Contrôle 2: Informatique 11 janvier 2017

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	26	
1.2	18	
1.3	60	
2	96	
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, nº 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 trois classes publiques Java, à savoir ***Services***, ***Logement*** et ***Villa***.

1.1 La classe publique ***Services*** fait partie du package ***cms_ctr2*** et définit deux méthodes publiques nommées ***dupliquer*** et ***chercher*** qui pourront être appelées avec le nom de la classe (donc sans l'aide d'un objet appelant) et qui seront utilisées dans la classe ***Villa***.

La méthode ***dupliquer*** permet d'obtenir l'adresse d'un nouveau tableau de chaînes de caractères qui a les mêmes éléments qu'un tableau de chaînes de caractères donné. Plus précisément, la méthode ***dupliquer*** a un seul argument de type tableau de chaînes de caractères, retourne un résultat de type tableau de chaînes de caractères et peut être écrite de la manière suivante :

- on crée une variable locale de type tableau de chaînes de caractères et on y stocke l'adresse d'un nouveau tableau de chaînes de caractères de même taille que le tableau argument ;
- on copie les valeurs des éléments du tableau argument dans les éléments correspondants du tableau local (par exemple, soit par un appel de la méthode prédéfinie ***arraycopy*** soit à l'aide d'une boucle appropriée) ;
- on retourne l'adresse du tableau local.

La méthode ***chercher*** permet de savoir si un tableau de chaînes de caractères donné contient une certaine chaîne de caractères. Plus précisément, la méthode ***chercher*** a deux arguments (le premier de type tableau de chaînes de caractères et le deuxième de type chaîne de caractères), retourne un résultat de type primitif logique et doit respecter les consignes suivantes :

- si le premier ou le deuxième argument de la méthode a la valeur spéciale ***null***, la méthode retourne la valeur logique **faux** ;
- (autrement) si (au moins) un des éléments du tableau correspondant au premier argument correspond à la même chaîne de caractères que le deuxième argument, la méthode retourne la valeur logique **vrai** ;
- (autrement), la méthode retourne la valeur logique **faux**.

Ecrivez ci-dessous le code complet de la classe ***Services***, à savoir : la déclaration du package, l'en-tête de la classe, les définitions (c'est-à-dire les en-têtes et les corps) des méthodes ***dupliquer*** et ***chercher***.

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

1.2 La classe publique ***Logement*** fait partie du package ***cms_ctr2*** et permet de créer et de manipuler des objets correspondant à des logements caractérisés par leur superficie. Le code source de cette classe est donné ci-dessous et vous devez y ajouter la définition d'une nouvelle méthode ***choisir*** selon les indications précisées plus loin.

```
package cms_ctr2;

public class Logement {
	private double superficie;

	public double getSuperficie() {
		return superficie;
	}

	public void setSuperficie(double s) {
		if(s>0) {
			superficie = s;
		}
	}

	public Logement(double surface) {
		setSuperficie(surface);
	}

	//la méthode choisir que vous allez définir sera ajoutée ci-dessous

}//fin de la classe Logement
```

La méthode publique ***choisir*** permet le choix entre l'objet argument et l'objet appelant la méthode (en fonction de certains critères qui seront mentionnés ci-après). Plus précisément, cette méthode a un seul arguent de type ***Object*** nommé ***obj***, retourne un résultat de type ***Object*** et doit respecter les consignes suivantes :

- si l'argument de la méthode a la valeur spéciale ***null***, la méthode retourne l'adresse correspondant à l'objet appelant la méthode ;
- (autrement) si le type effectif de l'argument de la méthode est la classe ***Logement*** ou un autre type compatible par polymorphisme :
    - si la valeur du champ ***superficie*** de l'objet argument est plus grande ou égale à la valeur du champ ***superficie*** de l'objet appelant, la méthode retourne l'adresse correspondant à l'objet argument ;
    - (autrement) la méthode retourne l'adresse correspondant à l'objet appelant ;
- (autrement) la méthode retourne l'adresse correspondant à l'objet appelant.

Ecrivez ci-dessous la définition complète de la méthode ***choisir***, à savoir son en-tête et son corps.

1.3 La classe publique ***Villa*** fait partie du package ***cms_ctr2*** et permet de créer et de manipuler des objets correspondant à des villas caractérisées par leur superficie et par certains éléments supplémentaires (par exemple : jardin, piscine, garage, etc.). Vu qu'une villa peut être regardée comme un logement qui offre un confort supplémentaire, la classe ***Villa*** doit dériver de la classe ***Logement***. Autrement dit, la classe ***Villa*** doit être la classe fille (ou la classe dérivée) de la classe mère (ou de la classe de base) ***Logement***.

