Somethingable sable) {
      sable.XXX();
}

void doo(Whateverable sable) {
      sable.YYY();
}

void fer(TheThing tt) {
      tt.ZZZ();
}

public static void main(String[] args) {
      Doofer doofer = new Doofer();
      TheThing the = new TheThing();
      doofer.doo(the);
      doofer.fer(the);
}
```

Par quels niveaux d'accès pourrait-on remplacer PPP?

r ar queis inveau	a d acces pourrant-on remplacer FFF:
private	package-private
public	protected
Question 8 $\bigoplus$ placer XXX ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem
Whatever	something

Question 9 $\bigoplus$ placer YYY ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem
something	whatever
Question 10 $\bigoplus$ remplacer ZZZ ?	Pour le code de la question précédente, par quelles méthodes pourrait-or
whatever	something

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit





Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r \cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r \sin(\theta)$   $\theta = \tan^{-1} \frac{y}{x}$ 

En Java ce dernier est calculé par theta = Math.atan2(y, x)

- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.





uestion 13	Continuez votre réponse à la question précédente ici si nécessaire.	



Question 14	Continuez	votre réponse	à la question	précédente i	ci si nécessaire.	



Question 15 Pour la question précédente, développez la méthode Polar#equals.  $\boxed{0}$   $\boxed{1}$   $\boxed{2}$   $\boxed{3}$ Une classe abstraite peut implémenter une interface Une classe peut étendre plusieurs classes Une interface peut contenir des méthodes privées Une classe abstraite peut contenir des constructeurs Une interface peut contenir des constructeurs

Une classe peut étendre une classe et implémenter plusieurs interfaces

Une classe peut étendre une classe abstraite



## Question $17 \oplus$ Soit le code ci-dessous :

```
@FunctionalInterface
interface ComparePerson {
    /**
    * @return negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :

```
ComparePerson byAge = new ComparePersonByAge();

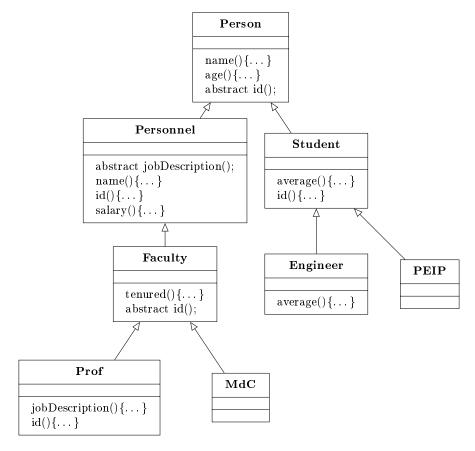
ComparePerson byAge = (p1, p2) -> p1.age - p2.age;

ComparePerson byAge
= new ComparePersonByAge::compare;
```

```
ComparePerson byAge = new ComparePersonByAge() {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
};
```



## Question 18 $\bigoplus$ Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

PEIP	☐ MdC
<pre>Engineer</pre>	Student
Person	Personnel
Prof	Faculty

Affiche seulement "Finallied exception"

QCMTEST Nom et prénom: Introduction à la programmation orientée objet 9/01/2020 Groupe: ..... Cochez les cases en mettant une  $\times$ . Le symbole  $\bigoplus$  indique que la question peut avoir zéro, une ou plusieurs bonnes réponses. Pour ces questions, cocher une bonne réponse apporte des points positifs ; cocher une mauvaise réponse peut apporter des points négatifs. Dans tout le code, les package et les import sont censés être correctement déclarés. Toute classe est supposée être dans le bon package, dans le bon fichier, avec les bons import.Question 1 La classe donnée class ClasseDonnee {} pourrait être sous-classée par extends pourrait être instanciée new ClasseDonnee ClasseDonnee() La classe donnée Question 2 ( class ClasseDonnee { private ClasseDonnee() {} pourrait être sous-classée par extends pourrait être instanciée par new ClasseDonnee() ClasseDonnee Question 3 \( \bigcap \) La classe donnée abstract class ClasseDonnee { ClasseDonnee() {} pourrait être instanciée pourrait être sous-classée par extends par new ClasseDonnee() ClasseDonnee Question 4 Erreur d'exécution Soit le code ci-dessous ; que se passe-t-il ?

public void throwup() throws RuntimeException {
 throw new RuntimeException();
}
public static void main(String[] args) {
 Epsilon epsilon = new Epsilon();
 try {
 epsilon.throwup();
 } catch (Exception e) {
 System.out.println("Caught exception");
} finally {
 System.out.println("Finallied exception");
}

Affiche "Caught exception" et "Finallied exception"

Affiche seulement "Caught exception"

Affiche seulement "Caught exception"

public class Epsilon {

}



#### Question 5 $\bigoplus$

Soit le code ci-dessous ; que se passe-t-il ?	Erreur de compilation
<pre>public class Gamma {     public void throwup() throws Exception {         throw new Exception();     }     public static void main(String[] args) {         Gamma gamma = new Gamma();         try {             gamma.throwup();      } catch (Exception e) {             System.out.println("Caught exception");      } finally {             System.out.println("Finallied exception");       }     } }</pre> Question 6 \(\hat{\Phi}\)	<ul> <li>□ Erreur d'exécution</li> <li>□ Affiche "Caught exception" et "Finallied exception"</li> <li>□ Affiche seulement "Finallied exception"</li> <li>□ Affiche seulement "Caught exception"</li> </ul>
Soit le code ci-dessous ; que se passe-t-il ?	☐ Affiche "Done"
<pre>public class Alpha {    public void throwup() throws Exception {       throw new Exception(); }</pre>	Erreur de compilation  Erreur d'exécution

#### **Question 7** Soit le code :

System.out.println("Done");

alpha.throwup();

}

public static void main(String[] args) {
 Alpha alpha = new Alpha();

```
interface Somethingable {
    void something();
}

interface Whateverable extends Somethingable {
    void whatever();
}

class TheThing implements Whateverable{
    PPP void something() {
        System.out.println("Somethinging");
    }

    PPP void whatever() {
        System.out.println("Whatevering");
    }
}
```

```
class Doofer {
    void doo(Somethingable sable) {
        sable.XXX();
    }

    void doo(Whateverable sable) {
        sable.YYY();
    }

    void fer(TheThing tt) {
        tt.ZZZ();
    }

    public static void main(String[] args) {
        Doofer doofer = new Doofer();
        TheThing the = new TheThing();
        doofer.doo(the);
        doofer.fer(the);
    }
}
```

Par quels niveaux d'accès pourrait-on remplacer PPP?

$oxedsymbol{\Box}$ $ exttt{private}$	package-private
public	protected
Question 8 $\bigoplus$ placer XXX ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
whatever	something

ιEΛ	/2	/10
+50	73	/48-

Question 9 $\bigoplus$ placer YYY ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
whatever	something
Question 10 $\bigoplus$ remplacer ZZZ ?	Pour le code de la question précédente, par quelles méthodes pourrait-on
whatever	something

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
    public double re;
    public Complex(double a, double b) {
        re = a;
        im = b;
    }

    public double getRe() {
        return re;
    }

    public void setRe(double re) {
        this.re = re;
    }

    public double getIm() {
        return im;
    }

    public void setIm(double im) {
        this.im = im;
    }
}
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit





Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r \cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r \sin(\theta)$   $\theta = \tan^{-1} \frac{y}{x}$ 

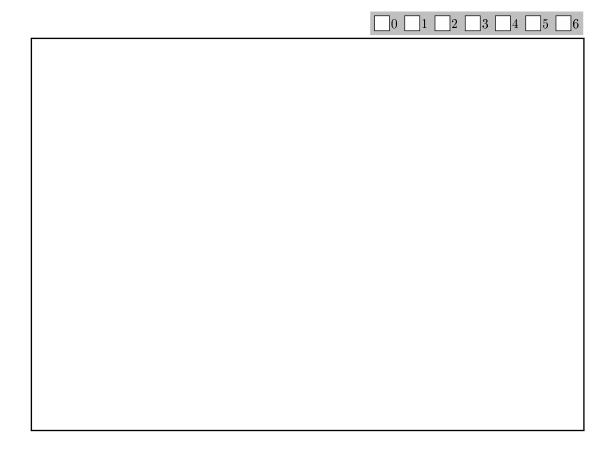
En Java ce dernier est calculé par theta = Math.atan2(y, x)

- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.





Question 13	Continuez votre réponse à la question précédente ici si nécessaire.	



Question 14	Continuez	votre réponse	e à la questior	n précédente	e ici si néces	ssaire.	0 0



Une interface peut contenir des constructeurs

Une classe abstraite peut implémenter une interface

Question 15 Pour la question précédente, développez la méthode Polar#equals.  $\boxed{0}$   $\boxed{1}$   $\boxed{2}$   $\boxed{3}$ Une classe peut étendre une classe abstraite Une classe peut étendre plusieurs classes Une interface peut contenir des méthodes privées Une classe abstraite peut contenir des constructeurs Une classe peut étendre une classe et implémenter plusieurs interfaces



#### Question $17 \oplus$ Soit le code ci-dessous :

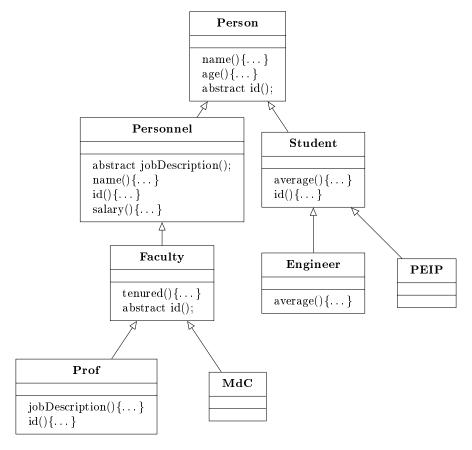
```
@FunctionalInterface
interface ComparePerson {
    /**
    * Oreturn negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :

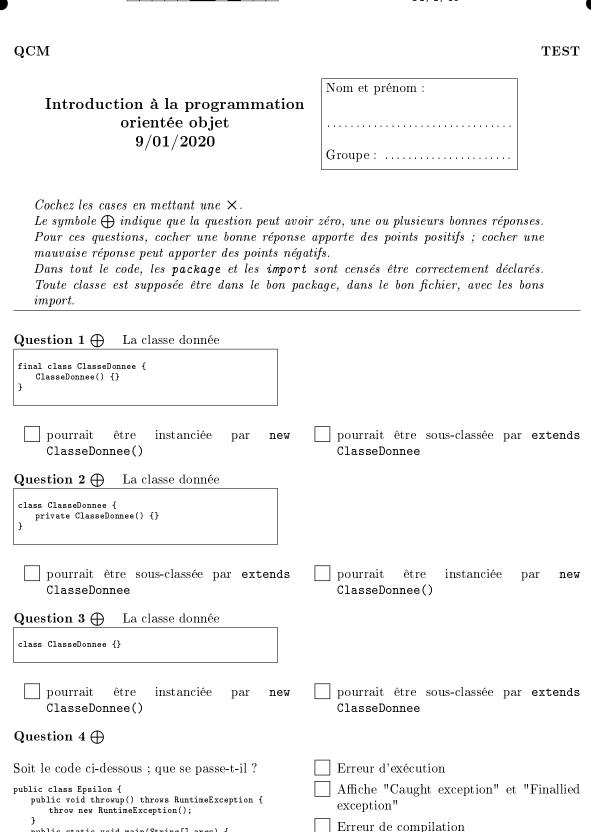


## Question 18 Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

Personnel	MdC
Faculty	Engineer
Student	Prof
Person	PEIP



System.out.println("Finallied exception");
}
}

Affiche seulement "Finallied exception"

Affiche seulement "Caught exception"

public static void main(String[] args) {
 Epsilon epsilon = new Epsilon();

System.out.println("Caught exception");

epsilon.throwup();

} catch (Exception e) {

try {

finally {

#### Question 5 $\bigoplus$

Affiche seulement "Finallied exception" Soit le code ci-dessous ; que se passe-t-il? public class Delta { Erreur d'exécution public void throwup() throws Exception { Affiche seulement "Caught exception" throw new Exception(); public static void main(String[] args) { Affiche "Caught exception" et "Finallied Delta delta = new Delta(); exception" try { delta.throwup(); Erreur de compilation } catch (Exception e) { System.out.println("Caught exception"); return; } finally { System.out.println("Finallied exception"); } Question 6 Affiche "Done" Soit le code ci-dessous ; que se passe-t-il? public class Alpha { Erreur de compilation public void throwup() throws Exception {

#### **Question 7** Soit le code :

throw new Exception();

Alpha alpha = new Alpha();
alpha.throwup();

System.out.println("Done");

public static void main(String[] args) {

```
interface Somethingable {
   void something();
}

interface Whateverable extends Somethingable {
   void whatever();
}

class TheThing implements Whateverable{
   PPP void something() {
       System.out.println("Somethinging");
   }

   PPP void whatever() {
       System.out.println("Whatevering");
   }
}
```

```
class Doofer {
    void doo(Somethingable sable) {
        sable.XXX();
}

void doo(Whateverable sable) {
        sable.YYY();
}

void fer(TheThing tt) {
        tt.ZZZ();
}

public static void main(String[] args) {
        Doofer doofer = new Doofer();
        TheThing the = new TheThing();
        doofer.doo(the);
        doofer.fer(the);
}
```

Erreur d'exécution

Par quels niveaux d'accès pourrait-on remplacer PPP ?

pacl	kage-pri	vate						protected
pri	vate							public
	о Ф	ъ	,	,			,	

Question 8 
Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer XXX ?

whatever something

Question 9 $\bigoplus$ placer YYY ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem
something	whatever
Question 10 $\bigoplus$ remplacer ZZZ ?	Pour le code de la question précédente, par quelles méthodes pourrait-or
Whatever	Something

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

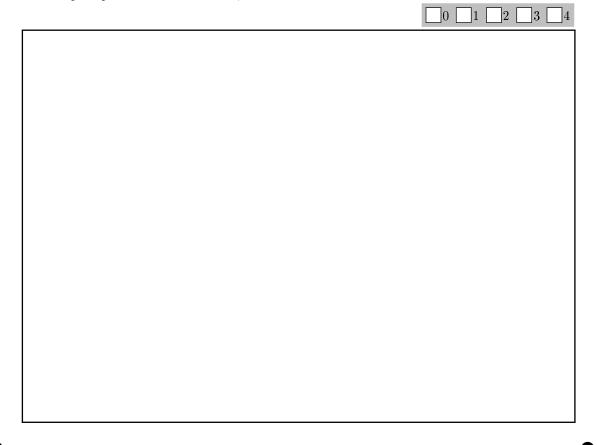
public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit





Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r\cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r\sin(\theta)$   $\theta = \tan^{-1}\frac{y}{x}$ 

En Java ce dernier est calculé par theta = Math.atan2(y, x)

- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.





Question 13	Continuez	votre réponse à	la question p	récédente i	ci si nécessaire.	



Question 14	Continuez	votre réponse	à la question	précédente	ici si nécessaire	e. 0 0



Une classe peut étendre plusieurs classes

Question	15	Pour	la	question	précédente,	développez	la	méthode	Polar	#equa	als.	
									1		3	$\Box 4$
Question Une o	_				tions ci-desso menter une i:		elle	s qui sont	vraies	:		
			_	=	méthodes pr							
Une o	classe p	eut éte	endi	re une cla	sse et implén	nenter plusie	eurs	interfaces	3			
Une o	classe p	eut éte	endi	re une cla	sse abstraite							
Une o	classe al	ostrait	ер	eut contei	nir des consti	ructeurs						
Une i	$_{ m nterface}$	e peut	cor	ntenir des	constructeur	'S						



### Question $17 \oplus$ Soit le code ci-dessous :

```
OFunctionalInterface
interface ComparePerson {
    /**
    * Oreturn negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :

```
ComparePerson byAge = (p1, p2) -> p1.age - p2.age;

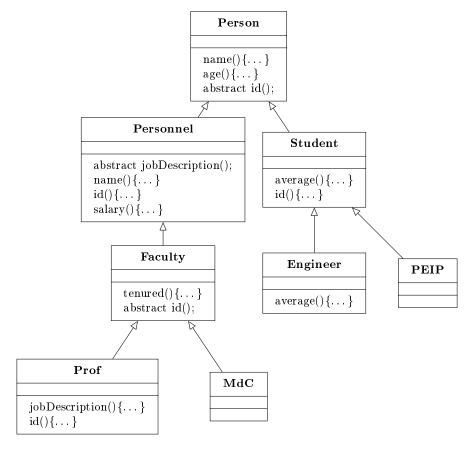
ComparePerson byAge = new ComparePersonByAge::compare;

ComparePerson byAge = new ComparePersonByAge();
```

```
ComparePerson byAge = new ComparePersonByAge() {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
};
```



Question 18  $\bigoplus$  Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

Faculty	Personnel
Prof	PEIP
Engineer	MdC
Student	Person

QCMTEST Nom et prénom: Introduction à la programmation orientée objet 9/01/2020 Groupe: ..... Cochez les cases en mettant une  $\times$ . Le symbole  $\bigoplus$  indique que la question peut avoir zéro, une ou plusieurs bonnes réponses. Pour ces questions, cocher une bonne réponse apporte des points positifs ; cocher une mauvaise réponse peut apporter des points négatifs. Dans tout le code, les package et les import sont censés être correctement déclarés. Toute classe est supposée être dans le bon package, dans le bon fichier, avec les bons import.Question 1 

La classe donnée abstract class ClasseDonnee { ClasseDonnee() {} pourrait être sous-classée par extends pourrait être instanciée par ClasseDonnee() ClasseDonnee La classe (enum) donnée Question 2 ( enum ClasseDonnee { INSTANCE; } pourrait être sous-classée par extends pourrait être instanciée par new ClasseDonnee ClasseDonnee() Question 3 ( La classe donnée final class ClasseDonnee { ClasseDonnee() {} pourrait être sous-classée par extends pourrait être instanciée par new ClasseDonnee ClasseDonnee() Question 4 Erreur d'exécution Soit le code ci-dessous ; que se passe-t-il?

public class Beta {

}

public void throwup() throws RuntimeException {

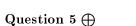
throw new RuntimeException();
}
public static void main(String[] args) {

System.out.println("Done");

Beta beta = new Beta();
beta.throwup();

Erreur de compilation

Affiche "Done"



Erreur de compilation Soit le code ci-dessous ; que se passe-t-il? Affiche seulement "Finallied exception" public class Gamma { public void throwup() throws Exception { Affiche seulement "Caught exception" throw new Exception(); public static void main(String[] args) { Erreur d'exécution Gamma gamma = new Gamma(); try { Affiche "Caught exception" et "Finallied gamma.throwup(); exception" } catch (Exception e) { System.out.println("Caught exception"); } finally { System.out.println("Finallied exception"); } } Question 6  $\bigoplus$ Affiche "Done" Soit le code ci-dessous ; que se passe-t-il? public class Alpha { Erreur de compilation public void throwup() throws Exception { throw new Exception(); Erreur d'exécution public static void main(String[] args) {
 Alpha alpha = new Alpha(); alpha.throwup(); System.out.println("Done");

#### Question 7 Soit le code :

}

```
interface Somethingable {
    void something();
}

interface Whateverable extends Somethingable {
    void whatever();
}

class TheThing implements Whateverable{
    PPP void something() {
        System.out.println("Somethinging");
    }

    PPP void whatever() {
        System.out.println("Whatevering");
    }
}
```

```
class Doofer {
    void doo(Somethingable sable) {
        sable.XXX();
    }

    void doo(Whateverable sable) {
        sable.YYY();
    }

    void fer(TheThing tt) {
        tt.ZZZ();
    }

    public static void main(String[] args) {
        Doofer doofer = new Doofer();
        TheThing the = new TheThing();
        doofer.doo(the);
        doofer.fer(the);
    }
}
```

Par quels niveaux d'accès pourrait-on remplacer PPP?

package-priv	vate private
protected	public
Question 8 $\oplus$ placer XXX ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
something	whatever

Question 9 $\bigoplus$ placer YYY ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
whatever	$\square$ something
Question 10 $\bigoplus$ remplacer ZZZ ?	Pour le code de la question précédente, par quelles méthodes pourrait-on
something	whatever

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit





Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r\cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r\sin(\theta)$   $\theta = \tan^{-1}\frac{y}{x}$ 

En Java ce dernier est calculé par theta = Math.atan2(y, x)

- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.





Question 13	Continuez votre réponse à la question précédente ici s	i nécessaire.	



Question 14	Continuez	votre réponse	à la question	précédente	ici si nécessaire.	



Question 15 Pour la question précédente, développez la méthode Polar#equals.  $\boxed{0}$   $\boxed{1}$   $\boxed{2}$   $\boxed{3}$ Question 16  $\bigoplus$ Parmi les affirmations ci-dessous, cocher celles qui sont vraies : Une classe abstraite peut contenir des constructeurs Une interface peut contenir des méthodes privées Une classe peut étendre plusieurs classes Une classe abstraite peut implémenter une interface Une classe peut étendre une classe et implémenter plusieurs interfaces

Une interface peut contenir des constructeurs
Une classe peut étendre une classe abstraite

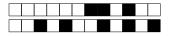


### Question $17 \oplus$ Soit le code ci-dessous :

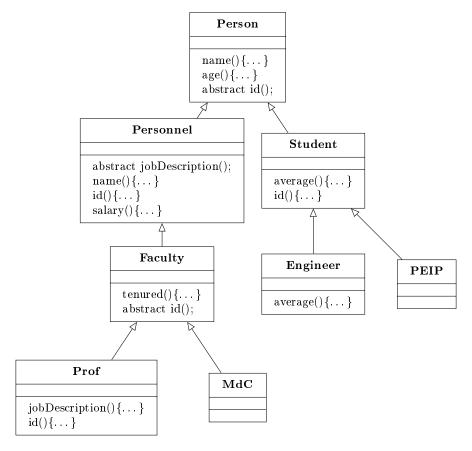
```
@FunctionalInterface
interface ComparePerson {
    /**
    * Oreturn negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :



# Question 18 $\bigoplus$ Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

PEIP	MdC
<pre>Engineer</pre>	Personnel
Prof	Student
Person	$\Box$ Faculty



 $\mathbf{QCM}$ TEST

Introduction à la programmation	Nom et prénom :
orientée objet	
9/01/2020	Groupe:
lacher les cases en mettent une Y	

Cochez les cases en mettant une  $\times$ .

 $Le\ symbole\ \bigoplus\ indique\ que\ la\ question\ peut\ avoir\ z\'ero,\ une\ ou\ plusieurs\ bonnes\ r\'eponses.$ Pour ces questions, cocher une bonne réponse apporte des points positifs ; cocher une mauvaise réponse peut apporter des points négatifs.

 $Dans\ tout\ le\ code,\ les\ \textit{package}\ et\ les\ \textit{import}\ sont\ cens\'es\ \^etre\ correctement\ d\'eclar\'es.$ Toute classe est supposée être dans le bon package, dans le bon fichier, avec les bons import.

Question 1   La classe donnée	
class ClasseDonnee {}	
pourrait être sous-classée par extends ClasseDonnee	pourrait être instanciée par neu ClasseDonnee()
Question 2    La classe donnée	
class ClasseDonnee {    private ClasseDonnee() {} }	
pourrait être sous-classée par <b>extends</b> ClasseDonnee	pourrait être instanciée par neu ClasseDonnee()
Question 3	
abstract class ClasseDonnee {     ClasseDonnee() {} }	
pourrait être instanciée par <b>new</b>	pourrait être sous-classée par extende ClasseDonnee

```
Affiche seulement "Finallied exception"
Soit le code ci-dessous ; que se passe-t-il?
public class Delta {
                                                            Erreur de compilation
   public void throwup() throws Exception {
                                                            Erreur d'exécution
      throw new Exception();
   public static void main(String[] args) {
                                                            Affiche "Caught exception" et "Finallied
      Delta delta = new Delta();
                                                            exception"
      try {
          delta.throwup();
                                                            Affiche seulement "Caught exception"
      } catch (Exception e) {
          System.out.println("Caught exception");
          return;
      } finally {
          System.out.println("Finallied exception");
}
Question 5
```

```
Soit le code ci-dessous ; que se passe-t-il?
public class Epsilon {
   public void throwup() throws RuntimeException {
       throw new RuntimeException();
   public static void main(String[] args) {
       Epsilon epsilon = new Epsilon();
       try {
           epsilon.throwup();
       } catch (Exception e) {
           System.out.println("Caught exception");
       } finally {
   System.out.println("Finallied exception");
   }
}
```

### Question 6

Soit le code ci-dessous ; que se passe-t-il? public class Beta { public void throwup() throws RuntimeException { throw new RuntimeException(); public static void main(String[] args) { Beta beta = new Beta(); beta.throwup(); System.out.println("Done"); }

	Affiche seulement	"Caught	exception"

Affiche seulement "Finallied exception"

Erreur de compilation

Affiche "Caught exception" et "Finallied exception"

Erreur d'exécution

Erreur d'exécution

Affiche "Done"

Erreur de compilation



### Question 7 Soit le code :

```
interface Somethingable {
   void something();
}

interface Whateverable extends Somethingable {
   void whatever();
}

class TheThing implements Whateverable{
   PPP void something() {
       System.out.println("Somethinging");
   }

   PPP void whatever() {
       System.out.println("Whatevering");
   }
}
```

```
class Doofer {
    void doo(Somethingable sable) {
        sable.XXX();
    }

    void doo(Whateverable sable) {
        sable.YYY();
    }

    void fer(TheThing tt) {
        tt.ZZZ();
    }

    public static void main(String[] args) {
        Doofer doofer = new Doofer();
        TheThing the = new TheThing();
        doofer.doo(the);
        doofer.fer(the);
    }
}
```

Par quels niveaux d'accès pourrait-on remplacer PPP? public package-private protected private Question 8 Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer XXX? whatever something Question 9 Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer YYY? something whatever Question 10  $\bigoplus$ Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer ZZZ? whatever something

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit





Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r\cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r\sin(\theta)$   $\theta = \tan^{-1}\frac{y}{x}$ 

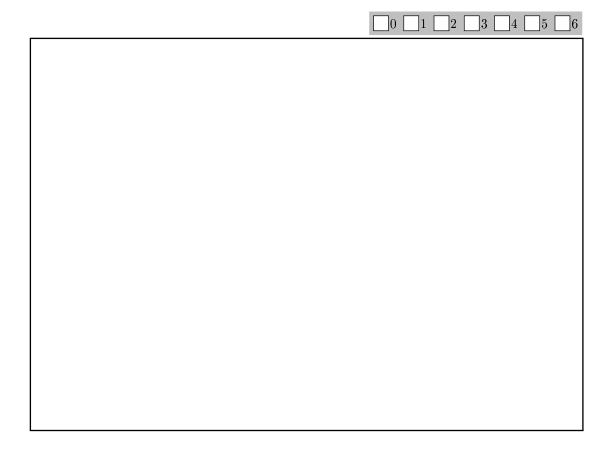
En Java ce dernier est calculé par theta = Math.atan2(y, x)

- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.





Question 13	Continuez votre réponse à la question précédente ici si nécessaire.	0 0



Question 14	Continuez votre réponse à la question précédente ici si nécessaire.	0 0



Une classe peut étendre plusieurs classes

Question	15	Pour	la	question	précédente,	développez	la	méthode	Polar	#equa	ıls.	
										$\square 2$	3	$\Box$ 4
Question Une	_				tions ci-desso sse abstraite	ous, cocher c	elle	s qui sont	vraies	:		
_	_				constructeur	`S						
					sse et implén		eurs	interfaces				
_					méthodes pr							
=		_			menter une i							
Une	classe al	ostrait	ер	eut conter	nir des consti	ructeurs						



### Question $17 \oplus$ Soit le code ci-dessous :

```
@FunctionalInterface
interface ComparePerson {
    /**
    * @return negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :

```
ComparePerson byAge = new ComparePersonByAge();

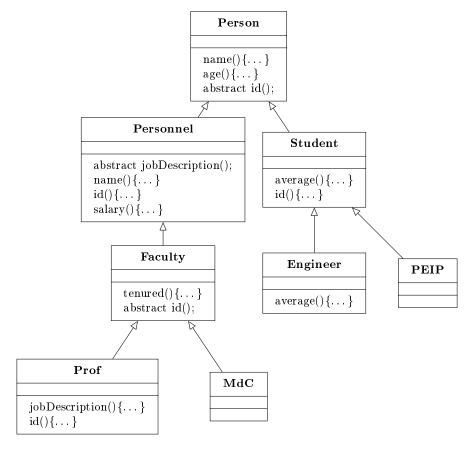
ComparePerson byAge = new ComparePersonByAge::compare;

ComparePerson byAge = (p1, p2) -> p1.age - p2.age;
```

```
ComparePerson byAge = new ComparePersonByAge() {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
};
```

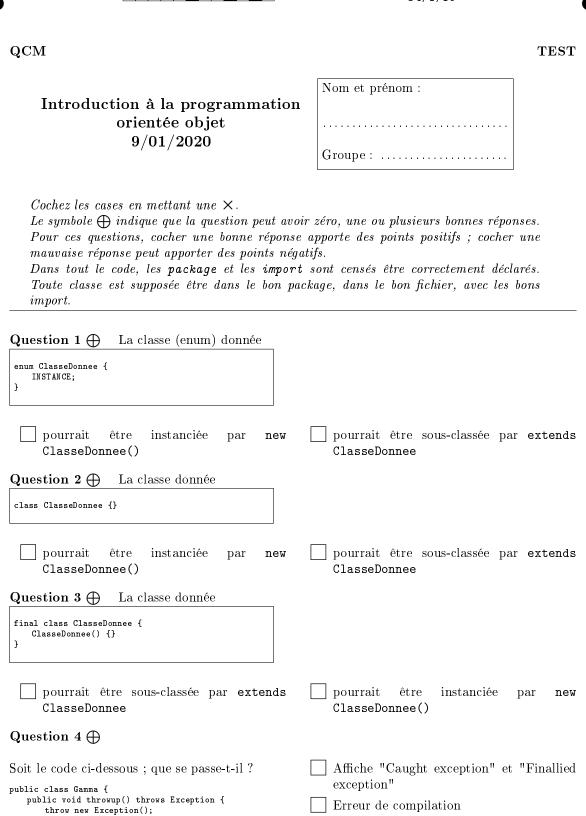


Question 18  $\bigoplus$  Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

Student	PEIP
MdC	$oxedsymbol{\Box}$ Faculty
Person	Personnel
Prof	Engineer



public static void main(String[] args) {
 Gamma gamma = new Gamma();

System.out.println("Caught exception");

System.out.println("Finallied exception");

gamma.throwup();

} catch (Exception e) {

try {

}

finally {

Erreur d'exécution

Affiche seulement "Finallied exception"

Affiche seulement "Caught exception"



## Question 5 $\bigoplus$

```
Soit le code ci-dessous ; que se passe-t-il?
                                                          Erreur de compilation
                                                            Affiche seulement "Caught exception"
public class Delta {
   public void throwup() throws Exception {
                                                            Affiche seulement "Finallied exception"
      throw new Exception();
   public static void main(String[] args) {
                                                            Erreur d'exécution
      Delta delta = new Delta();
      try {
                                                            Affiche "Caught exception" et "Finallied
          delta.throwup();
                                                            exception"
      } catch (Exception e) {
          System.out.println("Caught exception");
          return;
      } finally {
          System.out.println("Finallied exception");
   }
}
Question 6
                                                            Affiche "Done"
Soit le code ci-dessous ; que se passe-t-il?
public class Alpha {
                                                            Erreur de compilation
   public void throwup() throws Exception {
```

#### Question 7 Soit le code :

throw new Exception();

Alpha alpha = new Alpha();
alpha.throwup();

System.out.println("Done");

public static void main(String[] args) {

```
interface Somethingable {
   void something();
}

interface Whateverable extends Somethingable {
   void whatever();
}

class TheThing implements Whateverable{
   PPP void something() {
      System.out.println("Somethinging");
   }

   PPP void whatever() {
      System.out.println("Whatevering");
   }
}
```

```
class Doofer {
    void doo(Somethingable sable) {
        sable.XXX();
    }

    void doo(Whateverable sable) {
        sable.YYY();
    }

    void fer(TheThing tt) {
        tt.ZZZ();
    }

    public static void main(String[] args) {
        Doofer doofer = new Doofer();
        TheThing the = new TheThing();
        doofer.doo(the);
        doofer.fer(the);
    }
}
```

Erreur d'exécution

Par quels niveaux d'accès pourrait-on remplacer PPP ?

<pre>public private</pre>	package-private protected
Question 8 $\bigoplus$ placer XXX ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem
something	whatever

$\Box$	$\perp$				

Question 9 $\bigoplus$ placer YYY ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
something	whatever
Question 10 $\bigoplus$ remplacer ZZZ ?	Pour le code de la question précédente, par quelles méthodes pourrait-on
whatever	something

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

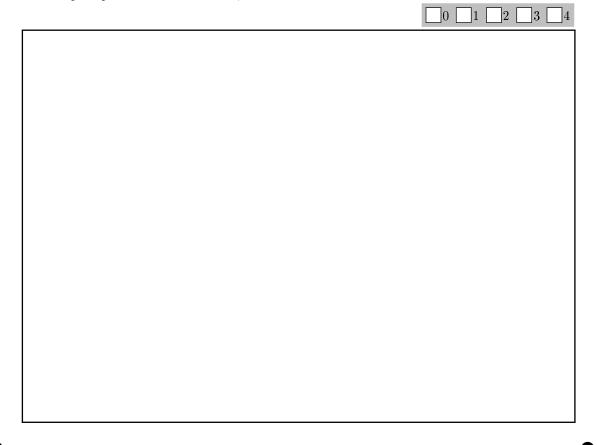
public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

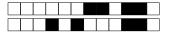
@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit





Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r\cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r\sin(\theta)$   $\theta = \tan^{-1}\frac{y}{x}$ 

En Java ce dernier est calculé par theta = Math.atan2(y, x)

- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.

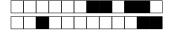




Question 13	Continuez	votre réponse à	la question p	orécédente i	ci si nécessaire.	



uestion 14	Continuez votre réponse à la question précédente ici si nécessaire.	



Une interface peut contenir des méthodes privées
Une interface peut contenir des constructeurs

Question 15 Pour la question précédente, développez la méthode Polar#equals.  $\boxed{0}$   $\boxed{1}$   $\boxed{2}$   $\boxed{3}$ Une classe abstraite peut implémenter une interface Une classe peut étendre une classe et implémenter plusieurs interfaces Une classe abstraite peut contenir des constructeurs Une classe peut étendre plusieurs classes Une classe peut étendre une classe abstraite



### Question $17 \oplus$ Soit le code ci-dessous :

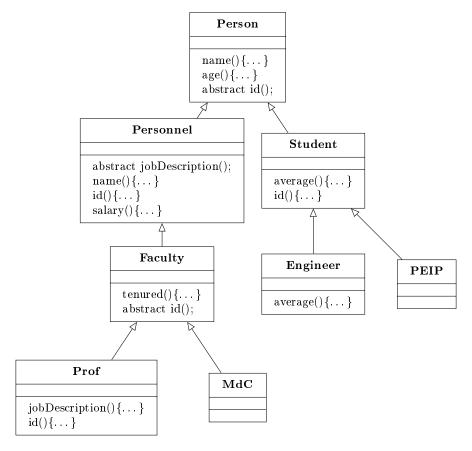
```
@FunctionalInterface
interface ComparePerson {
    /**
    * Oreturn negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :



Question 18  $\bigoplus$  Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

<pre>Faculty</pre>	Student
Engineer	PEIP
Prof	MdC
Personnel	Person

QCMTEST Nom et prénom: Introduction à la programmation orientée objet 9/01/2020 Groupe: ..... Cochez les cases en mettant une  $\times$ . Le symbole  $\bigoplus$  indique que la question peut avoir zéro, une ou plusieurs bonnes réponses. Pour ces questions, cocher une bonne réponse apporte des points positifs ; cocher une mauvaise réponse peut apporter des points négatifs. Dans tout le code, les package et les import sont censés être correctement déclarés. Toute classe est supposée être dans le bon package, dans le bon fichier, avec les bons import.Question 1 

La classe donnée abstract class ClasseDonnee { ClasseDonnee() {} pourrait être sous-classée par extends pourrait être instanciée par new ClasseDonnee() ClasseDonnee Question 2 La classe donnée class ClasseDonnee { private ClasseDonnee() {} } être pourrait être sous-classée par extends pourrait instanciée new par ClasseDonnee() ClasseDonnee Question 3 ( La classe (enum) donnée enum ClasseDonnee { INSTANCE; pourrait être sous-classée par extends pourrait être instanciée par new ClasseDonnee ClasseDonnee() Question 4 Erreur d'exécution Soit le code ci-dessous ; que se passe-t-il? Erreur de compilation public class Beta { public void throwup() throws RuntimeException {

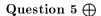
Affiche "Done"

throw new RuntimeException();
}
public static void main(String[] args) {

System.out.println("Done");

Beta beta = new Beta();
beta.throwup();

}



```
Erreur de compilation
Soit le code ci-dessous ; que se passe-t-il?
public class Epsilon {
                                                             Affiche seulement "Finallied exception"
   public void throwup() throws RuntimeException {
                                                             Affiche "Caught exception" et "Finallied
      throw new RuntimeException();
   public static void main(String[] args) {
      Epsilon epsilon = new Epsilon();
                                                             Erreur d'exécution
      try {
          epsilon.throwup();
                                                             Affiche seulement "Caught exception"
      } catch (Exception e) {
         System.out.println("Caught exception");
      } finally {
          System.out.println("Finallied exception");
   }
}
Question 6 \bigoplus
```

Soit le code ci-dessous ; que se passe-t-il ?

public class Gamma {
 public void throwup() throws Exception {
 throw new Exception();
 }
 public static void main(String[] args) {
 Gamma gamma = new Gamma();
 try {
 gamma.throwup();
 } catch (Exception e) {
 System.out.println("Caught exception");
 } finally {
 System.out.println("Finallied exception");
 }
}

# Erreur de compilation

Affiche "Caught exception" et "Finallied exception"

Affiche seulement "Finallied exception"

Affiche seulement "Caught exception"

Erreur d'exécution

## Question 7 Soit le code :

```
interface Somethingable {
    void something();
}

interface Whateverable extends Somethingable {
    void whatever();
}

class TheThing implements Whateverable{
    PPP void something() {
        System.out.println("Somethinging");
    }

    PPP void whatever() {
        System.out.println("Whatevering");
    }
}
```

```
class Doofer {
    void doo(Somethingable sable) {
        sable.XXX();
    }

    void doo(Whateverable sable) {
        sable.YYY();
    }

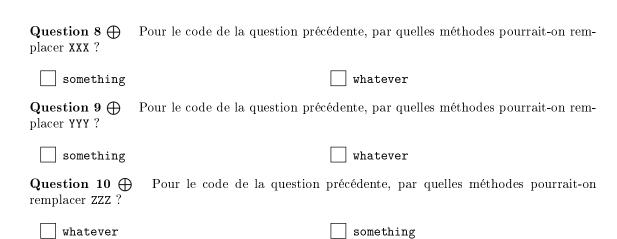
    void fer(TheThing tt) {
        tt.ZZZ();
    }

    public static void main(String[] args) {
        Doofer doofer = new Doofer();
        TheThing the = new TheThing();
        doofer.doo(the);
        doofer.fer(the);
    }
}
```

Par quels niveaux d'accès pourrait-on remplacer PPP?

□ private
□ protected

□ public
□ package-private



Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit



Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r\cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r\sin(\theta)$   $\theta = \tan^{-1}\frac{y}{x}$ 

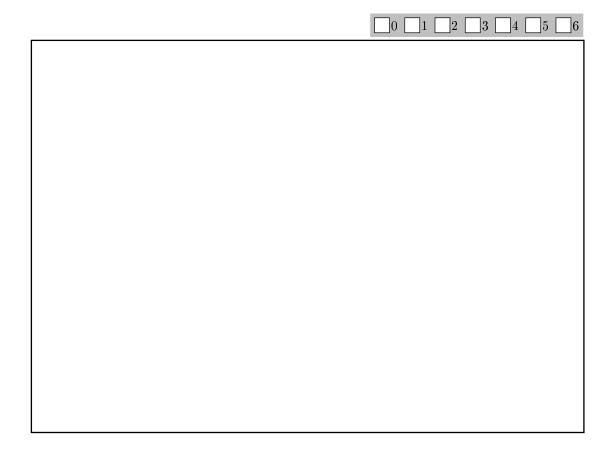
En Java ce dernier est calculé par theta = Math.atan2(y, x)

- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
       return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.





Question 13	Continuez	votre réponse	à la question	précédente i	ci si nécessaire.	0 0



Question 14	Continuez	votre réponse	à la question	précédente	ici si nécessaire.	



${f Question}$	15	Pour	la	question	précédente,	développez	la	_	Polari $0                   $	#equals.
${f Question}$	16 ⊕	Parn	ni le	es affirma	tions ci-dess	ous, cocher c	elles	qui son	t vraies	:
Une o	classe p	eut éte	endı	e une cla	sse abstraite	<u>)</u>				
Une o	classe al	bstrait	ере	eut conte	nir des const	ructeurs				
Une o	classe p	eut éte	endr	e une cla	sse et implé	menter plusie	eurs	interfac	es	
Une i	interface	e peut	con	tenir des	méthodes p	rivées				
Une i	interface	e peut	con	tenir des	constructeu	rs				
Une o	classe p	eut éte	endr	e plusieu	rs classes					
Une o	classe al	bstrait	е ре	eut implé	menter une	interface				

# Question $17 \oplus$ Soit le code ci-dessous :

```
@FunctionalInterface
interface ComparePerson {
    /**
    * @return negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :



## Question 18 $\bigoplus$ Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

Student	Engineer
Faculty	PEIP
Personnel	Person
MdC	Prof

QCMTEST Nom et prénom: Introduction à la programmation orientée objet 9/01/2020 Groupe: ..... Cochez les cases en mettant une  $\times$ . Le symbole  $\bigoplus$  indique que la question peut avoir zéro, une ou plusieurs bonnes réponses. Pour ces questions, cocher une bonne réponse apporte des points positifs ; cocher une mauvaise réponse peut apporter des points négatifs. Dans tout le code, les package et les import sont censés être correctement déclarés. Toute classe est supposée être dans le bon package, dans le bon fichier, avec les bons import.Question 1 La classe donnée class ClasseDonnee {} pourrait être sous-classée par extends pourrait être instanciée par new ClasseDonnee() ClasseDonnee Question 2 La classe donnée class ClasseDonnee { private ClasseDonnee() {} pourrait être sous-classée par extends pourrait être instanciée new par ClasseDonnee ClasseDonnee() Question 3  $\bigoplus$ La classe donnée final class ClasseDonnee { ClasseDonnee() {} pourrait être sous-classée par extends pourrait être instanciée new ClasseDonnee ClasseDonnee() Question 4 Erreur de compilation Soit le code ci-dessous ; que se passe-t-il? Erreur d'exécution public class Beta {

Affiche "Done"

public void throwup() throws RuntimeException {

throw new RuntimeException();
}
public static void main(String[] args) {

Beta beta = new Beta();
beta.throwup();

}

System.out.println("Done");



Affiche seulement "Finallied exception" Soit le code ci-dessous ; que se passe-t-il? public class Epsilon { Erreur de compilation public void throwup() throws RuntimeException { Erreur d'exécution throw new RuntimeException(); public static void main(String[] args) { Affiche seulement "Caught exception" Epsilon epsilon = new Epsilon(); try { Affiche "Caught exception" et "Finallied epsilon.throwup(); exception" } catch (Exception e) { System.out.println("Caught exception"); } finally { System.out.println("Finallied exception"); } } Question 6  $\bigoplus$ Affiche "Caught exception" et "Finallied Soit le code ci-dessous ; que se passe-t-il? exception" public class Gamma {

## Question 7 Soit le code :

public void throwup() throws Exception {

public static void main(String[] args) {
 Gamma gamma = new Gamma();

System.out.println("Caught exception");

System.out.println("Finallied exception");

throw new Exception();

gamma.throwup();

} catch (Exception e) {

try {

}

} finally {

```
interface Somethingable {
    void something();
}

interface Whateverable extends Somethingable {
    void whatever();
}

class TheThing implements Whateverable{
    PPP void something() {
        System.out.println("Somethinging");
    }

    PPP void whatever() {
        System.out.println("Whatevering");
    }
}
```

```
class Doofer {
   void doo(Somethingable sable) {
      sable.XXX();
}

void doo(Whateverable sable) {
   sable.YYY();
}

void fer(TheThing tt) {
   tt.ZZZ();
}

public static void main(String[] args) {
   Doofer doofer = new Doofer();
   TheThing the = new TheThing();
   doofer.doo(the);
   doofer.fer(the);
}
```

Affiche seulement "Finallied exception"

Affiche seulement "Caught exception"

Erreur d'exécution

Erreur de compilation

Par quels niveaux d'accès pourrait-on remplacer PPP?

