



QCM

TEST

Introduction à la programmation  
orientée objet  
30/09/2016

Nom et prénom :

BEROUK. Zaki.....

Groupe : ...4.....

*Vous apporterez un très grand soin à la présentation car elle interviendra dans la notation. Par exemple, les réponses très peu lisibles ou contenant du code non indenté seront considérées comme fausses. Par ailleurs, la qualité du code proposé et la complexité des solutions interviendront dans la notation. Les questions à choix multiples marquées d'un trèfle admettent plusieurs réponses possibles alors que les autres questions n'en admettent qu'une. Les bonnes réponses apportent des points positifs, les mauvaises réponses peuvent apporter des points négatifs.*

*Toute classe est supposée être dans le bon package, dans le bon fichier, avec les bons imports.*

**Question 1** Le résultat de l'exécution du code ci dessous :

```
public class Foo {
    private String aboutMe = "Je suis Foo";

    @Override
    public String toString() {
        return aboutMe;
    }
}

public class Bar extends Foo {
    private String me = "J'en ai Bar";

    @Override
    public String toString() {
        return me;
    }
}

public class Main {
    public static void main(String[] args) {
        Foo foo = new Bar();
        System.out.println(foo);
    }
}
```

est ceci : J'en ai Bar

Hélas, ce n'est pas ce que veut la cliente ! Elle veut que soit affiché quelque chose du genre :

Je suis Foo

Mon adresse : Bar@1feed786

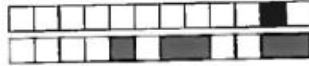
J'en ai Bar

La modification ci-dessous, donnera-t-elle le résultat souhaité ?

```
public class Bar extends Foo {
    private String me = "J'en ai Bar";

    @Override
    public String toString() {
        return super.toString() + "\n" + me;
    }
}
```

☐ vrai☒ faux

**Question 2 ♣** Pour que le code ci-dessous compile

```
public void toxiq() throws MyException {  
    // some code  
    if (someCondition()) {  
        throw new MyException("Oops, it just happened!");  
    }  
    // more code  
}
```

lesquelles des affirmations suivantes peuvent être vraies :

- ☒ MyException extends RuntimeException    ☐ MyException est une *checked exception*  
☐ MyException est une *unchecked exception*    ☒ MyException extends Exception

**Question 3 ♣** Prenez la méthode toxiq de la question précédente. Pour que le code ci-dessous compile

```
public void usingToxiq() {  
    // some code  
    toxiq();  
    // some more code  
}
```

lesquelles des affirmations suivantes doivent obligatoirement être vraies :

- ☐ MyException extends RuntimeException    ☐ MyException est une *checked exception*  
☐ MyException est une *unchecked exception*    ☒ MyException extends Exception

**Question 4** Le code ci-dessous :

```
public class A {  
    private String str;  
  
    public A(String str) {  
        this.str = str;  
    }  
  
    String getString() {  
        return str;  
    }  
  
    String toString(String str) {  
        return "A says " + this.str + " and " + str;  
    }  
}  
  
public class B extends A {  
    public B(String str) {  
        super(str);  
    }  
  
    protected String toString(String str) {  
        return "B says " + getString() + " and " + str;  
    }  
}  
  
public class Main {  
    public static void main(String[] args) {  
        A a = new B("Yo!");  
        System.out.print(a);  
    }  
}
```

- ☐ compile mais donne une erreur d'exécution    ☒ compile et exécute  
☐ ne compile pas

**Question 5 ♣** Dans la question précédente, pour quelles méthodes est-il correct de rajouter l'annotation @Override :

- ☒ A#toString    ☒ B#toString



**Question 6 ♣** Pour le code ci-dessous :

```
public abstract class Animal {  
    protected abstract String speak();  
    // other code  
}  
  
public class Toad extends Animal {  
    XXXX String speak() {  
        // speak like a toad  
    }  
}
```

quels niveaux d'accès sont admissibles pour remplacer XXXX :

☐ private

☒ public

☒ protected

☒ niveau package

**Question 7 ♣** Pour le code ci-dessous :

```
public interface Toadlike {  
    String speak();  
}  
  
abstract public class Animal {  
    protected abstract String speak();  
    // other code  
}  
  
public class Toad extends Animal implements Toadlike {  
    XXXX String speak() {  
        // speak like a toad  
    }  
}
```

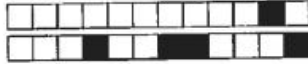
quels niveaux d'accès sont admissibles pour remplacer XXXX :

☐ public

☒ niveau package

☐ private

☒ protected



**Question 8 ♣** Vous essayez de compiler et d'exécuter le code ci-dessous :

```
import extension.Extender;

class Main {
    Main() {
        new Extender().display();
    }

    public static void main(String[] args) {
        new Main();
    }
}
```

Ce code utilise la classe **Original** définie dans le package **origin**:

```
package origin;

public abstract class Original {
    public abstract void display();
    public void print() {
        System.out.println("original");
    }
}
```

et étendue par la classe **Extender** dans le package **extension**:

```
package extension;
import origin.Original;

public class Extender extends Original {
    @Override
    public void display() {
        System.out.println("extender");
    }
}
```

Quelles affirmations sont **fausses** parmi les phrases suivantes ?