Ainsi, la classe ***Villa*** :

- définit un champ (d'instance propre) privé nommé ***tabPlus*** de type tableau de chaînes de caractères, sans valeur initiale explicite et qui sert à stocker les éléments supplémentaires qui caractérisent une certaine villa ;
- définit une méthode publique (d'instance) "**getter**" nommée en respectant la convention Java pour le nommage des getters et qui, à l'aide d'un appel approprié de la méthode ***dupliquer*** de la classe ***Services***, retourne l'adresse d'un nouveau tableau de chaînes de caractères qui a les mêmes éléments que le tableau correspondant au champ privé ***tabPlus***;
- définit une méthode publique (d'instance) "**setter**" nommée en respectant la convention Java pour le nommage des setters et qui, à l'aide d'un appel approprié de la méthode ***dupliquer*** de la classe ***Services***, stocke dans le champ privé ***tabPlus*** l'adresse d'un nouveau tableau de chaînes de caractères qui a les mêmes éléments que le tableau correspondant à son argument ;
- définit un constructeur public avec deux arguments (le premier de type numérique réel nommé ***surface*** et le deuxième de type tableau de chaînes de caractères nommé ***tab***) et qui permet la création des objets correspondant à des villas ; ce constructeur doit respecter les consignes suivantes :
    - par un appel au constructeur de la classe mère, la valeur du premier argument ***surface*** est "stockée" dans le champ hérité ***superficie*** de l'objet créé ;
    - par un appel à la méthode **setter** précisée plus haut, l'adresse d'un nouveau tableau de chaînes de caractères qui a les mêmes éléments que le tableau correspondant à l'argument ***tab*** est stockée dans le champ privé ***tabPlus*** ;
- redéfinit la méthode héritée ***choisir*** en respectant les consignes suivantes :
    - si l'argument de la méthode a la valeur spéciale ***null***, la méthode retourne l'adresse correspondant à l'objet appelant la méthode ;
    - (autrement) si le type effectif de l'argument de la méthode est exactement la classe (fille) ***Villa***, on procède ainsi :

        - si un élément du tableau correspondant au champ ***tabPlus*** de l'objet argument correspond à la chaîne de caractère ***jardin*** (et cette information peut être obtenue par un appel approprié de la méthode ***chercher*** de la classe ***Services***), la méthode retourne l'adresse correspondant à l'objet argument ;
        - (autrement) si un élément du tableau correspondant au champ ***tabPlus*** de l'objet appelant correspond à la chaîne de caractère ***piscine*** (et cette information peut être obtenue par un appel approprié de la méthode ***chercher*** de la classe ***Services***), la méthode retourne l'adresse correspondant à l'objet appelant ;
    - (autrement) si le type effectif de l'argument de la méthode est exactement la classe (mère) ***Logement***, la méthode retourne le résultat obtenu par un appel de la méthode d'origine de la classe mère pour le même argument ;
    - (autrement) la méthode retourne l'adresse correspondant à l'objet appelant.

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

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

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

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 qui définit trois classes Java, à savoir : ***Cercle***, ***Cylindre*** et ***CP_Ctr2Exo2***. 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 Cercle {
	public static int NB = 0;
	private double r;

	public double getR() {
		return r;
	}

	public void setR(double ra) {
		if(ra>0) r = ra;
	}

	public Cercle() {
		this(1);
		System.out.println("Un cercle unitaire !");
	}

	public Cercle(double rayon) {
		setR(rayon);
		if(++NB < 6) {
			System.out.println("Un cercle quelconque !");
		}
	}

	public void grandir(double k) {
		setR(k*r);
	}
	public Cercle combattre(Cercle arg) {
		if(r>arg.r) {
			System.out.println("Le cercle appelant gagne.");
		}else if(r<arg.r) {
			System.out.println("Le cercle argument gagne.");
		}else {
			System.out.println("Les deux cercles à égalité");
		}
		return arg;
	}

	public void quiSuisJe() {
		System.out.println("Un cercle de rayon " + r + " !");
	}
}//fin de la classe Cercle
```

```
class Cylindre extends Cercle {
	private double h = 1;

	public double getH() { return h; }
	public void setH(double ha) { if(ha>0) h = ha; }

	public Cylindre() {
		System.out.println("Cylindre de hauteur 1 !");
	}

	Cylindre(double hauteur) {
		this(3, hauteur);
		System.out.println("Cylindre avec un argument !");
	}