 □ package-private
 □ private

 □ protected
 □ public

Question 8 $\bigoplus$ placer XXX ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
whatever	something
Question 9 $\bigoplus$ placer YYY ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
something	whatever
$\begin{array}{c} \mathbf{Question}  10  \bigoplus \\ \mathbf{remplacer}  \mathbf{ZZZ} \ ? \end{array}$	Pour le code de la question précédente, par quelles méthodes pourrait-on
whatever	something

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit



Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r\cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r\sin(\theta)$   $\theta = \tan^{-1}\frac{y}{x}$ 

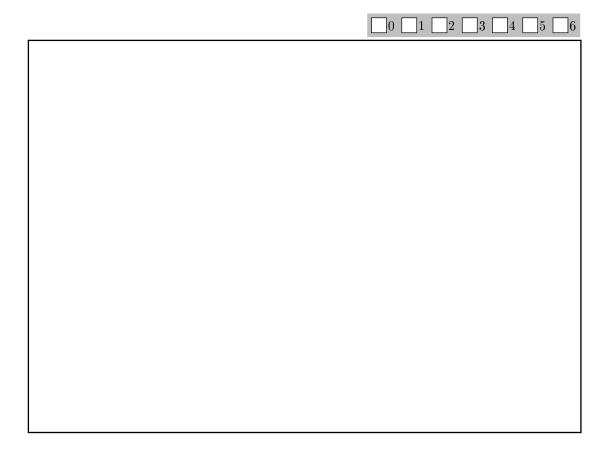
En Java ce dernier est calculé par theta = Math.atan2(y, x)

- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
       return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.





Question 13	Continuez	votre réponse	à la question	précédente i	ci si nécessaire.	0 0



${f Question}$	14	Continuez	votre	réponse	à la	question	précéde	nte ici s	si nécessai	re. 0	0



Une classe abstraite peut implémenter une interface

Une classe peut étendre une classe abstraite

Question	15	Pour	la	question	précédente,	développez	la	méthode Polar#equals.
${f Question}$	16 ⊕	Parm	ni le	es affirma	tions ci-desse	ous, cocher c	elle	s qui sont vraies :
Une o	classe p	eut éte	ndı	re une cla	sse et implér	nenter plusie	eurs	interfaces
Une i	nterfac	e peut	cor	ntenir des	constructeur	rs		
=			_		nir des const			
					méthodes pi	rivées		
$\bigsqcup$ Une $\alpha$	Une classe peut étendre plusieurs classes							



## Question $17 \oplus$ Soit le code ci-dessous :

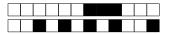
```
@FunctionalInterface
interface ComparePerson {
    /**
    * @return negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :

```
ComparePerson byAge = new ComparePersonByAge::compare;

ComparePerson byAge = new ComparePersonByAge() {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
};
```

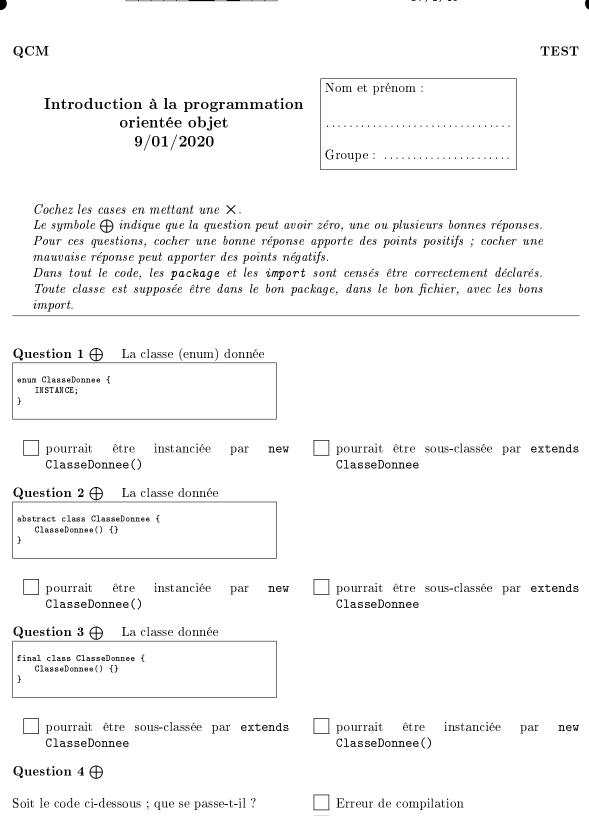


## Question 18 $\bigoplus$ Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

Prof	Engineer
PEIP	Personnel
MdC	Person
Faculty	Student



Affiche "Done"

Erreur d'exécution

public class Alpha {

}

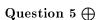
public void throwup() throws Exception {

public static void main(String[] args) {

throw new Exception();

Alpha alpha = new Alpha();
alpha.throwup();

System.out.println("Done");



```
Soit le code ci-dessous ; que se passe-t-il?
public class Delta {
   public void throwup() throws Exception {
       throw new Exception();
   public static void main(String[] args) {
       Delta delta = new Delta();
       try {
          delta.throwup();
                                                               Erreur d'exécution
       } catch (Exception e) {
          System.out.println("Caught exception");
          return;
       } finally {
          System.out.println("Finallied exception");
   }
}
```

Affiche seulement "Finallied exception"
Affiche seulement "Caught exception"
Affiche "Caught exception" et "Finallied exception"
Erreur de compilation

```
Soit le code ci-dessous ; que se passe-t-il ?
public class Epsilon {
   public void throwup() throws RuntimeException {
       throw new RuntimeException();
   public static void main(String[] args) {
       Epsilon epsilon = new Epsilon();
       try {
          epsilon.throwup();
       } catch (Exception e) {
          System.out.println("Caught exception");
       } finally {
          System.out.println("Finallied exception");
}
```

Affiche seulement "Finallied exception"
Affiche seulement "Caught exception"
Erreur d'exécution
Affiche "Caught exception" et "Finallied exception"
Erreur de compilation

#### Question 7 Soit le code:

```
interface Somethingable {
   void something();
}
interface Whateverable extends Somethingable {
   void whatever();
class TheThing implements Whateverable{
   PPP void something() {
      System.out.println("Somethinging");
   PPP void whatever() {
       System.out.println("Whatevering");
}
```

```
class Doofer {
   void doo(Somethingable sable) {
        sable.XXX();
    void doo(Whateverable sable) {
        sable.YYY();
   void fer(TheThing tt) {
       tt.ZZZ();
   public static void main(String[] args) {
       Doofer doofer = new Doofer();
TheThing the = new TheThing();
        doofer.doo(the);
        doofer.fer(the);
}
```

Par quels niveaux d'accès pourrait-on remplacer PPP?

```
private
                                             public
 package-private
                                             protected
```



Question 8 $\oplus$ placer XXX ?	Pour le code de la question précé	édente, par quelles méthodes pourrait-on rem-
something		whatever
Question 9 $\bigoplus$ placer YYY ?	Pour le code de la question précé	édente, par quelles méthodes pourrait-on rem-
something		whatever
Question 10 $\bigoplus$ remplacer ZZZ ?	Pour le code de la question p	précédente, par quelles méthodes pourrait-on
something		whatever



Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
    public double re;
    public Complex(double a, double b) {
        re = a;
        im = b;
    }

    public double getRe() {
        return re;
    }

    public void setRe(double re) {
        this.re = re;
    }

    public double getIm() {
        return im;
    }

    public void setIm(double im) {
        this.im = im;
    }
}
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit





Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r\cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r\sin(\theta)$   $\theta = \tan^{-1}\frac{y}{x}$ 

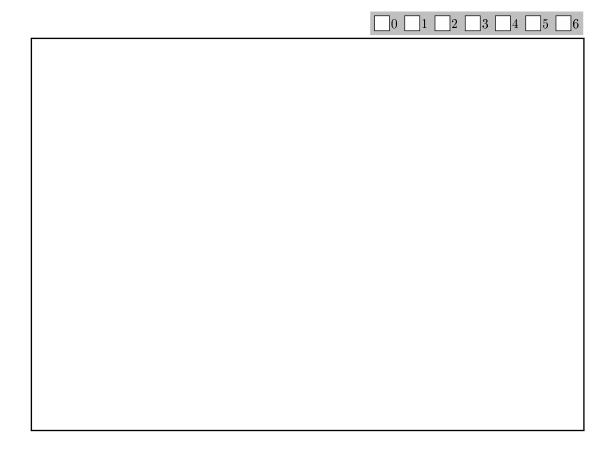
En Java ce dernier est calculé par theta = Math.atan2(y, x)

- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.





Question 13	Continuez votre réponse à la question précédente ici s	i nécessaire.	



Question 14	Continuez votre réponse à la question précédente ici si nécessaire.	0 0



Question 15 Pour la question précédente, développez la méthode Polar#equals.

	12	<u>3</u> 4

 ${\bf Question} \ \ {\bf 16} \ \bigoplus \ \ {\bf Parmi} \ {\bf les} \ {\bf affirmations} \ {\bf ci-dessous}, \ {\bf cocher} \ {\bf celles} \ {\bf qui} \ {\bf sont} \ {\bf vraies}:$ 

Une	e classe peut étendre une classe abstraite
Une	e classe peut étendre plusieurs classes
Une	e classe abstraite peut contenir des constructeurs
Une	e classe peut étendre une classe et implémenter plusieurs interfaces
Une	e interface peut contenir des méthodes privées
Une	e classe abstraite peut implémenter une interface
Une	e interface peut contenir des constructeurs



## Question $17 \oplus$ Soit le code ci-dessous :

```
@FunctionalInterface
interface ComparePerson {
    /**
    * Oreturn negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :

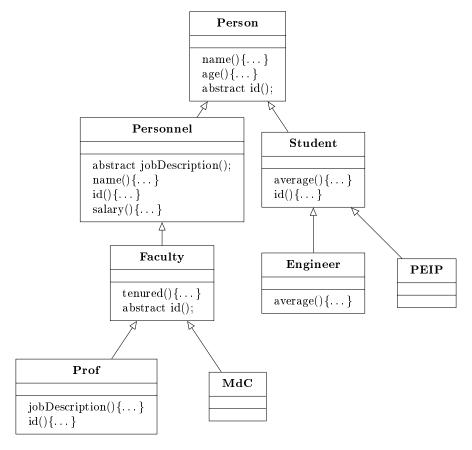
```
ComparePerson byAge = new ComparePersonByAge() {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
};

ComparePerson byAge = (p1, p2) -> p1.age - p2.age;

ComparePerson byAge
= new ComparePersonByAge::compare;
```



Question 18  $\oplus$ 



Soit l'organisation ci-dessous de Polytech'Groland :

Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

Student	Person
Prof	Engineer
PEIP	Personnel
<pre>Faculty</pre>	MdC

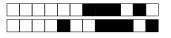
QCMTEST Nom et prénom: Introduction à la programmation orientée objet 9/01/2020 Groupe: ..... Cochez les cases en mettant une  $\times$ . Le symbole  $\bigoplus$  indique que la question peut avoir zéro, une ou plusieurs bonnes réponses. Pour ces questions, cocher une bonne réponse apporte des points positifs ; cocher une mauvaise réponse peut apporter des points négatifs. Dans tout le code, les package et les import sont censés être correctement déclarés. Toute classe est supposée être dans le bon package, dans le bon fichier, avec les bons import.Question 1 La classe donnée class ClasseDonnee {} pourrait être sous-classée par extends pourrait être instanciée par new ClasseDonnee() ClasseDonnee Question 2 La classe donnée class ClasseDonnee { private ClasseDonnee() {} pourrait être sous-classée par extends pourrait être instanciée par new ClasseDonnee ClasseDonnee() Question 3 La classe (enum) donnée enum ClasseDonnee { INSTANCE; } pourrait être pourrait être sous-classée par extends instanciée par new ClasseDonnee() ClasseDonnee Question 4 Erreur de compilation Soit le code ci-dessous ; que se passe-t-il? Affiche "Done" public class Beta { public void throwup() throws RuntimeException { Erreur d'exécution throw new RuntimeException();

public static void main(String[] args) {

Beta beta = new Beta();
beta.throwup();

}

System.out.println("Done");



## Question 5 $\bigoplus$

```
Erreur de compilation
Soit le code ci-dessous ; que se passe-t-il?
                                                               Affiche seulement "Finallied exception"
public class Gamma {
   public void throwup() throws Exception {
                                                               Erreur d'exécution
       throw new Exception();
   public static void main(String[] args) {
                                                               Affiche "Caught exception" et "Finallied
       Gamma gamma = new Gamma();
                                                               exception"
       try {
          gamma.throwup();
                                                               Affiche seulement "Caught exception"
       } catch (Exception e) {
          System.out.println("Caught exception");
       } finally {
          System.out.println("Finallied exception");
   }
}
Question 6 \bigoplus
Soit le code ci-dessous ; que se passe-t-il?
                                                               Erreur de compilation
                                                               Affiche "Done"
public class Alpha {
   public void throwup() throws Exception {
      throw new Exception();
                                                               Erreur d'exécution
   public static void main(String[] args) {
   Alpha alpha = new Alpha();
```

#### Question 7 Soit le code :

System.out.println("Done");

alpha.throwup();

}

```
interface Somethingable {
    void something();
}

interface Whateverable extends Somethingable {
    void whatever();
}

class TheThing implements Whateverable{
    PPP void something() {
        System.out.println("Somethinging");
    }

    PPP void whatever() {
        System.out.println("Whatevering");
    }
}
```

```
class Doofer {
    void doo(Somethingable sable) {
        sable.XXX();
    }

    void doo(Whateverable sable) {
        sable.YYY();
    }

    void fer(TheThing tt) {
        tt.ZZZ();
    }

    public static void main(String[] args) {
        Doofer doofer = new Doofer();
        TheThing the = new TheThing();
        doofer.doo(the);
        doofer.fer(the);
    }
}
```

Par quels niveaux d'accès pourrait-on remplacer PPP?

protected	public
package-priv	vate private
Question 8 $\bigoplus$ placer XXX ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
whatever	something



Question 9 $\bigoplus$ placer YYY ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
something	whatever
Question 10 $\bigoplus$ remplacer ZZZ ?	Pour le code de la question précédente, par quelles méthodes pourrait-on
something	whatever

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit





Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r\cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r\sin(\theta)$   $\theta = \tan^{-1}\frac{y}{x}$ 

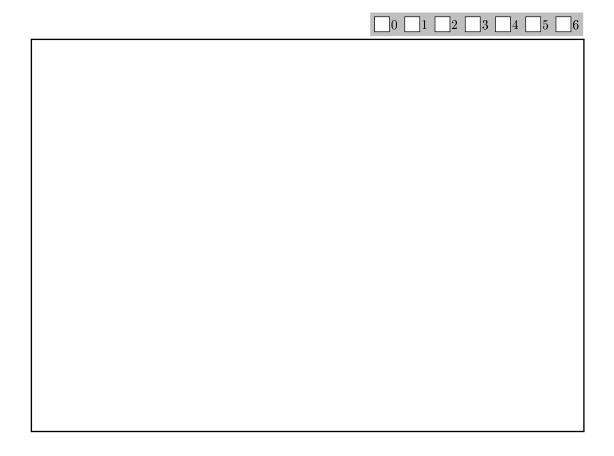
En Java ce dernier est calculé par theta = Math.atan2(y, x)

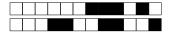
- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.





Question 13	Continuez votre réponse à la question précédente ici s	i nécessaire.	



Question 14	Continuez votre réponse à la question précédente ici si nécessaire.	
1		



Question 15 Pour la question précédente, développez la méthode Polar#equals.  $\boxed{0}$   $\boxed{1}$   $\boxed{2}$   $\boxed{3}$ Une classe peut étendre une classe abstraite Une classe peut étendre plusieurs classes Une classe abstraite peut contenir des constructeurs Une classe abstraite peut implémenter une interface Une interface peut contenir des constructeurs

Une classe peut étendre une classe et implémenter plusieurs interfaces

Une interface peut contenir des méthodes privées



## Question 17 Soit le code ci-dessous :

```
OFunctionalInterface
interface ComparePerson {
    /**
    * Oreturn negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

```
class Person {
    private int age;
    private String name;

Person(String name, int age) {
        this.age = age;
        this.name = name;
    }
}

class Main {
    public static void main(String[] args) {
        // code extract here

        System.out.print(byAge.compare(
            new Person("Fred", 23),
            new Person("Not Fred", 24)));
        System.out.print(byAge.compare(
                  new Person("Fred", 23),
                  new Person("Fred", 23));
        System.out.print(byAge.compare(
                  new Person("Not Fred", 23));
        System.out.print(byAge.compare(
                  new Person("Fred", 23),
                  new Person("Fred", 23),
                  new Person("Fred", 23);
                  new Person("Not Fred", 22)));
}
```

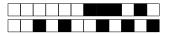
Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :

```
ComparePerson byAge = new ComparePersonByAge::compare;

ComparePerson byAge = new ComparePersonByAge();

ComparePerson byAge = (p1, p2) -> p1.age - p2.age;
```

```
ComparePerson byAge = new ComparePersonByAge() {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
};
```



# Question 18 $\bigoplus$ Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

Prof	Person
Faculty	<pre>Engineer</pre>
MdC	Personnel
Student	☐ PEIP



QCM

Introduction à la programmation
orientée objet
9/01/2020

Nom et	pı	é	n	0	m								
					٠								
$\operatorname{Groupe}$	:												

Cochez les cases en mettant une  $\times$ .

Le symbole  $\bigoplus$  indique que la question peut avoir zéro, une ou plusieurs bonnes réponses. Pour ces questions, cocher une bonne réponse apporte des points positifs ; cocher une mauvaise réponse peut apporter des points négatifs.

Dans tout le code, les package et les import sont censés être correctement déclarés. Toute classe est supposée être dans le bon package, dans le bon fichier, avec les bons import.

Question $1 \bigoplus$	La classe	donnée
------------------------	-----------	--------

abstract class ClasseDonnee {     ClasseDonnee() {} }
---

pourrait être sous-classée par extend	s 🗌 pourrait être	$instanci\'ee$	par	new
ClasseDonnee	ClasseDonnee()			

# 

		<pre>final class ClasseDonnee {    ClasseDonnee() {} }</pre>	
--	--	--	--

pourrait	${ m \hat{e}tre}$	$instanci\'ee$	par	new	pourrait	$\hat{\mathrm{e}}\mathrm{tre}$	sous-classée	par	extends
ClasseDon	nee()				ClasseDo	onnee			

# 

<pre>class ClasseDonnee {    private ClasseDonnee() {} }</pre>
--

pourrait	$\hat{\mathrm{e}}\mathrm{tre}$	${\bf sous\text{-}class\'{e}e}$	par	extends	pourrait	être	$instanci\'ee$	par	new
ClasseDo	onnee				ClasseDo	nnee()			

"Finallied

## Question 4

Soit le code ci-dessous ; que se passe-t-il ?	Erreur de compilation
<pre>public class Delta {    public void throwup() throws Exception {       throw new Exception();</pre>	Affiche "Caught exception" et "Finalli exception"
<pre>} public static void main(String[] args) {</pre>	Affiche seulement "Caught exception"
Delta delta = new Delta(); try {	Erreur d'exécution
<pre>delta.throwup(); } catch (Exception e) {     System.out.println("Caught exception");     return; } finally {     System.out.println("Finallied exception");</pre>	Affiche seulement "Finallied exception"
}	
}	
Question 5 $\bigoplus$	

Soit le code ci-dessous ; que se passe-t-il ? Erreur de compilation public class Beta { Erreur d'exécution public void throwup() throws RuntimeException { Affiche "Done" throw new RuntimeException(); public static void main(String[] args) { Beta beta = new Beta(); beta.throwup(); System.out.println("Done");

### Question 6

Soit le code ci-dessous ; que se passe-t-il ? public class Epsilon { public void throwup() throws RuntimeException { throw new RuntimeException(); public static void main(String[] args) {
 Epsilon epsilon = new Epsilon(); try { epsilon.throwup(); } catch (Exception e) { System.out.println("Caught exception"); } finally { System.out.println("Finallied exception"); }

Affiche seulement "Finallied exception"

Affiche "Caught exception" et "Finallied exception"

Erreur de compilation

Erreur d'exécution

Affiche seulement "Caught exception"



### Question 7 Soit le code :

```
interface Somethingable {
    void something();
}

interface Whateverable extends Somethingable {
    void whatever();
}

class TheThing implements Whateverable{
    PPP void something() {
        System.out.println("Somethinging");
    }

    PPP void whatever() {
        System.out.println("Whatevering");
    }
}
```

```
class Doofer {
    void doo(Somethingable sable) {
        sable.XXX();
    }

    void doo(Whateverable sable) {
        sable.YYY();
    }

    void fer(TheThing tt) {
        tt.ZZZ();
    }

    public static void main(String[] args) {
        Doofer doofer = new Doofer();
        TheThing the = new TheThing();
        doofer.doo(the);
        doofer.fer(the);
    }
}
```

Par quels niveaux d'accès pourrait-on remplacer PPP? package-private private protected public Question 8 Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer XXX? whatever something Question 9 Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer YYY? something whatever Question 10  $\bigoplus$ Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer ZZZ? something whatever

•

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit





Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r \cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r \sin(\theta)$   $\theta = \tan^{-1} \frac{y}{x}$ 

En Java ce dernier est calculé par theta = Math.atan2(y, x)

- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.





Question 13	Continuez	votre réponse à la question précédente ici si nécessaire.	



Question 14	Continuez	votre réponse à la question précédente ici si nécessaire.	



Une interface peut contenir des méthodes privées

Une classe peut étendre une classe et implémenter plusieurs interfaces

Question	15	Pour	la	question	précédente,	développez	la	${ m m\acute{e}thode}$ <code>Polar#equals</code> .		
•				•	,					
Question	16 ⊕	Parm	ıi l€	es affirma	tions ci-desso	ous, cocher c	elle	s qui sont vraies :		
Une o	classe p	eut éte	ndr	e une cla	sse abstraite					
					nir des const	ructeurs				
				e plusieu		C				
_					menter une i					
∟ ∪ne i	meriac	e peut	con	nemr des	constructeur	rs				



## Question $17 \oplus$ Soit le code ci-dessous :

```
@FunctionalInterface
interface ComparePerson {
    /**
    * Oreturn negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :



# Question 18 $\bigoplus$ Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

Person	Prof
MdC	Engineer
PEIP	Personnel
Student	$\square$ Faculty

QCMTEST Nom et prénom: Introduction à la programmation orientée objet 9/01/2020 Groupe: ..... Cochez les cases en mettant une  $\times$ . Le symbole  $\bigoplus$  indique que la question peut avoir zéro, une ou plusieurs bonnes réponses. Pour ces questions, cocher une bonne réponse apporte des points positifs ; cocher une mauvaise réponse peut apporter des points négatifs. Dans tout le code, les package et les import sont censés être correctement déclarés. Toute classe est supposée être dans le bon package, dans le bon fichier, avec les bons import.Question 1 La classe donnée class ClasseDonnee {} pourrait être sous-classée par extends pourrait être instanciée new ClasseDonnee ClasseDonnee() La classe (enum) donnée Question 2 ( enum ClasseDonnee { INSTANCE; pourrait être sous-classée par extends pourrait être instanciée ClasseDonnee ClasseDonnee() Question 3  $\bigoplus$  La classe donnée abstract class ClasseDonnee { ClasseDonnee() {} pourrait être sous-classée par extends pourrait être instanciée new ClasseDonnee ClasseDonnee() Question 4 Affiche seulement "Caught exception" Soit le code ci-dessous ; que se passe-t-il? public class Gamma { Affiche seulement "Finallied exception" public void throwup() throws Exception { Erreur d'exécution throw new Exception(); Affiche "Caught exception" et "Finallied

exception"

Erreur de compilation

public static void main(String[] args) { Gamma gamma = new Gamma();

System.out.println("Caught exception");

System.out.println("Finallied exception");

gamma.throwup();

} catch (Exception e) {

try {

} }

finally {



## Question 5 $\bigoplus$

Affiche "Done" Soit le code ci-dessous ; que se passe-t-il? public class Alpha { Erreur d'exécution public void throwup() throws Exception { Erreur de compilation throw new Exception(); public static void main(String[] args) { Alpha alpha = new Alpha(); alpha.throwup(); System.out.println("Done"); Question 6 Affiche seulement "Caught exception" Soit le code ci-dessous ; que se passe-t-il? public class Delta { Affiche seulement "Finallied exception"

public class Delta {
 public void throwup() throws Exception {
 throw new Exception();
 }
 public static void main(String[] args) {
 Delta delta = new Delta();
 try {
 delta.throwup();
 } catch (Exception e) {
 System.out.println("Caught exception");
 return;
 } finally {
 System.out.println("Finallied exception");
 }
 }
}

# Affiche "Caught exception" et "Finallied exception"

Erreur de compilation

Erreur d'exécution

### Question 7 Soit le code :

```
interface Somethingable {
    void something();
}

interface Whateverable extends Somethingable {
    void whatever();
}

class TheThing implements Whateverable{
    PPP void something() {
        System.out.println("Somethinging");
    }

    PPP void whatever() {
        System.out.println("Whatevering");
    }
}
```

```
class Doofer {
    void doo(Somethingable sable) {
        sable.XXX();
    }

    void doo(Whateverable sable) {
        sable.YYY();
    }

    void fer(TheThing tt) {
        tt.ZZZ();
    }

    public static void main(String[] args) {
        Doofer doofer = new Doofer();
        TheThing the = new TheThing();
        doofer.doo(the);
        doofer.fer(the);
    }
}
```

Par quels niveaux d'accès pourrait-on remplacer PPP?

1	ı r
public	package-private
protected	private
Question 8 $\oplus$ placer XXX ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
whatever	something

Question 9 $\bigoplus$ placer YYY ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
whatever	$\square$ something
Question 10 ⊕ remplacer ZZZ ?	Pour le code de la question précédente, par quelles méthodes pourrait-on
something	whatever

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit





Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r \cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r \sin(\theta)$   $\theta = \tan^{-1} \frac{y}{x}$ 

En Java ce dernier est calculé par theta = Math.atan2(y, x)

- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
       return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.

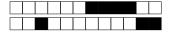




Question 13	Continuez votre réponse à la question précédente ici si nécessaire.	



Question 14	Continuez votre réponse à la question précédente ici si nécessaire.	



Une interface peut contenir des méthodes privées
Une interface peut contenir des constructeurs
Une classe peut étendre plusieurs classes

Question	15	Pour	la	question	précédente,	développez	la n	méthode Polar#equals.
Question	<b>16</b> $\oplus$	Parm	ni l€	es affirma	tions ci-desso	ous, cocher c	elle	s qui sont vraies :
Une o	classe p	eut éte	ndr	e une cla	sse abstraite			
Une o	classe a	bstraite	ере	eut conte	nir des const	ructeurs		
					sse et implér		eurs	interfaces
Une o	classe a	$\operatorname{bstrait}_{\mathbf{c}}$	е ре	eut implé	menter une i	$_{ m nterface}$		



## Question $17 \oplus$ Soit le code ci-dessous :

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :



Question 18  $\bigoplus$  Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

Faculty	Person
PEIP	Prof
Personnel	MdC
Student	Engineer

QCMTEST Nom et prénom: Introduction à la programmation orientée objet 9/01/2020 Groupe: ..... Cochez les cases en mettant une  $\times$ . Le symbole  $\bigoplus$  indique que la question peut avoir zéro, une ou plusieurs bonnes réponses. Pour ces questions, cocher une bonne réponse apporte des points positifs ; cocher une mauvaise réponse peut apporter des points négatifs. Dans tout le code, les package et les import sont censés être correctement déclarés. Toute classe est supposée être dans le bon package, dans le bon fichier, avec les bons import.Question 1 La classe donnée class ClasseDonnee {} pourrait être sous-classée par extends pourrait être instanciée new par ClasseDonnee ClasseDonnee() Question 2 La classe donnée final class ClasseDonnee { ClasseDonnee() {} pourrait être sous-classée par extends pourrait être instanciée par new ClasseDonnee ClasseDonnee() Question 3 La classe donnée abstract class ClasseDonnee { ClasseDonnee() {} pourrait être sous-classée par extends pourrait être instanciée new ClasseDonnee ClasseDonnee() Question 4 Erreur de compilation Soit le code ci-dessous ; que se passe-t-il? Erreur d'exécution public class Beta { public void throwup() throws RuntimeException { Affiche "Done" throw new RuntimeException();

public static void main(String[] args) {

Beta beta = new Beta();
beta.throwup();

}

System.out.println("Done");



Soit le code ci-dessous ; que se passe-t-il? Erreur d'exécution public class Delta { Affiche seulement "Finallied exception" public void throwup() throws Exception { Erreur de compilation throw new Exception(); public static void main(String[] args) { Affiche seulement "Caught exception" Delta delta = new Delta(); try { Affiche "Caught exception" et "Finallied delta.throwup(); exception" } catch (Exception e) { System.out.println("Caught exception"); return; } finally { System.out.println("Finallied exception"); } Question 6 Soit le code ci-dessous ; que se passe-t-il? Erreur d'exécution Affiche "Done" public class Alpha { public void throwup() throws Exception {

#### **Question 7** Soit le code :

throw new Exception();

Alpha alpha = new Alpha();
alpha.throwup();

System.out.println("Done");

public static void main(String[] args) {

```
interface Somethingable {
    void something();
}

interface Whateverable extends Somethingable {
    void whatever();
}

class TheThing implements Whateverable{
    PPP void something() {
        System.out.println("Somethinging");
    }

    PPP void whatever() {
        System.out.println("Whatevering");
    }
}
```

```
class Doofer {
    void doo(Somethingable sable) {
        sable.XXX();
}

void doo(Whateverable sable) {
        sable.YYY();
}

void fer(TheThing tt) {
        tt.ZZZ();
}

public static void main(String[] args) {
        Doofer doofer = new Doofer();
        TheThing the = new TheThing();
        doofer.doo(the);
        doofer.fer(the);
}
```

Erreur de compilation

Par quels niveaux d'accès pourrait-on remplacer PPP ?

package-private	private
protected	public

Question 8 
Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer XXX ?

whatever something

Question 9 ⊕ placer YYY ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
something	whatever
$\begin{array}{c} \mathbf{Question}  10 \ \bigoplus \\ \mathbf{remplacer} \ \mathbf{ZZZ} \ ? \end{array}$	Pour le code de la question précédente, par quelles méthodes pourrait-on
something	whatever

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

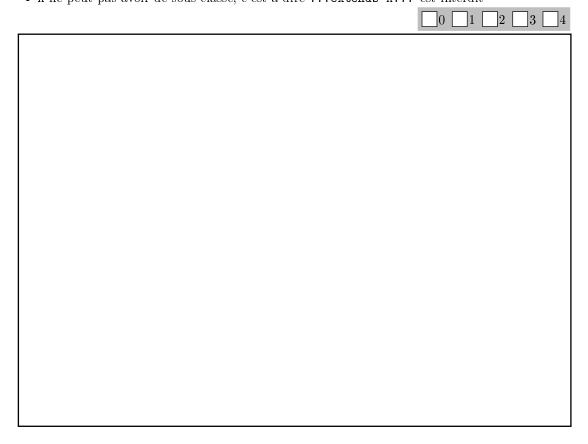
public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit



Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r \cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r \sin(\theta)$   $\theta = \tan^{-1} \frac{y}{x}$ 

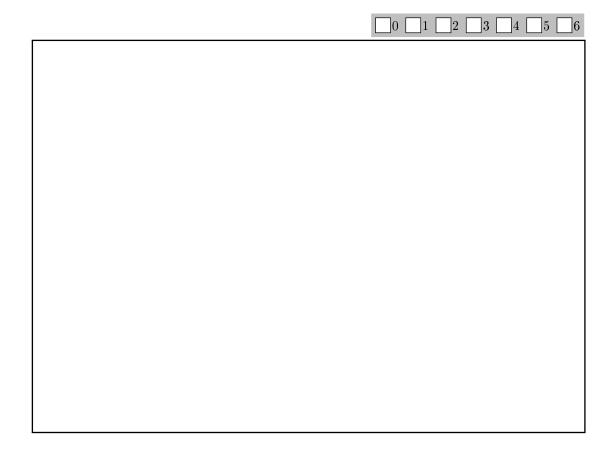
En Java ce dernier est calculé par theta = Math.atan2(y, x)

- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.





Question 13	Continuez	votre réponse	à la question	précédente i	ci si nécessaire.	0 0



Question 14	Continuez votre réponse à la question précédente ici si nécessaire.	



 $\hfill \Box$  Une classe abstraite peut implémenter une interface

Question	15	Pour	la	question	précédente,	développez	la	méthode	Polar	:#equa	ls.	
									1		3 [	$\Box$ 4
Question	<b>16</b> $\bigoplus$	Parn	ni le	es affirma	tions ci-desso	ous, cocher c	elle	s qui sont	vraies	:		
Une	classe p	eut éte	endi	re une cla	sse abstraite							
_					sse et implén		eurs	interfaces	3			
=		_			constructeur							
_					nir des consti							
				nemr des re plusieu	méthodes pr rs classes	rvees						



## Question $17 \oplus$ Soit le code ci-dessous :

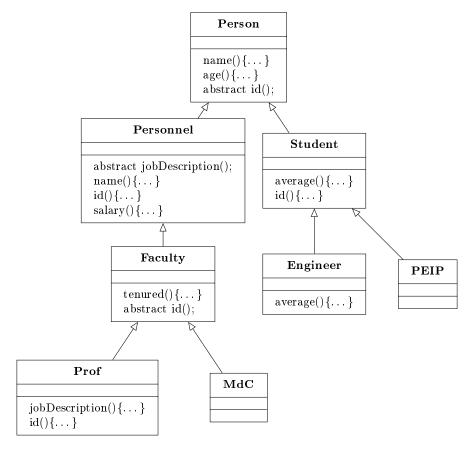
```
@FunctionalInterface
interface ComparePerson {
    /**
    * @return negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :

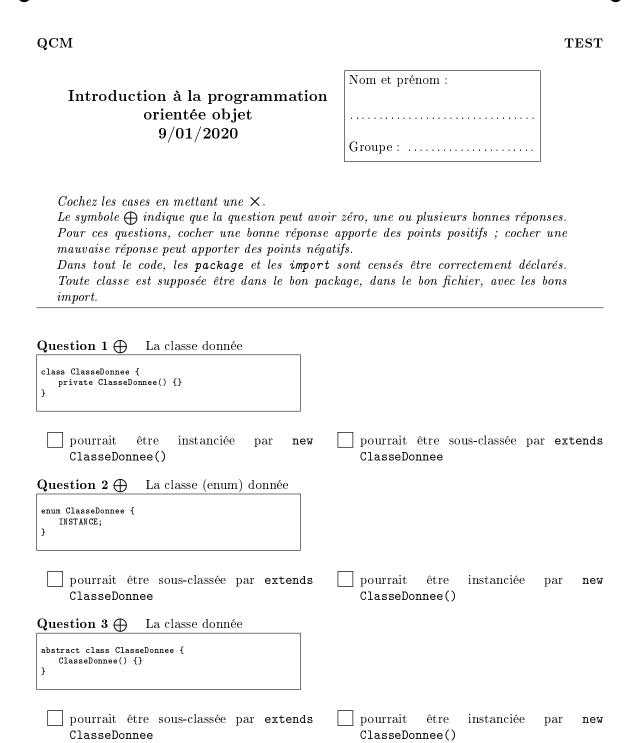


Question 18  $\bigoplus$  Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

Student	PEIP
Engineer	Personnel
Person	MdC
Prof	$oxedsymbol{\Box}$ Faculty





```
Erreur de compilation
Soit le code ci-dessous ; que se passe-t-il?
public class Gamma {
                                                             Affiche "Caught exception" et "Finallied
   public void throwup() throws Exception {
                                                             exception"
      throw new Exception();
                                                          Affiche seulement "Caught exception"
   public static void main(String[] args) {
      Gamma gamma = new Gamma();
                                                            Erreur d'exécution
      try {
         gamma.throwup();
                                                            Affiche seulement "Finallied exception"
      } catch (Exception e) {
         System.out.println("Caught exception");
      } finally {
         System.out.println("Finallied exception");
}
```

## Question 5 $\bigoplus$

```
Soit le code ci-dessous ; que se passe-t-il?
public class Epsilon {
   public void throwup() throws RuntimeException {
       throw new RuntimeException();
   public static void main(String[] args) {
    Epsilon epsilon = new Epsilon();
        try {
           epsilon.throwup();
        } catch (Exception e) {
           System.out.println("Caught exception");
       } finally {
           System.out.println("Finallied exception");
   }
}
```

# Question 6 $\bigoplus$

Soit le code ci-dessous ; que se passe-t-il? public class Beta { public void throwup() throws RuntimeException { throw new RuntimeException(); public static void main(String[] args) { Beta beta = new Beta(); beta.throwup(); System.out.println("Done"); }

Erreur	de	compilation

Affiche seulement "Finallied exception" Affiche seulement "Caught exception"

Erreur d'exécution

Affiche "Caught exception" et "Finallied exception"

Erreur d'exécution

Erreur de compilation

Affiche "Done"



### Question 7 Soit le code :

something

```
interface Somethingable {
   void something();
}

interface Whateverable extends Somethingable {
   void whatever();
}

class TheThing implements Whateverable{
   PPP void something() {
       System.out.println("Somethinging");
   }

   PPP void whatever() {
       System.out.println("Whatevering");
   }
}
```

Par quels niveaux d'accès pourrait-on remplacer PPP?

<pre>class Doofer {   void doo(Somethingable sable) {      sable.XXX();   }</pre>	
<pre>void doo(Whateverable sable) {    sable.YYY(); }</pre>	
<pre>void fer(TheThing tt) {    tt.ZZZ(); }</pre>	
<pre>public static void main(String[] args) {   Doofer doofer = new Doofer();   TheThing the = new TheThing();   doofer.doo(the);   doofer.fer(the);</pre>	
}	

protected  $\rfloor$  private public package-private Question 8  $\bigoplus$ Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer XXX? whatever something Question 9 Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer YYY? whatever something Question 10  $\bigoplus$ Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer ZZZ ?

whatever

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
}
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit



Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r\cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r\sin(\theta)$   $\theta = \tan^{-1}\frac{y}{x}$ 

En Java ce dernier est calculé par theta = Math.atan2(y, x)

- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
       return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.





Question 13	Continuez	votre réponse	à la question	précédente i	ci si nécessaire.	0 0



Question 14	Continuez	votre réponse à l	la question pré	cédente ici s	i nécessaire.	0 0



Une interface peut contenir des méthodes privées

Une classe peut étendre une classe et implémenter plusieurs interfaces

Question 15 Pour la question précédente, développez la méthode Polar#equals.  $\boxed{0}$   $\boxed{1}$   $\boxed{2}$   $\boxed{3}$ Question 16  $\bigoplus$ Parmi les affirmations ci-dessous, cocher celles qui sont vraies : Une classe peut étendre une classe abstraite Une classe abstraite peut implémenter une interface Une classe abstraite peut contenir des constructeurs Une interface peut contenir des constructeurs Une classe peut étendre plusieurs classes

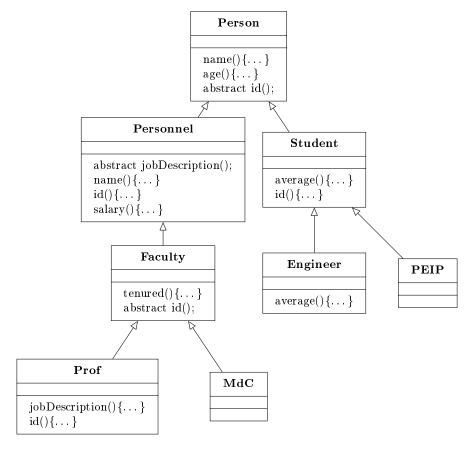


# Question $17 \oplus$ Soit le code ci-dessous :

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :



Question 18  $\bigoplus$  Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

Person	Engineer
PEIP	Personnel
Faculty	Student
MdC	Prof

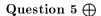
QCMTEST Nom et prénom: Introduction à la programmation orientée objet 9/01/2020 Groupe: ..... Cochez les cases en mettant une  $\times$ . Le symbole  $\bigoplus$  indique que la question peut avoir zéro, une ou plusieurs bonnes réponses. Pour ces questions, cocher une bonne réponse apporte des points positifs ; cocher une mauvaise réponse peut apporter des points négatifs. Dans tout le code, les package et les import sont censés être correctement déclarés. Toute classe est supposée être dans le bon package, dans le bon fichier, avec les bons import.Question 1 La classe donnée class ClasseDonnee {} pourrait être sous-classée par extends pourrait être instanciée new par ClasseDonnee ClasseDonnee() Question 2 La classe donnée final class ClasseDonnee { ClasseDonnee() {} pourrait être sous-classée par extends pourrait être instanciée new par ClasseDonnee ClasseDonnee() Question 3  $\bigoplus$ La classe donnée class ClasseDonnee { private ClasseDonnee() {} pourrait  $\hat{\mathrm{e}}\mathrm{tre}$ pourrait être sous-classée par extends instanciée par new ClasseDonnee() ClasseDonnee Question 4 Erreur d'exécution Soit le code ci-dessous ; que se passe-t-il? Erreur de compilation public class Alpha { public void throwup() throws Exception { throw new Exception(); Affiche "Done"

public static void main(String[] args) {

Alpha alpha = new Alpha();
alpha.throwup();

System.out.println("Done");

}



```
Affiche "Caught exception" et "Finallied
Soit le code ci-dessous ; que se passe-t-il?
                                                            exception"
public class Delta {
   public void throwup() throws Exception {
                                                            Erreur de compilation
      throw new Exception();
                                                            Affiche seulement "Caught exception"
   public static void main(String[] args) {
      Delta delta = new Delta();
                                                            Erreur d'exécution
      try {
          delta.throwup();
                                                            Affiche seulement "Finallied exception"
      } catch (Exception e) {
          System.out.println("Caught exception");
          return:
      } finally {
          System.out.println("Finallied exception");
   }
}
Question 6
                                                          Affiche "Caught exception" et "Finallied
Soit le code ci-dessous ; que se passe-t-il?
                                                            exception"
public class Gamma {
   public void throwup() throws Exception {
                                                            Erreur d'exécution
```

```
throw new Exception();
}
public static void main(String[] args) {
    Gamma gamma = new Gamma();
    try {
        gamma.throwup();
    } catch (Exception e) {
            System.out.println("Caught exception");
    } finally {
            System.out.println("Finallied exception");
    }
}
```

# Question 7 Soit le code :

```
interface Somethingable {
    void something();
}

interface Whateverable extends Somethingable {
    void whatever();
}

class TheThing implements Whateverable{
    PPP void something() {
        System.out.println("Somethinging");
    }

    PPP void whatever() {
        System.out.println("Whatevering");
    }
}
```

```
class Doofer {
    void doo(Somethingable sable) {
        sable.XXX();
    }

    void doo(Whateverable sable) {
        sable.YYY();
    }

    void fer(TheThing tt) {
        tt.ZZZ();
    }

    public static void main(String[] args) {
        Doofer doofer = new Doofer();
        TheThing the = new TheThing();
        doofer.doo(the);
        doofer.fer(the);
    }
}
```

Erreur de compilation

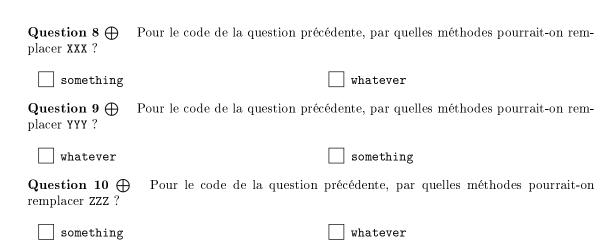
Affiche seulement "Caught exception"

Affiche seulement "Finallied exception"

Par quels niveaux d'accès pourrait-on remplacer PPP?

