Contrôle 2: Informatique 10 janvier 2018

Cours de mathématiques spéciales Semestre d'automne

(écrire lisiblement et avec au maximum trois couleurs différentes s.v.p.)

Nom : ……...

Prénom : ……..….

Groupe : …….

Question	Barème	Points
1.1	23	
1.2	58	
2	20	
3	99	
Total	200	

Note :

Indications générales

- Durée de l'examen : **105 minutes**.
- Posez votre **carte d'étudiant** sur la table.
- La réponse à chaque question doit être rédigée **à l'encre** sur la place réservée à cet effet à la suite de la question.

  Si la place prévue ne suffit pas, vous pouvez demander des feuilles supplémentaires aux surveillants ; chaque feuille supplémentaire doit porter **nom, prénom, n° du contrôle, branche, groupe et date**. Elle ne peut être utilisée que pour **une seule question**.
- Les feuilles de brouillon ne sont pas à rendre ; elles ne seront **pas corrigées** ; des feuilles de brouillon supplémentaires peuvent être demandées en cas de besoin auprès des surveillants.
- Les feuilles d'examen doivent être rendues **agrafées**.

Indications spécifiques

- Toutes les questions qui suivent se réfèrent au langage de programmation **Java** (à partir du **JDK 8.0**).
- A part les polycopiés du cours, **aucune** autre source bibliographique n'est autorisée et ne doit donc être consultée durant le contrôle.

Remarques initiales :

- Il est conseillé de lire chaque question jusqu'à la fin, avant de commencer la rédaction de la solution.
- Les sous-points d'une même question peuvent être résolus (éventuellement) séparément, mais après avoir lu et compris l'ensemble des énoncés.
- Si vous ne savez pas implémenter une méthode, écrivez son en-tête, réduisez son corps à un simple commentaire et passez à la suite.
- La classe ***Object*** est une classe prédéfinie (ou standard) du package ***java.lang*** ; elle est la classe "racine", ancêtre de toute autre classe Java.
- Les indices des éléments des tableaux commencent à zéro.
- Faites attention aux éventuels **casts** obligatoires ainsi qu'aux accès aux champs privés.

Question 1

On considère un projet Java qui contient un package nommé ***cms_ctr2*** et y définit deux classes publiques Java, à savoir ***Instant*** et ***InstantPrecis***.

1.1 La classe publique ***Instant*** fait partie du package ***cms_ctr2*** et permet de créer et de manipuler des objets correspondant à des instants caractérisés par une heure donnée. Le code source de cette classe est donné ci-dessous et vous devez y ajouter la définition d'une nouvelle méthode ***arriver*** selon les indications précisées plus loin.

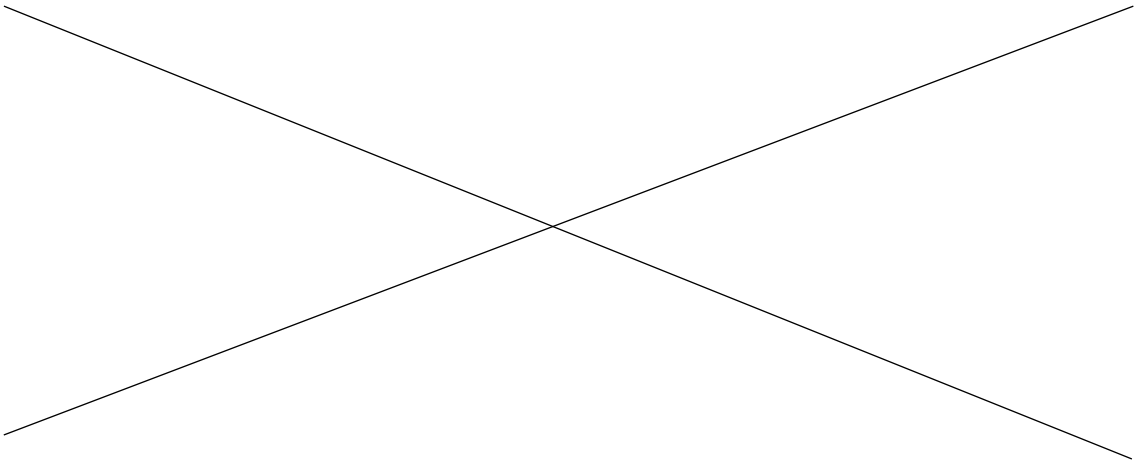
```
package cms_ctr2;

public class Instant {
	private int heure;

	public int getHeure() {
		return heure;
	}

	public void setHeure(int heure) {
		if(heure >= 0 && heure <= 23)
			this.heure = heure;
	}

	public Instant(int heure) {
		setHeure(heure);
	}
```

```
    //la méthode arriver que vous allez définir serait placée ici
}//fin de la classe Instant
```