	Cylindre(double ray, double haut) {
		super(ray);
		if(NB < 6) {
			System.out.println("Cylindre quelconque !");
		}
		setH(haut);
	}

	public void grandir(int k) {
		setH(k*h);
	}

	public Cylindre combattre(Cercle arg) {
		if(arg instanceof Cylindre) {
			System.out.println("Les jeunes discutent !");
			if(h>((Cylindre)arg).h) {
				System.out.println("L'appelant propose la paix.");
			}else if(h<((Cylindre)arg).h) {
				System.out.println("L'argument propose la paix.");
			}else {
				System.out.println("Les deux font la paix.");
			}
		}else {
			System.out.println("Un autre combat !");
			super.combattre(arg);
		}
		return this;
	}

	public Cylindre combattre(Cylindre arg) {
		System.out.println("Les enfants luttent pour la paix !");
		return this;
	}

	public void quiSuisJe() {
		System.out.println("Cylindre de rayon " + getR()
					+ " et de hauteur " + h + " !");
	}
}//fin de la classe Cylindre
```

```
public class CP_Ctr2Exo2 {
	public static void main(String[] args) {
		System.out.println("On crée des objets !");
		Cercle[] tab = new Cercle[5];
		tab[0] = new Cylindre();
		System.out.println("----------------------------------------");
		tab[1] = new Cercle(10);
		System.out.println("----------------------------------------");
		tab[2] = new Cercle();
		System.out.println("----------------------------------------");
		tab[3] = new Cylindre(4, 5);
		System.out.println("----------------------------------------");
		tab[4] = new Cylindre(20);

		System.out.println("On mélange des objets !");
		for(int i=0; i<4; i++) tab[i] = tab[i+1];
		for(int i=0; i<5; i++) tab[i].quiSuisJe();

		System.out.println("On fait grandir des objets!");
		Cercle ce_1 = new Cercle(10);
		ce_1.grandir(4);                    ce_1.quiSuisJe();
		System.out.println("----------------------------------------");
		Cylindre cy_1 = new Cylindre(50, 8);
		cy_1.grandir(3);                    cy_1.quiSuisJe();
		cy_1.grandir(4.);                   cy_1.quiSuisJe();
		System.out.println("----------------------------------------");
		Cercle mixte = new Cylindre(25,1);
		mixte.grandir(3);                   mixte.quiSuisJe();
		mixte.grandir(4.);                  mixte.quiSuisJe();
		((Cylindre)mixte).grandir(5);       mixte.quiSuisJe();

		System.out.println("Combats homogènes");
		Cercle cer_1 = new Cercle(10);   Cercle cer_2 = new Cercle(15);
		Cercle cer_res = cer_1.combattre(cer_2);
		cer_res.quiSuisJe();
		System.out.println("----------------------------------------");
		Cylindre cyl_1= new Cylindre(50, 8);
		Cylindre cyl_2 = new Cylindre(100, 3);
		Cylindre cyl_res = cyl_2.combattre(cyl_1);
		cyl_res.quiSuisJe();

		System.out.println("Combats croisés");
		Cercle cer_cro = cer_1.combattre(cyl_1); cer_cro.quiSuisJe();
		cer_cro = cyl_2.combattre(cer_2);        cer_cro.quiSuisJe();

		System.out.println("Combats mixtes");
		Cercle mixte_1 = new Cylindre(25, 1);
		Cercle mixte_2 = new Cylindre(20, 2);
		(mixte_1.combattre(mixte_2)).quiSuisJe();
		System.out.println("----------------------------------------");
		(((Cylindre)mixte_1).combattre(mixte_2)).quiSuisJe();
		System.out.println("----------------------------------------");
		(mixte_1.combattre((Cylindre)mixte_2)).quiSuisJe();
		System.out.println("----------------------------------------");
		(((Cylindre)mixte_1).combattre((Cylindre)mixte_2)).quiSuisJe();
}}//fin de la méthode main et de la classe principale
```

2.1 Écrivez ci-dessous les lignes affichées dans la fenêtre console après le message ***On crée des objets !*** et avant le message ***On mélange des objets !***.

2.2 Écrivez ci-dessous les lignes affichées dans la fenêtre console après le message ***On mélange des objets !*** et avant le message ***On fait grandir des objets !***.

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

2.3 Écrivez ci-dessous les lignes affichées dans la fenêtre console après le message ***On fait grandir des objets !*** et avant le message ***Combats homogènes !***.

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

2.4 Écrivez ci-dessous toutes les lignes affichées dans la fenêtre console après le message ***Combats homogènes*** (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).