```
□ public
□ package-private

□ private
□ protected
```



Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
    public double re;
    public Complex(double a, double b) {
        re = a;
        im = b;
    }

    public double getRe() {
        return re;
    }

    public void setRe(double re) {
        this.re = re;
    }

    public double getIm() {
        return im;
    }

    public void setIm(double im) {
        this.im = im;
    }
}
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit



Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r\cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r\sin(\theta)$   $\theta = \tan^{-1}\frac{y}{x}$ 

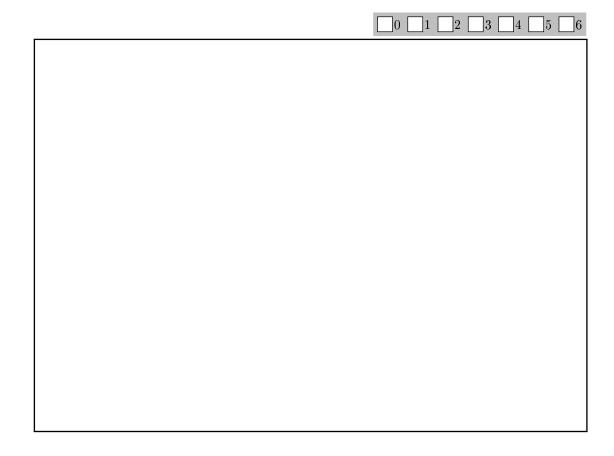
En Java ce dernier est calculé par theta = Math.atan2(y, x)

- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
       return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.





Question 13	Continuez votre réponse à la question précédente ici si nécessaire.	



Question 14	Continuez	votre réponse à la question précédente ici si nécessaire.	



Une interface peut contenir des constructeurs

Une classe peut étendre une classe et implémenter plusieurs interfaces

Question 15 Pour la question précédente, développez la méthode Polar#equals.  $\boxed{0}$   $\boxed{1}$   $\boxed{2}$   $\boxed{3}$ Une interface peut contenir des méthodes privées Une classe peut étendre plusieurs classes Une classe peut étendre une classe abstraite Une classe abstraite peut implémenter une interface Une classe abstraite peut contenir des constructeurs



## Question $17 \oplus$ Soit le code ci-dessous :

```
@FunctionalInterface
interface ComparePerson {
    /**
    * Oreturn negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :

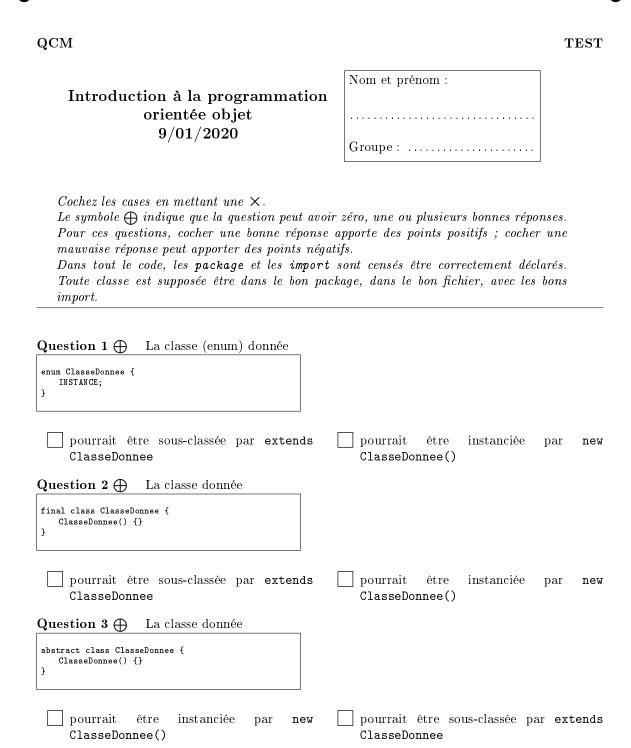


Question 18  $\bigoplus$  Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

Prof	$\Box$ Faculty
PEIP	Engineer
Student	Personnel
Person	MdC



## Question 4 $\bigoplus$

Soit le code ci-dessous ; que se passe-t-il !	Err€
<pre>public class Epsilon {     public void throwup() throws RuntimeException {</pre>	Erre
throw new RuntimeException(); }	Affic
public static void main(String[] args) {     Epsilon epsilon = new Epsilon();	Affic
try {	exce
<pre>epsilon.throwup(); } catch (Exception e) {</pre>	Affic
<pre>System.out.println("Caught exception"); } finally {</pre>	
<pre>System.out.println("Finallied exception"); }</pre>	
}	

# Erreur de compilation

Erreur d'exécution

Affiche seulement "Finallied exception"

Affiche "Caught exception" et "Finallied exception"

Affiche seulement "Caught exception"

# Question $5 \bigoplus$

```
Soit le code ci-dessous ; que se passe-t-il ?

public class Delta {
    public void throwup() throws Exception {
        throw new Exception();
    }

    public static void main(String[] args) {
        Delta delta = new Delta();
        try {
            delta.throwup();
    } catch (Exception e) {
            System.out.println("Caught exception");
            return;
    } finally {
            System.out.println("Finallied exception");
        }
}
```

# Affiche seulement "Finallied exception"

- Affiche "Caught exception" et "Finallied exception"
- Affiche seulement "Caught exception"
- Erreur d'exécution
- Erreur de compilation

## Question 6 $\bigoplus$

Soit le code ci-dessous ; que se passe-t-il ?
public class Beta {
 public void throwup() throws RuntimeException {
 throw new RuntimeException();
 }
 public static void main(String[] args) {
 Beta beta = new Beta();
 beta.throwup();
 System.out.println("Done");
 }
}

Erreur de compilation

Affiche "Done"

Erreur d'exécution



### Question 7 Soit le code :

```
interface Somethingable {
   void something();
}

interface Whateverable extends Somethingable {
   void whatever();
}

class TheThing implements Whateverable{
   PPP void something() {
       System.out.println("Somethinging");
   }

   PPP void whatever() {
       System.out.println("Whatevering");
   }
}
```

```
class Doofer {
   void doo(Somethingable sable) {
      sable.XXX();
   }

   void doo(Whateverable sable) {
      sable.YYY();
   }

   void fer(TheThing tt) {
      tt.ZZZ();
   }

   public static void main(String[] args) {
      Doofer doofer = new Doofer();
      TheThing the = new TheThing();
      doofer.doo(the);
      doofer.fer(the);
   }
}
```

Par quels niveaux d'accès pourrait-on remplacer PPP? public private package-private protected Question 8 Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer XXX? whatever something Question 9 Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer YYY? whatever something Question 10  $\bigoplus$ Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer ZZZ? whatever something

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit





Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r\cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r\sin(\theta)$   $\theta = \tan^{-1}\frac{y}{x}$ 

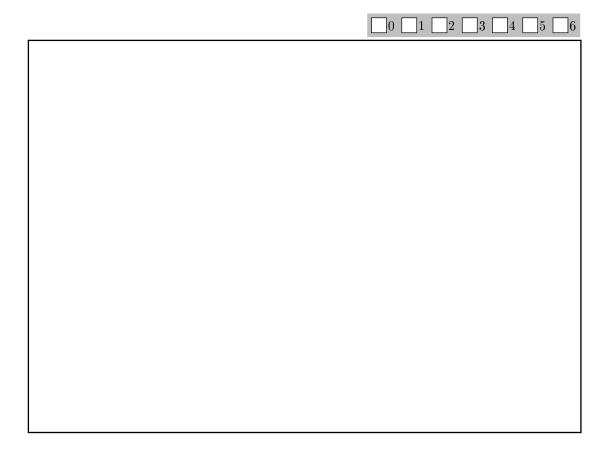
En Java ce dernier est calculé par theta = Math.atan2(y, x)

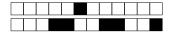
- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.

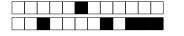




Question 13	Continuez	votre réponse	à la question	précédente i	ci si nécessaire.	0 0



Question 14	Continuez	votre réponse	à la question	précédente i	ci si nécessaire.	



Une classe peut étendre une classe abstraite

Une classe abstraite peut implémenter une interface

Question 15 Pour la question précédente, développez la méthode Polar#equals.  $\boxed{0}$   $\boxed{1}$   $\boxed{2}$   $\boxed{3}$ Question 16  $\bigoplus$ Parmi les affirmations ci-dessous, cocher celles qui sont vraies : Une interface peut contenir des constructeurs Une classe peut étendre plusieurs classes Une classe abstraite peut contenir des constructeurs Une interface peut contenir des méthodes privées Une classe peut étendre une classe et implémenter plusieurs interfaces



## Question $17 \oplus$ Soit le code ci-dessous :

```
@FunctionalInterface
interface ComparePerson {
    /**
    * Oreturn negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

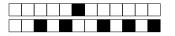
Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :

```
ComparePerson byAge = new ComparePersonByAge() {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
};

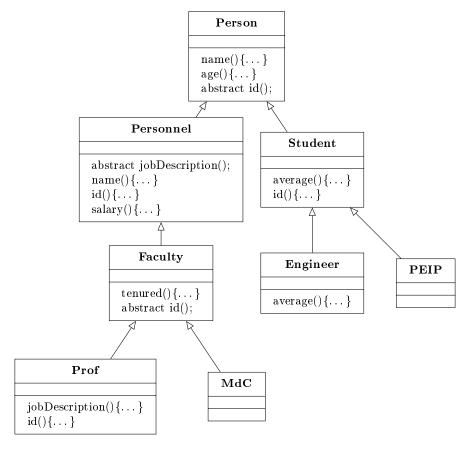
ComparePerson byAge
    = new ComparePersonByAge::compare;

ComparePerson byAge = (p1, p2) -> p1.age - p2.age;

ComparePerson byAge = (p1, p2) -> p1.age - p2.age;
```

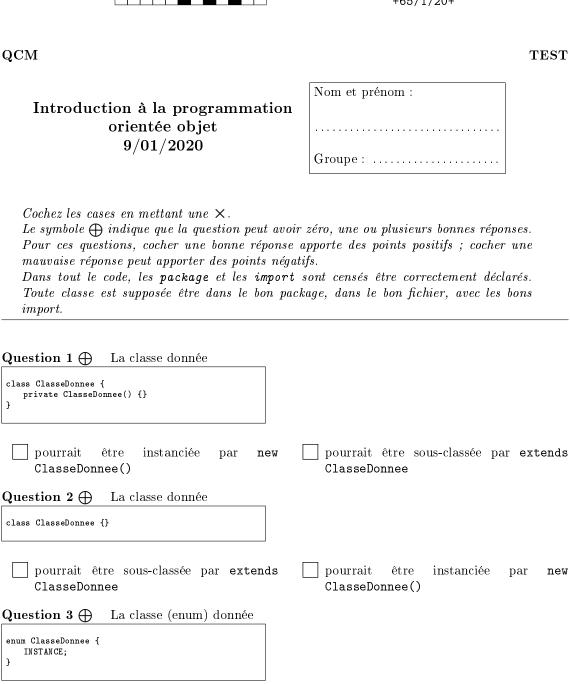


Question 18  $\bigoplus$  Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

PEIP	Personnel
Faculty	Prof
Engineer	Student
Person	MdC



ClasseDonnee

Soit le code ci-dessous ; que se passe-t-il ?
public class Alpha {
 public void throwup() throws Exception {
 throw new Exception();
 }
 public static void main(String[] args) {
 Alpha alpha = new Alpha();
 alpha.throwup();
 System.out.println("Done");

pourrait être sous-classée par extends

Erreur d'exécution

ClasseDonnee()

pourrait

Erreur de compilation

être

instanciée

new

Affiche "Done"



## Question 5 $\bigoplus$

```
Affiche seulement "Caught exception"
Soit le code ci-dessous ; que se passe-t-il?
public class Gamma {
                                                             Affiche seulement "Finallied exception"
   public void throwup() throws Exception {
                                                             Affiche "Caught exception" et "Finallied
      throw new Exception();
   public static void main(String[] args) {
      Gamma gamma = new Gamma();
                                                             Erreur de compilation
      try {
          gamma.throwup();
                                                             Erreur d'exécution
      } catch (Exception e) {
         System.out.println("Caught exception");
      } finally {
          System.out.println("Finallied exception");
   }
}
```

## Question 6 $\bigoplus$

```
Soit le code ci-dessous ; que se passe-t-il ?

public class Epsilon {
    public void throwup() throws RuntimeException {
        throw new RuntimeException();
    }
    public static void main(String[] args) {
        Epsilon epsilon = new Epsilon();
        try {
            epsilon.throwup();
      } catch (Exception e) {
            System.out.println("Caught exception");
      } finally {
            System.out.println("Finallied exception");
      }
}
```

# Affiche seulement "Finallied exception"

Affiche seulement "Caught exception"

Erreur d'exécution

Affiche "Caught exception" et "Finallied exception"

Erreur de compilation

#### Question 7 Soit le code :

```
interface Somethingable {
    void something();
}

interface Whateverable extends Somethingable {
    void whatever();
}

class TheThing implements Whateverable{
    PPP void something() {
        System.out.println("Somethinging");
    }

    PPP void whatever() {
        System.out.println("Whatevering");
    }
}
```

```
class Doofer {
    void doo(Somethingable sable) {
        sable.XXX();
    }

    void doo(Whateverable sable) {
        sable.YYY();
    }

    void fer(TheThing tt) {
        tt.ZZZ();
    }

    public static void main(String[] args) {
        Doofer doofer = new Doofer();
        TheThing the = new TheThing();
        doofer.doo(the);
        doofer.fer(the);
    }
}
```

Par quels niveaux d'accès pourrait-on remplacer PPP?

□ public
□ private

□ protected
□ package-private



Question 8 $\bigoplus$ placer XXX ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
something	whatever
Question 9 $\bigoplus$ placer YYY ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
whatever	$\square$ something
Question 10 $\bigoplus$ remplacer ZZZ ?	Pour le code de la question précédente, par quelles méthodes pourrait-on
something	whatever



Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
    public double re;
    public Complex(double a, double b) {
        re = a;
        im = b;
    }

    public double getRe() {
        return re;
    }

    public void setRe(double re) {
        this.re = re;
    }

    public double getIm() {
        return im;
    }

    public void setIm(double im) {
        this.im = im;
    }
}
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit





Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r\cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r\sin(\theta)$   $\theta = \tan^{-1}\frac{y}{x}$ 

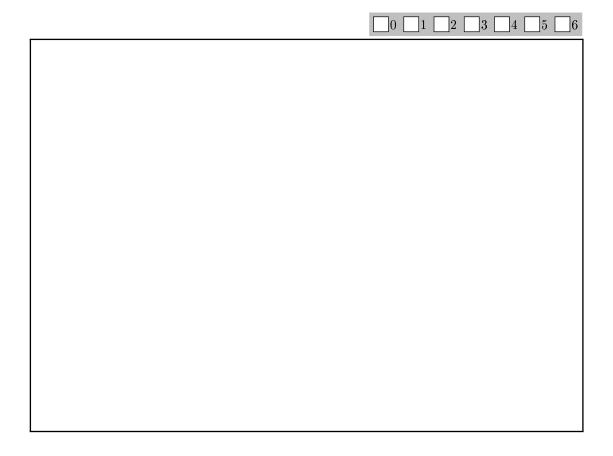
En Java ce dernier est calculé par theta = Math.atan2(y, x)

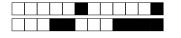
- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.

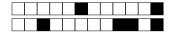




Question 13	Continuez	votre réponse	à la question	précédente i	ci si nécessaire.	0 0



Question 14	Continuez	votre réponse	à la question	précédente	ici si nécessaire.	



Question	<b>15</b>	Pour	la	question	précédente,	${\rm d\'eveloppez}$	la	$m\acute{e}thode$	Polar#equals.		
									$\bigcap_1$	$\neg_2$ $\lceil$	3

	L	]0 _	<u>_</u> 1	$\rfloor 2 \  $	<u></u> 3 ∟	$\rfloor 4$

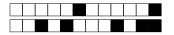
 ${\bf Question} \ {\bf 16} \ \bigoplus \quad {\bf Parmi \ les \ affirmations \ ci-dessous, \ cocher \ celles \ qui \ sont \ vraies: }$ 

Une interface peut contenir des constructeurs
Une classe peut étendre une classe abstraite
Une classe peut étendre plusieurs classes
Une classe peut étendre une classe et implémenter plusieurs interfaces
Une classe abstraite peut contenir des constructeurs
Une classe abstraite peut implémenter une interface
Une interface peut contenir des méthodes privées



## Question $17 \oplus$ Soit le code ci-dessous :

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :

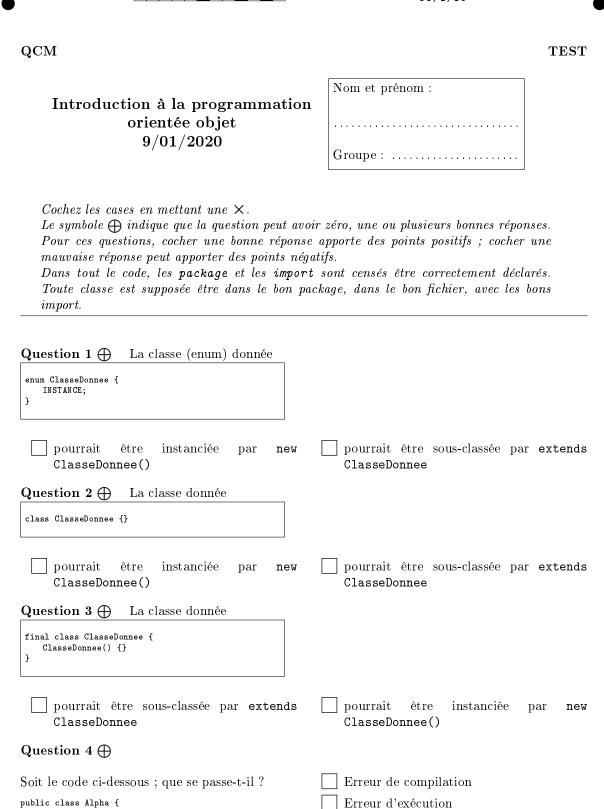


### Question 18 $\bigoplus$ Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

MdC	Personnel
Student	Person
Engineer	$oxedsymbol{\Box}$ Faculty
PEIP	Prof



Affiche "Done"

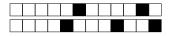
public void throwup() throws Exception {

public static void main(String[] args) {

throw new Exception();

Alpha alpha = new Alpha();
alpha.throwup();

System.out.println("Done");



#### Question 5 $\bigoplus$

Affiche "Done" Soit le code ci-dessous ; que se passe-t-il? public class Beta { Erreur d'exécution public void throwup() throws RuntimeException { Erreur de compilation throw new RuntimeException(); public static void main(String[] args) { Beta beta = new Beta(); beta.throwup(); System.out.println("Done"); } Question 6 Soit le code ci-dessous ; que se passe-t-il? Erreur de compilation public class Epsilon { Affiche "Caught exception" et "Finallied public void throwup() throws RuntimeException { exception" throw new RuntimeException(); Affiche seulement "Caught exception" public static void main(String[] args) { Epsilon epsilon = new Epsilon(); Affiche seulement "Finallied exception" try { epsilon.throwup(); Erreur d'exécution } catch (Exception e) { System.out.println("Caught exception"); } finally { System.out.println("Finallied exception");

#### Question 7 Soit le code :

}

```
interface Somethingable {
    void something();
}

interface Whateverable extends Somethingable {
    void whatever();
}

class TheThing implements Whateverable{
    PPP void something() {
        System.out.println("Somethinging");
    }

    PPP void whatever() {
        System.out.println("Whatevering");
    }
}
```

```
class Doofer {
    void doo(Somethingable sable) {
        sable.XXX();
    }

    void doo(Whateverable sable) {
        sable.YYY();
    }

    void fer(TheThing tt) {
        tt.ZZZ();
    }

    public static void main(String[] args) {
        Doofer doofer = new Doofer();
        TheThing the = new TheThing();
        doofer.doo(the);
        doofer.fer(the);
    }
}
```

Par quels niveaux d'accès pourrait-on remplacer PPP?

$\square$ private	protected
public	package-private
Question 8 $\bigoplus$ placer XXX ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
something	whatever

+66/3/8+

Question 9 $\bigoplus$ placer YYY ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
whatever	something
Question 10 $\bigoplus$ remplacer ZZZ ?	Pour le code de la question précédente, par quelles méthodes pourrait-or
something	whatever

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

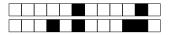
@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit





Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r \cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r \sin(\theta)$   $\theta = \tan^{-1} \frac{y}{x}$ 

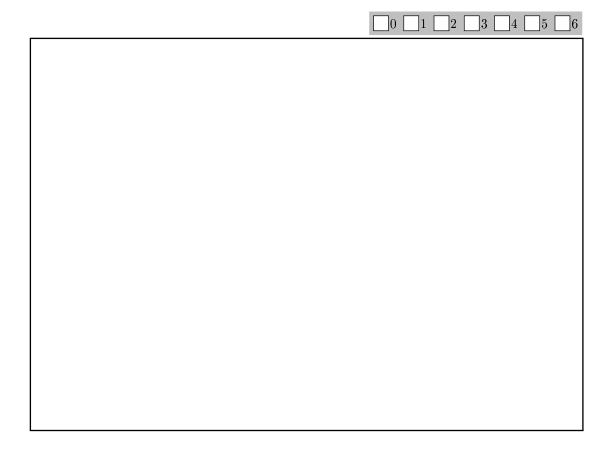
En Java ce dernier est calculé par theta = Math.atan2(y, x)

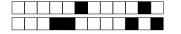
- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

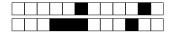
Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.

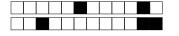




Question 13	Continuez votre rép	oonse à la questior	n précédente ici si n	écessaire.	0 0

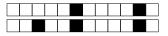


Question 14	Continuez	votre réponse	à la question	précédente	ici si nécessaire	e. 0 0



Une classe abstraite peut contenir des constructeurs

Question	15	Pour	la	question	précédente,	développez	la :	méthode	Pol	ar#eqı	ıals.	
									0	$ 1 \square 2$		
${f Question}$	16 ⊕	Parn	ni le	es affirma	tions ci-desso	ous, cocher c	elles	qui sont	vrai	es :		
Une o	classe ab	strait	ер	eut implé	menter une i	$_{ m nterface}$						
Une o	classe pe	eut éte	endi	re plusieu	rs classes							
Une o	classe pe	eut éte	endi	re une cla	sse abstraite							
Une i	nterface	peut	cor	ntenir des	constructeur	's						
Une o	classe pe	eut éte	endi	re une cla	sse et implén	nenter plusie	eurs	interface	S			
Une i		peut	cor	ntenir des	méthodes pr	ivées						



#### Question $17 \oplus$ Soit le code ci-dessous :

```
@FunctionalInterface
interface ComparePerson {
    /**
    * Oreturn negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

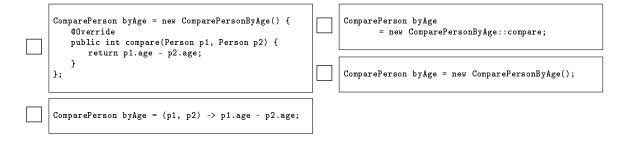
```
class Person {
    private int age;
    private String name;

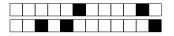
Person(String name, int age) {
        this.age = age;
        this.name = name;
    }
}

class Main {
    public static void main(String[] args) {
        // code extract here

        System.out.print(byAge.compare(
            new Person("Fred", 23),
            new Person("Not Fred", 24)));
        System.out.print(byAge.compare(
            new Person("Fred", 23),
            new Person("Not Fred", 23));
        System.out.print(byAge.compare(
            new Person("Not Fred", 23));
        System.out.print(byAge.compare(
            new Person("Fred", 23),
            new Person("Fred", 23),
            new Person("Not Fred", 22)));
    }
}
```

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :





### Question 18 $\bigoplus$ Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

PEIP	Personnel
MdC	<pre>Engineer</pre>
Student	Person
Faculty	Prof

QCMTEST Nom et prénom: Introduction à la programmation orientée objet 9/01/2020Groupe: ..... Cochez les cases en mettant une  $\times$ . Le symbole  $\bigoplus$  indique que la question peut avoir zéro, une ou plusieurs bonnes réponses. Pour ces questions, cocher une bonne réponse apporte des points positifs ; cocher une mauvaise réponse peut apporter des points négatifs. Dans tout le code, les package et les import sont censés être correctement déclarés. Toute classe est supposée être dans le bon package, dans le bon fichier, avec les bons import.Question 1 La classe donnée class ClasseDonnee { private ClasseDonnee() {} pourrait pourrait être sous-classée par extends être instanciée new ClasseDonnee ClasseDonnee() Question 2 La classe donnée abstract class ClasseDonnee { ClasseDonnee() {} pourrait être sous-classée par extends pourrait être instanciée par new ClasseDonnee ClasseDonnee() Question 3 ( La classe donnée final class ClasseDonnee { ClasseDonnee() {} pourrait être sous-classée par extends pourrait être instanciée par new ClasseDonnee ClasseDonnee()

#### Question 4 $\bigoplus$

Soit le code ci-dessous ; que se passe-t-il ?	Affiche "Caught exception" et "Finallied
<pre>public class Delta {     public void throwup() throws Exception {         throw new Exception();     }     public static void main(String[] args) {         Delta delta = new Delta();         try {             delta.throwup();         } catch (Exception e) {             System.out.println("Caught exception");             return;         } finally {                 System.out.println("Finallied exception");         }     } }</pre>	exception"  Affiche seulement "Caught exception"  Erreur de compilation  Erreur d'exécution  Affiche seulement "Finallied exception"
Question 5 $\bigoplus$	
Soit le code ci-dessous ; que se passe-t-il ?	Erreur de compilation

# Question 6 $\bigoplus$

}

public class Gamma {

public void throwup() throws Exception {

public static void main(String[] args) {
 Gamma gamma = new Gamma();
 try {
 gamma.throwup();

System.out.println("Caught exception");
} finally {
 System.out.println("Finallied exception");

throw new Exception();

} catch (Exception e) {

Soit le code ci-dessous ; que se passe-t-il ?

public class Epsilon {
 public void throwup() throws RuntimeException {
 throw new RuntimeException();
 }
 public static void main(String[] args) {
 Epsilon epsilon = new Epsilon();
 try {
 epsilon.throwup();
 } catch (Exception e) {
 System.out.println("Caught exception");
 } finally {
 System.out.println("Finallied exception");
 }
 }
}

Affiche "Caugh	ıt e	xcepti	on"	et	"Finallie
exception"					
Erreur d'exécu	ion	Į.			

Affiche seulement "Caught exception"

Affiche seulement "Finallied exception"

Affiche seulement "Caught exception"

Affiche "Caught exception" et "Finallied exception"

Erreur de compilation
Erreur d'exécution

Affiche seulement "Finallied exception"



#### **Question 7** Soit le code :

```
interface Somethingable {
   void something();
}

interface Whateverable extends Somethingable {
   void whatever();
}

class TheThing implements Whateverable{
   PPP void something() {
       System.out.println("Somethinging");
   }

   PPP void whatever() {
       System.out.println("Whatevering");
   }
}
```

```
class Doofer {
    void doo(Somethingable sable) {
        sable.XXX();
    }

    void doo(Whateverable sable) {
        sable.YYY();
    }

    void fer(TheThing tt) {
        tt.ZZZ();
    }

    public static void main(String[] args) {
        Doofer doofer = new Doofer();
        TheThing the = new TheThing();
        doofer.doo(the);
        doofer.fer(the);
    }
}
```

Par quels niveaux d'accès pourrait-on remplacer PPP? package-private public protected private Question 8 Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer XXX? something whatever Question 9 Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer YYY? whatever something Question 10  $\bigoplus$ Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer ZZZ? whatever something

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

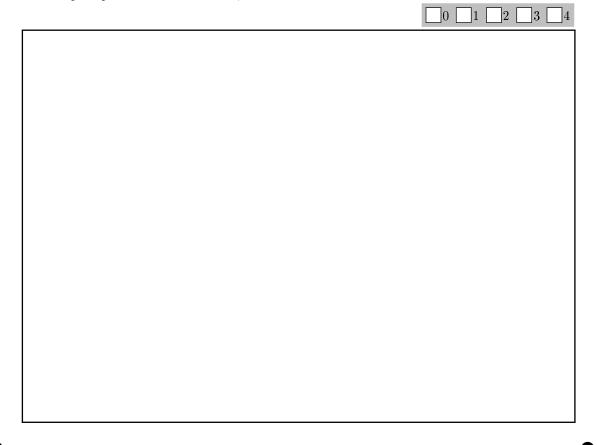
public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit





Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r \cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r \sin(\theta)$   $\theta = \tan^{-1} \frac{y}{x}$ 

En Java ce dernier est calculé par theta = Math.atan2(y, x)

- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.





Question	13	Continuez	votre	réponse	à la qu	estion	précédent	e ici s	i nécessaire.	



Question 14	Continuez votre réponse à la question précédente ici si nécessaire.	
1		



Une interface peut contenir des méthodes privées
Une classe abstraite peut implémenter une interface

Une classe peut étendre une classe et implémenter plusieurs interfaces

Question 15 Pour la question précédente, développez la méthode Polar#equals.  $\boxed{0}$   $\boxed{1}$   $\boxed{2}$   $\boxed{3}$ Question 16  $\bigoplus$ Parmi les affirmations ci-dessous, cocher celles qui sont vraies : Une interface peut contenir des constructeurs Une classe peut étendre plusieurs classes Une classe abstraite peut contenir des constructeurs Une classe peut étendre une classe abstraite



#### Question $17 \oplus$ Soit le code ci-dessous :

```
@FunctionalInterface
interface ComparePerson {
    /**
    * Oreturn negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :

```
ComparePerson byAge = (p1, p2) -> p1.age - p2.age;

ComparePerson byAge = new ComparePersonByAge() {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
};

ComparePerson byAge = new ComparePersonByAge();
```



### Question 18 $\bigoplus$ Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

Personnel	Engineer
MdC	$\square$ Faculty
Prof	Person
Student	PEIP

QCMTEST Nom et prénom: Introduction à la programmation orientée objet 9/01/2020 Groupe: ..... Cochez les cases en mettant une  $\times$ . Le symbole  $\bigoplus$  indique que la question peut avoir zéro, une ou plusieurs bonnes réponses. Pour ces questions, cocher une bonne réponse apporte des points positifs ; cocher une mauvaise réponse peut apporter des points négatifs. Dans tout le code, les package et les import sont censés être correctement déclarés. Toute classe est supposée être dans le bon package, dans le bon fichier, avec les bons import.Question 1 ( La classe (enum) donnée enum ClasseDonnee { INSTANCE; pourrait être sous-classée par extends pourrait être instanciée par new ClasseDonnee() ClasseDonnee Question 2 La classe donnée class ClasseDonnee {} pourrait être sous-classée par extends pourrait être instanciée par new ClasseDonnee ClasseDonnee() Question 3 La classe donnée class ClasseDonnee { private ClasseDonnee() {} pourrait être sous-classée par extends pourrait être instanciée new ClasseDonnee ClasseDonnee() Question 4 Erreur de compilation Soit le code ci-dessous ; que se passe-t-il?

public class Alpha {

public void throwup() throws Exception {

public static void main(String[] args) {

throw new Exception();

Alpha alpha = new Alpha();
alpha.throwup();

System.out.println("Done");

Erreur d'exécution

Affiche "Done"



#### Question 5 $\bigoplus$

Soit le code ci-dessous ; que se passe-t-il? Erreur d'exécution public class Beta { Affiche "Done" public void throwup() throws RuntimeException { Erreur de compilation throw new RuntimeException(); public static void main(String[] args) { Beta beta = new Beta(); beta.throwup(); System.out.println("Done"); }

#### Question 6

Soit le code ci-dessous ; que se passe-t-il? public class Delta { public void throwup() throws Exception { throw new Exception(); public static void main(String[] args) { Delta delta = new Delta(); try { delta.throwup(); } catch (Exception e) { System.out.println("Caught exception"); return; } finally { System.out.println("Finallied exception"); } }

# Affiche seulement "Caught exception" Affiche seulement "Finallied exception" Erreur d'exécution Erreur de compilation

Affiche "Caught exception" et "Finallied exception"

#### Question 7 Soit le code :

```
interface Somethingable {
   void something();
interface Whateverable extends Somethingable {
   void whatever();
class The Thing implements Whateverable {
   PPP void something() {
       System.out.println("Somethinging");
   PPP void whatever() {
       System.out.println("Whatevering");
}
```

```
class Doofer {
   void doo(Somethingable sable) {
       sable.XXX();
    void doo(Whateverable sable) {
        sable.YYY();
   void fer(TheThing tt) {
        tt.ZZZ();
   public static void main(String[] args) {
       Doofer doofer = new Doofer();
TheThing the = new TheThing();
        doofer.doo(the);
        doofer.fer(the);
}
```

Par quels niveaux d'accès pourrait-on remplacer PPP?

private protected public package-private Question 8 Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer XXX? whatever something

+68/3/48+

Question 9 $\bigoplus$ placer YYY ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
something	whatever
Question 10 ⊕ remplacer ZZZ ?	Pour le code de la question précédente, par quelles méthodes pourrait-on
something	whatever

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit



Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r \cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r \sin(\theta)$   $\theta = \tan^{-1} \frac{y}{x}$ 

En Java ce dernier est calculé par theta = Math.atan2(y, x)

- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.





Question 13	Continuez votre réponse à la question précédente ici si nécessaire.	0 0



Question 14	Continuez votre réponse à la question précédente ici si nécessaire.	
1		



Question 15 Pour la question précédente, développez la méthode Polar#equals.  $\boxed{0}$   $\boxed{1}$   $\boxed{2}$   $\boxed{3}$ Parmi les affirmations ci-dessous, cocher celles qui sont vraies : Question 16  $\bigoplus$ Une classe peut étendre une classe et implémenter plusieurs interfaces Une interface peut contenir des méthodes privées Une classe peut étendre une classe abstraite Une classe peut étendre plusieurs classes Une interface peut contenir des constructeurs Une classe abstraite peut contenir des constructeurs

Une classe abstraite peut implémenter une interface



#### Question $17 \oplus$ Soit le code ci-dessous :

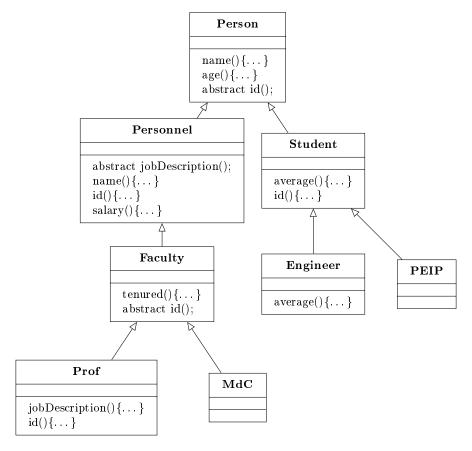
```
@FunctionalInterface
interface ComparePerson {
    /**
    * @return negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :

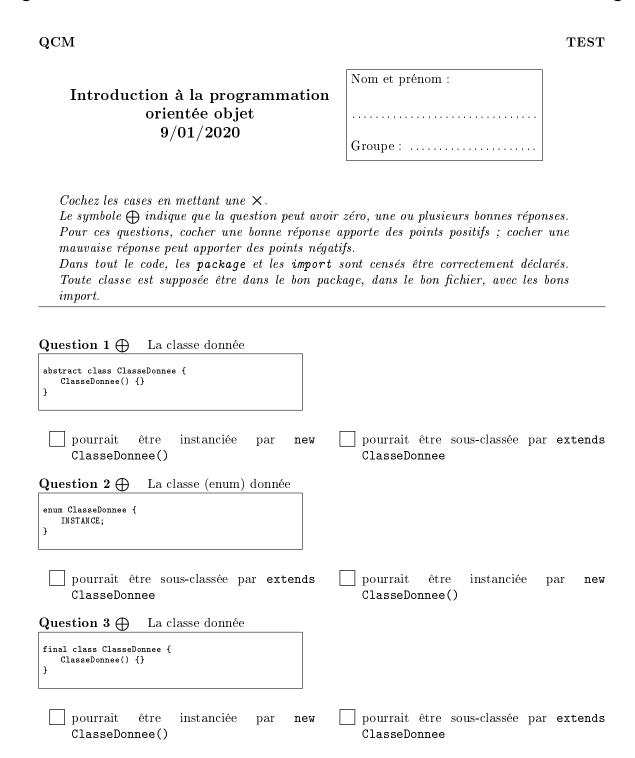


Question 18  $\bigoplus$  Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

Student	Prof
PEIP	Person
Faculty	Personnel
MdC	Engineer



et "Finallied

exception"

#### Question 4

Soit le code ci-dessous ; que se passe-t-il ?	Erreur d'exécution		
<pre>public class Gamma {     public void throwup() throws Exception {         throw new Exception();     }     public static void main(String[] args) {         Gamma gamma = new Gamma();         try {             gamma.throwup();       } catch (Exception e) {             System.out.println("Caught exception");       } finally {             System.out.println("Finallied exception");       }     } }</pre>	Affiche "Caught exception" et "Final exception"  Erreur de compilation  Affiche seulement "Caught exception"  Affiche seulement "Finallied exception"		
Question 5 $\bigoplus$			

Soit le code ci-dessous ; que se passe-t-il ? Erreur de compilation Affiche "Done" public class Alpha { public void throwup() throws Exception { throw new Exception(); Erreur d'exécution public static void main(String[] args) {
 Alpha alpha = new Alpha(); alpha.throwup(); System.out.println("Done"); }

#### Question 6 $\bigoplus$

Soit le code ci-dessous ; que se passe-t-il? public class Epsilon {
 public void throwup() throws RuntimeException {
 throw new RuntimeException(); public static void main(String[] args) {
 Epsilon epsilon = new Epsilon(); try { epsilon.throwup(); } catch (Exception e) {
 System.out.println("Caught exception"); } finally { System.out.println("Finallied exception"); } }

Affiche	seulement	"Finallied	exception"

Affiche seulement "Caught exception"

Affiche "Caught exception" et "Finallied exception"

| Erreur de compilation

Erreur d'exécution



```
interface Somethingable {
    void something();
}

interface Whateverable extends Somethingable {
    void whatever();
}

class TheThing implements Whateverable{
    PPP void something() {
        System.out.println("Somethinging");
    }

    PPP void whatever() {
        System.out.println("Whatevering");
    }
}
```

```
class Doofer {
    void doo(Somethingable sable) {
        sable.XXX();
    }

    void doo(Whateverable sable) {
        sable.YYY();
    }

    void fer(TheThing tt) {
        tt.ZZZ();
    }

    public static void main(String[] args) {
        Doofer doofer = new Doofer();
        TheThing the = new TheThing();
        doofer.doo(the);
        doofer.fer(the);
    }
}
```

Par quels niveaux d'accès pourrait-on remplacer PPP? package-private private protected public Question 8 Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer XXX? whatever something Question 9 Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer YYY? something whatever Question 10  $\bigoplus$ Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer ZZZ? whatever something



Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
}
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit





Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r\cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r\sin(\theta)$   $\theta = \tan^{-1}\frac{y}{x}$ 

En Java ce dernier est calculé par theta = Math.atan2(y, x)

- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.





Question 13	Continuez votre rép	oonse à la questior	n précédente ici si n	écessaire.	0 0



Question 14	Continuez votre réponse à la question précédente ici si nécessaire.	



Question	15	Pour	la	question	précédente,	développez	la	méthode	_		s.
Question	16 🕀	Parm	ni le	es affirma	tions ci-desso	ous, cocher c	elle	s qui sont	vraies	:	
					nir des const						
=		_			méthodes pr						
					constructeur sse abstraite						
_					menter une i						

Une classe peut étendre une classe et implémenter plusieurs interfaces

Une classe peut étendre plusieurs classes



## Question $17 \oplus$ Soit le code ci-dessous :

```
@FunctionalInterface
interface ComparePerson {
    /**
    * @return negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

```
class Person {
    private int age;
    private String name;

Person(String name, int age) {
        this.age = age;
        this.name = name;
    }
}

class Main {
    public static void main(String[] args) {
        // code extract here

        System.out.print(byAge.compare(
            new Person("Fred", 23),
            new Person("Not Fred", 24)));
        System.out.print(byAge.compare(
                  new Person("Fred", 23),
                  new Person("Fred", 23));
        System.out.print(byAge.compare(
                  new Person("Not Fred", 23));
        System.out.print(byAge.compare(
                  new Person("Fred", 23),
                  new Person("Fred", 23),
                  new Person("Fred", 23);
                  new Person("Not Fred", 22)));
}
```

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :

```
ComparePerson byAge = (p1, p2) -> p1.age - p2.age;

ComparePerson byAge = new ComparePersonByAge() {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
};

ComparePerson byAge = new ComparePersonByAge();
```



# Question 18 $\bigoplus$ Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

Prof	Student
Person	Engineer
MdC	Faculty
PEIP	Personnel

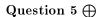
QCMTEST Nom et prénom: Introduction à la programmation orientée objet 9/01/2020 Groupe: ..... Cochez les cases en mettant une  $\times$ . Le symbole  $\bigoplus$  indique que la question peut avoir zéro, une ou plusieurs bonnes réponses. Pour ces questions, cocher une bonne réponse apporte des points positifs ; cocher une mauvaise réponse peut apporter des points négatifs. Dans tout le code, les package et les import sont censés être correctement déclarés. Toute classe est supposée être dans le bon package, dans le bon fichier, avec les bons import.Question 1 La classe donnée class ClasseDonnee {} pourrait être sous-classée par extends pourrait être instanciée new par ClasseDonnee ClasseDonnee()

### Question 2 La classe donnée class ClasseDonnee { private ClasseDonnee() {} pourrait être sous-classée par extends pourrait être instanciée new par ClasseDonnee ClasseDonnee() Question 3 $\bigoplus$ La classe donnée abstract class ClasseDonnee { ClasseDonnee() {} pourrait être sous-classée par extends pourrait être instanciée new ClasseDonnee ClasseDonnee() Question 4 Affiche "Done" Soit le code ci-dessous ; que se passe-t-il? Erreur de compilation public class Beta { public void throwup() throws RuntimeException { Erreur d'exécution throw new RuntimeException(); public static void main(String[] args) { Beta beta = new Beta();

beta.throwup();

}

System.out.println("Done");



```
Soit le code ci-dessous ; que se passe-t-il?
                                                           Erreur de compilation
public class Delta {
                                                            Affiche "Caught exception" et "Finallied
   public void throwup() throws Exception {
                                                            exception"
      throw new Exception();
                                                        Affiche seulement "Finallied exception"
   public static void main(String[] args) {
      Delta delta = new Delta();
                                                            Affiche seulement "Caught exception"
      try {
          delta.throwup();
                                                            Erreur d'exécution
      } catch (Exception e) {
          System.out.println("Caught exception");
          return:
      } finally {
          System.out.println("Finallied exception");
}
Question 6
                                                            Affiche seulement "Finallied exception"
Soit le code ci-dessous ; que se passe-t-il?
public class Gamma {
                                                            Erreur de compilation
   public void throwup() throws Exception {
                                                            Erreur d'exécution
      throw new Exception();
                                                           Affiche seulement "Caught exception"
   public static void main(String[] args) {
```

#### **Question 7** Soit le code :

Gamma gamma = new Gamma();

System.out.println("Caught exception");

System.out.println("Finallied exception");

gamma.throwup();

} catch (Exception e) {

try {

}

} finally {

```
interface Somethingable {
    void something();
}

interface Whateverable extends Somethingable {
    void whatever();
}

class TheThing implements Whateverable{
    PPP void something() {
        System.out.println("Somethinging");
    }

    PPP void whatever() {
        System.out.println("Whatevering");
    }
}
```

```
class Doofer {
    void doo(Somethingable sable) {
        sable.XXX();
    }

    void doo(Whateverable sable) {
        sable.YYY();
    }

    void fer(TheThing tt) {
        tt.ZZZ();
    }

    public static void main(String[] args) {
        Doofer doofer = new Doofer();
        TheThing the = new TheThing();
        doofer.doo(the);
        doofer.fer(the);
    }
}
```

Affiche "Caught exception" et "Finallied

exception"

Par quels niveaux d'accès pourrait-on remplacer PPP?

package-private	protected
private	public

Question 8 $\bigoplus$	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
whatever	something
Question 9 $\bigoplus$ placer YYY ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
something	whatever
Question 10 $\bigoplus$ remplacer ZZZ ?	Pour le code de la question précédente, par quelles méthodes pourrait-on

something

\_\_\_ whatever

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit





Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r \cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r \sin(\theta)$   $\theta = \tan^{-1} \frac{y}{x}$ 

En Java ce dernier est calculé par theta = Math.atan2(y, x)

- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.





Question 13	Continuez	votre réponse à	la question p	récédente i	ci si nécessaire.	



Question 14	Continuez	votre réponse	à la question	précédente i	ci si nécessaire.	



Une classe peut étendre plusieurs classes

Une classe abstraite peut contenir des constructeurs
 Une classe abstraite peut implémenter une interface

Question	15	Pour l	la question	n précédente,	développez	la mé			3. <u>4</u>
${f Question}$	<b>16</b> $\oplus$	Parmi	i les affirm	ations ci-dess	ous, cocher c	elles qı	ıi sont vrai	es :	
				es constructeu					
				lasse abstraite lasse et implér		eurs int	erfaces		
☐ Une i	interface	e peut c	contenir de	s méthodes p	rivées				



```
@FunctionalInterface
interface ComparePerson {
    /**
    * Oreturn negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    OOverride
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

```
class Person {
    private int age;
    private String name;

Person(String name, int age) {
        this.age = age;
        this.name = name;
    }
}

class Main {
    public static void main(String[] args) {
        // code extract here

        System.out.print(byAge.compare(
            new Person("Fred", 23),
            new Person("Not Fred", 24)));
        System.out.print(byAge.compare(
                  new Person("Fred", 23),
                  new Person("Fred", 23));
        System.out.print(byAge.compare(
                  new Person("Not Fred", 23));
        System.out.print(byAge.compare(
                  new Person("Fred", 23),
                  new Person("Fred", 23),
                  new Person("Fred", 23);
                  new Person("Not Fred", 22)));
}
```

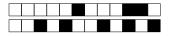
Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :

```
ComparePerson byAge = new ComparePersonByAge() {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
};

ComparePerson byAge = new ComparePersonByAge();

ComparePerson byAge
    = new ComparePersonByAge::compare;

ComparePerson byAge
    = new ComparePersonByAge::compare;
```