La méthode d'instance publique ***arriver*** permet de comparer l'heure correspondant à l'objet argument et l'heure correspondant à l'objet appelant la méthode (en fonction de certains critères qui seront mentionnés ci-après). Plus précisément, cette méthode a un seul argument de type ***Object*** nommé ***inst***, retourne un résultat de type chaîne de caractères et doit respecter les consignes suivantes :

- si l'objet argument est le même que l'objet appelant (c'est-à-dire si l'adresse de l'objet argument est la même que l'adresse de l'objet appelant), la méthode retourne la chaîne de caractères ***Même instant !*** ;
- (autrement) si l'argument de la méthode a la valeur spéciale ***null***, la méthode retourne la chaîne de caractères ***Un instant unique !*** ;
- (autrement) si le type effectif de l'argument de la méthode **n**'est **pas** la classe ***Instant*** ou une autre classe descendante (donc compatible par polymorphisme), la méthode retourne la chaîne de caractères ***Un instant sans pareil !*** ;
- (autrement) si le champ ***heure*** de l'objet argument a la même valeur que le champ ***heure*** de l'objet appelant, la méthode retourne la chaîne de caractères ***A l'heure !*** ;
- (autrement) si la valeur du champ heure de l'objet argument est (strictement) plus petite que la valeur du champ heure de l'objet appelant, la méthode retourne la chaîne de caractères ***Avant l'heure !*** ;
- (autrement) la méthode retourne la chaîne de caractères ***Après l'heure !***.

Ecrivez à la page suivante la définition complète de la méthode ***arriver***, à savoir son en-tête et son corps.



.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

1.2 La classe publique ***InstantPrecis*** fait partie du package ***cms_ctr2*** et permet de créer et de manipuler des objets correspondant à des instants plus précis caractérisés par une heure et une minute données. Vu qu'un instant précis peut être regardé comme un instant qui offre une précision supplémentaire, la classe ***InstantPrecis*** doit dériver de la classe ***Instant***. Autrement dit, la classe ***InstantPrecis*** doit être la classe fille (ou la classe dérivée) de la classe mère (ou de la classe de base) ***Instant***.

Ainsi, la classe ***InstantPrecis*** :

- définit un champ (d'instance propre) privé nommé ***minute*** de type numérique entier, sans valeur initiale explicite et qui sert à stocker le nombre de minute compris entre **0** et **59** ;
- définit une méthode publique (d'instance) "**getter**" nommée en respectant la convention Java pour le nommage des getters et qui retourne la valeur du champ ***minute*** (sans aucune vérification particulière) ;
- définit une méthode publique (d'instance) "**setter**" nommée en respectant la convention Java pour le nommage des setters et qui stocke la valeur de son argument dans le champ propre ***minute*** seulement si cette valeur est bien comprise entre **0** et **59** ;
- définit un constructeur public avec deux arguments de type numérique entier nommées ***heure*** et ***minute*** et qui permet la création des objets correspondant à des instants précis ; ce constructeur doit respecter les consignes suivantes :
    - par un appel au **constructeur** de la classe mère, la valeur du premier argument ***heure*** est "stockée" dans le champ homonyme hérité de l'objet créé ;
    - par un appel à la méthode **setter** précisée plus haut, la valeur du deuxième argument ***minute*** est "stockée" dans le champ homonyme propre de l'objet créé ;
- redéfinit la méthode héritée ***arriver*** en respectant les consignes suivantes :
    - si le type effectif de l'argument de la méthode est la classe (fille) ***InstantPrecis*** ou un autre type compatible par polymorphisme, on procède ainsi :
        - si un appel de la méthode ***arriver*** d'origine (c'est-à-dire de la classe mère ***Instant***) pour le même argument (que celui de la méthode qu'on est en train de définir) retourne une chaîne de caractère qui correspond au message ***A l'heure !***, on procède ainsi :
            - si le champ ***minute*** de l'objet argument et le champ ***minute*** de l'objet appelant ont la même valeur, la méthode retourne la chaîne de caractères ***Tout à fait à l'heure !*** ;