- |   |  |
|---|--|
| <input checked="" type="checkbox"/> Le compilateur indique que la classe <b>origin.Original</b> ne peut être redéfinie.                       | <input checked="" type="checkbox"/> Le programme affiche <b>original</b> à l'exécution.  |
| <input checked="" type="checkbox"/> Le compilateur indique que la méthode <b>Extender.display()</b> n'a pas de privilèges d'accès suffisants. | <input checked="" type="checkbox"/> Le compilateur indique que la méthode <b>display()</b> n'existe pas dans la superclasse de <b>extension.Extender</b> . |
| <input checked="" type="checkbox"/> L'exécution génère une <b>RuntimeException</b> au moment de l'appel de <b>display()</b> .                 | <input type="checkbox"/> Le programme affiche <b>extender</b> à l'exécution.   |



Question 9 ♣ Observez la classe suivante:

```
class Main {
    static int code;

    public void setCode(int code) {
        this.code = code;
    }

    public void display() {
        System.out.println(code);
    }

    public static void main(String[] args) {
        Main m1 = new Main(); Main m2 = new Main();
        m1.setCode(5);
        m1.display();
        m2.display();
    }
}
```

Quelles affirmations sont **vraies** parmi les phrases suivantes ?

- |  |  |
|--|--|
| <input checked="" type="checkbox"/> Une NullPointerException est générée par l'appel m2.display(). | avoir de variable statique dans une méthode dynamique.   |
| <input type="checkbox"/> m2.display() affiche des entiers.   | <input type="checkbox"/> Le code passe à la compilation.                                       |
| <input checked="" type="checkbox"/> code est une variable d'instance                               | <input type="checkbox"/> m1.setCode(5) génère une exception car la variable code est statique. |
| <input type="checkbox"/> Le compilateur indique qu'il ne peut y                                    |  |

Question 10 Printable est une interface. La déclaration suivante serait **légale** :  
public enum GroJours implements Printable

- |  |                               |
|--|-------------------------------|
| <input checked="" type="checkbox"/> vrai | <input type="checkbox"/> faux |
|--|-------------------------------|

Question 11 Jours est une classe. La déclaration suivante serait **légale** :  
public enum GroJours extends Jours

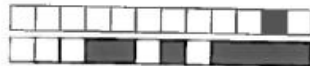
- |  |                               |
|--|-------------------------------|
| <input checked="" type="checkbox"/> vrai | <input type="checkbox"/> faux |
|--|-------------------------------|

Question 12 ♣ Soit le code ci-dessous.

```
public enum GroJourWE {
    SADI, GROMANCHE;
}
```

Quelles opérations donnent un résultat **true** :

- |   |   |
|---|---|
| <input checked="" type="checkbox"/> SADI instanceof Object    | <input type="checkbox"/> SADI instanceof Enum |
| <input checked="" type="checkbox"/> SADI instanceof GroJourWE |   |



Question 13 Polytech'Groland se sert du code ci-dessous pour gérer les notes des élèves :

```
/**
 * Marks are stored as subject, mark pairs.
 */
public class PolytechGroland {
    private final Map<String, Integer> marksmarks = new HashMap<>();

    public void put(String subject, int mark) {
        marks.put(subject, mark);
    }

    public Map<String, Integer> get() {
        return marks;
    }
}
```

Piètre élève, Haxxor a néanmoins des superpouvoirs informatiques, grâce auxquels il pense avoir trouvé comment améliorer ses notes insuffisantes. Complétez le code suivant avec sa solution, ainsi que la démo de son effet sur le cours donné :

```
/**
 * Student who wants to increment all PolytechGroland marks by exploiting
 * a flaw in the code.
 */
public class Haxxor {
    /**
     * Haxxor's secret exploit.
     */
    public void increment(// code à fournir marks) {
        // code à fournir
    }
}

public class Main {
    public static void main(String[] args) {
        PolytechGroland ptech = new PolytechGroland();
        ptech.put("Quantum Cooking", 9);
        // code à fournir
    }
}
```

☐ 0 ☐ 1 ☐ 2 ☒ 3 ☐ 4

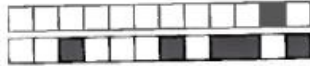


Question 14 Il faut choisir :

- `public class Ellipse extends Cercle`
- `public class Cercle extends Ellipse`

Justifiez votre choix.

☐ 0 ☐ 1 ☐ 2 ☐ 3 ☐ 4



### Question 15

Vous devez réaliser un logiciel permettant de tester des assertions logiques concernant des polygones. Ce logiciel conservera un certain nombre de polygones inconnus dans une liste. Vous devez pouvoir tester pour chacun d'entre eux s'ils comportent des symétries (méthode `hasSymmetries()`) et des angles droits (méthode `hasSquareAngles()`).

Par ailleurs, vous souhaitez pouvoir dessiner chacun des polygones de la liste. Vous disposez déjà de deux classes `Losange` et `Rectangle` qui fournissent des méthodes de dessin (`display()`). La classe `Losange` dispose d'ailleurs de deux accesseurs pour configurer la longueur de ses axes `setLongAxisLength()` et `setShortAxisLength()` et la classe `Rectangle` d'accesseurs `setLongSideLength()` et `setShortSideLength()`. Vous souhaitez ajouter un polygone `Carré`. Vous souhaitez réutiliser les méthodes implémentées dans les classes `Losange` et `Rectangle`. Quelle

architecture proposez-vous ? Justifiez vos choix de conception.

☐ 0 ☐ 1 ☐ 2 ☐ 3 ☐ 4