Question 18  $\bigoplus$  Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

Person	Personnel
MdC	$oxedsymbol{\Box}$ Faculty
Student	PEIP
Prof	Engineer

QCMTEST Nom et prénom: Introduction à la programmation orientée objet 9/01/2020 Groupe: ..... Cochez les cases en mettant une  $\times$ . Le symbole  $\bigoplus$  indique que la question peut avoir zéro, une ou plusieurs bonnes réponses. Pour ces questions, cocher une bonne réponse apporte des points positifs ; cocher une mauvaise réponse peut apporter des points négatifs. Dans tout le code, les package et les import sont censés être correctement déclarés. Toute classe est supposée être dans le bon package, dans le bon fichier, avec les bons import.Question 1 La classe (enum) donnée enum ClasseDonnee { INSTANCE; pourrait pourrait être sous-classée par extends être instanciée new ClasseDonnee ClasseDonnee() Question 2 La classe donnée class ClasseDonnee { private ClasseDonnee() {} pourrait être pourrait être sous-classée par extends instanciée par new ClasseDonnee() ClasseDonnee Question 3 ( La classe donnée class ClasseDonnee {} \_\_\_\_ pourrait être sous-classée par extends pourrait être instanciée new ClasseDonnee ClasseDonnee() Question 4 (

Soit le code ci-dessous ; que se passe-t-il?

throw new RuntimeException();
}
public static void main(String[] args) {
Beta beta = new Beta();
beta.throwup();

System.out.println("Done");

public void throwup() throws RuntimeException {

public class Beta {

}

Erreur d'exécution

Affiche "Done"

Erreur de compilation



## Question 5 $\bigoplus$

Affiche seulement "Finallied exception" Soit le code ci-dessous ; que se passe-t-il? public class Gamma { Erreur de compilation public void throwup() throws Exception { Affiche seulement "Caught exception" throw new Exception(); public static void main(String[] args) { Affiche "Caught exception" et "Finallied Gamma gamma = new Gamma(); exception" try { gamma.throwup(); Erreur d'exécution } catch (Exception e) { System.out.println("Caught exception"); } finally { System.out.println("Finallied exception"); } } Question 6  $\bigoplus$ Soit le code ci-dessous ; que se passe-t-il? Erreur d'exécution public class Alpha { Erreur de compilation public void throwup() throws Exception {

#### Question 7 Soit le code :

alpha.throwup();

}

throw new Exception();

System.out.println("Done");

public static void main(String[] args) {
 Alpha alpha = new Alpha();

```
interface Somethingable {
    void something();
}

interface Whateverable extends Somethingable {
    void whatever();
}

class TheThing implements Whateverable{
    PPP void something() {
        System.out.println("Somethinging");
    }

    PPP void whatever() {
        System.out.println("Whatevering");
    }
}
```

```
class Doofer {
    void doo(Somethingable sable) {
        sable.XXX();
    }

    void doo(Whateverable sable) {
        sable.YYY();
    }

    void fer(TheThing tt) {
        tt.ZZZ();
    }

    public static void main(String[] args) {
        Doofer doofer = new Doofer();
        TheThing the = new TheThing();
        doofer.doo(the);
        doofer.fer(the);
    }
}
```

Affiche "Done"

Par quels niveaux d'accès pourrait-on remplacer PPP?

private	package-private
public	protected
Question 8 $\bigoplus$ placer XXX ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
something	whatever



Question 9 $\bigoplus$ placer YYY ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
whatever	something
Question 10 $\bigoplus$ remplacer ZZZ ?	Pour le code de la question précédente, par quelles méthodes pourrait-on
something	whatever

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

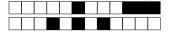
@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit





Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r\cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r\sin(\theta)$   $\theta = \tan^{-1}\frac{y}{x}$ 

En Java ce dernier est calculé par theta = Math.atan2(y, x)

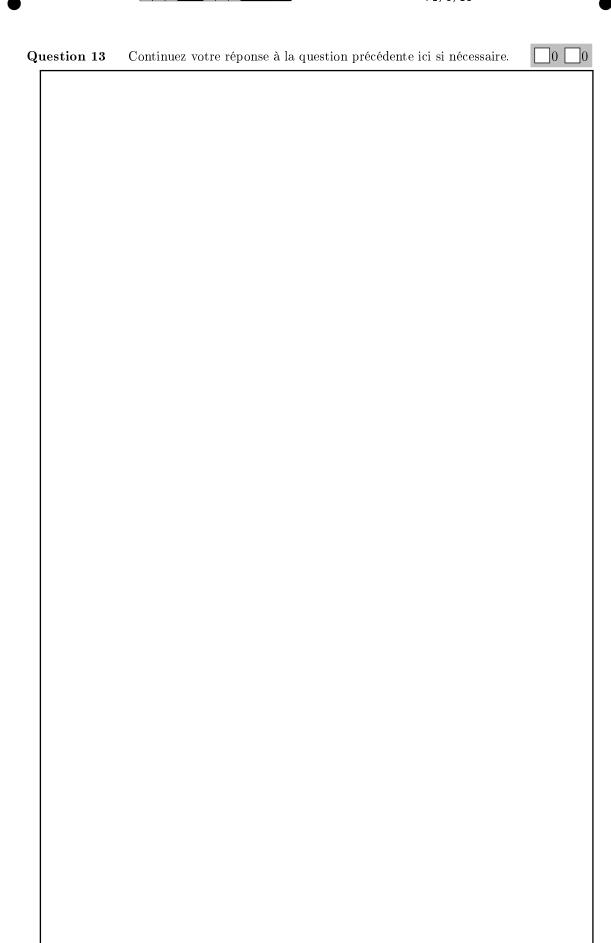
- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

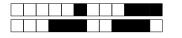
```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

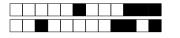
Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.







Question 14	Continuez	votre répo	nse à la que	estion précéd	ente ici si n	écessaire.	0 0



Question	15	Pour	la	auestion	précédente,	développez	la	méthod	e <b>Pola</b>	r#egua	ls.
<b>V</b>				1	<b>F</b>				0 1		3 4
Question	16 🔿	Down	n; 1	og offinns	tions ci-desso	ous acaber a	allac	. aui aor	nt tracio	a .	
_	•					ous, coener e	enes	qui soi	n viaie	ъ.	
_				re plusieu							
					constructeur						
_					méthodes pr sse et implér		ourg	interfac	OG		
_					menter une i		surs	merrac	es		
=			_	=	$rac{1}{2} = 1$						
=					sse abstraite						

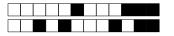


## Question $17 \oplus$ Soit le code ci-dessous :

```
@FunctionalInterface
interface ComparePerson {
    /**
    * @return negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :



# Question 18 $\bigoplus$ Soit l'organisation ci-dessous de Polytech'Groland :

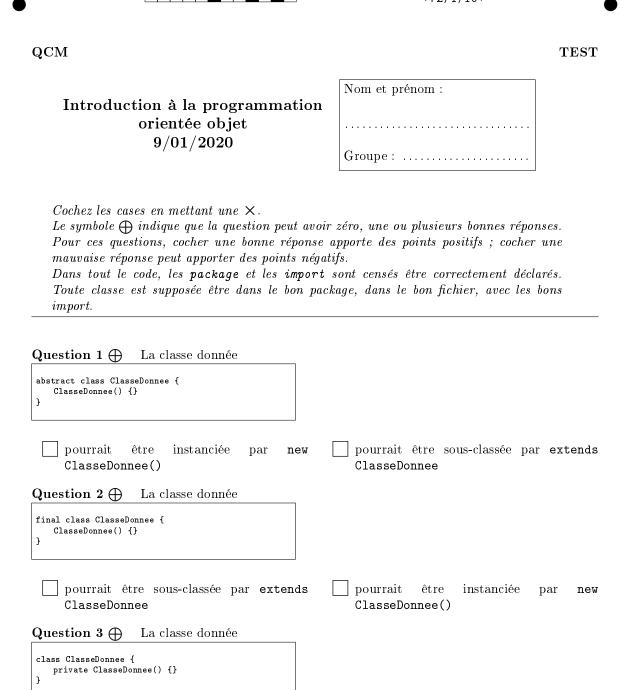


Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

MdC	Prof
Faculty	Student
PEIP	Person
Engineer	Personnel

pourrait être sous-classée par extends

ClasseDonnee



pourrait

ClasseDonnee()

être

instanciée

par

new

Soit le code ci-dessous ; que se passe-t-il ?	Erreur de compilation
<pre>public class Epsilon {     public void throwup() throws RuntimeException {         throw new RuntimeException();     }     public static void main(String[] args) {         Epsilon epsilon = new Epsilon();         try {             epsilon.throwup();     } catch (Exception e) {             System.out.println("Caught exception");     } finally {             System.out.println("Finallied exception");     } }</pre>	<ul> <li>Erreur d'exécution</li> <li>Affiche "Caught exception" et "Finallied exception"</li> <li>Affiche seulement "Caught exception"</li> <li>Affiche seulement "Finallied exception"</li> </ul>
Question 5 $\bigoplus$	

```
Soit le code ci-dessous ; que se passe-t-il?
public class Delta {
   public void throwup() throws Exception {
       throw new Exception();
   public static void main(String[] args) {
    Delta delta = new Delta();
        try {
    delta.throwup();
        } catch (Exception e) {
            System.out.println("Caught exception");
            return;
        } finally {
   System.out.println("Finallied exception");
```

## Question 6 $\bigoplus$

Soit le code ci-dessous ; que se passe-t-il? public class Beta { public void throwup() throws RuntimeException { throw new RuntimeException(); public static void main(String[] args) { Beta beta = new Beta();
beta.throwup(); System.out.println("Done"); }

	Affiche	seule	$\mathbf{ment}$	"Finall	ied	exception	on"
$\neg$			_				

 $\_$  Affiche "Caught exception" et "Finallied exception"

Erreur d'exécution

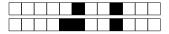
Erreur de compilation

Affiche seulement "Caught exception"

Erreur de compilation

Affiche "Done"

Erreur d'exécution



## Question 7 Soit le code :

```
interface Somethingable {
   void something();
}

interface Whateverable extends Somethingable {
   void whatever();
}

class TheThing implements Whateverable{
   PPP void something() {
       System.out.println("Somethinging");
   }

   PPP void whatever() {
       System.out.println("Whatevering");
   }
}
```

```
class Doofer {
   void doo(Somethingable sable) {
      sable.XXX();
   }

   void doo(Whateverable sable) {
      sable.YYY();
   }

   void fer(TheThing tt) {
      tt.ZZZ();
   }

  public static void main(String[] args) {
      Doofer doofer = new Doofer();
      TheThing the = new TheThing();
      doofer.doo(the);
      doofer.fer(the);
   }
}
```

Par quels niveaux d'accès pourrait-on remplacer PPP? package-private public protected private Question 8 Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer XXX? whatever something Question 9 Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer YYY? something whatever Question 10  $\bigoplus$ Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer ZZZ? something whatever

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

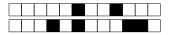
@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit





Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r\cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r\sin(\theta)$   $\theta = \tan^{-1}\frac{y}{x}$ 

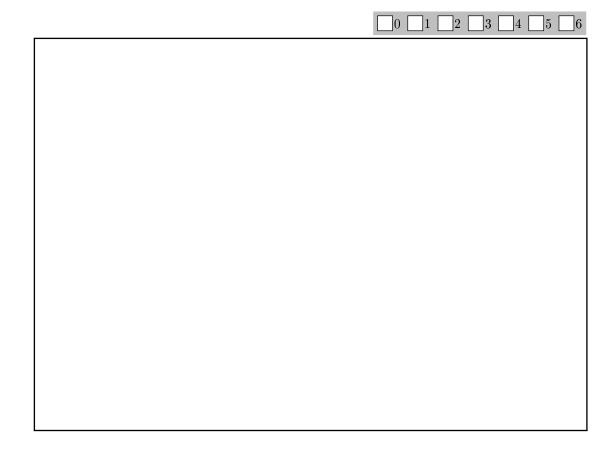
En Java ce dernier est calculé par theta = Math.atan2(y, x)

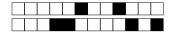
- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.





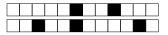
Question 13	Continuez votre réponse à la question précédente ici si nécessaire.	



Question 14	Continuez votre réponse à la question précédente ici si nécessaire.	
1		



Question 15 Pour la question précédente, développez la méthode Polar#equals.  $\boxed{0}$   $\boxed{1}$   $\boxed{2}$   $\boxed{3}$ Question 16  $\bigoplus$ Parmi les affirmations ci-dessous, cocher celles qui sont vraies : Une interface peut contenir des méthodes privées Une classe abstraite peut implémenter une interface Une interface peut contenir des constructeurs Une classe peut étendre une classe et implémenter plusieurs interfaces Une classe peut étendre plusieurs classes Une classe peut étendre une classe abstraite Une classe abstraite peut contenir des constructeurs



# Question $17 \oplus$ Soit le code ci-dessous :

```
@FunctionalInterface
interface ComparePerson {
    /**
    * Oreturn negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :

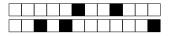
```
ComparePerson byAge = (p1, p2) -> p1.age - p2.age;

ComparePerson byAge = new ComparePersonByAge() {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
};

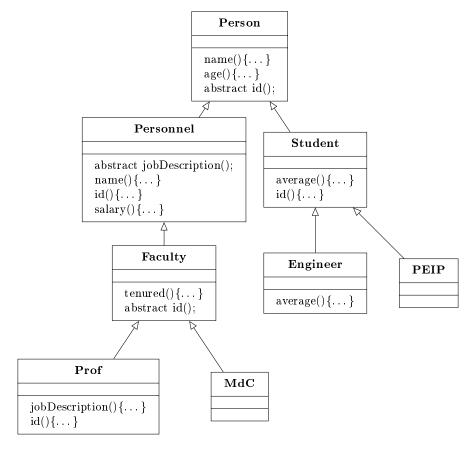
ComparePerson byAge = new ComparePersonByAge();

ComparePerson byAge = new ComparePersonByAge();

ComparePerson byAge = new ComparePersonByAge();
```



Question 18  $\bigoplus$  Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

<pre>Faculty</pre>	MdC
Person	PEIP
Prof	Personnel
Engineer	Student

 $\mathbf{QCM}$ TEST

Introduction à la programmation
orientée objet
9/01/2020

Nom e	t p	oré	'n	01	n	:						
Group	e :							 		٠		

Cochez les cases en mettant une  $\times$ .

Le symbole  $\bigoplus$  indique que la question peut avoir zéro, une ou plusieurs bonnes réponses.  $Pour \ ces \ questions, \ cocher \ une \ bonne \ r\'eponse \ apporte \ des \ points \ positifs \ ; \ cocher \ une$ mauvaise réponse peut apporter des points négatifs.

Dans tout le code, les package et les import sont censés être correctement déclarés.

Toute classe est supposée être dans le bon paimport.	ackage, dans le bon fichier, avec les bons
Question 1	
class ClasseDonnee {}	
pourrait être sous-classée par extends ClasseDonnee	pourrait être instanciée par new ClasseDonnee()
Question 2   La classe (enum) donnée	
enum ClasseDonnee {    INSTANCE; }	
pourrait être instanciée par <b>new</b> ClasseDonnee()	pourrait être sous-classée par extends ClasseDonnee
Question 3   La classe donnée	
abstract class ClasseDonnee {     ClasseDonnee() {} }	
pourrait être instanciée par <b>new</b> ClasseDonnee()	pourrait être sous-classée par <b>extends</b>
Question 4 $\bigoplus$	
Soit le code ci-dessous ; que se passe-t-il ?	Affiche "Caught exception" et "Finallied exception"
public class Gamma {     public void throwup() throws Exception {	Erreur de compilation
<pre>throw new Exception(); } public static void main(String[] args) {    Gamma gamma = new Gamma();    try {        gamma.throwup(); }</pre>	Affiche seulement "Caught exception"  Erreur d'exécution
<pre>} catch (Exception e) {     System.out.println("Caught exception"); } finally {     System.out.println("Finallied exception"); }</pre>	Affiche seulement "Finallied exception"



# Question 5 $\bigoplus$

Soit le code ci-dessous ; que se passe-t-il? Erreur de compilation public class Alpha { Erreur d'exécution public void throwup() throws Exception { Affiche "Done" throw new Exception(); public static void main(String[] args) { Alpha alpha = new Alpha(); alpha.throwup(); System.out.println("Done"); Question 6 Soit le code ci-dessous ; que se passe-t-il? Affiche seulement "Finallied exception" public class Epsilon { Affiche "Caught exception" et "Finallied public void throwup() throws RuntimeException { exception" throw new RuntimeException(); Affiche seulement "Caught exception" public static void main(String[] args) { Epsilon epsilon = new Epsilon(); Erreur d'exécution try { epsilon.throwup(); Erreur de compilation } catch (Exception e) { System.out.println("Caught exception"); } finally { System.out.println("Finallied exception");

#### Question 7 Soit le code :

}

```
interface Somethingable {
    void something();
}

interface Whateverable extends Somethingable {
    void whatever();
}

class TheThing implements Whateverable{
    PPP void something() {
        System.out.println("Somethinging");
    }

    PPP void whatever() {
        System.out.println("Whatevering");
    }
}
```

```
class Doofer {
    void doo(Somethingable sable) {
        sable.XXX();
    }

    void doo(Whateverable sable) {
        sable.YYY();
    }

    void fer(TheThing tt) {
        tt.ZZZ();
    }

    public static void main(String[] args) {
        Doofer doofer = new Doofer();
        TheThing the = new TheThing();
        doofer.doo(the);
        doofer.fer(the);
    }
}
```

Par quels niveaux d'accès pourrait-on remplacer PPP?

protected	private
package-pri	vate public
Question 8 $\bigoplus$ placer XXX ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem
something	whatever

Question 9 ⊕ placer YYY ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
something	whatever
$\begin{array}{c} \mathbf{Question}  10  \bigoplus \\ \mathbf{remplacer}  \mathbf{ZZZ} \ ? \end{array}$	Pour le code de la question précédente, par quelles méthodes pourrait-on
whatever	something

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit



Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r\cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r\sin(\theta)$   $\theta = \tan^{-1}\frac{y}{x}$ 

En Java ce dernier est calculé par theta = Math.atan2(y, x)

- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.





Question 13	Continuez	votre répons	e à la question	précédente ic	i si nécessaire.	0 0



Question 14	Continuez	votre réponse	à la question	précédente i	ci si nécessaire.	



Question 15 Pour la question précédente, développez la méthode Polar#equals.

 $\textbf{Question 16} \ \bigoplus \ \ \text{Parmi les affirmations ci-dessous, cocher celles qui sont vraies} :$ 

Une classe abstraite peut contenir des constructeurs
Une classe peut étendre une classe abstraite
Une classe peut étendre une classe et implémenter plusieurs interfaces
Une classe abstraite peut implémenter une interface
Une classe peut étendre plusieurs classes
Une interface peut contenir des constructeurs
Une interface neut contenir des méthodes privées



# Question $17 \oplus$ Soit le code ci-dessous :

```
@FunctionalInterface
interface ComparePerson {
    /**
    * Oreturn negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

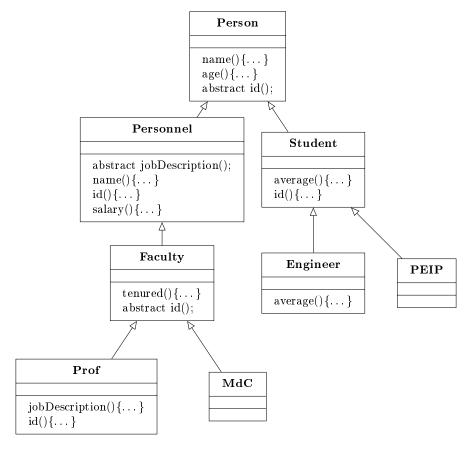
Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :

```
ComparePerson byAge = new ComparePersonByAge::compare;

ComparePerson byAge = new ComparePersonByAge() {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
};
```

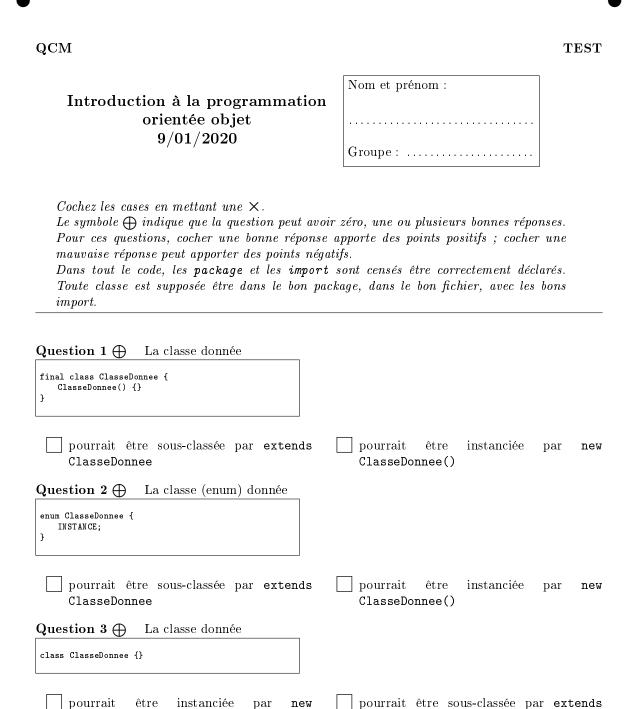


Question 18  $\bigoplus$  Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

Faculty	Engineer
Personnel	Student
MdC	Prof
Person	PEIP



être

ClasseDonnee()

instanciée

par

new

ClasseDonnee

Soit le code ci-dessous ; que se passe-t-il ?	Affiche seulement "Caught exception"
<pre>public class Delta {     public void throwup() throws Exception {         throw new Exception();     }     public static void main(String[] args) {         Delta delta = new Delta();         try {             delta.throwup();       } catch (Exception e) {             System.out.println("Caught exception");             return;       } finally {             System.out.println("Finallied exception");       }     } }</pre>	Affiche "Caught exception" et "Finallied exception"  Erreur d'exécution  Affiche seulement "Finallied exception"  Erreur de compilation
Question 5 $\bigoplus$	
Soit le code ci dessous : que se passe + il ?	Affiche soulement "Caught exception"

# Soit le code ci-dessous ; que se passe-t-il?

```
public class Epsilon {
   public void throwup() throws RuntimeException {
        throw new RuntimeException();
   public static void main(String[] args) {
    Epsilon epsilon = new Epsilon();
        try {
            epsilon.throwup();
        } catch (Exception e) {
            System.out.println("Caught exception");
        } finally {
   System.out.println("Finallied exception");
}
```

# Question 6 $\bigoplus$

Soit le code ci-dessous ; que se passe-t-il? public class Alpha { public void throwup() throws Exception { throw new Exception(); public static void main(String[] args) { Alpha alpha = new Alpha(); alpha.throwup(); System.out.println("Done");

] Affiche seulement "Caught ex	ception"	
--------------------------------	----------	--

Erreur de compilation

Affiche seulement "Finallied exception"

Affiche "Caught exception" et "Finallied exception"

Erreur d'exécution

Affiche "Done"

Erreur d'exécution

Erreur de compilation



### **Question 7** Soit le code :

```
interface Somethingable {
   void something();
}

interface Whateverable extends Somethingable {
   void whatever();
}

class TheThing implements Whateverable{
   PPP void something() {
       System.out.println("Somethinging");
   }

   PPP void whatever() {
       System.out.println("Whatevering");
   }
}
```

<pre>class Doofer {   void doo(Somethingable sable) {      sable.XXX();   }</pre>
<pre>void doo(Whateverable sable) {     sable.YYY(); }</pre>
<pre>void fer(TheThing tt) {    tt.ZZZ(); }</pre>
<pre>public static void main(String[] args) {    Doofer doofer = new Doofer();    TheThing the = new TheThing();    doofer.doo(the);    doofer.fer(the);</pre>
}

Par quels niveaux d'accès pourrait-on remplacer PPP ?

private protected

public package-private

something whatever

**Question 9**  $\bigoplus$  Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer YYY ?

whatever something

**Question 10**  $\bigoplus$  Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer ZZZ ?

whatever something

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit



Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r \cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r \sin(\theta)$   $\theta = \tan^{-1} \frac{y}{x}$ 

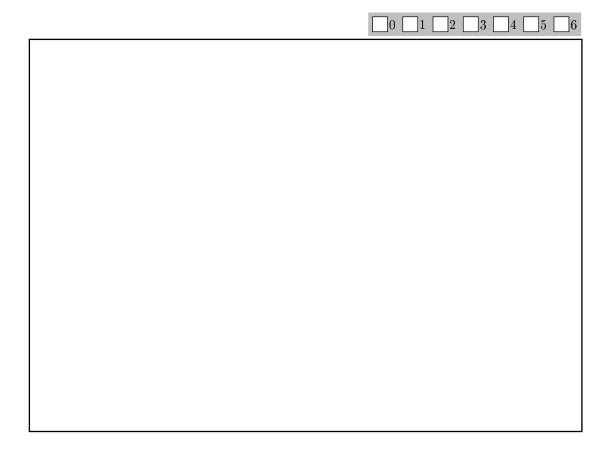
En Java ce dernier est calculé par theta = Math.atan2(y, x)

- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.





Question 13	Continuez votre réponse à la question précédente ici si nécessaire.	





Une interface peut contenir des constructeurs
Une classe peut étendre une classe abstraite

Une classe abstraite peut implémenter une interface
Une interface peut contenir des méthodes privées



# Question $17 \oplus$ Soit le code ci-dessous :

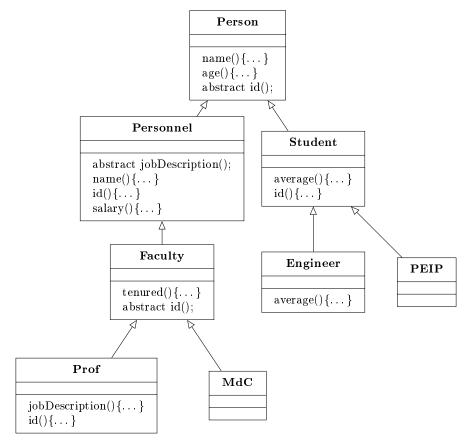
```
@FunctionalInterface
interface ComparePerson {
    /**
    * Oreturn negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :

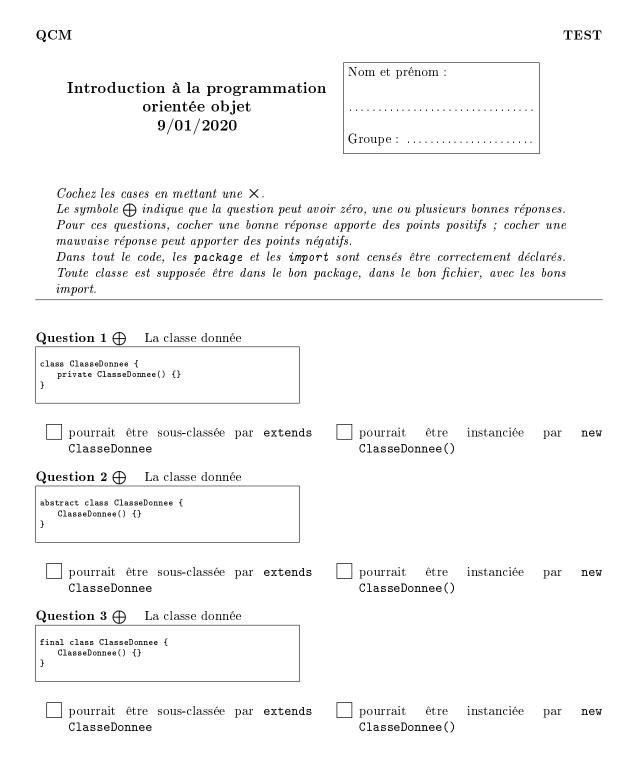


Question 18  $\bigoplus$  Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

Engineer	Personnel
Prof	Faculty
Student	MdC
PEIP	Person



```
Affiche seulement "Caught exception"
Soit le code ci-dessous ; que se passe-t-il?
public class Gamma {
                                                             Erreur d'exécution
   public void throwup() throws Exception {
                                                             Affiche seulement "Finallied exception"
      throw new Exception();
   public static void main(String[] args) {
                                                             Erreur de compilation
      Gamma gamma = new Gamma();
      try {
                                                             Affiche "Caught exception" et "Finallied
          gamma.throwup();
                                                             exception"
      } catch (Exception e) {
          System.out.println("Caught exception");
       } finally {
          System.out.println("Finallied exception");
   }
}
Question 5 \bigoplus
```

```
Soit le code ci-dessous ; que se passe-t-il?
public class Beta {
   public void throwup() throws RuntimeException {
       throw new RuntimeException();
   public static void main(String[] args) {
   Beta beta = new Beta();
       beta.throwup();
       System.out.println("Done");
}
```

# Erreur de compilation

Affiche "Done"

Erreur d'exécution

# Question 6 $\bigoplus$

Soit le code ci-dessous ; que se passe-t-il? public class Delta { public void throwup() throws Exception { throw new Exception(); public static void main(String[] args) { Delta delta = new Delta(); try { delta.throwup(); } catch (Exception e) { System.out.println("Caught exception"); return; } finally { System.out.println("Finallied exception"); }

Affiche	"Caught	exception"	$\operatorname{et}$	"Finallied
exception	on"			

Erreur d'exécution

Affiche seulement "Caught exception"

Affiche seulement "Finallied exception"

Erreur de compilation



```
interface Somethingable {
   void something();
}

interface Whateverable extends Somethingable {
   void whatever();
}

class TheThing implements Whateverable{
   PPP void something() {
       System.out.println("Somethinging");
   }

   PPP void whatever() {
       System.out.println("Whatevering");
   }
}
```

```
class Doofer {
    void doo(Somethingable sable) {
        sable.XXX();
    }

    void doo(Whateverable sable) {
        sable.YYY();
    }

    void fer(TheThing tt) {
        tt.ZZZ();
    }

    public static void main(String[] args) {
        Doofer doofer = new Doofer();
        TheThing the = new TheThing();
        doofer.doo(the);
        doofer.fer(the);
    }
}
```

Par quels niveaux d'accès pourrait-on remplacer PPP? public private package-private protected Question 8 Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer XXX? whatever something Question 9 Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer YYY? whatever something Question 10  $\bigoplus$ Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer ZZZ? something whatever

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit



Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r\cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r\sin(\theta)$   $\theta = \tan^{-1}\frac{y}{x}$ 

En Java ce dernier est calculé par theta = Math.atan2(y, x)

- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.





Question 13	Continuez	votre réponse à la question précédente ici si nécessaire.	



Question 14	Continuez	votre réponse à la question précédente ici si nécessaire.	



Question	15	Pour	la	question	précédente,	développez	la	méthode Polar#equals.

Question 16 $\bigoplus$	Parmi les affirmations ci-dessous, cocher celles qui sont vraies :
Une interface	peut contenir des méthodes privées
Une classe ab	estraite peut contenir des constructeurs
$\Box$ Une classe pe	eut étendre plusieurs classes
Une classe ab	straite peut implémenter une interface
Une classe pe	eut étendre une classe abstraite
Une interface	peut contenir des constructeurs
Une classe pe	eut étendre une classe et implémenter plusieurs interfaces



# Question $17 \oplus$ Soit le code ci-dessous :

```
@FunctionalInterface
interface ComparePerson {
    /**
    * Oreturn negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

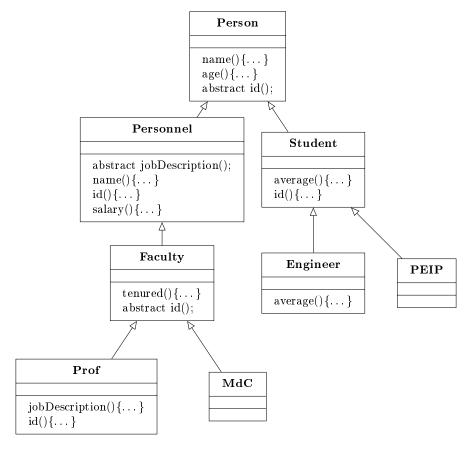
Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :

```
ComparePerson byAge = new ComparePersonByAge::compare;

ComparePerson byAge = new ComparePersonByAge() {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
};
```



Question 18  $\bigoplus$  Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

Personnel	Student
Engineer	Person
PEIP	MdC
<pre>Faculty</pre>	Prof

QCM

Introduction à la programmation
orientée objet
9/01/2020

Nom et	pı	ré	n	О	m	1 :								
				•										
Groupe	:													

Cochez les cases en mettant une  $\times$ .

Le symbole  $\bigoplus$  indique que la question peut avoir zéro, une ou plusieurs bonnes réponses. Pour ces questions, cocher une bonne réponse apporte des points positifs ; cocher une mauvaise réponse peut apporter des points négatifs.

Dans tout le code, les package et les import sont censés être correctement déclarés. Toute classe est supposée être dans le bon package, dans le bon fichier, avec les bons import.

Question 1 $\bigoplus$	La classe	(enum)	) donnée
------------------------	-----------	--------	----------

pourrait	$\hat{\mathrm{e}}\mathrm{tre}$	sous-classée	par	extends	pourrait	${\rm \hat{e}tre}$	$instanci\'ee$	par	new
ClasseDo	onnee				ClasseDor	nee()			

# Question 2 La classe donnée

class ClasseDonnee {}
-----------------------

pourrait	${ m \hat{e}tre}$	$instanci\'ee$	par	new	pourrait	être	sous-classée	par	extends
ClasseDon	nee()				ClasseDo	nnee			

# 

```
final class ClasseDonnee {
   ClasseDonnee() {}
}
```

pourrait	${ m \hat{e}tre}$	sous-classée	par	extends	pourrait	être	$instanci\'{e}e$	par	new
ClasseDo	onnee				ClasseDo	nnee()			

### Question 4

}

```
Soit le code ci-dessous ; que se passe-t-il ?

public class Gamma {
    public void throwup() throws Exception {
        throw new Exception();
    }

public static void main(String[] args) {
        Gamma gamma = new Gamma();
        try {
            gamma.throwup();
      } catch (Exception e) {
            System.out.println("Caught exception");
      } finally {
            System.out.println("Finallied exception");
      }
}
```

Affiche	"Caught	exception "	$\operatorname{et}$	"Finallied
exception	on"			

Erreur de compilation

Affiche seulement "Caught exception"

Affiche seulement "Finallied exception"

Erreur d'exécution

# Question 5 $\bigoplus$

Soit le code ci-dessous ; que se passe-t-il ?

public class Epsilon {
 public void throwup() throws RuntimeException {
 throw new RuntimeException();
 }

 public static void main(String[] args) {
 Epsilon epsilon = new Epsilon();
 try {
 epsilon.throwup();
 } catch (Exception e) {
 System.out.println("Caught exception");
 } finally {
 System.out.println("Finallied exception");
 }
 }
}

Affiche seulement "Finallied exception"
Affiche seulement "Caught exception"
Affiche "Caught exception" et "Finallied exception"
Erreur de compilation
Erreur d'exécution

# Question 6 $\bigoplus$

Soit le code ci-dessous ; que se passe-t-il ?
public class Delta {
 public void throwup() throws Exception {
 throw new Exception();
 }
 public static void main(String[] args) {
 Delta delta = new Delta();
 try {
 delta.throwup();
 } catch (Exception e) {
 System.out.println("Caught exception");
 return;
 } finally {
 System.out.println("Finallied exception");
 }
 }
}

	_ Errei	ır d'exé	cutio	n		
		he "Caı otion"	ight	exception"	et	"Finallied
г	¬	1	.,	. •		

Erreur de compilation

Affiche seulement "Caught exception"

Affiche seulement "Finallied exception"

#### **Question 7** Soit le code :

```
interface Somethingable {
    void something();
}

interface Whateverable extends Somethingable {
    void whatever();
}

class TheThing implements Whateverable{
    PPP void something() {
        System.out.println("Somethinging");
    }

    PPP void whatever() {
        System.out.println("Whatevering");
    }
}
```

```
class Doofer {
    void doo(Somethingable sable) {
        sable.XXX();
    }

    void doo(Whateverable sable) {
        sable.YYY();
    }

    void fer(TheThing tt) {
        tt.ZZZ();
    }

    public static void main(String[] args) {
        Doofer doofer = new Doofer();
        TheThing the = new TheThing();
        doofer.doo(the);
        doofer.fer(the);
    }
}
```

Par quels niveaux d'accès pourrait-on remplacer PPP?

 ☐ package-private
 ☐ private

 ☐ protected
 ☐ public

Question 8 $\bigoplus$ placer XXX ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem
something	whatever
Question 9 $\bigoplus$ placer YYY ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem
whatever	something
Question 10 $\bigoplus$ remplacer ZZZ ?	Pour le code de la question précédente, par quelles méthodes pourrait-or
whatever	something

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

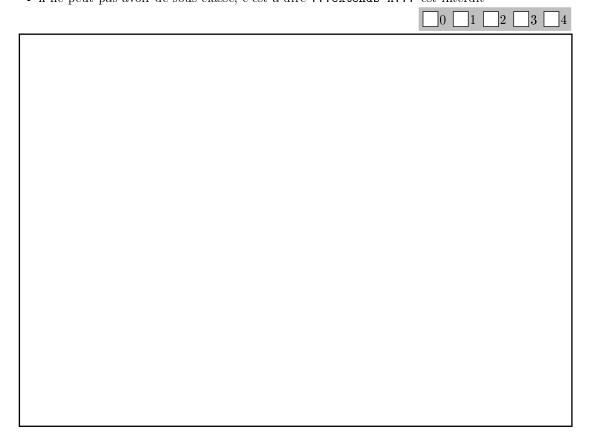
public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit





Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r\cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r\sin(\theta)$   $\theta = \tan^{-1}\frac{y}{x}$ 

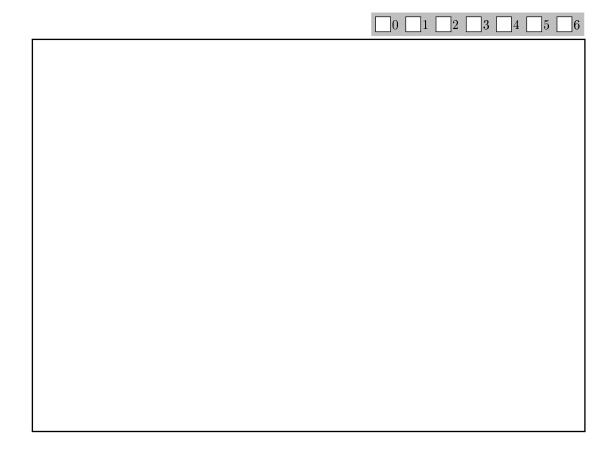
En Java ce dernier est calculé par theta = Math.atan2(y, x)

- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.



Question 13	Continuez	votre réponse	à la question	précédente i	ci si nécessaire.	0 0



Question 14	Continuez	votre réponse	à la question	précédente	ici si nécessaire.	



Une classe abstraite peut contenir des constructeurs

Une classe peut étendre une classe abstraite
 Une interface peut contenir des méthodes privées



# Question $17 \oplus$ Soit le code ci-dessous :

```
@FunctionalInterface
interface ComparePerson {
    /**
    * @return negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :

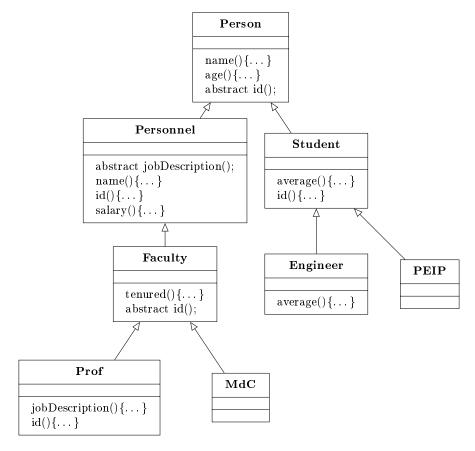
```
ComparePerson byAge = new ComparePersonByAge();

ComparePerson byAge = (p1, p2) -> p1.age - p2.age;

ComparePerson byAge = new ComparePersonByAge() {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
};
```

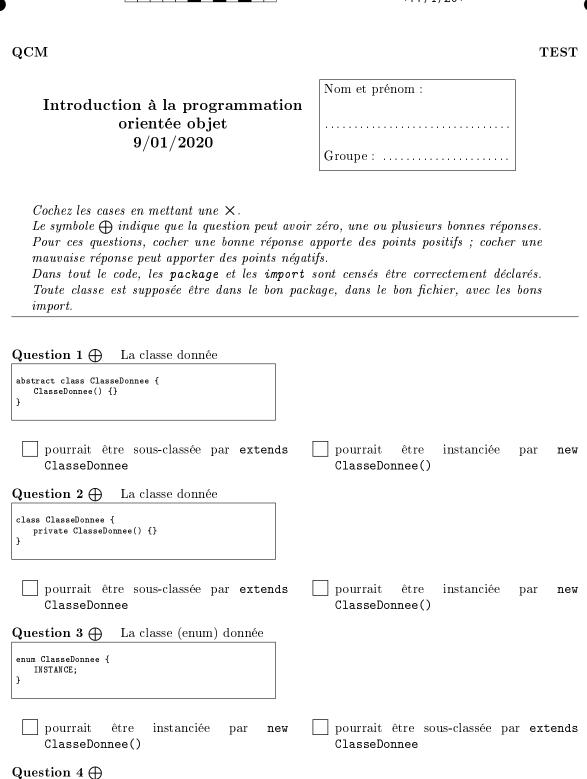


Question 18  $\bigoplus$  Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

PEIP	MdC
Person	Personnel
Prof	Engineer
Faculty	Student



Soit le code ci-dessous ; que se passe-t-il?

public void throwup() throws Exception {

public static void main(String[] args) {

throw new Exception();

Alpha alpha = new Alpha();
alpha.throwup();

System.out.println("Done");

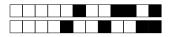
public class Alpha {

}

Erreur d'exécution

Affiche "Done"

Erreur de compilation



# Question 5 $\bigoplus$

Erreur d'exécution Soit le code ci-dessous ; que se passe-t-il? public class Beta { Erreur de compilation public void throwup() throws RuntimeException { Affiche "Done" throw new RuntimeException(); public static void main(String[] args) { Beta beta = new Beta(); beta.throwup(); System.out.println("Done"); } Question 6 Soit le code ci-dessous ; que se passe-t-il? Erreur de compilation public class Gamma { Affiche seulement "Caught exception" public void throwup() throws Exception { Affiche "Caught exception" et "Finallied throw new Exception(); public static void main(String[] args) { Gamma gamma = new Gamma(); Affiche seulement "Finallied exception" try { gamma.throwup(); Erreur d'exécution } catch (Exception e) { System.out.println("Caught exception"); } finally { System.out.println("Finallied exception");

#### Question 7 Soit le code :

}

```
interface Somethingable {
    void something();
}

interface Whateverable extends Somethingable {
    void whatever();
}

class TheThing implements Whateverable{
    PPP void something() {
        System.out.println("Somethinging");
    }

    PPP void whatever() {
        System.out.println("Whatevering");
    }
}
```

```
class Doofer {
    void doo(Somethingable sable) {
        sable.XXX();
    }

    void doo(Whateverable sable) {
        sable.YYY();
    }

    void fer(TheThing tt) {
        tt.ZZZ();
    }

    public static void main(String[] args) {
        Doofer doofer = new Doofer();
        TheThing the = new TheThing();
        doofer.doo(the);
        doofer.fer(the);
    }
}
```

Par quels niveaux d'accès pourrait-on remplacer PPP?

package-priv	vate public
protected	private
Question 8 $\bigoplus$ placer XXX ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
whatever	something



Question 9 $\bigoplus$ placer YYY ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
whatever	something
Question 10 $\bigoplus$ remplacer ZZZ ?	Pour le code de la question précédente, par quelles méthodes pourrait-on
something	whatever



Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

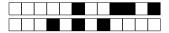
@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit





Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r \cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r \sin(\theta)$   $\theta = \tan^{-1} \frac{y}{x}$ 

En Java ce dernier est calculé par theta = Math.atan2(y, x)

- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.





Question 13	Continuez votre réponse à la question précédente ici si nécessaire.	



Question 14	Continuez	votre réponse	à la question	précédente	ici si nécessaire.	



Une classe peut étendre une classe abstraite

Question	15	Pour	la	question	précédente,	développez	la	méthode	Polar	#equal	s.	
									1 1	2	]3 [	4
=	classe pe	eut éte	endi	re plusieu:						:		
					sse et implén		eurs	interfaces	5			
					menter une in nir des consti							
					${ m constructeur}$							
=	_	=		_	méthodes pr							

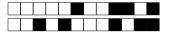


# Question $17 \oplus$ Soit le code ci-dessous :

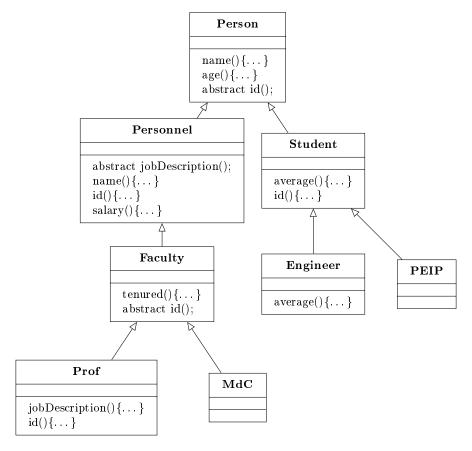
```
@FunctionalInterface
interface ComparePerson {
    /**
    * @return negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :



# Question 18 $\bigoplus$ Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

MdC	Person
Student	$\Box$ Faculty
Engineer	Personnel
PEIP	Prof

Introduction à la programmation orientée objet
9/01/2020

Cochez les cases en mettant une ×.