- (autrement) la méthode retourne la chaîne de caractères ***Pas vraiment à l'heure !*** ;

- (autrement) la méthode retourne la chaîne de caractères ***Deux instants distincts !*** ;

- (autrement) si le type effectif de l'argument de la méthode est la classe (mère) ***Instant*** ou un autre type compatible par polymorphisme, on procède ainsi :
  - on déclare une variable locale nommée ***instPrec*** de type ***InstantPrecis*** et on y stocke l'adresse d'un nouvel objet de type ***InstantPrecis*** créé pour l'occasion par un appel au constructeur de la classe ***InstantPrecis*** ; ce nouvel objet doit avoir la valeur du champ ***heure*** égale à la valeur du champ ***heure*** de l'objet argument et la valeur du champ ***minute*** égale à zéro ;
  - la méthode retourne la chaîne de caractères obtenue par un appel (récursif) de la méthode ***arriver*** (de la classe fille ***InstantPrecis***) ayant comme argument (effectif) la variable locale ***instPrec*** ;
- (autrement) la méthode retourne la chaîne de caractères obtenue par un appel de la méthode ***arriver*** d'origine (c'est-à-dire de la classe mère ***Instant***) pour le même argument que celui de la méthode qu'on est en train de définir.

Ecrivez ci-dessous le code complet de la classe ***InstantPrecis***.

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Question 2

On considère un projet Java dont le code source est regroupé dans un seul fichier nommé **CP_Ctr2Exo2.java**. Plus précisément, on donne ci-dessous le contenu de ce fichier et on vous demande de préciser (à la page suivante) quels seront les messages affichés dans la fenêtre console suite à l'exécution de ce projet.

```
package cms_ctr2;
class Fruit {}
class Pomme extends Fruit {}
class McIntosh extends Pomme {}    class Golden extends Pomme {}
class Prune extends Fruit {}
class Nectarine extends Prune {}   class Mirabelle extends Prune {}

public class CP_Ctr2Exo2 {
  public static int[] compter(Fruit[] pannier) {
    int[] tab = new int[4];
    for(int i=0; i<pannier.length; i++) {
      if(pannier[i].getClass().getName().equals("cms_ctr2.Pomme"))
        tab[0]++;
      else if(pannier[i].getClass().getName().equals("cms_ctr2.Golden"))
        tab[1]++;
      else if(pannier[i] instanceof Prune)
        tab[2]++;
      else if(pannier[i] instanceof Nectarine)
        tab[3]++;
    }
    return tab;
  }//fin de la méthode compter

  public static void main(String[] args) {
    Fruit[] basket = { new Fruit(), new Prune(), new Nectarine(),
                new Mirabelle(), new McIntosh(), new Prune(),
                new Pomme(), new Golden(), new Prune(), new Nectarine(),
                new Nectarine(), new Golden(), new McIntosh(),
                new Pomme(), new Golden(), new Fruit() };
    int[] nombres = compter(basket);
    for(int i=0; i<nombres.length; i++) {
      System.out.println("Le nombre de fruits de type " + i + 1 + ":");
      System.out.println(nombres[i] + ".");
    }
  }//fin de la méthode main
}//fin de la classe principale
```