Le symbole ⊕ indique que la question peut avoir zéro, une ou plusieurs bonnes réponses.

Pour ces questions, cocher une bonne réponse apporte des points positifs ; cocher une mauvaise réponse peut apporter des points négatifs.

Dans tout le code, les package et les import sont censés être correctement déclarés.

Toute classe est supposée être dans le hon nackage dans le hon fichier, avec les hons

Toute classe est supposée être dans le bon package, dans le bon fichier, avec les bons import.Question 1 ( La classe donnée final class ClasseDonnee { ClasseDonnee() {} pourrait être sous-classée par extends pourrait être instanciée par new ClasseDonnee ClasseDonnee() Question 2 La classe donnée class ClasseDonnee {} pourrait être sous-classée par extends pourrait être instanciée ClasseDonnee ClasseDonnee() Question 3  $\bigoplus$  La classe donnée abstract class ClasseDonnee { ClasseDonnee() {} pourrait être sous-classée par extends pourrait être instanciée new ClasseDonnee ClasseDonnee() Question 4 Soit le code ci-dessous ; que se passe-t-il? Erreur de compilation public class Epsilon { Affiche "Caught exception" et "Finallied public void throwup() throws RuntimeException { exception" throw new RuntimeException(); Affiche seulement "Caught exception" public static void main(String[] args) { Epsilon epsilon = new Epsilon(); Affiche seulement "Finallied exception" try { epsilon.throwup(); Erreur d'exécution } catch (Exception e) { System.out.println("Caught exception"); finally { System.out.println("Finallied exception");

}



```
Affiche "Caught exception" et "Finallied
Soit le code ci-dessous ; que se passe-t-il?
                                                             exception"
public class Delta {
   public void throwup() throws Exception {
                                                            Affiche seulement "Finallied exception"
      throw new Exception();
                                                            Affiche seulement "Caught exception"
   public static void main(String[] args) {
      Delta delta = new Delta();
                                                            Erreur d'exécution
      try {
          delta.throwup();
                                                            Erreur de compilation
      } catch (Exception e) {
          System.out.println("Caught exception");
          return;
      } finally {
          System.out.println("Finallied exception");
}
Question 6
                                                            Erreur de compilation
Soit le code ci-dessous ; que se passe-t-il?
                                                             Affiche "Done"
public class Alpha {
   public void throwup() throws Exception {
                                                             Erreur d'exécution
      throw new Exception();
   public static void main(String[] args) {
```

#### Question 7 Soit le code :

Alpha alpha = new Alpha();
alpha.throwup();

System.out.println("Done");

```
interface Somethingable {
    void something();
}

interface Whateverable extends Somethingable {
    void whatever();
}

class TheThing implements Whateverable{
    PPP void something() {
        System.out.println("Somethinging");
    }

    PPP void whatever() {
        System.out.println("Whatevering");
    }
}
```

```
class Doofer {
    void doo(Somethingable sable) {
        sable.XXX();
}

void doo(Whateverable sable) {
        sable.YYY();
}

void fer(TheThing tt) {
        tt.ZZZ();
}

public static void main(String[] args) {
        Doofer doofer = new Doofer();
        TheThing the = new TheThing();
        doofer.doo(the);
        doofer.fer(the);
}
```

Par quels niveaux d'accès pourrait-on remplacer PPP ?

package-priv	protected public
Question 8 $\bigoplus$ placer XXX ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem
something	whatever

Question 9 $\bigoplus$ placer YYY ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
something	whatever
Question 10 ⊕ remplacer ZZZ ?	Pour le code de la question précédente, par quelles méthodes pourrait-on
whatever	something

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit





Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r \cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r \sin(\theta)$   $\theta = \tan^{-1} \frac{y}{x}$ 

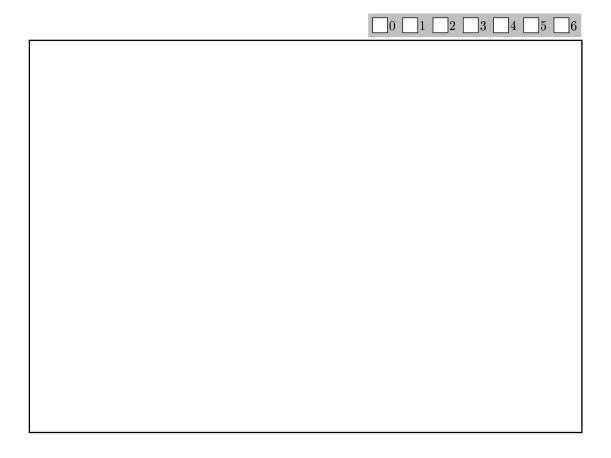
En Java ce dernier est calculé par theta = Math.atan2(y, x)

- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.





Question 13	Continuez	votre réponse à la question précédente ici si nécessaire.	



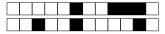
Question 14	Continuez	votre réponse à l	la question pré	cédente ici s	si nécessaire.	0 0



Question	<b>15</b>	Pour	la	question	précédente,	développez	la	méthode	Polar#equ	ıals.
									$\square 1 \square 2$	3 4
Question	<b>16</b> $\bigoplus$	Parn	ni l	es affirma	tions ci-desso	ous, cocher c	elle	s qui sont	vraies :	
Une o	classe p	eut éte	end	re plusieu	rs classes					
Une i	interfac	e peut	coı	ntenir des	méthodes pr	rivées				
Une o	classe a	bstrait	ер	eut conte	nir des const	ructeurs				
Une i	Une interface peut contenir des constructeurs									
Une o	Une classe abstraite peut implémenter une interface									

Une classe peut étendre une classe et implémenter plusieurs interfaces

Une classe peut étendre une classe abstraite



# Question $17 \oplus$ Soit le code ci-dessous :

```
@FunctionalInterface
interface ComparePerson {
    /**
    * Oreturn negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

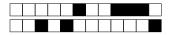
class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :

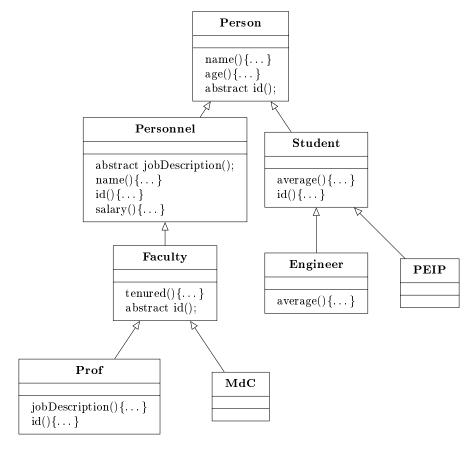
```
ComparePerson byAge = new ComparePersonByAge();

ComparePerson byAge = new ComparePersonByAge() {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
};

ComparePerson byAge = new ComparePersonByAge() {
    ComparePerson byAge = (p1, p2) -> p1.age - p2.age;
}
```



# Question 18 $\bigoplus$ Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

Person	MdC
Faculty	Student
Engineer	Personnel
PEIP	Prof

QCMTEST Nom et prénom: Introduction à la programmation orientée objet 9/01/2020 Groupe: ..... Cochez les cases en mettant une  $\times$ . Le symbole  $\bigoplus$  indique que la question peut avoir zéro, une ou plusieurs bonnes réponses. Pour ces questions, cocher une bonne réponse apporte des points positifs ; cocher une mauvaise réponse peut apporter des points négatifs. Dans tout le code, les package et les import sont censés être correctement déclarés. Toute classe est supposée être dans le bon package, dans le bon fichier, avec les bons import.Question 1 La classe donnée class ClasseDonnee { private ClasseDonnee() {} pourrait être sous-classée par extends pourrait être instanciée par new ClasseDonnee() ClasseDonnee Question 2 La classe donnée abstract class ClasseDonnee { ClasseDonnee() {} pourrait pourrait être sous-classée par extends être instanciée par new ClasseDonnee ClasseDonnee() Question 3 ( La classe donnée class ClasseDonnee {} pourrait être pourrait être sous-classée par extends instanciée par new ClasseDonnee() ClasseDonnee Question 4 Erreur d'exécution Soit le code ci-dessous ; que se passe-t-il? Affiche "Done" public class Beta {

Erreur de compilation

public void throwup() throws RuntimeException {

throw new RuntimeException();
}
public static void main(String[] args) {
Beta beta = new Beta();
beta.throwup();

System.out.println("Done");

}



Affiche seulement "Caught exception" Soit le code ci-dessous ; que se passe-t-il? public class Epsilon { Erreur d'exécution public void throwup() throws RuntimeException { Affiche seulement "Finallied exception" throw new RuntimeException(); public static void main(String[] args) { Erreur de compilation Epsilon epsilon = new Epsilon(); try { Affiche "Caught exception" et "Finallied epsilon.throwup(); exception" } catch (Exception e) { System.out.println("Caught exception"); } finally { System.out.println("Finallied exception"); } } Question 6  $\bigoplus$ Soit le code ci-dessous ; que se passe-t-il? Erreur d'exécution public class Alpha { Erreur de compilation public void throwup() throws Exception { throw new Exception(); Affiche "Done"

#### Question 7 Soit le code :

System.out.println("Done");

alpha.throwup();

}

public static void main(String[] args) {
 Alpha alpha = new Alpha();

```
interface Somethingable {
    void something();
}

interface Whateverable extends Somethingable {
    void whatever();
}

class TheThing implements Whateverable{
    PPP void something() {
        System.out.println("Somethinging");
    }

    PPP void whatever() {
        System.out.println("Whatevering");
    }
}
```

```
class Doofer {
    void doo(Somethingable sable) {
        sable.XXX();
    }

    void doo(Whateverable sable) {
        sable.YYY();
    }

    void fer(TheThing tt) {
        tt.ZZZ();
    }

    public static void main(String[] args) {
        Doofer doofer = new Doofer();
        TheThing the = new TheThing();
        doofer.doo(the);
        doofer.fer(the);
    }
}
```

Par quels niveaux d'accès pourrait-on remplacer PPP?

package-priv	rate public
protected	private
Question 8 $\oplus$ placer XXX ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-

whatever something

Question 9 $\bigoplus$ placer YYY ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
whatever	something
Question 10 $\bigoplus$ remplacer ZZZ ?	Pour le code de la question précédente, par quelles méthodes pourrait-on
something	whatever

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
}
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit



Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r \cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r \sin(\theta)$   $\theta = \tan^{-1} \frac{y}{x}$ 

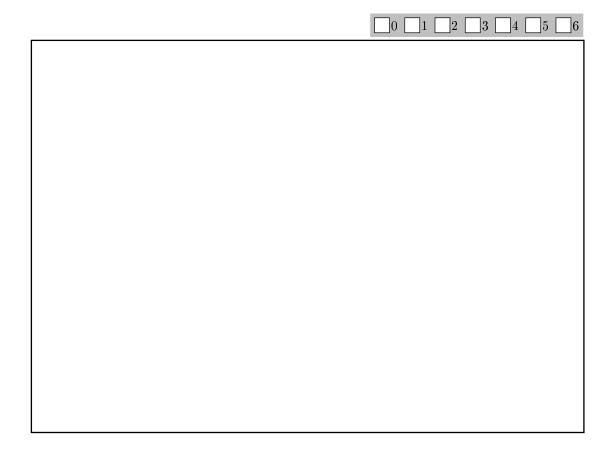
En Java ce dernier est calculé par theta = Math.atan2(y, x)

- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.





Question 13	Continuez	votre réponse	à la question	précédente i	ci si nécessaire.	0 0



Question 14	Continuez	votre réponse à l	la question pré	cédente ici s	si nécessaire.	0 0



Une classe peut étendre plusieurs classes

Question	15	Pour	la	question	précédente,	développez	la	méthode	Pola	r#eau	als	
& destion	10	rour	100	question	procedence,	четегорред	100					$\Box 4$
${f Question}$	16 ⊕	Parn	ni l	es affirma	tions ci-desso	ous, cocher c	elle	s qui sont	vraie	s :		
Une o	classe p	eut éte	$\operatorname{end}$	re une cla	sse abstraite							
Une o	classe a	bstrait	ер	eut conte	nir des const	ructeurs						
=		=			méthodes pr							
					constructeur							
	=				sse et implén menter une i	=	eurs	interfaces	5			
∪ пе с	rasse a	ustran	eр	ear mpre	шешег ипе 1	merrace						

# Question $17 \oplus$ Soit le code ci-dessous :

```
@FunctionalInterface
interface ComparePerson {
    /**
    * @return negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :

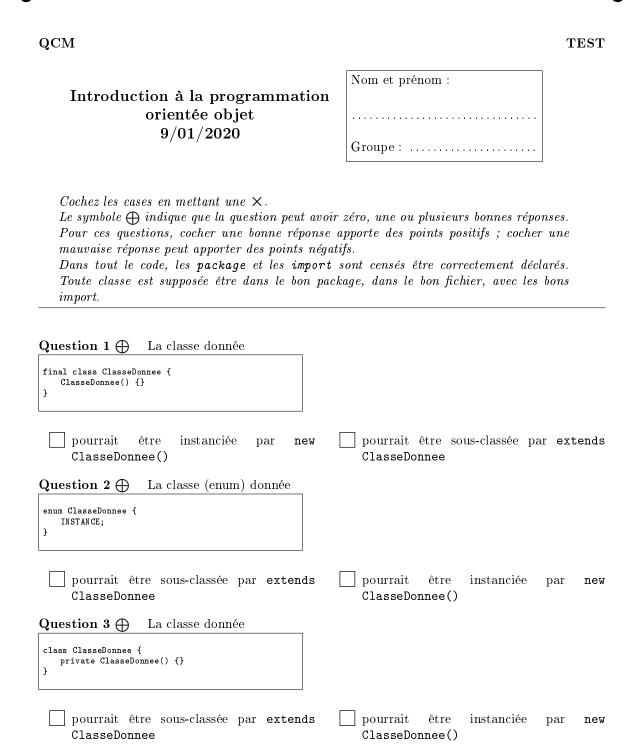


# Question 18 $\bigoplus$ Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

Person	$oxedsymbol{\square}$ Faculty
PEIP	Student
MdC	Personnel
Prof	Engineer



Soit le code ci-dessous ; que se passe-t-il ?	Erreur de compilation
<pre>public class Delta {     public void throwup() throws Exception {         throw new Exception();     }     public static void main(String[] args) {         Delta delta = new Delta();         try {             delta.throwup();         } catch (Exception e) {             System.out.println("Caught exception");             return;         } finally {                 System.out.println("Finallied exception");         }     } }</pre>	<ul> <li>□ Erreur d'exécution</li> <li>□ Affiche seulement "Caught exception"</li> <li>□ Affiche seulement "Finallied exception"</li> <li>□ Affiche "Caught exception" et "Finallied exception"</li> </ul>
Question $5 \oplus$	

```
Soit le code ci-dessous ; que se passe-t-il ?

public class Gamma {
    public void throwup() throws Exception {
        throw new Exception();
    }
    public static void main(String[] args) {
        Gamma gamma = new Gamma();
        try {
            gamma.throwup();
    } catch (Exception e) {
            System.out.println("Caught exception");
    } finally {
            System.out.println("Finallied exception");
    }
}
```

# Question 6 $\bigoplus$

Soit le code ci-dessous ; que se passe-t-il ?
public class Beta {
 public void throwup() throws RuntimeException {
 throw new RuntimeException();
 }
 public static void main(String[] args) {
 Beta beta = new Beta();
 beta.throwup();
 System.out.println("Done");
 }
}

## Erreur d'exécution

Affiche "Caught exception" et "Finallied exception"

\_\_\_ Affiche seulement "Finallied exception"

Erreur de compilation

Affiche seulement "Caught exception"

Erreur d'exécution

Erreur de compilation

Affiche "Done"



# Question 7 Soit le code :

```
interface Somethingable {
   void something();
}

interface Whateverable extends Somethingable {
   void whatever();
}

class TheThing implements Whateverable{
   PPP void something() {
       System.out.println("Somethinging");
   }

   PPP void whatever() {
       System.out.println("Whatevering");
   }
}
```

```
class Doofer {
    void doo(Somethingable sable) {
        sable.XXX();
    }

    void doo(Whateverable sable) {
        sable.YYY();
    }

    void fer(TheThing tt) {
        tt.ZZZ();
    }

    public static void main(String[] args) {
        Doofer doofer = new Doofer();
        TheThing the = new TheThing();
        doofer.doo(the);
        doofer.fer(the);
    }
}
```

Par quels niveaux d'accès pourrait-on remplacer PPP ?

protected public	<pre>package-private private</pre>
Question 8 $\bigoplus$ placer XXX ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
something	whatever
Question 9 $\bigoplus$ placer YYY ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
something	whatever
Question 10 $\bigoplus$ remplacer ZZZ ?	Pour le code de la question précédente, par quelles méthodes pourrait-on
whatever	something

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit





Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r\cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r\sin(\theta)$   $\theta = \tan^{-1}\frac{y}{x}$ 

En Java ce dernier est calculé par theta = Math.atan2(y, x)

- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.





Question 13	Continuez	votre réponse à la question précédente ici si nécessaire.	



Question 14	Continuez	votre réponse	à la question	précédente i	ci si nécessaire.	



Question 15 Pour la question précédente, développez la méthode Polar#equals.  $\boxed{0}$   $\boxed{1}$   $\boxed{2}$   $\boxed{3}$ Question 16  $\bigoplus$ Parmi les affirmations ci-dessous, cocher celles qui sont vraies : Une interface peut contenir des méthodes privées Une classe abstraite peut contenir des constructeurs Une classe abstraite peut implémenter une interface Une classe peut étendre plusieurs classes Une classe peut étendre une classe abstraite Une interface peut contenir des constructeurs Une classe peut étendre une classe et implémenter plusieurs interfaces



# Question $17 \oplus$ Soit le code ci-dessous :

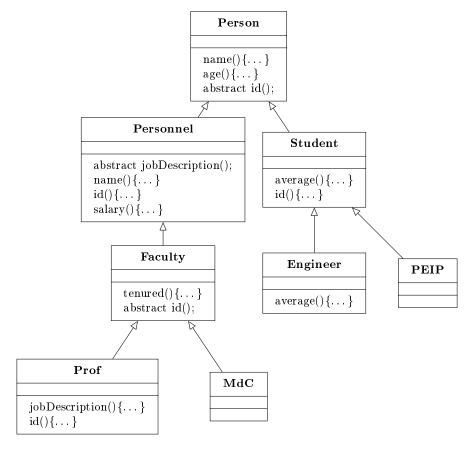
```
@FunctionalInterface
interface ComparePerson {
    /**
    * Oreturn negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :



Question 18  $\bigoplus$  Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

	Personnel	Student
	Faculty	MdC
	PEIP	Prof
ſ	Person	Engineer

QCMTEST Nom et prénom: Introduction à la programmation orientée objet 9/01/2020 Groupe: ..... Cochez les cases en mettant une  $\times$ . Le symbole  $\bigoplus$  indique que la question peut avoir zéro, une ou plusieurs bonnes réponses. Pour ces questions, cocher une bonne réponse apporte des points positifs ; cocher une mauvaise réponse peut apporter des points négatifs. Dans tout le code, les package et les import sont censés être correctement déclarés. Toute classe est supposée être dans le bon package, dans le bon fichier, avec les bons import.Question 1 La classe donnée final class ClasseDonnee { ClasseDonnee() {} pourrait être instanciée pourrait être sous-classée par extends par new ClasseDonnee ClasseDonnee() Question 2 La classe donnée class ClasseDonnee {} pourrait être sous-classée par extends pourrait être instanciée par new ClasseDonnee() ClasseDonnee Question 3 La classe (enum) donnée enum ClasseDonnee { INSTANCE; pourrait être sous-classée par extends pourrait être instanciée new ClasseDonnee ClasseDonnee() Erreur de compilation Affiche seulement "Caught exception"

Question 4 Soit le code ci-dessous ; que se passe-t-il ? public class Gamma { public void throwup() throws Exception { Affiche "Caught exception" et "Finallied throw new Exception(); public static void main(String[] args) { Gamma gamma = new Gamma(); Erreur d'exécution try { gamma.throwup(); Affiche seulement "Finallied exception" } catch (Exception e) { System.out.println("Caught exception"); finally { System.out.println("Finallied exception"); } }



# Question 5 $\bigoplus$

Soit le code ci-dessous ; que se passe-t-il ?	Erreur d'exécution
<pre>public class Alpha {     public void throwup() throws Exception {         throw new Exception();     }     public static void main(String[] args) {         Alpha alpha = new Alpha();         alpha.throwup();         System.out.println("Done");     } }</pre>	☐ Affiche "Done" ☐ Erreur de compilation
Question 6 $\bigoplus$	
Soit le code ci-dessous ; que se passe-t-il ?	Affiche "Caught exception" et "Finallied exception"
<pre>public void throwup() throws Exception {     throw new Exception();</pre>	Affiche seulement "Finallied exception"
} public static void main(String[] args) {	Erreur d'exécution
<pre>Delta delta = new Delta(); try {</pre>	Affiche seulement "Caught exception"
delta.throwup(); } catch (Exception e) {	Erreur de compilation

#### Question 7 Soit le code:

} catch (Exception e) {

return; } finally {

}

System.out.println("Caught exception");

System.out.println("Finallied exception");

```
interface Somethingable {
   void something();
interface Whateverable extends Somethingable {
   void whatever();
class The Thing implements Whateverable {
  PPP void something() {
    System.out.println("Somethinging");
   PPP void whatever() {
       System.out.println("Whatevering");
}
```

```
class Doofer {
   void doo(Somethingable sable) {
       sable.XXX();
    void doo(Whateverable sable) {
        sable.YYY();
   void fer(TheThing tt) {
      tt.ZZZ();
   public static void main(String[] args) {
        Doofer doofer = new Doofer();
TheThing the = new TheThing();
doofer.doo(the);
        doofer.fer(the);
}
```

Par quels niveaux d'accès pourrait-on remplacer PPP ?

private package-pri	vate		public protected
Question 8 $\bigoplus$ blacer XXX ?	Pour le code de la question pré	céde	nte, par quelles méthodes pourrait-on rem
whatever			something

Question 9 ⊕ placer YYY ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
whatever	something
Question 10 $\bigoplus$ remplacer ZZZ ?	Pour le code de la question précédente, par quelles méthodes pourrait-on
whatever	something



Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit





Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r\cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r\sin(\theta)$   $\theta = \tan^{-1}\frac{y}{x}$ 

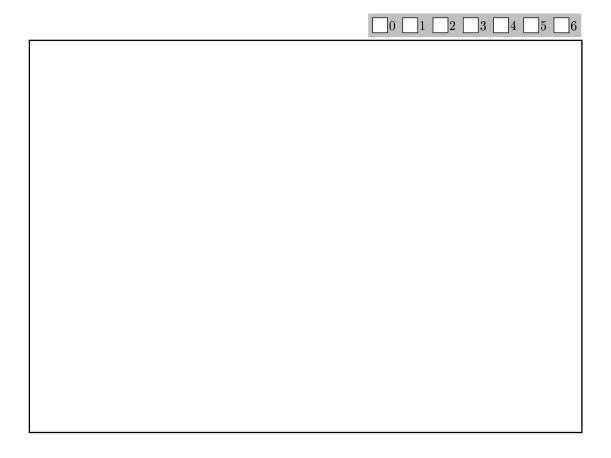
En Java ce dernier est calculé par theta = Math.atan2(y, x)

- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.





Question 13	Continuez	votre réponse	à la question	précédente i	ci si nécessaire.	0 0



Question 14	Continuez	votre réponse	à la question	précédente	ici si nécessaire.	



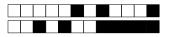
Une classe peut étendre plusieurs classes
Une classe peut étendre une classe abstraite

Question	15	Pour	la	question	précédente,	développez	la	$     \text{m\'ethode} $		quals. $2  \boxed{3}$	4
Question	16 🕀	Parn	ni le	es affirma	tions ci-desso	ous, cocher c	elle	s qui sont	vraies :		
Une	classe p	eut éte	endi	re une cla	sse et implén	nenter plusie	eurs	interfaces			
Une		e peut	cor	ntenir des	constructeur	`S					
Une	interface	e peut	cor	ntenir des	méthodes pr	rivées					
Une	classe al	bstrait	ер	eut conte	nir des consti	ructeurs					
$\Box$ Une	classe al	bstrait	ер	eut implé	menter une i	${ m nterface}$					

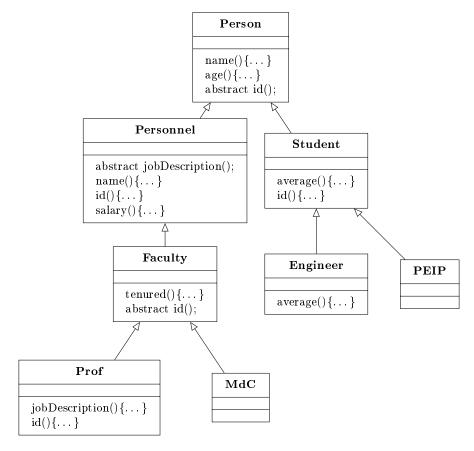


# Question $17 \oplus$ Soit le code ci-dessous :

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :



Question 18  $\bigoplus$  Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

Engineer	Prof
Personnel	Person
Student	Faculty
MdC	PEIP

QCMTEST Nom et prénom: Introduction à la programmation orientée objet 9/01/2020 Groupe: ..... Cochez les cases en mettant une  $\times$ . Le symbole  $\bigoplus$  indique que la question peut avoir zéro, une ou plusieurs bonnes réponses. Pour ces questions, cocher une bonne réponse apporte des points positifs ; cocher une mauvaise réponse peut apporter des points négatifs. Dans tout le code, les package et les import sont censés être correctement déclarés. Toute classe est supposée être dans le bon package, dans le bon fichier, avec les bons import.Question 1 

La classe donnée abstract class ClasseDonnee { ClasseDonnee() {} pourrait être sous-classée par extends pourrait être instanciée par new ClasseDonnee() ClasseDonnee Question 2 La classe donnée class ClasseDonnee { private ClasseDonnee() {} pourrait pourrait être sous-classée par extends être instanciée par new ClasseDonnee ClasseDonnee() Question 3 ( La classe donnée class ClasseDonnee {} pourrait être pourrait être sous-classée par extends instanciée par new ClasseDonnee() ClasseDonnee Question 4 Affiche "Done" Soit le code ci-dessous ; que se passe-t-il? public class Beta { Erreur d'exécution

Erreur de compilation

public void throwup() throws RuntimeException {

throw new RuntimeException();
}
public static void main(String[] args) {
Beta beta = new Beta();
beta.throwup();

System.out.println("Done");

}



# Question 5 $\bigoplus$

Affiche "Caught exception" et "Finallied Soit le code ci-dessous ; que se passe-t-il? exception" public class Epsilon { public void throwup() throws RuntimeException { Erreur d'exécution throw new RuntimeException(); Erreur de compilation public static void main(String[] args) { Epsilon epsilon = new Epsilon(); Affiche seulement "Caught exception" try { epsilon.throwup(); Affiche seulement "Finallied exception" } catch (Exception e) { System.out.println("Caught exception"); } finally { System.out.println("Finallied exception"); } } Question 6  $\bigoplus$ Affiche "Done" Soit le code ci-dessous ; que se passe-t-il? Erreur d'exécution public class Alpha { public void throwup() throws Exception { throw new Exception(); Erreur de compilation

#### Question 7 Soit le code :

System.out.println("Done");

alpha.throwup();

}

public static void main(String[] args) {
 Alpha alpha = new Alpha();

```
interface Somethingable {
    void something();
}

interface Whateverable extends Somethingable {
    void whatever();
}

class TheThing implements Whateverable{
    PPP void something() {
        System.out.println("Somethinging");
    }

    PPP void whatever() {
        System.out.println("Whatevering");
    }
}
```

```
class Doofer {
    void doo(Somethingable sable) {
        sable.XXX();
    }

    void doo(Whateverable sable) {
        sable.YYY();
    }

    void fer(TheThing tt) {
        tt.ZZZ();
    }

    public static void main(String[] args) {
        Doofer doofer = new Doofer();
        TheThing the = new TheThing();
        doofer.doo(the);
        doofer.fer(the);
    }
}
```

Par quels niveaux d'accès pourrait-on remplacer PPP?

package-pri	vate public
$\square$ private	protected
Question $8 \oplus$ placer XXX ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-

something whatever

Question 9 ⊕ placer YYY ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
whatever	<pre>something</pre>
Question 10 $\bigoplus$ remplacer ZZZ ?	Pour le code de la question précédente, par quelles méthodes pourrait-on

 $\square$  something

\_\_\_ whatever

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
    public double re;
    public Complex(double a, double b) {
        re = a;
        im = b;
    }

    public double getRe() {
        return re;
    }

    public void setRe(double re) {
        this.re = re;
    }

    public double getIm() {
        return im;
    }

    public void setIm(double im) {
        this.im = im;
    }
}
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

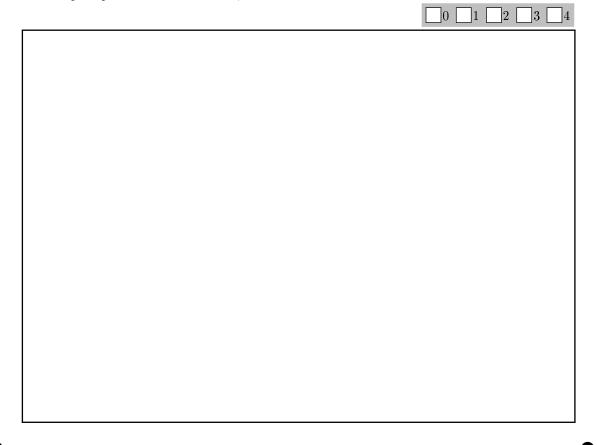
public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

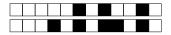
@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit





Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r\cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r\sin(\theta)$   $\theta = \tan^{-1}\frac{y}{x}$ 

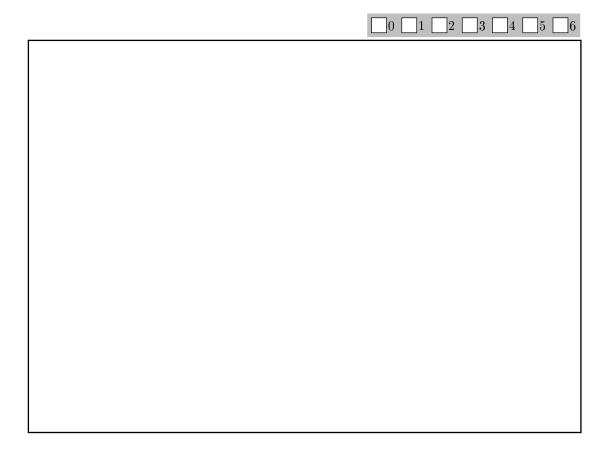
En Java ce dernier est calculé par theta = Math.atan2(y, x)

- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.





Question 13	Continuez	votre réponse	à la question	précédente i	ci si nécessaire.	0 0



Question 14	Continuez	votre réponse à la question précédente ici si nécessaire.	



Une classe abstraite peut contenir des constructeurs
Une classe abstraite peut implémenter une interface

Question 15 Pour la question précédente, développez la méthode Polar#equals.  $\boxed{0}$   $\boxed{1}$   $\boxed{2}$   $\boxed{3}$ Une classe peut étendre une classe abstraite Une interface peut contenir des constructeurs Une classe peut étendre plusieurs classes Une classe peut étendre une classe et implémenter plusieurs interfaces Une interface peut contenir des méthodes privées



# Question $17 \oplus$ Soit le code ci-dessous :

```
@FunctionalInterface
interface ComparePerson {
    /**
    * @return negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

```
class Person {
    private int age;
    private String name;

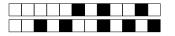
Person(String name, int age) {
        this.age = age;
        this.name = name;
    }
}

class Main {
    public static void main(String[] args) {

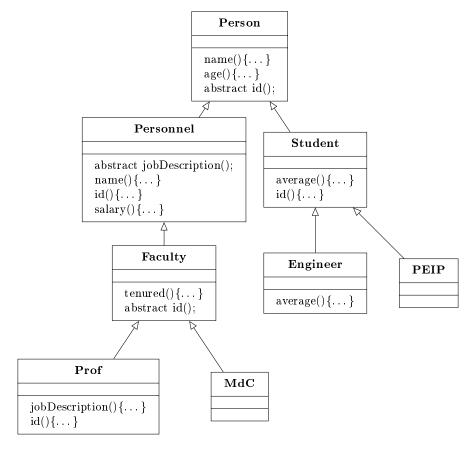
        // code extract here

        System.out.print(byAge.compare(
            new Person("Fred", 23),
            new Person("Not Fred", 24)));
        System.out.print(byAge.compare(
            new Person("Fred", 23),
            new Person("Not Fred", 23));
        System.out.print(byAge.compare(
            new Person("Fred", 23),
            new Person("Fred", 23),
            new Person("Fred", 23),
            new Person("Fred", 23),
            new Person("Not Fred", 22)));
}
```

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :

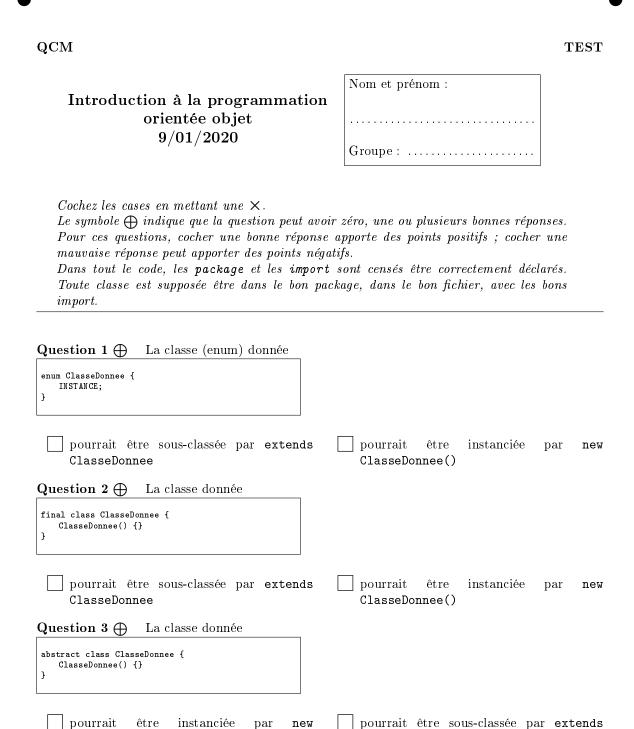


Question 18  $\bigoplus$  Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

Prof	Personnel
Person	PEIP
Student	$oxedsymbol{\Box}$ Faculty
MdC	Engineer



ClasseDonnee

ClasseDonnee()

```
Erreur de compilation
Soit le code ci-dessous ; que se passe-t-il?
public class Gamma {
                                                              Affiche seulement "Finallied exception"
   public void throwup() throws Exception {
      throw new Exception();
   public static void main(String[] args) {
      Gamma gamma = new Gamma();
                                                              Erreur d'exécution
      try {
          gamma.throwup();
      } catch (Exception e) {
          System.out.println("Caught exception");
       } finally {
          System.out.println("Finallied exception");
}
```

# Question 5 $\bigoplus$

```
Soit le code ci-dessous ; que se passe-t-il?
public class Delta {
   public void throwup() throws Exception {
       throw new Exception();
   public static void main(String[] args) {
   Delta delta = new Delta();
       try {
           delta.throwup();
        } catch (Exception e) {
            System.out.println("Caught exception");
            return;
       } finally {
   System.out.println("Finallied exception");
```

# Question 6 $\bigoplus$

Soit le code ci-dessous ; que se passe-t-il? public class Beta { public void throwup() throws RuntimeException { throw new RuntimeException(); public static void main(String[] args) { Beta beta = new Beta(); beta.throwup(); System.out.println("Done"); }

Effeur	de compna	uton
A 600 - 1	1 4	0172: 11: d

_	THITCH	bearemen	o i mama	0210	cpolon
	Affiche	"Caught	exception"	$\operatorname{et}$	"Finallied"
	exception	on"			

Affiche seulement "Caught exception"

Affiche seulement "Finallied exception"

Affiche "Caught exception" et "Finallied exception"

Affiche seulement "Caught exception"

Erreur d'exécution

Erreur de compilation

Erreur d'exécution

Affiche "Done"

Erreur de compilation



# Question 7 Soit le code :

```
interface Somethingable {
    void something();
}

interface Whateverable extends Somethingable {
    void whatever();
}

class TheThing implements Whateverable{
    PPP void something() {
        System.out.println("Somethinging");
    }

    PPP void whatever() {
        System.out.println("Whatevering");
    }
}
```

```
class Doofer {
  void doo(Somethingable sable) {
    sable.XXX();
  }
  void doo(Whateverable sable) {
    sable.YYY();
  }
  void fer(TheThing tt) {
    tt.ZZZ();
  }
  public static void main(String[] args) {
    Doofer doofer = new Doofer();
    TheThing the = new TheThing();
    doofer.doo(the);
    doofer.fer(the);
  }
}
```

Par quels niveaux d'accès pourrait-on remplacer PPP? package-private private public protected Question 8 Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer XXX? whatever something Question 9 Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer YYY? whatever something Question 10  $\bigoplus$ Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer ZZZ? whatever something

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit





Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r \cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r \sin(\theta)$   $\theta = \tan^{-1} \frac{y}{x}$ 

En Java ce dernier est calculé par theta = Math.atan2(y, x)

- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.





Question 13	Continuez votre réponse à la question précédente ici si nécessaire.	



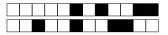
Continuez votre réponse à la question précédente ici si nécessaire.	
	Continuez votre réponse à la question précédente ici si nécessaire.



Une interface peut contenir des constructeurs

Une classe peut étendre une classe et implémenter plusieurs interfaces

Question	15	Pour	la	question	précédente,	développez	la	méthode Polar#equals.
Question	<b>16</b> $\oplus$	Parm	i le	es affirma	tions ci-desso	ous, cocher c	elle	s qui sont vraies:
					nir des const			
					sse abstraite			
				re plusieu				
					menter une i			
Une i		e peut	cor	ntenir des	méthodes pr	rivées		



#### Question $17 \oplus$ Soit le code ci-dessous :

```
@FunctionalInterface
interface ComparePerson {
    /**
    * Oreturn negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :

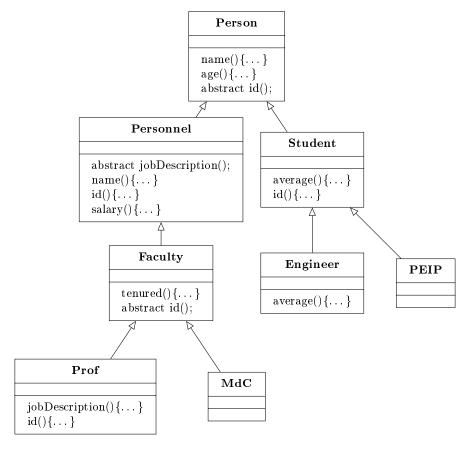
```
ComparePerson byAge = new ComparePersonByAge();

ComparePerson byAge = new ComparePersonByAge();

ComparePerson byAge = new ComparePersonByAge() {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
};
```



## Question 18 $\bigoplus$ Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

Person	MdC
Engineer	Personnel
Faculty	Student
Prof	PEIP

QCMTEST Nom et prénom: Introduction à la programmation orientée objet 9/01/2020 Groupe: ..... Cochez les cases en mettant une  $\times$ . Le symbole  $\bigoplus$  indique que la question peut avoir zéro, une ou plusieurs bonnes réponses. Pour ces questions, cocher une bonne réponse apporte des points positifs ; cocher une mauvaise réponse peut apporter des points négatifs. Dans tout le code, les package et les import sont censés être correctement déclarés. Toute classe est supposée être dans le bon package, dans le bon fichier, avec les bons import.Question 1 ( La classe donnée class ClasseDonnee { private ClasseDonnee() {} pourrait être pourrait être sous-classée par extends instanciée par new ClasseDonnee ClasseDonnee() Question 2 La classe donnée abstract class ClasseDonnee { ClasseDonnee() {} } pourrait être sous-classée par extends pourrait être instanciée par new ClasseDonnee ClasseDonnee() Question 3 ( La classe donnée class ClasseDonnee {} pourrait être sous-classée par extends pourrait être instanciée new ClasseDonnee() ClasseDonnee Question 4 Soit le code ci-dessous ; que se passe-t-il ? Affiche "Caught exception" et "Finallied exception" public class Epsilon { public void throwup() throws RuntimeException { Affiche seulement "Caught exception" throw new RuntimeException();

public static void main(String[] args) {
 Epsilon epsilon = new Epsilon();

System.out.println("Caught exception");

System.out.println("Finallied exception");

epsilon.throwup();

} catch (Exception e) {

try {

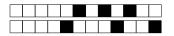
}

finally {

Erreur de compilation

Erreur d'exécution

Affiche seulement "Finallied exception"



## Question 5 $\bigoplus$

Affiche "Done" Soit le code ci-dessous ; que se passe-t-il? public class Beta { Erreur d'exécution public void throwup() throws RuntimeException { Erreur de compilation throw new RuntimeException(); public static void main(String[] args) { Beta beta = new Beta(); beta.throwup(); System.out.println("Done"); } Question 6 Soit le code ci-dessous ; que se passe-t-il? Affiche "Caught exception" et "Finallied exception" public class Gamma { public void throwup() throws Exception { Erreur de compilation throw new Exception(); Affiche seulement "Caught exception" public static void main(String[] args) { Gamma gamma = new Gamma(); Erreur d'exécution try { gamma.throwup(); Affiche seulement "Finallied exception" } catch (Exception e) { System.out.println("Caught exception"); } finally { System.out.println("Finallied exception");

#### Question 7 Soit le code :

}

```
interface Somethingable {
    void something();
}

interface Whateverable extends Somethingable {
    void whatever();
}

class TheThing implements Whateverable{
    PPP void something() {
        System.out.println("Somethinging");
    }

    PPP void whatever() {
        System.out.println("Whatevering");
    }
}
```

```
class Doofer {
    void doo(Somethingable sable) {
        sable.XXX();
    }

    void doo(Whateverable sable) {
        sable.YYY();
    }

    void fer(TheThing tt) {
        tt.ZZZ();
    }

    public static void main(String[] args) {
        Doofer doofer = new Doofer();
        TheThing the = new TheThing();
        doofer.doo(the);
        doofer.fer(the);
    }
}
```

Par quels niveaux d'accès pourrait-on remplacer PPP?

public	protected
package-pri	vate private
Question 8 $\bigoplus$ placer XXX ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem
whatever	something

+84	/2	10	
+84	/.≾	/ ಜ	+

Question 9 $\bigoplus$ placer YYY ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
whatever	<pre>something</pre>
Question 10 $\bigoplus$ remplacer ZZZ ?	Pour le code de la question précédente, par quelles méthodes pourrait-on
something	whatever

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit





Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r \cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r \sin(\theta)$   $\theta = \tan^{-1} \frac{y}{x}$ 

En Java ce dernier est calculé par theta = Math.atan2(y, x)

- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.





Question 13	Continuez votre réponse à la question précédente ici si nécessaire.	



Question 14	Continuez	votre réponse à l	la question pré	cédente ici s	i nécessaire.	0 0



Question 15 Pour la question précédente, développez la méthode Polar#equals.  $\boxed{0}$   $\boxed{1}$   $\boxed{2}$   $\boxed{3}$ Question 16  $\bigoplus$ Parmi les affirmations ci-dessous, cocher celles qui sont vraies : Une interface peut contenir des méthodes privées Une interface peut contenir des constructeurs Une classe peut étendre une classe abstraite Une classe abstraite peut implémenter une interface Une classe peut étendre une classe et implémenter plusieurs interfaces Une classe abstraite peut contenir des constructeurs Une classe peut étendre plusieurs classes



#### Question $17 \oplus$ Soit le code ci-dessous :

```
@FunctionalInterface
interface ComparePerson {
    /**
    * Oreturn negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :

```
ComparePerson byAge = new ComparePersonByAge() {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
};

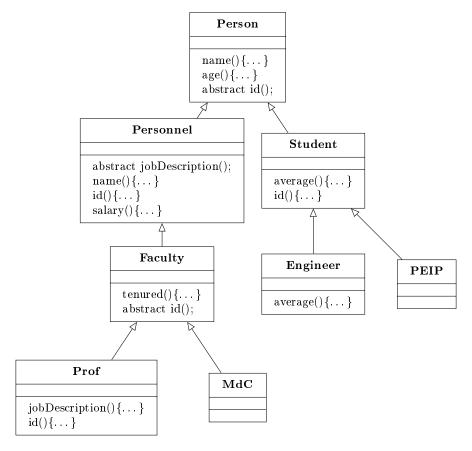
ComparePerson byAge = new ComparePersonByAge::compare;

ComparePerson byAge = new ComparePersonByAge();

ComparePerson byAge = new ComparePersonByAge();
```



## Question 18 $\bigoplus$ Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

Student	MdC
Person	PEIP
Engineer	Faculty
Prof	Personnel

QCMTEST Nom et prénom: Introduction à la programmation orientée objet 9/01/2020Groupe: ..... Cochez les cases en mettant une  $\times$ . Le symbole  $\bigoplus$  indique que la question peut avoir zéro, une ou plusieurs bonnes réponses. Pour ces questions, cocher une bonne réponse apporte des points positifs ; cocher une mauvaise réponse peut apporter des points négatifs. Dans tout le code, les package et les import sont censés être correctement déclarés. Toute classe est supposée être dans le bon package, dans le bon fichier, avec les bons import.Question 1 La classe (enum) donnée enum ClasseDonnee { INSTANCE; pourrait pourrait être sous-classée par extends être instanciée new ClasseDonnee ClasseDonnee() Question 2 ( La classe donnée final class ClasseDonnee { ClasseDonnee() {} pourrait être pourrait être sous-classée par extends instanciée par new ClasseDonnee ClasseDonnee() Question 3 ( La classe donnée class ClasseDonnee { private ClasseDonnee() {} pourrait pourrait être sous-classée par extends être instanciée par new ClasseDonnee ClasseDonnee()

Soit le code ci-dessous ; que se passe-t-il ?	Affiche seulement "Finallied exception"
<pre>public class Delta {     public void throwup() throws Exception {         throw new Exception();     }     public static void main(String[] args) {         Delta delta = new Delta();         try {             delta.throwup();     } catch (Exception e) {             System.out.println("Caught exception");             return;     } finally {             System.out.println("Finallied exception");         }     } }</pre>	<ul> <li>Erreur de compilation</li> <li>Erreur d'exécution</li> <li>Affiche seulement "Caught exception"</li> <li>Affiche "Caught exception" et "Finallied exception"</li> </ul>
Question $5 \oplus$	
Soit le code ci-dessous : que se passe-t-il ?	Affiche "Done"

## Soit le code ci-dessous ; que se passe-t-il?

```
public class Alpha {
   public void throwup() throws Exception {
       throw new Exception();
   public static void main(String[] args) {
     Alpha alpha = new Alpha();
     alpha.throwup();
     System.out.println("Done");
```

## Question 6

Soit le code ci-dessous ; que se passe-t-il ? public class Epsilon { public void throwup() throws RuntimeException { throw new RuntimeException(); public static void main(String[] args) {
 Epsilon epsilon = new Epsilon(); try { epsilon.throwup(); } catch (Exception e) { System.out.println("Caught exception"); } finally { System.out.println("Finallied exception"); }

Erreur de compilation

Erreur d'exécution

Affiche "Caught exception" et "Finallied exception" Erreur de compilation Affiche seulement "Caught exception" Erreur d'exécution Affiche seulement "Finallied exception"

#### Question 7 Soit le code :

```
interface Somethingable {
   void something();
}

interface Whateverable extends Somethingable {
   void whatever();
}

class TheThing implements Whateverable{
   PPP void something() {
       System.out.println("Somethinging");
   }

   PPP void whatever() {
       System.out.println("Whatevering");
   }
}
```

<pre>class Doofer {   void doo(Somethingable sable) {     sable.XXX(); }</pre>
<pre>void doo(Whateverable sable) {     sable.YYY(); }</pre>
<pre>void fer(TheThing tt) {     tt.ZZZ(); }</pre>
<pre>public static void main(String[] args) {     Doofer doofer = new Doofer();     TheThing the = new TheThing();     doofer.doo(the);     doofer.fer(the); }</pre>
}

Par quels niveaux d'accès pourrait-on remplacer PPP?  $\rfloor$  protected private public package-private Question 8  $\bigoplus$ Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer XXX? whatever something Question 9 Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer YYY? something whatever Question 10  $\bigoplus$ Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer ZZZ? whatever something

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

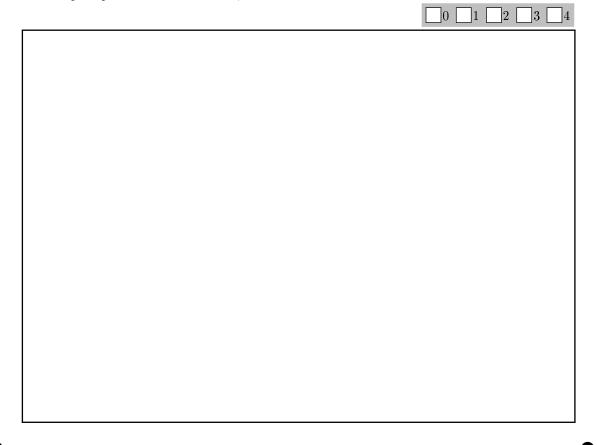
public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit



Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r \cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r \sin(\theta)$   $\theta = \tan^{-1} \frac{y}{x}$ 

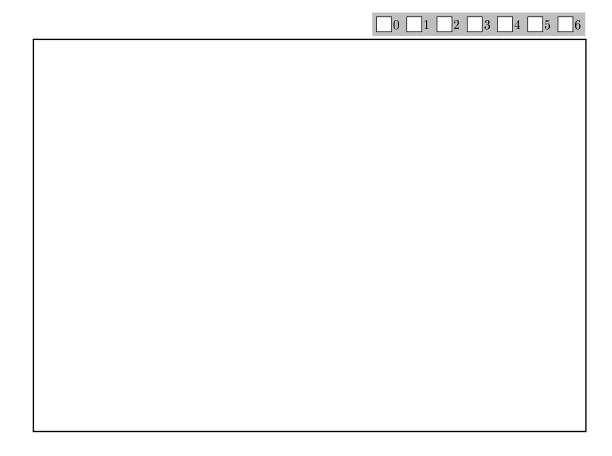
En Java ce dernier est calculé par theta = Math.atan2(y, x)

- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.







Question 14	Continuez	votre réponse	à la question	précédente i	ci si nécessaire.	



Une classe peut étendre plusieurs classes

Question	15	Pour	la	question	précédente,	développez	la	méthode	Polar	#equa	als.	
									01		3	$\Box 4$
Question	16 <b>⊕</b>	Parn	ni l	es affirma	tions ci-desso	ous, cocher c	elles	qui sont	vraies	:		
Une o	classe p	eut éte	end	re une cla	sse et implén	nenter plusie	eurs	interfaces	5			
Une o	classe al	bstrait	ер	eut implé	menter une i	nterface						
					constructeur							
_					nir des consti							
					méthodes pr	uvėes						
$\square$ one of	rasse p	eut ete	end:	re une cia	sse abstraite							



## Question $17 \oplus$ Soit le code ci-dessous :

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :

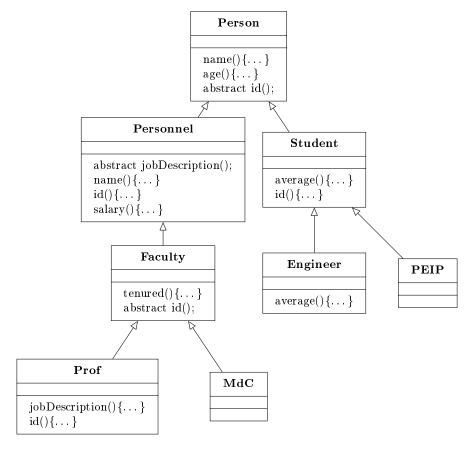
```
ComparePerson byAge = new ComparePersonByAge();

ComparePerson byAge = (p1, p2) -> p1.age - p2.age;

ComparePerson byAge = new ComparePersonByAge() {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
};
```



## Question 18 $\bigoplus$ Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

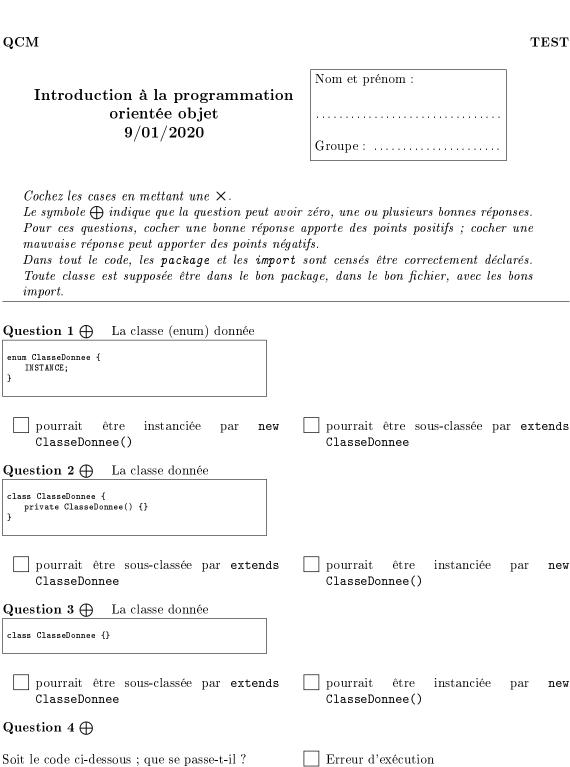
Personnel	Person
Prof	Engineer
MdC	Faculty
PEIP	Student

Affiche seulement "Caught exception"

Affiche "Caught exception" et "Finallied

Affiche seulement "Finallied exception"

Erreur de compilation



public class Epsilon {
 public void throwup() throws RuntimeException {
 throw new RuntimeException();
 }
 public static void main(String[] args) {
 Epsilon epsilon = new Epsilon();
 try {
 epsilon.throwup();
 } catch (Exception e) {
 System.out.println("Caught exception");
 } finally {
 System.out.println("Finallied exception");
 }
 }
}



```
Affiche seulement "Finallied exception"
Soit le code ci-dessous ; que se passe-t-il?
                                                             Affiche "Caught exception" et "Finallied
public class Gamma {
   public void throwup() throws Exception {
                                                             exception"
      throw new Exception();
                                                          Affiche seulement "Caught exception"
   public static void main(String[] args) {
      Gamma gamma = new Gamma();
                                                             Erreur de compilation
      try {
         gamma.throwup();
                                                             Erreur d'exécution
      } catch (Exception e) {
         System.out.println("Caught exception");
      } finally {
         System.out.println("Finallied exception");
   }
}
```

#### Question 6 $\bigoplus$

```
Soit le code ci-dessous ; que se passe-t-il ?

public class Delta {
    public void throwup() throws Exception {
        throw new Exception();
    }
    public static void main(String[] args) {
        Delta delta = new Delta();
        try {
            delta.throwup();
      } catch (Exception e) {
            System.out.println("Caught exception");
            return;
      } finally {
            System.out.println("Finallied exception");
        }
    }
}
```

# Affiche seulement "Finallied exception"

Erreur de compilation

Erreur d'exécution

Affiche "Caught exception" et "Finallied exception"

Affiche seulement "Caught exception"

#### **Question 7** Soit le code :

```
interface Somethingable {
    void something();
}

interface Whateverable extends Somethingable {
    void whatever();
}

class TheThing implements Whateverable{
    PPP void something() {
        System.out.println("Somethinging");
    }

    PPP void whatever() {
        System.out.println("Whatevering");
    }
}
```

```
class Doofer {
    void doo(Somethingable sable) {
        sable.XXX();
    }

    void doo(Whateverable sable) {
        sable.YYY();
    }

    void fer(TheThing tt) {
        tt.ZZZ();
    }

    public static void main(String[] args) {
        Doofer doofer = new Doofer();
        TheThing the = new TheThing();
        doofer.doo(the);
        doofer.fer(the);
    }
}
```

Par quels niveaux d'accès pourrait-on remplacer PPP?

 ☐ private
 ☐ protected

 ☐ public
 ☐ package-private

Question 8 $\oplus$ placer XXX ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
something	whatever
Question 9 $\bigoplus$ placer YYY ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
something	whatever
Question 10 ⊕ remplacer ZZZ ?	Pour le code de la question précédente, par quelles méthodes pourrait-on

something whatever

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit



Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r \cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r \sin(\theta)$   $\theta = \tan^{-1} \frac{y}{x}$ 

En Java ce dernier est calculé par theta = Math.atan2(y, x)

- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.





Question 13	Continuez	votre réponse	à la question	précédente i	ci si nécessaire.	0 0



Question 14	Continuez votre réponse à la question précédente ici si nécessaire.	0 0
1		



Une classe peut étendre plusieurs classes

Une classe abstraite peut implémenter une interface

${f Question}$	15	Pour la	a questi	on précédente	, développez	la	méthode Polar#equal	
								<u></u>
	_							
Question	16 ⊕	Parmi	les affiri	mations ci-des	sous, cocher c	elle	s qui sont vraies :	
$\square$ Une i	interface	e peut c	ontenir d	les constructe	ırs			
Une o	classe pe	eut éten	dre une	classe et implé	ementer plusie	eurs	interfaces	
Une o	classe pe	eut éten	dre une	classe abstrait	е			
Une i	Une interface peut contenir des méthodes privées							
Une o	Une classe abstraite peut contenir des constructeurs							



#### Question $17 \oplus$ Soit le code ci-dessous :

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :

```
ComparePerson byAge = new ComparePersonByAge();

ComparePerson byAge = (p1, p2) -> p1.age - p2.age;

ComparePerson byAge = new ComparePersonByAge() {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
};
```



## Question 18 $\bigoplus$ Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

<pre>Faculty</pre>	MdC
Student	Person
PEIP	Personnel
Engineer	Prof

QCMTEST Nom et prénom: Introduction à la programmation orientée objet 9/01/2020 Groupe: ..... Cochez les cases en mettant une  $\times$ . Le symbole  $\bigoplus$  indique que la question peut avoir zéro, une ou plusieurs bonnes réponses. Pour ces questions, cocher une bonne réponse apporte des points positifs ; cocher une mauvaise réponse peut apporter des points négatifs. Dans tout le code, les package et les import sont censés être correctement déclarés. Toute classe est supposée être dans le bon package, dans le bon fichier, avec les bons import.Question 1 

La classe donnée final class ClasseDonnee { ClasseDonnee() {} pourrait être sous-classée par extends pourrait être instanciée par new ClasseDonnee() ClasseDonnee Question 2 La classe donnée abstract class ClasseDonnee { ClasseDonnee() {} } pourrait être sous-classée par extends pourrait être instanciée par new ClasseDonnee() ClasseDonnee La classe (enum) donnée Question 3 ( enum ClasseDonnee { INSTANCE; pourrait être sous-classée par extends pourrait être instanciée par new ClasseDonnee ClasseDonnee() Question 4 Erreur de compilation Soit le code ci-dessous ; que se passe-t-il?

Affiche "Done"

Erreur d'exécution

public class Alpha {

}

public void throwup() throws Exception {

public static void main(String[] args) {

throw new Exception();

Alpha alpha = new Alpha();
alpha.throwup();

System.out.println("Done");



Erreur d'exécution Soit le code ci-dessous ; que se passe-t-il? public class Beta { Erreur de compilation public void throwup() throws RuntimeException { Affiche "Done" throw new RuntimeException(); public static void main(String[] args) { Beta beta = new Beta(); beta.throwup(); System.out.println("Done"); } Question 6 Soit le code ci-dessous ; que se passe-t-il? Affiche seulement "Caught exception" public class Delta { Erreur d'exécution public void throwup() throws Exception { Affiche "Caught exception" et "Finallied throw new Exception(); public static void main(String[] args) { Delta delta = new Delta(); Affiche seulement "Finallied exception" try { delta.throwup();

#### Question 7 Soit le code:

} catch (Exception e) {

return; } finally {

} }

System.out.println("Caught exception");

System.out.println("Finallied exception");

```
interface Somethingable {
   void something();
interface Whateverable extends Somethingable {
   void whatever();
class The Thing implements Whateverable {
   PPP void something() {
       System.out.println("Somethinging");
   PPP void whatever() {
       System.out.println("Whatevering");
}
```

```
class Doofer {
   void doo(Somethingable sable) {
        sable.XXX();
    void doo(Whateverable sable) {
        sable.YYY();
    void fer(TheThing tt) {
        tt.ZZZ();
    public static void main(String[] args) {
        Doofer doofer = new Doofer();
TheThing the = new TheThing();
doofer.doo(the);
        doofer.fer(the);
    }
}
```

Erreur de compilation

Par quels niveaux d'accès pourrait-on remplacer PPP?

☐ private ☐ package-private		public protected				
Question 8 $\bigoplus$ placer XXX ?	Pour le code de la question pré	cédente, par quelles méthodes pourrait-on rem				
Whatever		Something				

Question 9 $\bigoplus$ placer YYY ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem
whatever	something
Question 10 $\bigoplus$ remplacer ZZZ ?	Pour le code de la question précédente, par quelles méthodes pourrait-or
something	whatever

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
    public double re;
    public Complex(double a, double b) {
        re = a;
        im = b;
    }

    public double getRe() {
        return re;
    }

    public void setRe(double re) {
        this.re = re;
    }

    public double getIm() {
        return im;
    }

    public void setIm(double im) {
        this.im = im;
    }
}
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit



Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r \cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r \sin(\theta)$   $\theta = \tan^{-1} \frac{y}{x}$ 

En Java ce dernier est calculé par theta = Math.atan2(y, x)

- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.





Question 13	Continuez votre réponse à la question précédente ici s	i nécessaire.	



Question 14	Continuez	votre réponse	à la question	précédente	ici si nécessaire	e. 0 0



Question 15 Pour la question précédente, développez la méthode Polar#equals.

0 1 2 3 4

Question 16 ⊕ Parmi les affirmations ci-dessous, cocher celles qui sont vraies :

Une classe peut étendre une classe et implémenter plusieurs interfaces

Une classe abstraite peut implémenter une interface

Une classe abstraite peut contenir des constructeurs

Une classe peut étendre plusieurs classes

Une interface peut contenir des constructeurs

Une classe peut étendre une classe abstraite

Une interface peut contenir des méthodes privées

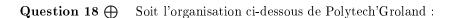


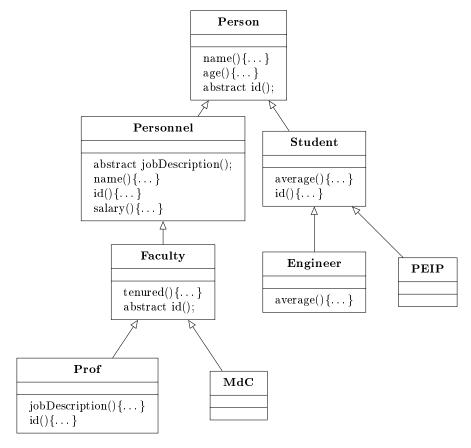
# Question $17 \oplus$ Soit le code ci-dessous :

```
@FunctionalInterface
interface ComparePerson {
    /**
    * Oreturn negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :





Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

Prof	PEIP
MdC	Person
Student	Engineer
Personnel	Faculty

 $\mathbf{QCM}$ 

Introduction à la programmation
orientée objet
9/01/2020

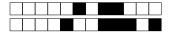
Nom et	pı	é	n	0	m	:									
		٠						٠	٠	٠					
Groupe	:			•						٠	٠				

Cochez les cases en mettant une  $\times$ .

Le symbole  $\bigoplus$  indique que la question peut avoir zéro, une ou plusieurs bonnes réponses. Pour ces questions, cocher une bonne réponse apporte des points positifs ; cocher une mauvaise réponse peut apporter des points négatifs.

Dans tout le code, les package et les import sont censés être correctement déclarés. Toute classe est supposée être dans le bon package, dans le bon fichier, avec les bons import.

import.	
Question 1	
<pre>private ClasseDonnee() {} }</pre>	
pourrait être instanciée par <b>new</b> ClasseDonnee()	pourrait être sous-classée par extenda ClasseDonnee
Question 2    La classe donnée	
class ClasseDonnee {}	
pourrait être instanciée par <b>new</b> ClasseDonnee()	pourrait être sous-classée par extende ClasseDonnee
Question 3   La classe donnée	
<pre>final class ClasseDonnee {    ClasseDonnee() {} }</pre>	
pourrait être instanciée par <b>new</b> ClasseDonnee()	pourrait être sous-classée par extend: ClasseDonnee
Question 4 $\bigoplus$	
Soit le code ci-dessous ; que se passe-t-il ?	Affiche "Caught exception" et "Finallied exception"
public void throwup() throws RuntimeException {     throw new RuntimeException();	Erreur d'exécution
} public static void main(String[] args) {	Affiche seulement "Finallied exception"
Epsilon epsilon = new Epsilon(); try {	Erreur de compilation
<pre>epsilon.throwup(); } catch (Exception e) {     System.out.println("Caught exception"); } finally {     System.out.println("Finallied exception");</pre>	Affiche seulement "Caught exception"
}	



# Question 5 $\bigoplus$

Erreur d'exécution Soit le code ci-dessous ; que se passe-t-il? public class Gamma { Affiche "Caught exception" et "Finallied public void throwup() throws Exception { exception" throw new Exception(); Erreur de compilation public static void main(String[] args) { Gamma gamma = new Gamma(); Affiche seulement "Caught exception" try { gamma.throwup(); Affiche seulement "Finallied exception" } catch (Exception e) { System.out.println("Caught exception"); } finally { System.out.println("Finallied exception"); } Question 6  $\bigoplus$ Soit le code ci-dessous ; que se passe-t-il? Erreur de compilation Erreur d'exécution public class Alpha { public void throwup() throws Exception { throw new Exception(); Affiche "Done" public static void main(String[] args) {
 Alpha alpha = new Alpha();

#### Question 7 Soit le code :

System.out.println("Done");

alpha.throwup();

}

```
interface Somethingable {
    void something();
}

interface Whateverable extends Somethingable {
    void whatever();
}

class TheThing implements Whateverable{
    PPP void something() {
        System.out.println("Somethinging");
    }

    PPP void whatever() {
        System.out.println("Whatevering");
    }
}
```

```
class Doofer {
    void doo(Somethingable sable) {
        sable.XXX();
    }

    void doo(Whateverable sable) {
        sable.YYY();
    }

    void fer(TheThing tt) {
        tt.ZZZ();
    }

    public static void main(String[] args) {
        Doofer doofer = new Doofer();
        TheThing the = new TheThing();
        doofer.doo(the);
        doofer.fer(the);
    }
}
```

Par quels niveaux d'accès pourrait-on remplacer PPP?

private		public
package-priv	vate	protected
Question 8 $\bigoplus$ placer XXX ?	Pour le code de la question précéde	ente, par quelles méthodes pourrait-on rem-
whatever		something

Question 9 $\bigoplus$ placer YYY ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem
something	whatever
Question 10 $\bigoplus$ remplacer ZZZ ?	Pour le code de la question précédente, par quelles méthodes pourrait-or
something	whatever

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

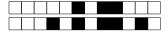
@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit





Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r\cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r\sin(\theta)$   $\theta = \tan^{-1}\frac{y}{x}$ 

En Java ce dernier est calculé par theta = Math.atan2(y, x)

- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.





Question 13	Continuez votre réponse à la question précédente ici si nécessaire.	0 0



Question 14	Continuez	votre réponse	à la question	précédente i	ci si nécessaire.	



Question 15 Pour la question précédente, développez la méthode Polar#equals.  $\boxed{0}$   $\boxed{1}$   $\boxed{2}$   $\boxed{3}$ Question 16  $\bigoplus$ Parmi les affirmations ci-dessous, cocher celles qui sont vraies : Une interface peut contenir des méthodes privées Une classe peut étendre plusieurs classes Une classe abstraite peut contenir des constructeurs Une interface peut contenir des constructeurs Une classe abstraite peut implémenter une interface

Une classe peut étendre une classe et implémenter plusieurs interfaces

Une classe peut étendre une classe abstraite



# Question $17 \oplus$ Soit le code ci-dessous :

```
@FunctionalInterface
interface ComparePerson {
    /**
    * Oreturn negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :

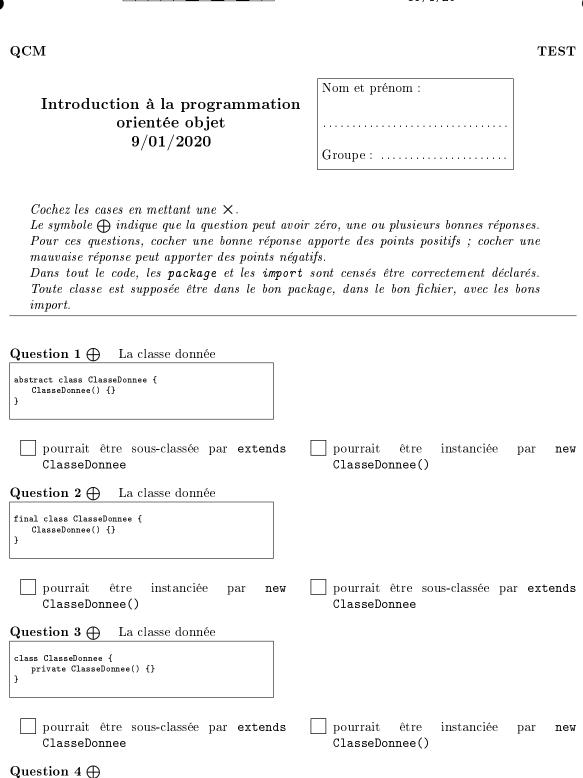


Question 18  $\bigoplus$  Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

<pre>Engineer</pre>	Prof
MdC	PEIP
Person	Student
Faculty	Personnel



Soit le code ci-dessous ; que se passe-t-il ?

public class Beta {
 public void throwup() throws RuntimeException {
 throw new RuntimeException();
 }
 public static void main(String[] args) {
 Beta beta = new Beta();
 beta.throwup();

System.out.println("Done");

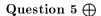
}

Erreur de compilation

Erreur d'exécution

exception"

Affiche "Caught exception" et "Finallied



```
Soit le code ci-dessous ; que se passe-t-il?
public class Delta {
                                                            Erreur d'exécution
   public void throwup() throws Exception {
                                                            Affiche "Caught exception" et "Finallied
      throw new Exception();
   public static void main(String[] args) {
      Delta delta = new Delta();
                                                            Affiche seulement "Finallied exception"
      try {
          delta.throwup();
                                                            Affiche seulement "Caught exception"
      } catch (Exception e) {
          System.out.println("Caught exception");
          return:
      } finally {
          System.out.println("Finallied exception");
   }
}
Question 6
                                                            Erreur de compilation
Soit le code ci-dessous ; que se passe-t-il?
public class Gamma {
                                                            Affiche seulement "Finallied exception"
   public void throwup() throws Exception {
                                                            Affiche seulement "Caught exception"
```

```
throw new Exception();
public static void main(String[] args) {
   Gamma gamma = new Gamma();
    try {
       gamma.throwup();
    } catch (Exception e) {
       System.out.println("Caught exception");
    } finally {
       System.out.println("Finallied exception");
}
```

#### Question 7 Soit le code:

}

```
interface Somethingable {
   void something();
}
interface Whateverable extends Somethingable {
   void whatever();
class TheThing implements Whateverable{
   PPP void something() {
       System.out.println("Somethinging");
   PPP void whatever() {
       System.out.println("Whatevering");
}
```

```
class Doofer {
   void doo(Somethingable sable) {
        sable.XXX();
    void doo(Whateverable sable) {
        sable.YYY();
    void fer(TheThing tt) {
       tt.ZZZ();
   public static void main(String[] args) {
       Doofer doofer = new Doofer();
TheThing the = new TheThing();
        doofer.doo(the);
        doofer.fer(the);
}
```

Par quels niveaux d'accès pourrait-on remplacer PPP?

protected	package-private
public	$oxedsymbol{\square}$ $\operatorname{\texttt{private}}$



Question 8 $\bigoplus$ placer XXX ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
whatever	<pre>something</pre>
Question 9 $\bigoplus$ placer YYY ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
whatever	$\square$ something
Question 10 ⊕ remplacer ZZZ ?	Pour le code de la question précédente, par quelles méthodes pourrait-on
whatever	<pre>something</pre>

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit





Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r\cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r\sin(\theta)$   $\theta = \tan^{-1}\frac{y}{x}$ 

En Java ce dernier est calculé par theta = Math.atan2(y, x)

- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.





Question 13	Continuez	votre réponse à	la question p	orécédente i	ci si nécessaire.	



Question 14	Continuez	votre réponse	e à la question	précédente	ici si nécessaire.	0 0



Une classe abstraite peut implémenter une interface

Une classe peut étendre plusieurs classes

${f Question}$	15	Pour	la	question	précédente,	développez	la	$\begin{array}{c ccccccccccccccccccccccccccccccccccc$
Question							elle	s qui sont vraies :
=	_				sse abstraite nir des const			
_					sse et implén		eurs	interfaces
					constructeur			
Une i	interfac	e peut	coi	ntenir des	méthodes pr	rivées		



# Question $17 \oplus$ Soit le code ci-dessous :

```
@FunctionalInterface
interface ComparePerson {
    /**
    * Oreturn negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

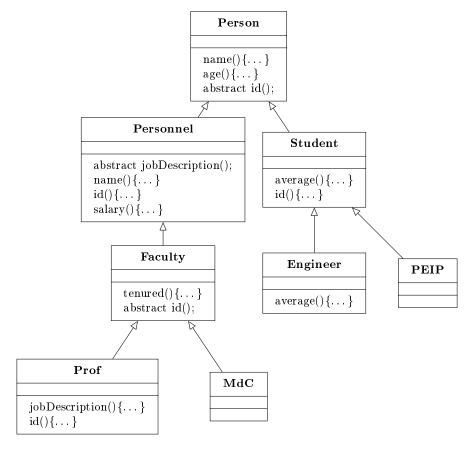
Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :

```
ComparePerson byAge = new ComparePersonByAge();

| ComparePerson byAge = new ComparePersonByAge();
| ComparePerson byAge = new ComparePersonByAge() {
| ComparePerson byAge = new ComparePersonByAge() {
| ComparePerson byAge = (p1, p2) -> p1.age - p2.age;
| ComparePerson byAge = (p1, p2) -> p1.age - p2.age;
| ComparePerson byAge = (p1, p2) -> p1.age - p2.age;
| ComparePerson byAge = new ComparePersonByAge();
| ComparePerson byAge = new CompareP
```

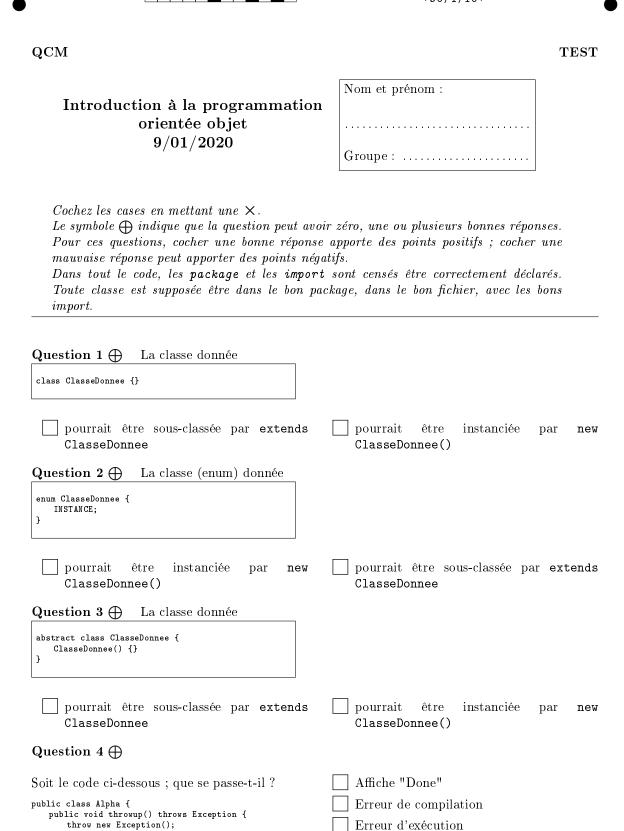


# Question 18 $\bigoplus$ Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

Personnel	Student
Engineer	Person
PEIP	Prof
Faculty	MdC

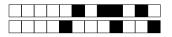


public static void main(String[] args) {

Alpha alpha = new Alpha();
alpha.throwup();

System.out.println("Done");

}



# Question 5 $\bigoplus$

Affiche seulement "Finallied exception" Soit le code ci-dessous ; que se passe-t-il? public class Epsilon { Erreur d'exécution public void throwup() throws RuntimeException { Affiche seulement "Caught exception" throw new RuntimeException(); public static void main(String[] args) { Erreur de compilation Epsilon epsilon = new Epsilon(); try { Affiche "Caught exception" et "Finallied epsilon.throwup(); exception" } catch (Exception e) { System.out.println("Caught exception"); } finally { System.out.println("Finallied exception"); } } Question 6  $\bigoplus$ Erreur de compilation Soit le code ci-dessous ; que se passe-t-il? Erreur d'exécution public class Beta { public void throwup() throws RuntimeException {

#### Question 7 Soit le code :

beta.throwup();

}

throw new RuntimeException();

System.out.println("Done");

public static void main(String[] args) {
 Beta beta = new Beta();

```
interface Somethingable {
    void something();
}

interface Whateverable extends Somethingable {
    void whatever();
}

class TheThing implements Whateverable{
    PPP void something() {
        System.out.println("Somethinging");
    }

    PPP void whatever() {
        System.out.println("Whatevering");
    }
}
```

```
class Doofer {
    void doo(Somethingable sable) {
        sable.XXX();
}

void doo(Whateverable sable) {
        sable.YYY();
}

void fer(TheThing tt) {
        tt.ZZZ();
}

public static void main(String[] args) {
        Doofer doofer = new Doofer();
        TheThing the = new TheThing();
        doofer.doo(the);
        doofer.fer(the);
}
```

Affiche "Done"

Par quels niveaux d'accès pourrait-on remplacer PPP?

private	protected
package-pri	vate public
Question 8 $\bigoplus$ placer XXX ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem
whatever	something

Question 9 ⊕ placer YYY ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
whatever	<pre>something</pre>
Question 10 $\bigoplus$ remplacer ZZZ ?	Pour le code de la question précédente, par quelles méthodes pourrait-on

\_\_\_ whatever

something

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

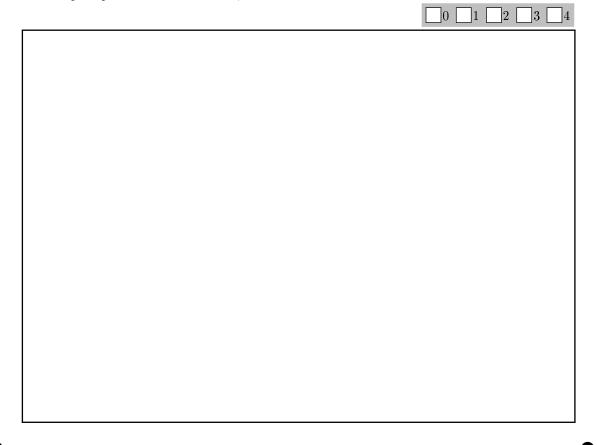
public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit





Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r\cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r\sin(\theta)$   $\theta = \tan^{-1}\frac{y}{x}$ 

En Java ce dernier est calculé par theta = Math.atan2(y, x)

- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.





Question 13	Continuez votre réponse à la question précédente ici si nécessaire.	



Question 14	Continuez votre réponse à la question précédente ici si nécessaire.	



Une classe abstraite peut contenir des constructeurs

Une interface peut contenir des constructeurs

Question 15 Pour la question précédente, développez la méthode Polar#equals.  $\boxed{0}$   $\boxed{1}$   $\boxed{2}$   $\boxed{3}$ Une classe peut étendre plusieurs classes Une interface peut contenir des méthodes privées Une classe peut étendre une classe abstraite Une classe peut étendre une classe et implémenter plusieurs interfaces Une classe abstraite peut implémenter une interface



# Question 17 Soit le code ci-dessous :

```
@FunctionalInterface
interface ComparePerson {
    /**
    * @return negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :

```
ComparePerson byAge = new ComparePersonByAge();

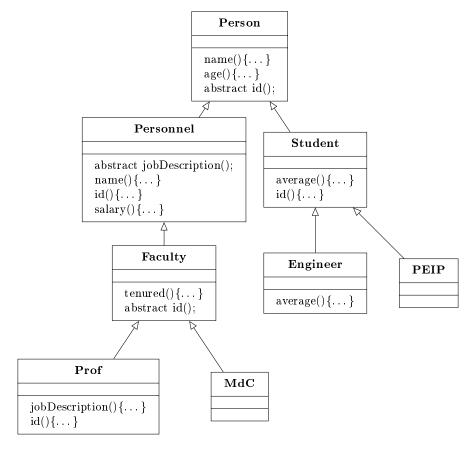
ComparePerson byAge = new ComparePersonByAge::compare;

ComparePerson byAge = (p1, p2) -> p1.age - p2.age;
```

```
ComparePerson byAge = new ComparePersonByAge() {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
};
```



# Question 18 $\bigoplus$ Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

Student	Prof
PEIP	<pre>Engineer</pre>
Personnel	Person
<pre>Faculty</pre>	☐ MdC

QCMTEST Nom et prénom: Introduction à la programmation orientée objet 9/01/2020 Groupe: ..... Cochez les cases en mettant une  $\times$ . Le symbole  $\bigoplus$  indique que la question peut avoir zéro, une ou plusieurs bonnes réponses. Pour ces questions, cocher une bonne réponse apporte des points positifs ; cocher une mauvaise réponse peut apporter des points négatifs. Dans tout le code, les package et les import sont censés être correctement déclarés. Toute classe est supposée être dans le bon package, dans le bon fichier, avec les bons import.Question 1 

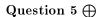
La classe donnée abstract class ClasseDonnee { ClasseDonnee() {} pourrait être sous-classée par extends pourrait être instanciée par new ClasseDonnee() ClasseDonnee Question 2 La classe donnée final class ClasseDonnee { ClasseDonnee() {} } pourrait pourrait être sous-classée par extends être instanciée par new ClasseDonnee() ClasseDonnee La classe (enum) donnée Question 3 ( enum ClasseDonnee { INSTANCE; pourrait être sous-classée par extends pourrait être instanciée par new ClasseDonnee() ClasseDonnee Question 4 Erreur de compilation Soit le code ci-dessous ; que se passe-t-il? public class Alpha { Affiche "Done" public void throwup() throws Exception { throw new Exception(); Erreur d'exécution

public static void main(String[] args) {

Alpha alpha = new Alpha();
alpha.throwup();

System.out.println("Done");

}



Affiche seulement "Finallied exception" Soit le code ci-dessous ; que se passe-t-il? Affiche "Caught exception" et "Finallied public class Epsilon { public void throwup() throws RuntimeException { exception" throw new RuntimeException(); Erreur de compilation public static void main(String[] args) { Epsilon epsilon = new Epsilon(); Affiche seulement "Caught exception" try { epsilon.throwup(); Erreur d'exécution } catch (Exception e) { System.out.println("Caught exception"); } finally { System.out.println("Finallied exception"); } } Question 6  $\bigoplus$ Affiche "Done" Soit le code ci-dessous ; que se passe-t-il? public class Beta { Erreur de compilation public void throwup() throws RuntimeException { throw new RuntimeException(); Erreur d'exécution public static void main(String[] args) {
 Beta beta = new Beta(); beta.throwup(); System.out.println("Done");

#### Question 7 Soit le code :

}

```
interface Somethingable {
    void something();
}

interface Whateverable extends Somethingable {
    void whatever();
}

class TheThing implements Whateverable{
    PPP void something() {
        System.out.println("Somethinging");
    }

    PPP void whatever() {
        System.out.println("Whatevering");
    }
}
```

```
class Doofer {
    void doo(Somethingable sable) {
        sable.XXX();
    }

    void doo(Whateverable sable) {
        sable.YYY();
    }

    void fer(TheThing tt) {
        tt.ZZZ();
    }

    public static void main(String[] args) {
        Doofer doofer = new Doofer();
        TheThing the = new TheThing();
        doofer.doo(the);
        doofer.fer(the);
    }
}
```

Par quels niveaux d'accès pourrait-on remplacer PPP?

public	package-private
protected	private
Question 8 $\oplus$ placer XXX ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
something	whatever

Question 9 $\bigoplus$ placer YYY ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem
something	whatever
Question 10 $\bigoplus$ remplacer ZZZ ?	Pour le code de la question précédente, par quelles méthodes pourrait-or
Something	Whatever

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit



Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r\cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r\sin(\theta)$   $\theta = \tan^{-1}\frac{y}{x}$ 

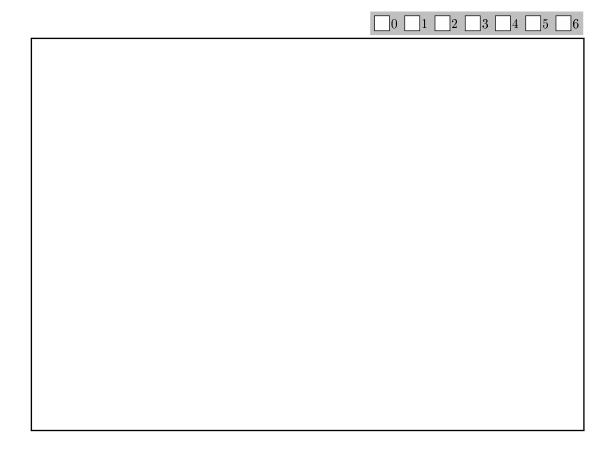
En Java ce dernier est calculé par theta = Math.atan2(y, x)

- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.





${f Question}$	13	Continuez	votre	réponse	à la	question	précéde	nte ici s	si nécessair	re. 0 0



Question 14	Continuez	votre réponse	à la question	précédente	ici si nécessaire.	