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Question 3

On considère un projet Java dont le code source est regroupé dans un seul fichier nommé **CP_Ctr2Exo3.java**. Plus précisément, on donne ci-dessous le contenu de ce fichier et on vous demande de préciser quels seront les messages affichés dans la fenêtre console suite à l'exécution de ce projet.

```
package cms_ctr2;

class Personne {
	String nom;
	static int nb = 0;

	Personne() {
		this("Anonymus");
		if(nb < 5)   System.out.println("Pas de nom connu !");
	}

	Personne(String nom) {
		this.nom = nom;
		if(nb++ < 5)   System.out.println("Un vrai nom !");
	}

	String saluer(Personne p) {
		return this.nom + " salue " + p.nom + " !";
	}

	public String toString() {
		return "La personne " + nom + ".";
	}
}

class Prof extends Personne {
	String nomCours = "Java";

	Prof() {
		if(nb < 5)   System.out.println("Le meilleur cours !");
	}

	Prof(String arg) {
		this("NoName", arg);
		if(nb < 5)   System.out.println("Un bon cours !");
	}

	Prof(String arg1, String arg2) {
		super(arg1);
		this.nomCours = arg2;
		if(nb < 5)   System.out.println("Un cours complet !");
	}
```

```
	String saluer(Personne p) {
		return "La/le prof " + this.nom + " présente ses hommages à " +
			p.nom + " !";
	}
	String saluer(Prof p) {
		return "Mes hommages pour ma/mon collègue " + p.nom + " !";
	}
	String saluer(Etudiant e) {
		if(this.nomCours.equals(e.nomCours))
			return "Bonjour à mon étudiant(e) " + e.nom + " !";
		return "Bonjour de la part de " + this.nom + " !";
	}

	public String toString() {
		return "Le prof " + nom + " donne le cours " + nomCours + ".";
	}
}

class Etudiant extends Personne {
	String nomCours = "Info";

	Etudiant(String arg1, String arg2) {
		super(arg1);
		this.nomCours = arg2;
	}
}

public class CP_Ctr2Exo3 {
	static void echanger(Object[] tab) {
		Object temp;
		int lon = tab.length;
		for(int i = 0; i < lon/2; i++) {
			temp = tab[i];
			tab[i] = tab[lon-i-1];
			tab[lon-i-1] = temp;
		}
	}
	public static void main(String[] args) {
		System.out.println("Partie 1");
		Personne[] personnes = new Personne[5];
		personnes[0] = new Prof("Euler", "Math");
		System.out.println("------------------------------");
		personnes[1] = new Personne("Gigi");
		System.out.println("------------------------------");
		personnes[2] = new Personne();
		System.out.println("------------------------------");
		personnes[3] = new Prof("Physique");
		System.out.println("------------------------------");
		personnes[4] = new Prof();
		System.out.println("------------------------------");
```

```
            System.out.println("Partie 2");
            echanger(personnes);
            for(int i=0; i<personnes.length; i++) {
                  System.out.println(personnes[i]);
            }

            System.out.println("Partie 3");
            Personne per1 = new Personne("Moi"),
                     per2 = new Personne("Toi");
            Prof  pro1 = new Prof("Newton", "Mécanique"),
                  pro2 = new Prof("Turing", "Info");
            Etudiant et1 = new Etudiant("Lui", "Info"),
                     et2 = new Etudiant("Elle", "Info");

            System.out.println(per1.saluer(per2));
            System.out.println(pro2.saluer(pro1));
            System.out.println(et1.saluer(et2));

            System.out.println("Partie 4");
            System.out.println(per1.saluer(pro2));
            System.out.println(per1.saluer(et1));

            System.out.println(pro2.saluer(per1));
            System.out.println(pro2.saluer(et1));

            System.out.println(et2.saluer(per2));
            System.out.println(et1.saluer(pro2));

            System.out.println("Partie 5");
            Personne mixte1 = new Prof("Poincaré", "Math"),
                     mixte2 = new Prof("Neumann", "Info"),
                     mixte3 = new Etudiant("Toi", "Info");
            System.out.println(mixte1.saluer(mixte2));
            System.out.println(((Prof)mixte1).saluer(((Prof)mixte2)));
            System.out.println(((Prof)mixte1).saluer(mixte2));
            System.out.println(mixte1.saluer(((Prof)mixte2)));
            System.out.println("**************************************");
            System.out.println(mixte2.saluer(mixte3));
            System.out.println(((Prof)mixte2).saluer(((Etudiant)mixte3)));
            System.out.println(mixte2.saluer(((Etudiant)mixte3)));
            System.out.println(((Prof)mixte2).saluer(mixte3));
      }//fin de la méthode main
}//fin de la classe principale
```

3.1 Écrivez ci-dessous les lignes affichées dans la fenêtre console <u>après</u> le message initial ***Partie 1*** et <u>avant</u> le message ***Partie 2***.

3.2 Écrivez ci-dessous les lignes affichées dans la fenêtre console après le message ***Partie 2*** et avant le message ***Partie 3***.

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

3.3 Écrivez ci-dessous les lignes affichées dans la fenêtre console après le message ***Partie 3*** et avant le message ***Partie 4***.

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

3.4 Écrivez ci-dessous toutes les lignes affichées dans la fenêtre console après le message ***Partie 4*** et avant le message ***Partie 5***.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

3.5 Écrivez ci-dessous toutes les lignes affichées dans la fenêtre console après le message ***Partie 5*** (et jusqu'à la fin des affichages).

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

A part ce texte, cette page doit rester normalement blanche (mais vous pouvez l'utiliser pour la rédaction de vos réponses si la place prévue à cet effet s'est avérée insuffisante).