Une classe abstraite peut contenir des constructeurs

Une classe peut étendre plusieurs classes

Question 15 Pour la question précédente, développez la méthode Polar#equals.  $\boxed{0}$   $\boxed{1}$   $\boxed{2}$   $\boxed{3}$ Une classe abstraite peut implémenter une interface Une classe peut étendre une classe abstraite Une classe peut étendre une classe et implémenter plusieurs interfaces Une interface peut contenir des méthodes privées Une interface peut contenir des constructeurs



# Question $17 \oplus$ Soit le code ci-dessous :

```
@FunctionalInterface
interface ComparePerson {
    /**
    * @return negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :

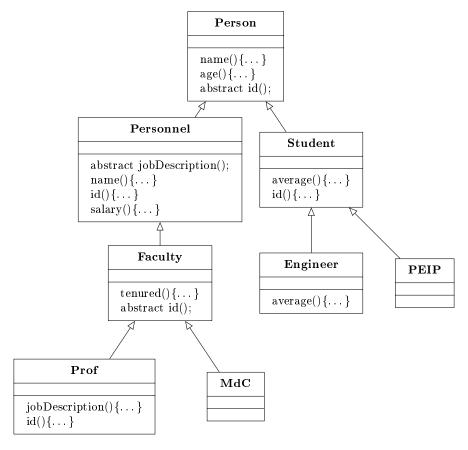
```
ComparePerson byAge = (p1, p2) -> p1.age - p2.age;

ComparePerson byAge = new ComparePersonByAge() {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
};

ComparePerson byAge = new ComparePersonByAge();
```

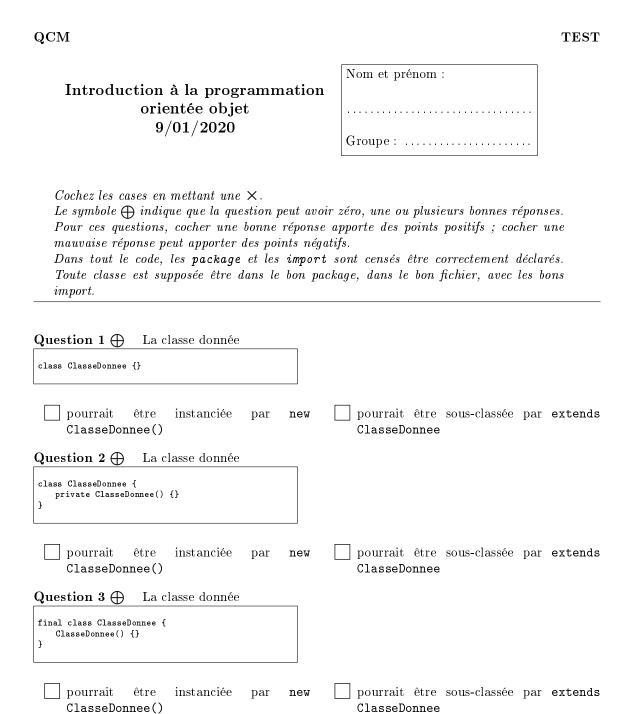


Question 18  $\bigoplus$  Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

PEIP	MdC
Personnel	Person
Engineer	$\square$ Faculty
Prof	Student



# Question 4 $\bigoplus$

Soit le code ci-dessous ; que se passe-t-il?	Affiche seulement "Finallied exception"
<pre>public class Delta {     public void throwup() throws Exception {         throw new Exception();     }     public static void main(String[] args) {         Delta delta = new Delta();         try {             delta.throwup();         } catch (Exception e) {             System.out.println("Caught exception");             return;         } finally {             System.out.println("Finallied exception");         }     } }</pre>	Affiche seulement "Caught exception"  Erreur d'exécution  Affiche "Caught exception" et "Finallied exception"  Erreur de compilation

## Question 5 $\bigoplus$

```
Soit le code ci-dessous ; que se passe-t-il ?

public class Gamma {
    public void throwup() throws Exception {
        throw new Exception();
    }
    public static void main(String[] args) {
        Gamma gamma = new Gamma();
        try {
            gamma.throwup();
    } catch (Exception e) {
            System.out.println("Caught exception");
    } finally {
            System.out.println("Finallied exception");
        }
    }
}
```

# Question 6 $\bigoplus$

Soit le code ci-dessous ; que se passe-t-il ?

public class Epsilon {
 public void throwup() throws RuntimeException {
 throw new RuntimeException();
 }
 public static void main(String[] args) {
 Epsilon epsilon = new Epsilon();
 try {
 epsilon.throwup();
 } catch (Exception e) {
 System.out.println("Caught exception");
 } finally {
 System.out.println("Finallied exception");
 }
 }
}

Erreur de compilation

Affiche seulement "Finallied exception"

Affiche seulement "Caught exception"

Erreur d'exécution

Affiche "Caught exception" et "Finallied exception"

Erreur d'exécution

Affiche seulement "Finallied exception"

| Erreur de compilation

Affiche "Caught exception" et "Finallied exception"

exception

Affiche seulement "Caught exception"



## Question 7 Soit le code :

```
interface Somethingable {
    void something();
}

interface Whateverable extends Somethingable {
    void whatever();
}

class TheThing implements Whateverable{
    PPP void something() {
        System.out.println("Somethinging");
    }

    PPP void whatever() {
        System.out.println("Whatevering");
    }
}
```

```
class Doofer {
    void doo(Somethingable sable) {
        sable.XXX();
    }

    void doo(Whateverable sable) {
        sable.YYY();
    }

    void fer(TheThing tt) {
        tt.ZZZ();
    }

    public static void main(String[] args) {
        Doofer doofer = new Doofer();
        TheThing the = new TheThing();
        doofer.doo(the);
        doofer.fer(the);
    }
}
```

Par quels niveaux d'accès pourrait-on remplacer PPP? private package-private protected public Question 8 Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer XXX? whatever something Question 9 Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer YYY? something whatever Question 10  $\bigoplus$ Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer ZZZ? something whatever

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit



Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r\cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r\sin(\theta)$   $\theta = \tan^{-1}\frac{y}{x}$ 

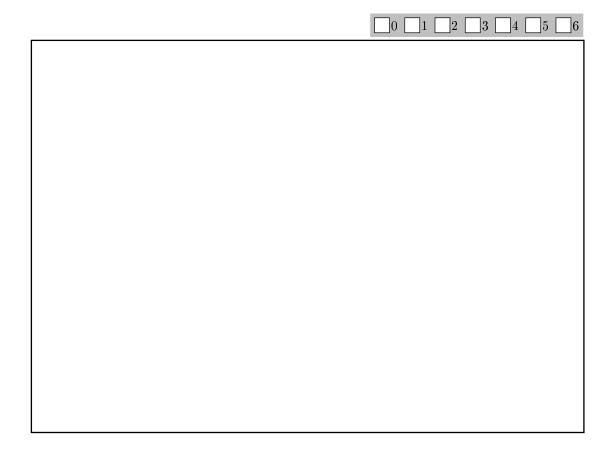
En Java ce dernier est calculé par theta = Math.atan2(y, x)

- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
       return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.





Question 13	Continuez votre réponse à la question précédente ici s	i nécessaire.	



Question 14	Continuez	votre réponse	à la question	précédente	ici si nécessaire.	



Une classe peut étendre plusieurs classes

Une classe abstraite peut contenir des constructeurs

Question 15 Pour la question précédente, développez la méthode Polar#equals.  $\boxed{0}$   $\boxed{1}$   $\boxed{2}$   $\boxed{3}$ Une classe peut étendre une classe et implémenter plusieurs interfaces Une interface peut contenir des méthodes privées Une classe peut étendre une classe abstraite Une interface peut contenir des constructeurs Une classe abstraite peut implémenter une interface



# Question $17 \oplus$ Soit le code ci-dessous :

```
@FunctionalInterface
interface ComparePerson {
    /**
    * Oreturn negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :

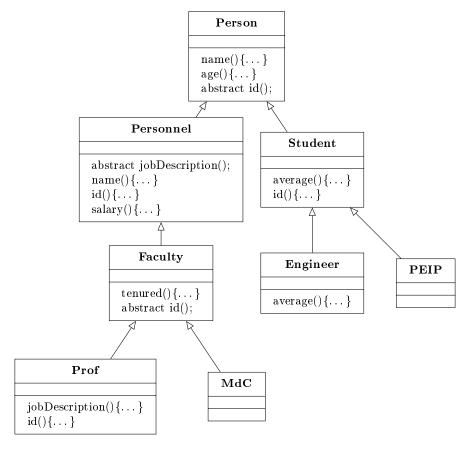
```
ComparePerson byAge = new ComparePersonByAge();

ComparePerson byAge = new ComparePersonByAge();

ComparePerson byAge = new ComparePersonByAge() {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
};
```



Question 18 Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

PEIP	Prof
Personnel	MdC
Student	Person
Engineer	Faculty

QCMTEST Nom et prénom: Introduction à la programmation orientée objet 9/01/2020 Groupe: ..... Cochez les cases en mettant une  $\times$ . Le symbole  $\bigoplus$  indique que la question peut avoir zéro, une ou plusieurs bonnes réponses. Pour ces questions, cocher une bonne réponse apporte des points positifs ; cocher une mauvaise réponse peut apporter des points négatifs. Dans tout le code, les package et les import sont censés être correctement déclarés. Toute classe est supposée être dans le bon package, dans le bon fichier, avec les bons import.Question 1 

La classe donnée abstract class ClasseDonnee { ClasseDonnee() {} pourrait être sous-classée par extends pourrait être instanciée par new ClasseDonnee() ClasseDonnee Question 2 La classe (enum) donnée enum ClasseDonnee { INSTANCE; } pourrait pourrait être sous-classée par extends être instanciée par new ClasseDonnee ClasseDonnee() Question 3 ( La classe donnée class ClasseDonnee {} \_\_\_\_ pourrait être sous-classée par extends pourrait être instanciée new ClasseDonnee ClasseDonnee() Question 4 ( Erreur d'exécution Soit le code ci-dessous ; que se passe-t-il? public class Beta { Erreur de compilation public void throwup() throws RuntimeException {

throw new RuntimeException();
}
public static void main(String[] args) {
Beta beta = new Beta();
beta.throwup();

System.out.println("Done");

}

Affiche "Done"



# Question 5 $\bigoplus$

Affiche "Done" Soit le code ci-dessous ; que se passe-t-il? public class Alpha { Erreur de compilation public void throwup() throws Exception { Erreur d'exécution throw new Exception(); public static void main(String[] args) { Alpha alpha = new Alpha(); alpha.throwup(); System.out.println("Done"); Question 6 Soit le code ci-dessous ; que se passe-t-il? Affiche "Caught exception" et "Finallied exception" public class Delta { public void throwup() throws Exception { Affiche seulement "Caught exception" throw new Exception(); Affiche seulement "Finallied exception" public static void main(String[] args) { Delta delta = new Delta(); Erreur de compilation try { delta.throwup(); Erreur d'exécution } catch (Exception e) {

## Question 7 Soit le code :

return;
} finally {

}

```
interface Somethingable {
    void something();
}

interface Whateverable extends Somethingable {
    void whatever();
}

class TheThing implements Whateverable{
    PPP void something() {
        System.out.println("Somethinging");
    }

    PPP void whatever() {
        System.out.println("Whatevering");
    }
}
```

System.out.println("Caught exception");

System.out.println("Finallied exception");

```
class Doofer {
    void doo(Somethingable sable) {
        sable.XXX();
    }

    void doo(Whateverable sable) {
        sable.YYY();
    }

    void fer(TheThing tt) {
        tt.ZZZ();
    }

    public static void main(String[] args) {
        Doofer doofer = new Doofer();
        TheThing the = new TheThing();
        doofer.doo(the);
        doofer.fer(the);
    }
}
```

Par quels niveaux d'accès pourrait-on remplacer PPP ?

protected public	package-private private
Question $8 \oplus$ placer XXX ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem
whatever	something

Question 9 $\bigoplus$ placer YYY ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
something	whatever
Question 10 $\bigoplus$ remplacer ZZZ?	Pour le code de la question précédente, par quelles méthodes pourrait-on

\_\_\_ whatever

something



Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit





Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r \cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r \sin(\theta)$   $\theta = \tan^{-1} \frac{y}{x}$ 

En Java ce dernier est calculé par theta = Math.atan2(y, x)

- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.





Question 13	Continuez votre réponse à la question précédente ici si nécessaire.	0 0



Question 14	Continuez votre réponse à la question précédente ici si nécessaire.	
1		



Question 15 Pour la question précédente, développez la méthode Polar#equals.

01	$\square 2 \square 3 \square 4$

Question 16 $\bigoplus$	Parmi les	affirmations	ci-dessous,	$\operatorname{cocher}$	celles	qui sont	vraies :
-------------------------	-----------	--------------	-------------	-------------------------	--------	----------	----------

Une interface peut contenir des constructeurs
Une classe peut étendre plusieurs classes
Une classe peut étendre une classe abstraite
Une classe abstraite peut contenir des constructeurs
Une interface peut contenir des méthodes privées
Une classe peut étendre une classe et implémenter plusieurs interfaces
Une classe abstraite neut implémenter une interface



# Question $17 \oplus$ Soit le code ci-dessous :

```
@FunctionalInterface
interface ComparePerson {
    /**
    * @return negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :

```
ComparePerson byAge = new ComparePersonByAge();

ComparePerson byAge = new ComparePersonByAge() {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
};

ComparePerson byAge = new ComparePersonByAge() {
    ComparePerson byAge = (p1, p2) -> p1.age - p2.age;
}
```



# Question 18 Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

Prof	Person
Faculty	Engineer
MdC	PEIP
Student	Personnel

QCMTEST Nom et prénom: Introduction à la programmation orientée objet 9/01/2020Groupe: ..... Cochez les cases en mettant une  $\times$ . Le symbole  $\bigoplus$  indique que la question peut avoir zéro, une ou plusieurs bonnes réponses. Pour ces questions, cocher une bonne réponse apporte des points positifs ; cocher une mauvaise réponse peut apporter des points négatifs. Dans tout le code, les package et les import sont censés être correctement déclarés. Toute classe est supposée être dans le bon package, dans le bon fichier, avec les bons import.Question 1 La classe donnée class ClasseDonnee { private ClasseDonnee() {} pourrait pourrait être sous-classée par extends être instanciée new ClasseDonnee ClasseDonnee() Question 2 La classe (enum) donnée enum ClasseDonnee { INSTANCE; } pourrait être sous-classée par extends pourrait être instanciée par new ClasseDonnee ClasseDonnee() Question 3 ( La classe donnée final class ClasseDonnee { ClasseDonnee() {} pourrait pourrait être sous-classée par extends être instanciée par new ClasseDonnee ClasseDonnee()

# Question 4 $\bigoplus$

```
Affiche "Caught exception" et "Finallied
Soit le code ci-dessous ; que se passe-t-il?
                                                             exception"
public class Gamma {
   public void throwup() throws Exception {
                                                            Erreur d'exécution
      throw new Exception();
                                                            Affiche seulement "Caught exception"
   public static void main(String[] args) {
      Gamma gamma = new Gamma();
                                                             Affiche seulement "Finallied exception"
      try {
         gamma.throwup();
                                                             Erreur de compilation
      } catch (Exception e) {
         System.out.println("Caught exception");
      } finally {
         System.out.println("Finallied exception");
   }
}
Question 5 \bigoplus
Soit le code ci-dessous ; que se passe-t-il?
                                                            Affiche seulement "Finallied exception"
```

# public class Delta { public void throwup() throws Exception { throw new Exception(); } public static void main(String[] args) { Delta delta = new Delta(); try { delta.throwup(); } catch (Exception e) { System.out.println("Caught exception"); return; } finally { System.out.println("Finallied exception"); } }

## Question 6

Soit le code ci-dessous ; que se passe-t-il ?

public class Epsilon {
 public void throwup() throws RuntimeException {
 throw new RuntimeException();
 }
 public static void main(String[] args) {
 Epsilon epsilon = new Epsilon();
 try {
 epsilon.throwup();
 } catch (Exception e) {
 System.out.println("Caught exception");
 } finally {
 System.out.println("Finallied exception");
 }
}

exception"
Erreur d'exécution
Erreur de compilation
Affiche seulement "Caught exception"

Affiche "Caught exception" et "Finallied

Erreur de compilation

Erreur d'exécution

Affiche seulement "Finallied exception"

Affiche seulement "Caught exception"

Affiche "Caught exception" et "Finallied exception"



## Question 7 Soit le code :

```
interface Somethingable {
    void something();
}

interface Whateverable extends Somethingable {
    void whatever();
}

class TheThing implements Whateverable{
    PPP void something() {
        System.out.println("Somethinging");
    }

    PPP void whatever() {
        System.out.println("Whatevering");
    }
}
```

<pre>class Doofer {   void doo(Somethingable sable) {      sable.XXX();   }</pre>	
<pre>void doo(Whateverable sable) {     sable.YYY(); }</pre>	
<pre>void fer(TheThing tt) {    tt.ZZZ(); }</pre>	
<pre>public static void main(String[] args) {    Doofer doofer = new Doofer();    TheThing the = new TheThing();    doofer.doo(the);    doofer.fer(the); }</pre>	
}	

Par quels niveaux d'accès pourrait-on remplacer PPP? package-private  $\rfloor$  private public protected Question  $8 \bigoplus$ Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer XXX? whatever something Question 9 Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer YYY? whatever something Question 10  $\bigoplus$ Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer ZZZ? whatever something

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit





Rappel:

Question 12

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r\cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r\sin(\theta)$   $\theta = \tan^{-1}\frac{y}{x}$ 

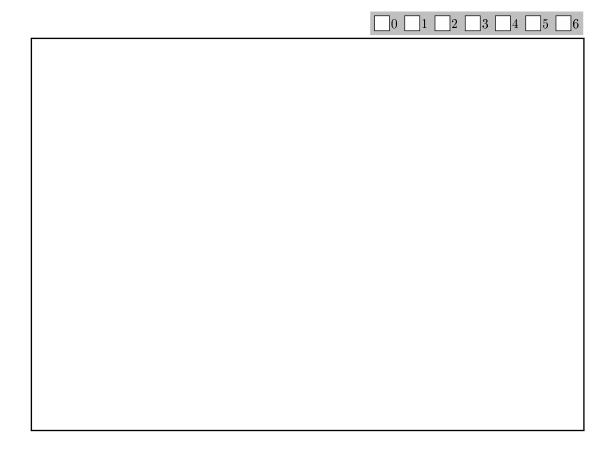
En Java ce dernier est calculé par theta = Math.atan2(y, x)

- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.





Question 13	Continuez votre réponse à la question précédente ici si nécessaire.	0 0



Question 14	Continuez	votre réponse	à la question	précédente	ici si nécessaire	e. 0 0



Une classe peut étendre plusieurs classes

Une classe abstraite peut implémenter une interface

Question 15 Pour la question précédente, développez la méthode Polar#equals.  $\boxed{0}$   $\boxed{1}$   $\boxed{2}$   $\boxed{3}$ Question 16  $\bigoplus$ Parmi les affirmations ci-dessous, cocher celles qui sont vraies : Une interface peut contenir des constructeurs Une interface peut contenir des méthodes privées Une classe peut étendre une classe et implémenter plusieurs interfaces Une classe abstraite peut contenir des constructeurs Une classe peut étendre une classe abstraite



### Question $17 \oplus$ Soit le code ci-dessous :

```
@FunctionalInterface
interface ComparePerson {
    /**
    * Oreturn negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :



## Question 18 $\bigoplus$ Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

MdC	Student
Personnel	Engineer
<pre>Faculty</pre>	Person
Prof	PEIP

QCMTEST Nom et prénom: Introduction à la programmation orientée objet 9/01/2020 Groupe: ..... Cochez les cases en mettant une  $\times$ . Le symbole  $\bigoplus$  indique que la question peut avoir zéro, une ou plusieurs bonnes réponses. Pour ces questions, cocher une bonne réponse apporte des points positifs ; cocher une mauvaise réponse peut apporter des points négatifs. Dans tout le code, les package et les import sont censés être correctement déclarés. Toute classe est supposée être dans le bon package, dans le bon fichier, avec les bons import.Question 1 La classe donnée class ClasseDonnee {} pourrait être sous-classée par extends pourrait être instanciée new par ClasseDonnee ClasseDonnee() Question 2 La classe donnée abstract class ClasseDonnee { ClasseDonnee() {} pourrait être sous-classée par extends pourrait être instanciée par new ClasseDonnee ClasseDonnee() Question 3 La classe donnée class ClasseDonnee { private ClasseDonnee() {} pourrait être sous-classée par extends pourrait être instanciée new ClasseDonnee ClasseDonnee() Question 4 Erreur d'exécution Soit le code ci-dessous ; que se passe-t-il? Affiche "Done" public class Beta {

Erreur de compilation

public void throwup() throws RuntimeException {

throw new RuntimeException();
}
public static void main(String[] args) {

Beta beta = new Beta();
beta.throwup();

}

System.out.println("Done");



Erreur de compilation Soit le code ci-dessous ; que se passe-t-il? Affiche "Done" public class Alpha { public void throwup() throws Exception { Erreur d'exécution throw new Exception(); public static void main(String[] args) { Alpha alpha = new Alpha(); alpha.throwup(); System.out.println("Done"); Question 6 Soit le code ci-dessous ; que se passe-t-il? Affiche seulement "Finallied exception" public class Gamma { Erreur d'exécution public void throwup() throws Exception { Affiche "Caught exception" et "Finallied throw new Exception(); public static void main(String[] args) { Gamma gamma = new Gamma(); Affiche seulement "Caught exception" try { gamma.throwup(); Erreur de compilation } catch (Exception e) { System.out.println("Caught exception"); } finally { System.out.println("Finallied exception"); }

#### Question 7 Soit le code :

```
interface Somethingable {
    void something();
}

interface Whateverable extends Somethingable {
    void whatever();
}

class TheThing implements Whateverable{
    PPP void something() {
        System.out.println("Somethinging");
    }

    PPP void whatever() {
        System.out.println("Whatevering");
    }
}
```

```
class Doofer {
    void doo(Somethingable sable) {
        sable.XXX();
    }

    void doo(Whateverable sable) {
        sable.YYY();
    }

    void fer(TheThing tt) {
        tt.ZZZ();
    }

    public static void main(String[] args) {
        Doofer doofer = new Doofer();
        TheThing the = new TheThing();
        doofer.doo(the);
        doofer.fer(the);
    }
}
```

Par quels niveaux d'accès pourrait-on remplacer PPP?

public protected	☐ package-private ☐ private
Question 8 $\bigoplus$	Pour le code de la question précédente, par quelles méthodes pourrait-on rem
whatever	something



Question 9 $\bigoplus$ placer YYY ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
something	whatever
Question 10 $\bigoplus$ remplacer ZZZ ?	Pour le code de la question précédente, par quelles méthodes pourrait-on
something	whatever

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit





Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r \cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r \sin(\theta)$   $\theta = \tan^{-1} \frac{y}{x}$ 

En Java ce dernier est calculé par theta = Math.atan2(y, x)

- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.





Question	13	Continuez	votre	réponse	à la qu	estion	précédent	e ici s	i nécessaire.	



Question 14	Continuez	votre réponse	à la question	précédente	ici si nécessaire.	



Une classe peut étendre plusieurs classes

Question	15	Pour	la	question	précédente,	développez	la		Polar	#equals	3 4
									, 🔲 1		Jo4
Question	16 🕀	Parn	ni l	es affirma	tions ci-desso	ous, cocher c	elles	qui sont	vraies	:	
Une o	classe al	ostrait	ер	eut conte	nir des const	ructeurs					
Une o	classe al	ostrait	ер	eut implé	menter une i	$\operatorname{nterface}$					
Une o	classe pe	eut éte	$\operatorname{end}$	re une cla	sse et implér	nenter plusie	eurs	interfaces	5		
Une i	$_{ m nterface}$	e peut	COI	ntenir des	méthodes pr	rivées					
Une i	$_{ m nterface}$	e peut	coı	ntenir des	constructeur	°S					
Une o	classe pe	eut éte	$\operatorname{end}$	re une cla	sse abstraite						

# Question $17 \oplus$ Soit le code ci-dessous :

```
@FunctionalInterface
interface ComparePerson {
    /**
    * @return negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :

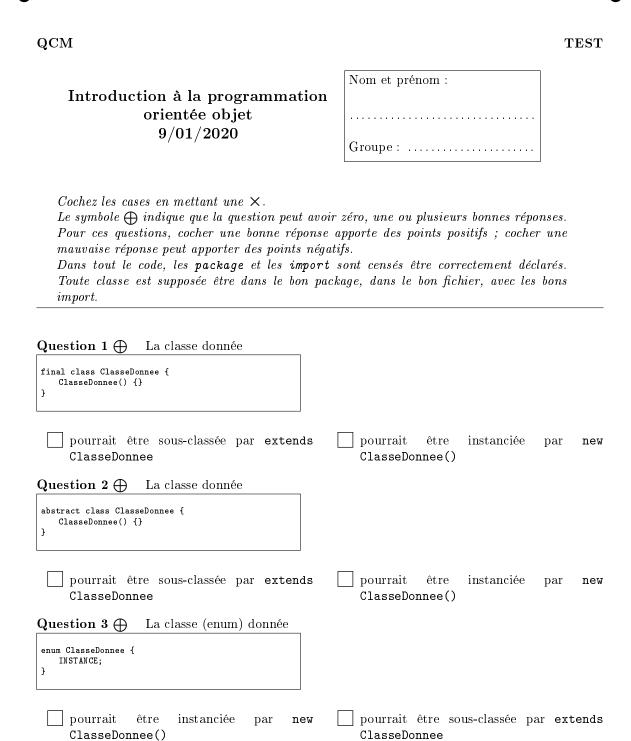


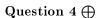
# Question 18 $\bigoplus$ Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

	Personnel	Person
	MdC	Prof
	PEIP	Faculty
ſ	Student	Engineer





Erreur d'exécution Soit le code ci-dessous ; que se passe-t-il? public class Delta { Affiche seulement "Caught exception" public void throwup() throws Exception { Affiche "Caught exception" et "Finallied throw new Exception(); public static void main(String[] args) { Delta delta = new Delta(); Affiche seulement "Finallied exception" try { delta.throwup(); Erreur de compilation } catch (Exception e) { System.out.println("Caught exception"); return; } finally { System.out.println("Finallied exception"); } Question 5

Soit le code ci-dessous ; que se passe-t-il? public class Epsilon { public void throwup() throws RuntimeException { throw new RuntimeException(); public static void main(String[] args) { Epsilon epsilon = new Epsilon(); try { epsilon.throwup(); } catch (Exception e) { System.out.println("Caught exception"); } finally {
 System.out.println("Finallied exception"); } }

#### Question 6

Soit le code ci-dessous ; que se passe-t-il? public class Alpha { public void throwup() throws Exception { throw new Exception(); public static void main(String[] args) { Alpha alpha = new Alpha(); alpha.throwup(); System.out.println("Done");

] Erreur	$_{\mathrm{de}}$	compilation
----------	------------------	-------------

Erreur d'exécution

Affiche seulement "Caught exception"

Affiche seulement "Finallied exception"

Affiche "Caught exception" et "Finallied exception"

Erreur de compilation

Affiche "Done"

Erreur d'exécution



#### Question 7 Soit le code :

```
interface Somethingable {
   void something();
}

interface Whateverable extends Somethingable {
   void whatever();
}

class TheThing implements Whateverable{
   PPP void something() {
       System.out.println("Somethinging");
   }

   PPP void whatever() {
       System.out.println("Whatevering");
   }
}
```

```
class Doofer {
  void doo(Somethingable sable) {
    sable.XXX();
  }
  void doo(Whateverable sable) {
    sable.YYY();
  }
  void fer(TheThing tt) {
    tt.ZZZ();
  }
  public static void main(String[] args) {
    Doofer doofer = new Doofer();
    TheThing the = new TheThing();
    doofer.doo(the);
    doofer.fer(the);
  }
}
```

Par quels niveaux d'accès pourrait-on remplacer PPP? private package-private public protected Question 8 Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer XXX? whatever something Question 9 Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer YYY? whatever something Question 10  $\bigoplus$ Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer ZZZ? something whatever

Rappel:

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

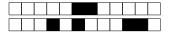
@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage – tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit





Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r \cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r \sin(\theta)$   $\theta = \tan^{-1} \frac{y}{x}$ 

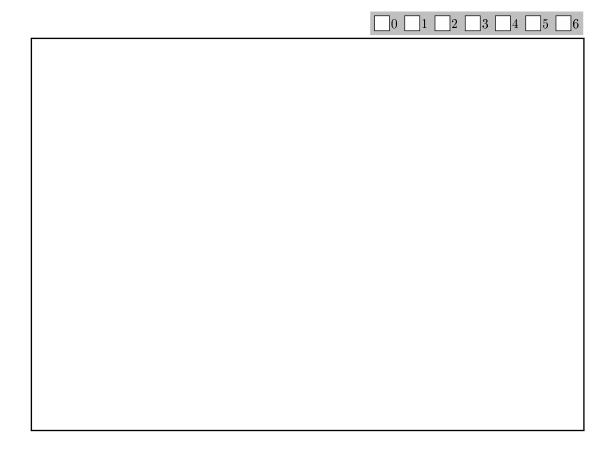
En Java ce dernier est calculé par theta = Math.atan2(y, x)

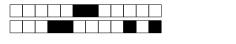
- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

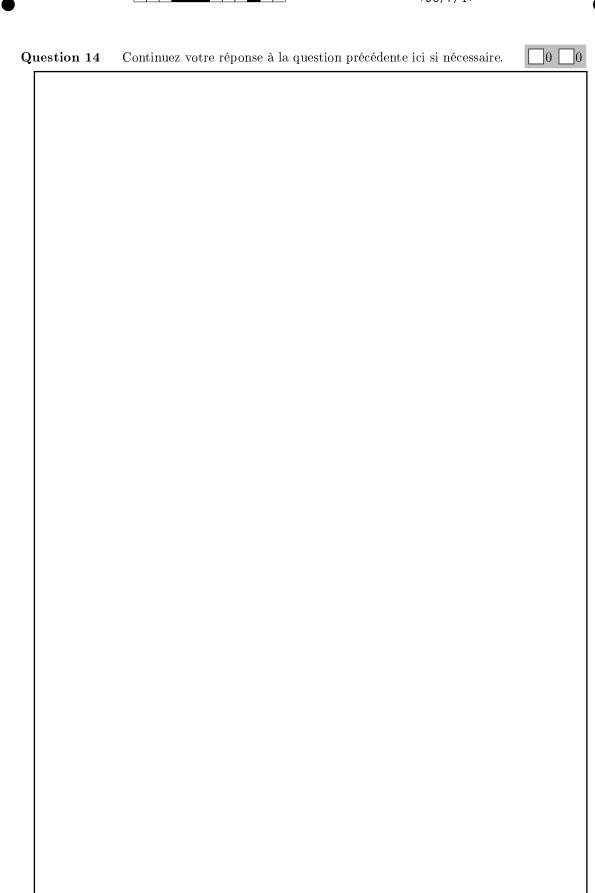
Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.



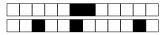


Question 13	Continuez votre réponse à la question précédente ici s	i nécessaire.	





Question 15 Pour la question précédente, développez la méthode Polar#equals.  $\boxed{0}$   $\boxed{1}$   $\boxed{2}$   $\boxed{3}$ Question 16  $\bigoplus$ Parmi les affirmations ci-dessous, cocher celles qui sont vraies : Une interface peut contenir des méthodes privées Une interface peut contenir des constructeurs Une classe peut étendre une classe et implémenter plusieurs interfaces Une classe abstraite peut implémenter une interface Une classe abstraite peut contenir des constructeurs Une classe peut étendre plusieurs classes Une classe peut étendre une classe abstraite



### Question $17 \oplus$ Soit le code ci-dessous :

```
@FunctionalInterface
interface ComparePerson {
    /**
    * Oreturn negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :



 $\textbf{Question 18} \ \bigoplus \quad \text{Soit l'organisation ci-dessous de Polytech'Groland}:$ 



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

Prof	Faculty
MdC	Person
Student	Engineer
Personnel	☐ PEIP

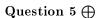
QCMTEST Nom et prénom: Introduction à la programmation orientée objet 9/01/2020 Groupe: ..... Cochez les cases en mettant une  $\times$ . Le symbole  $\bigoplus$  indique que la question peut avoir zéro, une ou plusieurs bonnes réponses. Pour ces questions, cocher une bonne réponse apporte des points positifs ; cocher une mauvaise réponse peut apporter des points négatifs. Dans tout le code, les package et les import sont censés être correctement déclarés. Toute classe est supposée être dans le bon package, dans le bon fichier, avec les bons import.Question 1 La classe donnée class ClasseDonnee {} pourrait être sous-classée par extends pourrait être instanciée par new ClasseDonnee() ClasseDonnee Question 2 La classe donnée class ClasseDonnee { private ClasseDonnee() {} pourrait être sous-classée par extends pourrait être instanciée par new ClasseDonnee ClasseDonnee() Question 3 La classe donnée abstract class ClasseDonnee { ClasseDonnee() {} pourrait être pourrait être sous-classée par extends instanciée par new ClasseDonnee() ClasseDonnee Question 4 Affiche "Done" Soit le code ci-dessous ; que se passe-t-il? Erreur d'exécution public class Beta { public void throwup() throws RuntimeException { Erreur de compilation throw new RuntimeException();

public static void main(String[] args) {

Beta beta = new Beta();
beta.throwup();

}

System.out.println("Done");



Affiche seulement "Finallied exception" Soit le code ci-dessous ; que se passe-t-il? Affiche "Caught exception" et "Finallied public class Gamma { public void throwup() throws Exception { exception" throw new Exception(); Erreur de compilation public static void main(String[] args) { Gamma gamma = new Gamma(); Affiche seulement "Caught exception" try { gamma.throwup(); Erreur d'exécution } catch (Exception e) { System.out.println("Caught exception"); } finally { System.out.println("Finallied exception"); } } Question 6  $\bigoplus$ 

Soit le code ci-dessous ; que se passe-t-il ?

public class Epsilon {
 public void throwup() throws RuntimeException {
 throw new RuntimeException();
 }
 public static void main(String[] args) {
 Epsilon epsilon = new Epsilon();
 try {
 epsilon.throwup();
 } catch (Exception e) {
 System.out.println("Caught exception");
 } finally {
 System.out.println("Finallied exception");
 }
}

# $oxedsymbol{oxed}$ Affiche seulement "Finallied exception"

Terreur d'exécution

Affiche seulement "Caught exception"

Affiche "Caught exception" et "Finallied exception"

Erreur de compilation

#### Question 7 Soit le code :

```
interface Somethingable {
    void something();
}

interface Whateverable extends Somethingable {
    void whatever();
}

class TheThing implements Whateverable{
    PPP void something() {
        System.out.println("Somethinging");
    }

    PPP void whatever() {
        System.out.println("Whatevering");
    }
}
```

```
class Doofer {
    void doo(Somethingable sable) {
        sable.XXX();
    }

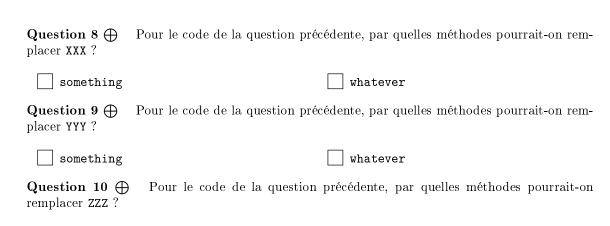
    void doo(Whateverable sable) {
        sable.YYY();
    }

    void fer(TheThing tt) {
        tt.ZZZ();
    }

    public static void main(String[] args) {
        Doofer doofer = new Doofer();
        TheThing the = new TheThing();
        doofer.doo(the);
        doofer.fer(the);
    }
}
```

Par quels niveaux d'accès pourrait-on remplacer PPP?

protected public package-private private



something

whatever

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

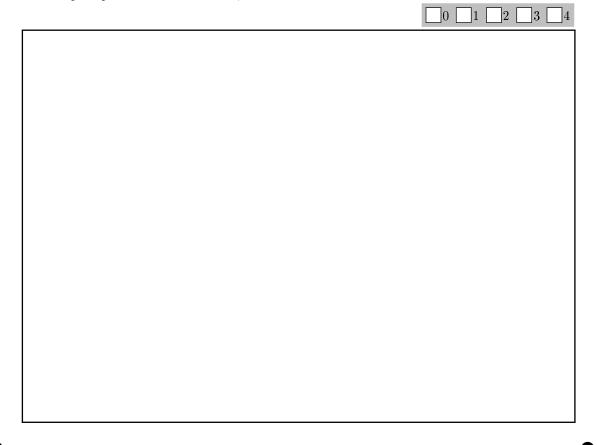
public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit





Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r \cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r \sin(\theta)$   $\theta = \tan^{-1} \frac{y}{x}$ 

En Java ce dernier est calculé par theta = Math.atan2(y, x)

- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.





${f Question}$	13	Continuez	votre	réponse	à la	question	précéde	nte ici s	si nécessair	re. 0 0



Question 14	Continuez	votre réponse	à la question	précédente i	ci si nécessaire.	



Une interface peut contenir des méthodes privées

Une classe peut étendre plusieurs classes

Question 15 Pour la question précédente, développez la méthode Polar#equals.  $\boxed{0}$   $\boxed{1}$   $\boxed{2}$   $\boxed{3}$ Une classe abstraite peut implémenter une interface Une classe peut étendre une classe et implémenter plusieurs interfaces Une classe peut étendre une classe abstraite Une classe abstraite peut contenir des constructeurs Une interface peut contenir des constructeurs



### Question $17 \oplus$ Soit le code ci-dessous :

```
@FunctionalInterface
interface ComparePerson {
    /**
    * Oreturn negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :

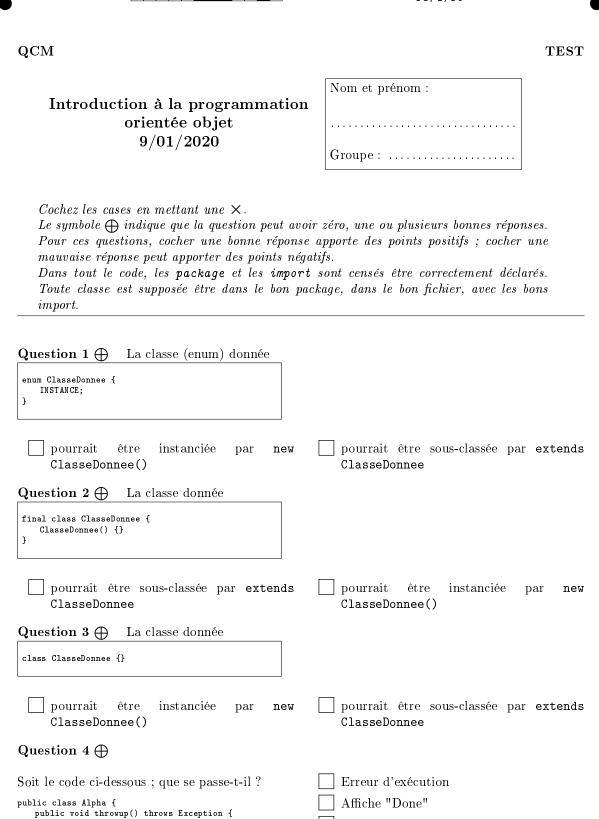


Question 18  $\bigoplus$  Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

Prof	Student
Person	PEIP
Faculty	Personnel
MdC	Engineer



Erreur de compilation

throw new Exception();

Alpha alpha = new Alpha();
alpha.throwup();

System.out.println("Done");

public static void main(String[] args) {



# Question $5 \bigoplus$

```
Soit le code ci-dessous ; que se passe-t-il?
                                                            Erreur de compilation
public class Delta {
                                                            Erreur d'exécution
   public void throwup() throws Exception {
                                                            Affiche seulement "Caught exception"
      throw new Exception();
   public static void main(String[] args) {
                                                            Affiche "Caught exception" et "Finallied
      Delta delta = new Delta();
                                                            exception"
      try {
          delta.throwup();
                                                            Affiche seulement "Finallied exception"
      } catch (Exception e) {
          System.out.println("Caught exception");
          return;
      } finally {
          System.out.println("Finallied exception");
}
Question 6
                                                            Erreur d'exécution
Soit le code ci-dessous ; que se passe-t-il?
                                                            Affiche "Done"
public class Beta {
   public void throwup() throws RuntimeException {
```

#### Question 7 Soit le code :

throw new RuntimeException();

Beta beta = new Beta();
beta.throwup();

System.out.println("Done");

public static void main(String[] args) {

```
interface Somethingable {
   void something();
}

interface Whateverable extends Somethingable {
   void whatever();
}

class TheThing implements Whateverable{
   PPP void something() {
       System.out.println("Somethinging");
   }

   PPP void whatever() {
       System.out.println("Whatevering");
   }
}
```

```
class Doofer {
    void doo(Somethingable sable) {
        sable.XXX();
    }

    void doo(Whateverable sable) {
        sable.YYY();
    }

    void fer(TheThing tt) {
        tt.ZZZ();
    }

    public static void main(String[] args) {
        Doofer doofer = new Doofer();
        TheThing the = new TheThing();
        doofer.doo(the);
        doofer.fer(the);
    }
}
```

Erreur de compilation

Par quels niveaux d'accès pourrait-on remplacer PPP ?

protected public	private package-private
public	package-private
Question 8 $\bigoplus$ placer XXX ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem
whatever	something

+98/3/48+

Question 9 $\bigoplus$ placer YYY ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem
whatever	something
Question 10 $\bigoplus$ remplacer ZZZ ?	Pour le code de la question précédente, par quelles méthodes pourrait-or
whatever	something

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage – tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit



Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r\cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r\sin(\theta)$   $\theta = \tan^{-1}\frac{y}{x}$ 

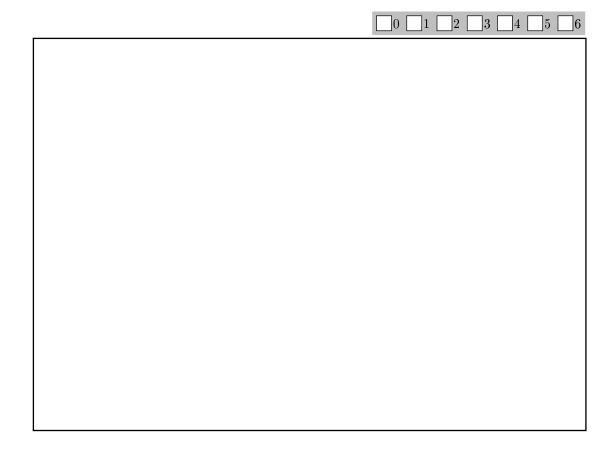
En Java ce dernier est calculé par theta = Math.atan2(y, x)

- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.





Question 13	Continuez	votre réponse	à la question	précédente i	ci si nécessaire.	0 0



Question 14	Continuez	votre réponse	à la question	précédente	ici si nécessaire.	



Une interface peut contenir des constructeurs
Une classe peut étendre une classe abstraite

Question 15 Pour la question précédente, développez la méthode Polar#equals.  $\boxed{0}$   $\boxed{1}$   $\boxed{2}$   $\boxed{3}$ Une classe abstraite peut implémenter une interface Une classe abstraite peut contenir des constructeurs Une interface peut contenir des méthodes privées Une classe peut étendre une classe et implémenter plusieurs interfaces Une classe peut étendre plusieurs classes



# Question $17 \oplus$ Soit le code ci-dessous :

```
@FunctionalInterface
interface ComparePerson {
    /**
    * Oreturn negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :

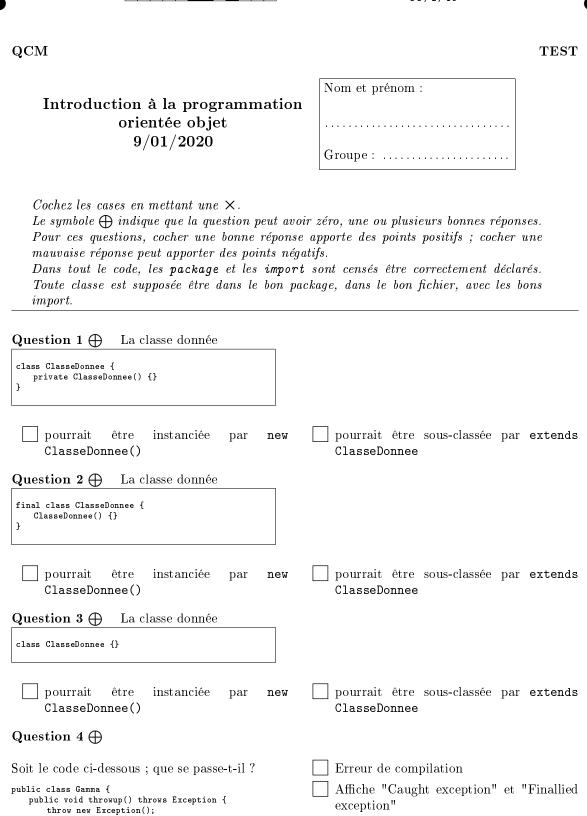


Question 18  $\bigoplus$  Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

Person	PEIP
Personnel	Student
MdC	Prof
Engineer	Faculty



gamma.throwup();
} catch (Exception e) {
 System.out.println("Caught exception");
} finally {
 System.out.println("Finallied exception");
}
Affiche seulement "Caught exception"

\*\*The content of the content of the

public static void main(String[] args) {
 Gamma gamma = new Gamma();

try {

}

Erreur d'exécution

Affiche seulement "Finallied exception"



Affiche seulement "Caught exception" Soit le code ci-dessous ; que se passe-t-il? public class Epsilon { Erreur d'exécution public void throwup() throws RuntimeException { Affiche seulement "Finallied exception" throw new RuntimeException(); public static void main(String[] args) { Affiche "Caught exception" et "Finallied Epsilon epsilon = new Epsilon(); exception" try { epsilon.throwup(); Erreur de compilation } catch (Exception e) { System.out.println("Caught exception"); } finally { System.out.println("Finallied exception"); } } Question 6  $\bigoplus$ Affiche "Done" Soit le code ci-dessous ; que se passe-t-il? public class Alpha { Erreur de compilation public void throwup() throws Exception { throw new Exception(); Erreur d'exécution

### Question 7 Soit le code :

System.out.println("Done");

alpha.throwup();

}

public static void main(String[] args) {
 Alpha alpha = new Alpha();

```
interface Somethingable {
    void something();
}

interface Whateverable extends Somethingable {
    void whatever();
}

class TheThing implements Whateverable{
    PPP void something() {
        System.out.println("Somethinging");
    }

    PPP void whatever() {
        System.out.println("Whatevering");
    }
}
```

```
class Doofer {
    void doo(Somethingable sable) {
        sable.XXX();
    }

    void doo(Whateverable sable) {
        sable.YYY();
    }

    void fer(TheThing tt) {
        tt.ZZZ();
    }

    public static void main(String[] args) {
        Doofer doofer = new Doofer();
        TheThing the = new TheThing();
        doofer.doo(the);
        doofer.fer(the);
    }
}
```

Par quels niveaux d'accès pourrait-on remplacer PPP?

protected	private
public	package-private
Question 8 $\bigoplus$ placer XXX ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
something	whatever

Question 9 $\bigoplus$ placer YYY ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem
whatever	something
Question 10 ⊕ remplacer ZZZ ?	Pour le code de la question précédente, par quelles méthodes pourrait-or
something	whatever

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit





Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r\cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r\sin(\theta)$   $\theta = \tan^{-1}\frac{y}{x}$ 

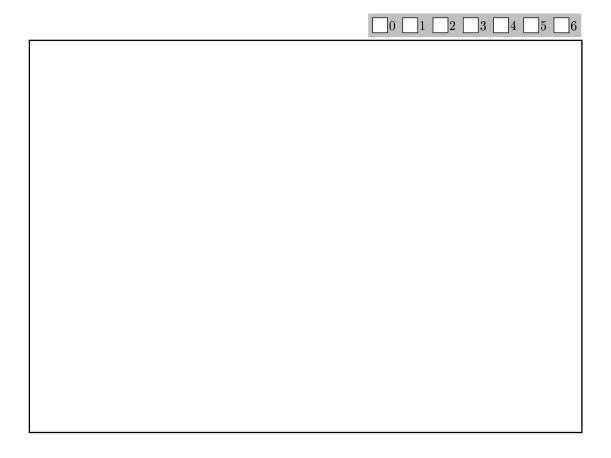
En Java ce dernier est calculé par theta = Math.atan2(y, x)

- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.





Question 13	Continuez	votre répons	e à la question	précédente ic	i si nécessaire.	0 0



Question 14	Continuez votre réponse à la question précédente ici si nécessaire.	
1		



Une classe abstraite peut contenir des constructeurs
Une interface peut contenir des méthodes privées

Question 15 Pour la question précédente, développez la méthode Polar#equals.  $\boxed{0}$   $\boxed{1}$   $\boxed{2}$   $\boxed{3}$ Une classe peut étendre une classe et implémenter plusieurs interfaces Une classe abstraite peut implémenter une interface Une classe peut étendre une classe abstraite Une interface peut contenir des constructeurs Une classe peut étendre plusieurs classes

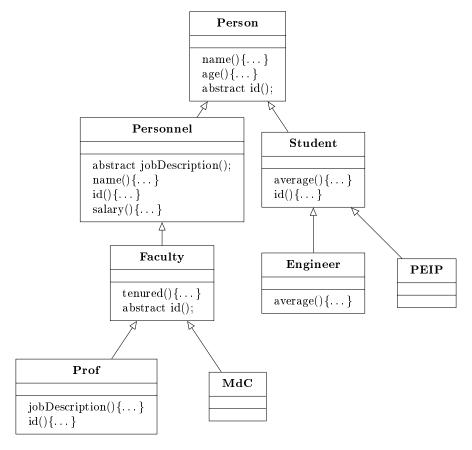


# Question $17 \oplus$ Soit le code ci-dessous :

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :



Question 18  $\bigoplus$  Soit l'organisation ci-dessous de Polytech'Groland :

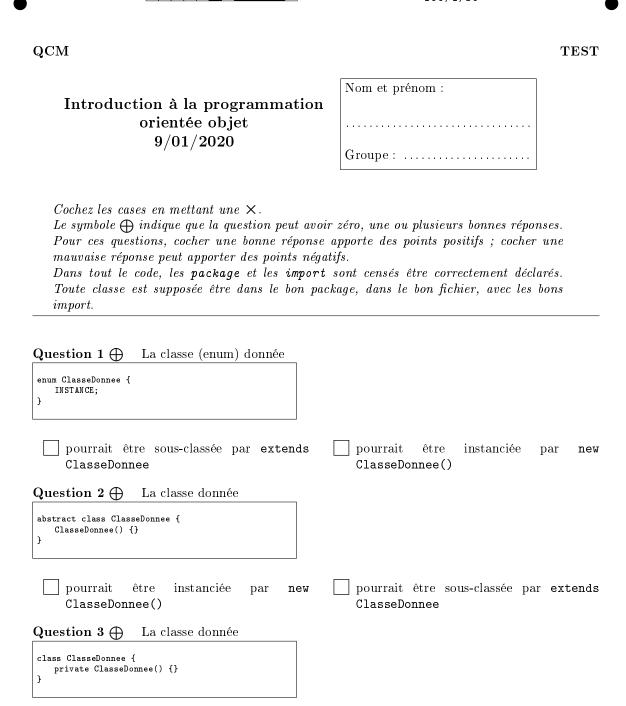


Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

Student	Prof
PEIP	Personnel
Person	<pre>Engineer</pre>
MdC	$oxedsymbol{\square}$ Faculty

pourrait être sous-classée par extends

ClasseDonnee



pourrait

ClasseDonnee()

être

instanciée

par

new

# Question 4 $\bigoplus$

Soit le code ci-dessous ; que se passe-t-il ?
<pre>public class Delta {    public void throwup() throws Exception {       throw new Exception();    }</pre>
<pre>public static void main(String[] args) {    Delta delta = new Delta();    try {       delta.throwup();    } catch (Exception e) {       System.out.println("Caught exception");       return;    } finally {       System.out.println("Finallied exception");</pre>
} } }

# Erreur de compilation Erreur d'exécution Affiche seulement "Caught exception" Affiche seulement "Finallied exception"

Affiche "Caught exception" et "Finallied exception"

# Question 5 $\bigoplus$

```
Soit le code ci-dessous ; que se passe-t-il ?
public class Beta {
    public void throwup() throws RuntimeException {
        throw new RuntimeException();
    }
    public static void main(String[] args) {
        Beta beta = new Beta();
        beta.throwup();
        System.out.println("Done");
    }
}
```

# Erreur de compilation

Erreur d'exécution
Affiche "Done"

# Question 6

Soit le code ci-dessous ; que se passe-t-il ?

public class Gamma {
 public void throwup() throws Exception {
 throw new Exception();
 }

 public static void main(String[] args) {
 Gamma gamma = new Gamma();
 try {
 gamma.throwup();
 } catch (Exception e) {
 System.out.println("Caught exception");
 } finally {
 System.out.println("Finallied exception");
 }
}

Affiche "Caught exception" et "Finallied
exception"
Affiche seulement "Finallied exception"
Affiche seulement "Caught exception"
Erreur de compilation

Erreur d'exécution



## Question 7 Soit le code :

```
interface Somethingable {
   void something();
}

interface Whateverable extends Somethingable {
   void whatever();
}

class TheThing implements Whateverable{
   PPP void something() {
       System.out.println("Somethinging");
   }

   PPP void whatever() {
       System.out.println("Whatevering");
   }
}
```

```
class Doofer {
   void doo(Somethingable sable) {
      sable.XXX();
   }

   void doo(Whateverable sable) {
      sable.YYY();
   }

   void fer(TheThing tt) {
      tt.ZZZ();
   }

  public static void main(String[] args) {
      Doofer doofer = new Doofer();
      TheThing the = new TheThing();
      doofer.doo(the);
      doofer.fer(the);
   }
}
```

Par quels niveaux d'accès pourrait-on remplacer PPP? package-private public protected private Question 8 Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer XXX? whatever something Question 9 Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer YYY? something whatever Question 10  $\bigoplus$ Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer ZZZ? whatever something

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit





Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r\cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r\sin(\theta)$   $\theta = \tan^{-1}\frac{y}{x}$ 

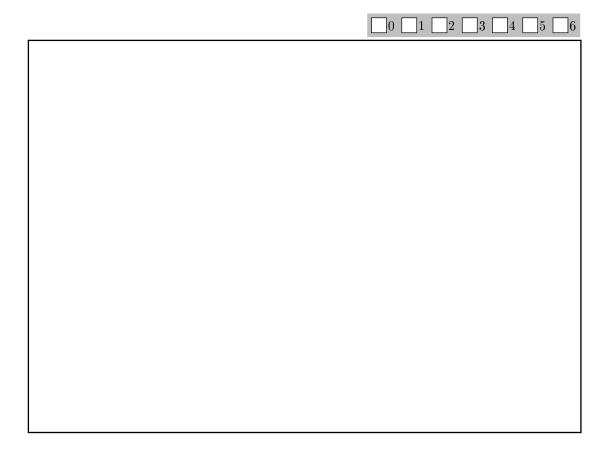
En Java ce dernier est calculé par theta = Math.atan2(y, x)

- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.





Question 13	Continuez votre réponse à la question précédente ici s	i nécessaire.	



uestion 14	Continuez votre réponse à la question précédente ici si nécessaire.	



Question 15 Pour la question précédente, développez la méthode Polar#equals.

	1		$\Box 4$

Question 16 
Parmi les affirmations ci-dessous, cocher celles qui sont vraies :

Une interface peut contenir des méthodes privées
Une classe peut étendre plusieurs classes
Une interface peut contenir des constructeurs
Une classe peut étendre une classe abstraite

Une classe abstraite peut contenir des constructeurs
Une classe abstraite peut implémenter une interface

Une classe peut étendre une classe et implémenter plusieurs interfaces



# Question $17 \oplus$ Soit le code ci-dessous :

```
@FunctionalInterface
interface ComparePerson {
    /**
    * Oreturn negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :

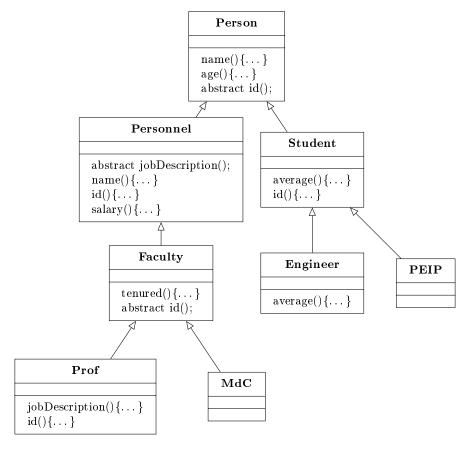
```
ComparePerson byAge = new ComparePersonByAge();

ComparePerson byAge = new ComparePersonByAge();

ComparePerson byAge = new ComparePersonByAge() {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
};
```

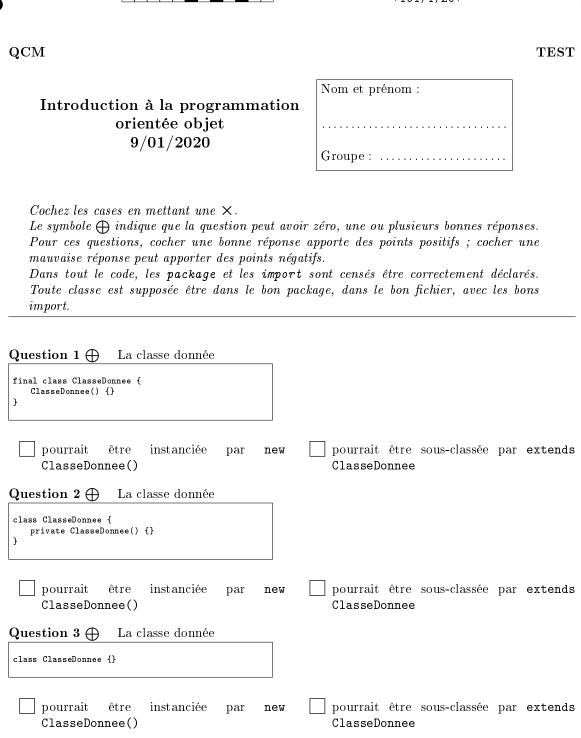


Question 18  $\bigoplus$  Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

Personnel	Person
Engineer	Student
Faculty	MdC
PEIP	Prof



public class Alpha {
 public void throwup() throws Exception {
 throw new Exception();
 }
 public static void main(String[] args) {
 Alpha alpha = new Alpha();
 alpha.throwup();
 System.out.println("Done");
 }
}
Affiche "Done"

Erreur d'exécution

System.out.println("Done");

Erreur de compilation

Question 4

}

Soit le code ci-dessous ; que se passe-t-il?



# Question 5 $\bigoplus$

Affiche "Caught exception" et "Finallied Soit le code ci-dessous ; que se passe-t-il? exception" public class Gamma { public void throwup() throws Exception { Erreur de compilation throw new Exception(); Affiche seulement "Finallied exception" public static void main(String[] args) { Gamma gamma = new Gamma(); Affiche seulement "Caught exception" try { gamma.throwup(); Erreur d'exécution } catch (Exception e) { System.out.println("Caught exception"); } finally { System.out.println("Finallied exception"); } } Question 6  $\bigoplus$ Erreur de compilation Soit le code ci-dessous ; que se passe-t-il? Erreur d'exécution public class Beta { public void throwup() throws RuntimeException { throw new RuntimeException(); Affiche "Done" public static void main(String[] args) {
 Beta beta = new Beta();

### Question 7 Soit le code :

System.out.println("Done");

beta.throwup();

}

```
interface Somethingable {
    void something();
}

interface Whateverable extends Somethingable {
    void whatever();
}

class TheThing implements Whateverable{
    PPP void something() {
        System.out.println("Somethinging");
    }

    PPP void whatever() {
        System.out.println("Whatevering");
    }
}
```

```
class Doofer {
    void doo(Somethingable sable) {
        sable.XXX();
    }

    void doo(Whateverable sable) {
        sable.YYY();
    }

    void fer(TheThing tt) {
        tt.ZZZ();
    }

    public static void main(String[] args) {
        Doofer doofer = new Doofer();
        TheThing the = new TheThing();
        doofer.doo(the);
        doofer.fer(the);
    }
}
```

Par quels niveaux d'accès pourrait-on remplacer PPP?

protected	public
$\square$ private	package-private
Question 8 $\bigoplus$ placer XXX ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
something	whatever

Question 9 $\bigoplus$ placer YYY ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
something	whatever
Question 10 $\bigoplus$ remplacer ZZZ ?	Pour le code de la question précédente, par quelles méthodes pourrait-on
something	whatever



Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit





Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r\cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r\sin(\theta)$   $\theta = \tan^{-1}\frac{y}{x}$ 

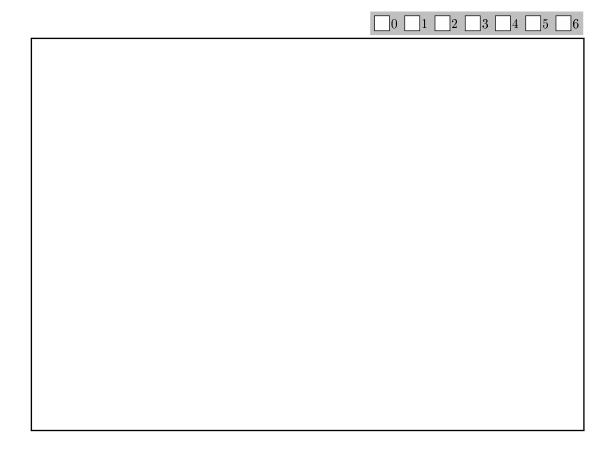
En Java ce dernier est calculé par theta = Math.atan2(y, x)

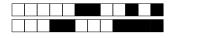
- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.





Question 13	Continuez	votre réponse à	la question p	orécédente i	ci si nécessaire.	



Question 14	Continuez	votre réponse	à la question	précédente	ici si nécessaire.	



Question 15 Pour la question précédente, développez la méthode Polar#equals.

0 1 2 3

 ${\bf Question} \ \, {\bf 16} \ \, \bigoplus \quad {\bf Parmi} \ \, {\bf les} \ \, {\bf affirmations} \ \, {\bf ci-dessous}, \ \, {\bf cocher} \ \, {\bf celles} \ \, {\bf qui} \ \, {\bf sont} \ \, {\bf vraies} :$ 

Une classe abstraite peut contenir des constructeurs
Une interface peut contenir des constructeurs
Une classe abstraite peut implémenter une interface
Une classe peut étendre une classe et implémenter plusieurs interfaces
Une classe peut étendre une classe abstraite
Une interface peut contenir des méthodes privées

Une classe peut étendre plusieurs classes



# Question $17 \oplus$ Soit le code ci-dessous :

```
@FunctionalInterface
interface ComparePerson {
    /**
    * Oreturn negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :

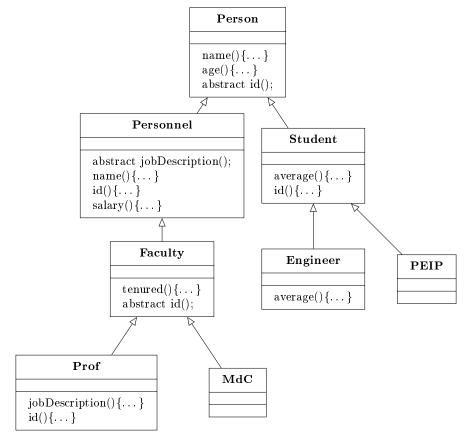
```
ComparePerson byAge = new ComparePersonByAge();

ComparePerson byAge = new ComparePersonByAge() {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
};

ComparePerson byAge = new ComparePersonByAge() {
    ComparePerson byAge = (p1, p2) -> p1.age - p2.age;
}
```

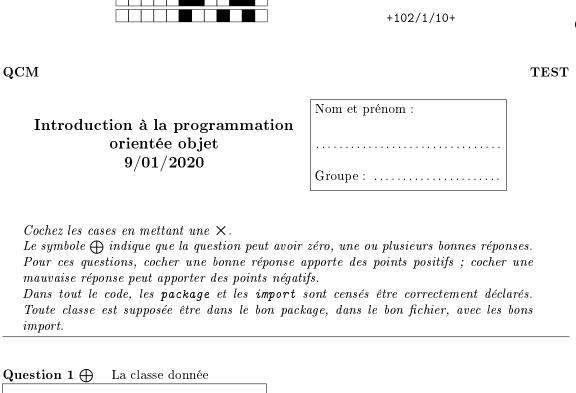


# Question 18 $\bigoplus$ Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

Person	$\Box$ Faculty
MdC	Prof
PEIP	Engineer
Personnel	Student



Question 1   La classe donnée	
abstract class ClasseDonnee {     ClasseDonnee() {} }	
pourrait être instanciée par <b>new</b> ClasseDonnee()	pourrait être sous-classée par extends ClasseDonnee
Question 2    La classe (enum) donnée	
enum ClasseDonnee {    INSTANCE; }	
pourrait être instanciée par <b>new</b> ClasseDonnee()	pourrait être sous-classée par extends ClasseDonnee
Question 3 ( La classe donnée	
class ClasseDonnee {    private ClasseDonnee() {} }	
pourrait être instanciée par <b>new</b> ClasseDonnee()	pourrait être sous-classée par <b>extends</b> ClasseDonnee

# Question 4 $\bigoplus$

Soit le code ci-dessous ; que se passe-t-il ?	Affiche seulement "Caught exception"				
<pre>public class Delta {     public void throwup() throws Exception {         throw new Exception();     }     public static void main(String[] args) {         Delta delta = new Delta();         try {             delta.throwup();     } catch (Exception e) {             System.out.println("Caught exception");     } }</pre>	Affiche "Caught exception" et "Finall exception"  Affiche seulement "Finallied exception"  Erreur d'exécution  Erreur de compilation				
<pre>return; } finally {           System.out.println("Finallied exception"); } } </pre>					
Question 5 $\bigoplus$					
Soit le code ci-dessous ; que se passe-t-il ?	Affiche seulement "Finallied exception"				

```
Affiche "Caught exception" et "Finallied
public class Epsilon {
   public void throwup() throws RuntimeException {
                                                                    exception"
       throw new RuntimeException();
                                                                    Erreur de compilation
   public static void main(String[] args) {
    Epsilon epsilon = new Epsilon();
                                                                    Affiche seulement "Caught exception"
       try {
          epsilon.throwup();
                                                                    Erreur d'exécution
       } catch (Exception e) {
          System.out.println("Caught exception");
       } finally {
   System.out.println("Finallied exception");
   }
}
```

# Question 6 $\bigoplus$

Soit le code ci-dessous ; que se passe-t-il ?

public class Alpha {
 public void throwup() throws Exception {
 throw new Exception();
 }

 public static void main(String[] args) {
 Alpha alpha = new Alpha();
 alpha.throwup();
 System.out.println("Done");
}

Erreur de compilation

Affiche "Done"

| Erreur d'exécution



# Question 7 Soit le code :

```
interface Somethingable {
   void something();
}

interface Whateverable extends Somethingable {
   void whatever();
}

class TheThing implements Whateverable{
   PPP void something() {
       System.out.println("Somethinging");
   }

   PPP void whatever() {
       System.out.println("Whatevering");
   }
}
```

```
class Doofer {
    void doo(Somethingable sable) {
        sable.XXX();
    }

    void doo(Whateverable sable) {
        sable.YYY();
    }

    void fer(TheThing tt) {
        tt.ZZZ();
    }

    public static void main(String[] args) {
        Doofer doofer = new Doofer();
        TheThing the = new TheThing();
        doofer.doo(the);
        doofer.fer(the);
    }
}
```

Par quels niveaux d'accès pourrait-on remplacer  ${\tt PPP}$  ?

protected public	☐ private ☐ package-private
Question 8 $\bigoplus$ placer XXX ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
whatever	something
Question 9 $\bigoplus$ placer YYY ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
whatever	something
Question 10 $\bigoplus$ remplacer ZZZ ?	Pour le code de la question précédente, par quelles méthodes pourrait-on
whatever	something

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit





Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r\cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r\sin(\theta)$   $\theta = \tan^{-1}\frac{y}{x}$ 

En Java ce dernier est calculé par theta = Math.atan2(y, x)

- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.





Question 13	Continuez	votre réponse	à la question	précédente i	ci si nécessaire.	0 0



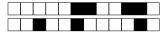
uestion 14	Continuez votre réponse à la question précédente ici si nécessaire.	



Une classe abstraite peut contenir des constructeurs

Une interface peut contenir des constructeurs

Question 15 Pour la question précédente, développez la méthode Polar#equals.  $\boxed{0}$   $\boxed{1}$   $\boxed{2}$   $\boxed{3}$ Une classe peut étendre une classe et implémenter plusieurs interfaces Une classe peut étendre plusieurs classes Une interface peut contenir des méthodes privées Une classe abstraite peut implémenter une interface Une classe peut étendre une classe abstraite



# Question $17 \oplus$ Soit le code ci-dessous :

```
@FunctionalInterface
interface ComparePerson {
    /**
    * Oreturn negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :

```
ComparePerson byAge = new ComparePersonByAge();

ComparePerson byAge = (p1, p2) -> p1.age - p2.age;

ComparePerson byAge
= new ComparePersonByAge::compare;
```

```
ComparePerson byAge = new ComparePersonByAge() {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
};
```



Question 18  $\bigoplus$  Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

Engineer	Prof
Student	PEIP
MdC	Personnel
Person	Faculty

QCM

Introduction à la programmation
orientée objet
9/01/2020

Nom et	р	ré	'n	o	m	;							
$\operatorname{Group} \epsilon$	:												

Cochez les cases en mettant une  $\times$ .

Le symbole  $\bigoplus$  indique que la question peut avoir zéro, une ou plusieurs bonnes réponses. Pour ces questions, cocher une bonne réponse apporte des points positifs ; cocher une mauvaise réponse peut apporter des points négatifs.

Dans tout le code, les package et les import sont censés être correctement déclarés. Toute classe est supposée être dans le bon package, dans le bon fichier, avec les bons import.

import.	
Question 1   La classe donnée	
class ClasseDonnee {}	
pourrait être sous-classée par extends ClasseDonnee	pourrait être instanciée par <b>new</b> ClasseDonnee()
Question 2   La classe donnée	
final class ClasseDonnee {     ClasseDonnee() {} }	
pourrait être instanciée par <b>new</b> ClasseDonnee()	pourrait être sous-classée par extends ClasseDonnee
Question 3   La classe donnée	
abstract class ClasseDonnee {     ClasseDonnee() {} }	
pourrait être instanciée par <b>new</b> ClasseDonnee()	pourrait être sous-classée par extends
Question $4 \oplus$	
Soit le code ci-dessous ; que se passe-t-il ?	Affiche seulement "Finallied exception"
<pre>public class Gamma {    public void throwup() throws Exception {       throw new Exception();</pre>	Affiche "Caught exception" et "Finallied exception"
<pre>} public static void main(String[] args) {    Gamma gamma = new Gamma();    try {       gamma.throwup();    } catch (Exception e) {       System.out.println("Caught exception"); }</pre>	Affiche seulement "Caught exception" Erreur de compilation Erreur d'exécution
<pre>} finally {     System.out.println("Finallied exception"); }</pre>	



Erreur d'exécution Soit le code ci-dessous ; que se passe-t-il? public class Beta { Erreur de compilation public void throwup() throws RuntimeException { Affiche "Done" throw new RuntimeException(); public static void main(String[] args) { Beta beta = new Beta(); beta.throwup(); System.out.println("Done"); } Question 6 Soit le code ci-dessous ; que se passe-t-il? Erreur de compilation public class Delta { Affiche seulement "Caught exception" public void throwup() throws Exception { Affiche "Caught exception" et "Finallied throw new Exception(); public static void main(String[] args) { Delta delta = new Delta(); Affiche seulement "Finallied exception" try {

#### Question 7 Soit le code :

delta.throwup();

} catch (Exception e) {

return;
} finally {

}

System.out.println("Caught exception");

System.out.println("Finallied exception");

```
interface Somethingable {
    void something();
}

interface Whateverable extends Somethingable {
    void whatever();
}

class TheThing implements Whateverable{
    PPP void something() {
        System.out.println("Somethinging");
    }

    PPP void whatever() {
        System.out.println("Whatevering");
    }
}
```

```
class Doofer {
    void doo(Somethingable sable) {
        sable.XXX();
    }

    void doo(Whateverable sable) {
        sable.YYY();
    }

    void fer(TheThing tt) {
        tt.ZZZ();
    }

    public static void main(String[] args) {
        Doofer doofer = new Doofer();
        TheThing the = new TheThing();
        doofer.doo(the);
        doofer.fer(the);
    }
}
```

Erreur d'exécution

Par quels niveaux d'accès pourrait-on remplacer PPP ?

protected package-pri	vate	private public
Question 8 $\oplus$ placer XXX ?		édente, par quelles méthodes pourrait-on rem-
Whatever		something

Question 9 $\bigoplus$ placer YYY ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
whatever	something
Question 10 $\bigoplus$ remplacer ZZZ ?	Pour le code de la question précédente, par quelles méthodes pourrait-on
something	Whatever

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit



Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r\cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r\sin(\theta)$   $\theta = \tan^{-1}\frac{y}{x}$ 

En Java ce dernier est calculé par theta = Math.atan2(y, x)

- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

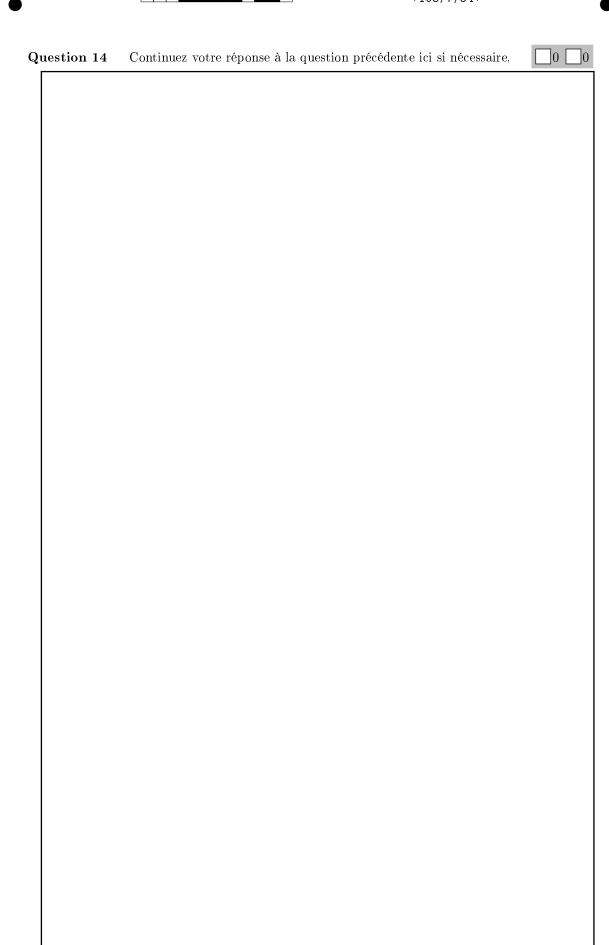
Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.





Question 13	Continuez	votre réponse à la question précédente ici si nécessaire.	





Une interface peut contenir des constructeurs
 Une interface peut contenir des méthodes privées

${f Question}$	15	Pour	la	question	précédente,	développez	la	$\begin{array}{c} \text{m\'ethode} \\ \hline  0 \end{array}$		equals.	
Question	<b>16</b> ⊕	Parn	ni le	es affirma	tions ci-desso	ous, cocher c	elle	s qui sont	vraies :		
					sse et implén		eurs	interfaces	5		
_					nir des consti						
_					sse abstraite menter une i						
_				eut impie re plusieu		merrace					

# Question $17 \oplus$ Soit le code ci-dessous :

```
@FunctionalInterface
interface ComparePerson {
    /**
    * @return negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

```
class Person {
    private int age;
    private String name;

Person(String name, int age) {
        this.age = age;
        this.name = name;
    }
}

class Main {
    public static void main(String[] args) {

        // code extract here

        System.out.print(byAge.compare(
            new Person("Fred", 23),
            new Person("Not Fred", 24)));
        System.out.print(byAge.compare(
            new Person("Not Fred", 23),
            new Person("Not Fred", 23));
        System.out.print(byAge.compare(
            new Person("Fred", 23),
            new Person("Fred", 23),
            new Person("Not Fred", 23),
            new Person("Not Fred", 23);
            new Person("Not Fred", 22)));
}
```

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :

```
ComparePerson byAge = new ComparePersonByAge() {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
};

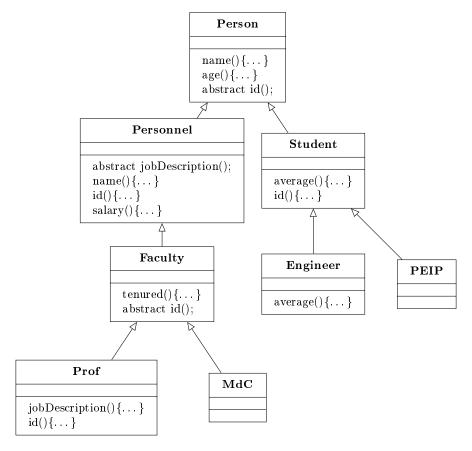
ComparePerson byAge
    = new ComparePersonByAge::compare;

ComparePerson byAge = new ComparePersonByAge();

ComparePerson byAge = new ComparePersonByAge();
```

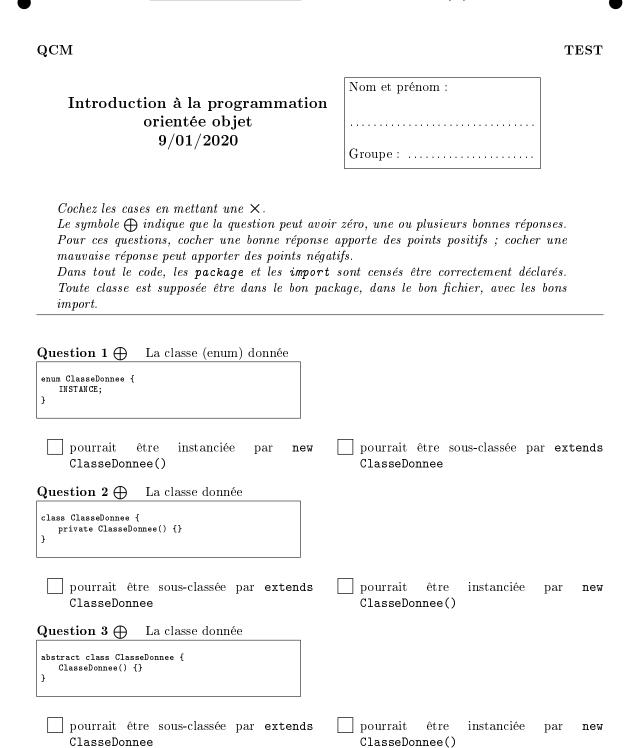


Question 18  $\bigoplus$  Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

Student	PEIP
Person	<pre>Engineer</pre>
MdC	$\Box$ Faculty
Prof	Personnel



Soit le code ci-dessous ; que se passe-t-il ?	Affiche "Caught exception" et "Finallied
public class Epsilon {	exception"
<pre>public void throwup() throws RuntimeException {    throw new RuntimeException();</pre>	Erreur d'exécution
} public static void main(String[] args) {	Erreur de compilation
<pre>Epsilon epsilon = new Epsilon(); try {</pre>	Affiche seulement "Finallied exception"
<pre>epsilon.throwup(); } catch (Exception e) {     System.out.println("Caught exception");</pre>	Affiche seulement "Caught exception"
<pre>} finally {     System.out.println("Finallied exception");</pre>	
}	
}	

# cécution compilation

# lement "Caught exception"

# Question $5 \bigoplus$

```
Soit le code ci-dessous ; que se passe-t-il?
public class Delta {
   public void throwup() throws Exception {
       throw new Exception();
   public static void main(String[] args) {
   Delta delta = new Delta();
        try {
    delta.throwup();
        } catch (Exception e) {
            System.out.println("Caught exception");
            return;
        } finally (
    System.out.println("Finallied exception");
```

# Erreur d'exécution Affiche "Caught exception" et "Finallied exception"

$oxedsymbol{oxed}$ Affiche seulement "Finallied exception	
---	--

- Erreur de compilation
- Affiche seulement "Caught exception"

# Question 6 $\bigoplus$

Soit le code ci-dessous ; que se passe-t-il? public class Alpha { public void throwup() throws Exception { throw new Exception(); public static void main(String[] args) { Alpha alpha = new Alpha();
alpha.throwup();
System.out.println("Done");

Erreur	${\rm d'ex\'ecution}$

Erreur de compilation

Affiche "Done"



#### Question 7 Soit le code :

```
interface Somethingable {
    void something();
}

interface Whateverable extends Somethingable {
    void whatever();
}

class TheThing implements Whateverable{
    PPP void something() {
        System.out.println("Somethinging");
    }

    PPP void whatever() {
        System.out.println("Whatevering");
    }
}
```

```
class Doofer {
   void doo(Somethingable sable) {
      sable.XXX();
   }

   void doo(Whateverable sable) {
      sable.YYY();
   }

   void fer(TheThing tt) {
      tt.ZZZ();
   }

   public static void main(String[] args) {
      Doofer doofer = new Doofer();
      TheThing the = new TheThing();
      doofer.doo(the);
      doofer.fer(the);
   }
}
```

Par quels niveaux d'accès pourrait-on remplacer PPP? public private package-private protected Question 8 Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer XXX? whatever something Question 9 Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer YYY? whatever something Question 10  $\bigoplus$ Pour le code de la question précédente, par quelles méthodes pourrait-on remplacer ZZZ? something whatever

Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit



Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r \cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r \sin(\theta)$   $\theta = \tan^{-1} \frac{y}{x}$ 

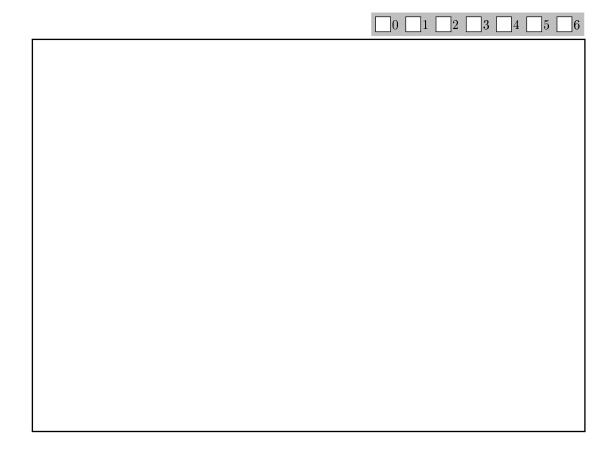
En Java ce dernier est calculé par theta = Math.atan2(y, x)

- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.





Question	13	Continuez	votre	réponse	à la qu	estion	précédent	e ici s	i nécessaire.	

Question 14	Continuez votre réponse à la question précédente ici si nécessaire.	0 0



Une classe abstraite peut implémenter une interface
 Une classe abstraite peut contenir des constructeurs

Une classe peut étendre une classe et implémenter plusieurs interfaces

Question	15	Pour	la	question	précédente,	développez	la	méthode	_	r#equal	.s.
Question	<b>16</b> $\oplus$	Parr	ni le	es affirma	tions ci-desso	ous, cocher c	elle	s qui sont	vraies	:	
Une i	nterfac	e peut	con	ntenir des	constructeur	°S					
_					sse abstraite						
_					rs classes						
Une i	interface	e peut	con	tenir des	méthodes pr	rivées					



# Question $17 \oplus$ Soit le code ci-dessous :

```
@FunctionalInterface
interface ComparePerson {
    /**
    * @return negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :

```
ComparePerson byAge = new ComparePersonByAge();

ComparePerson byAge = (p1, p2) -> p1.age - p2.age;

ComparePerson byAge
= new ComparePersonByAge::compare;
```

```
ComparePerson byAge = new ComparePersonByAge() {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
};
```



Question 18  $\bigoplus$  Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

Personnel	PEIP
Person	Prof
MdC	$\Box$ Faculty
Student	Engineer

QCMTEST Nom et prénom: Introduction à la programmation orientée objet 9/01/2020 Groupe: ..... Cochez les cases en mettant une  $\times$ . Le symbole  $\bigoplus$  indique que la question peut avoir zéro, une ou plusieurs bonnes réponses. Pour ces questions, cocher une bonne réponse apporte des points positifs ; cocher une mauvaise réponse peut apporter des points négatifs. Dans tout le code, les package et les import sont censés être correctement déclarés. Toute classe est supposée être dans le bon package, dans le bon fichier, avec les bons import.Question 1 

La classe donnée final class ClasseDonnee { ClasseDonnee() {} pourrait être sous-classée par extends pourrait être instanciée par new ClasseDonnee ClasseDonnee() Question 2 La classe donnée class ClasseDonnee {} pourrait être sous-classée par extends pourrait être instanciée new par ClasseDonnee() ClasseDonnee Question 3  $\bigoplus$ La classe (enum) donnée enum ClasseDonnee { INSTANCE; } pourrait être sous-classée par extends pourrait être instanciée new ClasseDonnee ClasseDonnee() Question 4 Erreur d'exécution Soit le code ci-dessous ; que se passe-t-il?

public class Beta {

}

public void throwup() throws RuntimeException {

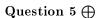
throw new RuntimeException();
}
public static void main(String[] args) {

Beta beta = new Beta();
beta.throwup();

System.out.println("Done");

Affiche "Done"

Erreur de compilation



```
Erreur d'exécution
Soit le code ci-dessous ; que se passe-t-il?
public class Gamma {
                                                             Affiche "Caught exception" et "Finallied
   public void throwup() throws Exception {
                                                             exception"
      throw new Exception();
                                                           Affiche seulement "Finallied exception"
   public static void main(String[] args) {
      Gamma gamma = new Gamma();
                                                             Erreur de compilation
      try {
         gamma.throwup();
                                                             Affiche seulement "Caught exception"
      } catch (Exception e) {
         System.out.println("Caught exception");
      } finally {
          System.out.println("Finallied exception");
   }
}
Question 6 \bigoplus
```

```
Soit le code ci-dessous ; que se passe-t-il ?

public class Epsilon {
    public void throwup() throws RuntimeException {
        throw new RuntimeException();
    }
    public static void main(String[] args) {
        Epsilon epsilon = new Epsilon();
        try {
            epsilon.throwup();
    } catch (Exception e) {
            System.out.println("Caught exception");
    } finally {
            System.out.println("Finallied exception");
    }
}
```

# Erreur de compilation

	Affiche	"Caught	exception"	et	"Finallied
	exception	on"			
$\neg$	ACC 1.	1 .	HT2: 11: 1		, · · · · · · · · · · · · · · · · · · ·

Affiche seulement "Finallied except	ion"
-------------------------------------	------

Erreur	d'exécution
--------	-------------

	Affiche	${\it seulement}$	"Caught	exception'
--	---------	-------------------	---------	------------

#### Question 7 Soit le code :

```
interface Somethingable {
   void something();
}

interface Whateverable extends Somethingable {
   void whatever();
}

class TheThing implements Whateverable{
   PPP void something() {
       System.out.println("Somethinging");
   }

  PPP void whatever() {
       System.out.println("Whatevering");
   }
}
```

```
class Doofer {
    void doo(Somethingable sable) {
        sable.XXX();
    }

    void doo(Whateverable sable) {
        sable.YYY();
    }

    void fer(TheThing tt) {
        tt.ZZZ();
    }

    public static void main(String[] args) {
        Doofer doofer = new Doofer();
        TheThing the = new TheThing();
        doofer.doo(the);
        doofer.fer(the);
    }
}
```

Par quels niveaux d'accès pourrait-on remplacer PPP?

Question 8 ⊕ placer XXX ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
something	whatever
Question 9 $\bigoplus$ placer YYY ?	Pour le code de la question précédente, par quelles méthodes pourrait-on rem-
whatever	something
Question 10 $\bigoplus$ remplacer ZZZ ?	Pour le code de la question précédente, par quelles méthodes pourrait-on
whatever	something



Rappel

• la forme cartésienne d'un nombre complexe s'écrit  $x+y\imath$ , où x est la partie réelle et y est la partie imaginaire.  $\imath^2=-1$ 

Java n'a pas de nombres complexes. Le stagiaire de l'année dernière avait développé la classe donnée :

```
public class Complex {
   public double re;
   public Complex(double a, double b) {
      re = a;
      im = b;
   }
   public double getRe() {
      return re;
   }
   public void setRe(double re) {
      this.re = re;
   }
   public double getIm() {
      return im;
   }
   public void setIm(double im) {
      this.im = im;
   }
```

```
public void add(Complex c) {
    re += c.re;
    im += c.im;
}

public void mult(Complex c) {
    re = re * c.re - im * c.im;
    im = re * c.im + im * c.re;
}

@Override
public String toString() {
    return "(" + re + ", " + im + ")";
}
```

A la fin du stage, la responsable de stage trouve le code pas adapté à l'usage — tout d'abord, elle aurait voulu que la classe Complex soit immuable, à l'image de Integer ou Double. Evidemment il revient à vous de le faire. Alors, faites-le.

Rappel: une classe A est immuable si

- un objet de classe A ne peut plus être modifié un fois construit
- A ne peut pas avoir de sous-classe, c'est à dire ...extends A... est interdit





Rappel:

- la forme polaire d'un nombre complexe s'écrit  $(r, \theta)$ , où r est le rayon (radius) et  $\theta$  est l'angle (angle).  $\theta$  est la lettre grecque theta
- la conversion entre les formes cartésienne et polaire se fait par

$$x = r\cos(\theta)$$
  $r = \sqrt{x^2 + y^2}$   
 $y = r\sin(\theta)$   $\theta = \tan^{-1}\frac{y}{x}$ 

En Java ce dernier est calculé par theta = Math.atan2(y, x)

- addition (de formes cartésiennes): (x,yi)+(u,vi)=(x+u,(y+v)i)• multiplication (de formes cartésiennes): (x,yi)+(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires): (x,yi)\*(u,vi)=(xu-yv,(xv+yu)i)• multiplication (de formes polaires):  $(r,\theta)*(s,\phi)=(r*s,\theta+\phi)$  pour simplifier, nous ne traitons pas les cas limites, par exemple où r=0, et nous nous limitons aux seules opérations

```
public class Cartesian {
   public Cartesian(double a, double b) {...}
   public Cartesian add(Cartesian c) {...}
   public Cartesian add(Polar p) {...}
   public Cartesian mult(Cartesian c) {...}
   public Cartesian mult(Polar p) {...}
   @Override
   public String toString() {
      return "Cartesian + ("
}
```

Le code donné ici avait été dévéloppé par un deuxième stagiaire, guère plus doué, pour représenter la forme cartésienne d'un nombre complexe (l'implémentation des méthodes est omise ici par souci d'économiser du papier). Il avait aussi développé la classe Polar, avec les mêmes méthodes mais renvoyant des objets de type Polar.

Hélas, la responsable est toujours très exigente, elle trouve qu'il y a beaucoup trop de redondances de code (pourquoi deux méthodes Cartesian#add par exemple? Une seule devrait suffir). Pas de bol, c'est encore à vous de refaire l'application. Vous pouvez refactoriser les classes Cartesian et Polar, et vous avez droit à toute autre modification de l'application que vous pouvez juger nécessaire. Implémentez toutes les méthodes de toutes vos classes.





Question 13	Continuez	votre réponse	à la question	précédente i	ci si nécessaire.	0 0



Question 14	Continuez	votre réponse	à la question	précédente	ici si nécessaire.	



Question	15	Pour	la	question	précédente,	développez	la	méthode Polar#equals.
Question	16 ⊕	Parm	i le	es affirma	tions ci-desso	ous, cocher c	elle	s qui sont vraies :
				re plusieu				
					constructeur			
					sse abstraite			
					menter une i			
Une interface peut contenir des méthodes privées								

Une classe peut étendre une classe et implémenter plusieurs interfaces

Une classe abstraite peut contenir des constructeurs

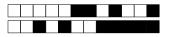


# Question $17 \oplus$ Soit le code ci-dessous :

```
@FunctionalInterface
interface ComparePerson {
    /**
    * Oreturn negative if p1 < p2, 0 if p1 = p2,
    * positive if p1 > p2
    */
    int compare(Person p1, Person P2);
}

class ComparePersonByAge implements ComparePerson {
    @Override
    public int compare(Person p1, Person p2) {
        return p1.age - p2.age;
    }
}
```

Lequel des extraits de code ci-dessous donne le résultat -101 quand substitué dans la méthode main :



# Question 18 $\bigoplus$ Soit l'organisation ci-dessous de Polytech'Groland :



Les méthodes sont toutes en niveau d'accès package-private. Pour que ce schéma compile, quelles classes doivent être déclarées abstract ?

Prof	Person
Engineer	MdC
Student	PEIP
Faculty	Personnel