Contrôle 1: Informatique 9 novembre 2016

Cours de mathématiques spéciales Semestre d'automne

(écrire lisiblement s.v.p.)

Nom : ……...

Prénom : ……..….

Groupe : …….

Question	Barème	Points
1.1	35	
1.2	27	
2.1	72	
2.2	66	
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 le polycopié du cours, **aucune** autre source bibliographique n'est autorisée et ne doit donc pas être consultée durant le contrôle.

Question 1

1.1 Soit le fichier ***CP_Ctr1Exo1_1.java*** contenant le code source suivant :

```
package cms_ctr1;

public class CP_Ctr1Exo1_1
{
	public static void main(String[] args)
	{
		int k;
		for(k = 5; k >= 0; k = k-1)
		{
			switch(k % 3){
				case 0 :
					System.out.println("Scénario 0");
				case 1 :
					System.out.println("Scénario 1");
					continue;
				case 2 :
					System.out.println("Scénario 2");
				default :
					System.out.println("Scénario par défaut");
			}
			System.out.println("********************");
		}
		System.out.println("k=" + k + ".");
	}//fin de la méthode main
}//fin de la classe principale
```

Sur la page suivante, précisez les messages qui seront affichés à l'écran suite à l'exécution du projet correspondant au code source indiqué ci-dessus.

```
Scénario 2
Scénario par défaut
******************
Scénario 1
Scénario 0
Scénario 1
Scénario 2
Scénario par défaut
******************
Scénario 1
Scénario 0
Scénario 1
k=-1.
```

1.2 Soit le fichier ***CP_Ctr1Exo1_2.java*** contenant le code source suivant :

```
package cms_ctr1;

public class CP_Ctr1Exo1_2
{
	public static void main(String[] args)
	{
		int i = 0;
		int compteur = 1;
		do
		{
			if(i < 2)
			{
				compteur = compteur + ++i;
			}else if(i < 5)
			{
				compteur = compteur - i++;
				System.out.println("************************");
			}else
			{
				compteur += 10;
				System.out.println("-----------------");
			}
			System.out.println("i=" + i +
					" et compteur=" + compteur + ".");
		}while(compteur <= 11);
		System.out.println("Au revoir !");
	}//fin de la méthode main
}//fin de la classe principale
```

Sur la page suivante, précisez les messages qui seront affichés à l'écran suite à l'exécution du projet correspondant au code source indiqué ci-dessus.

```
i=1 et compteur=2.
i=2 et compteur=4.
**********************
i=3 et compteur=2.
**********************
i=4 et compteur=-1.
**********************
i=5 et compteur=-5.
-----------------
i=5 et compteur=5.
-----------------
i=5 et compteur=15.
Au revoir !
```

Question 2

Le but de cet exercice est de réaliser une application autonome interactive qui aide l'utilisateur à manipuler des objets correspondant à des verres à boire. Cette application doit correspondre à un projet doté d'un package nommé ***cms_ctr1*** et qui contient deux classes publiques, définies dans deux fichiers distincts et appelées ***Verre*** et, respectivement, ***CP_Ctr1Exo2***. Par la suite, aux points **2.1** et **2.2**, on vous demande d'écrire le code complet de ces deux classes, en respectant les consignes précisées. Vous pouvez éventuellement répondre (d'abord) au point **2.2** en supposant le point **2.1** résolu correctement.

2.1 Une instance de la classe ***Verre*** est un objet de type ***Verre*** qui modélise un certain verre à boire et stocke le volume du ***contenu*** du verre correspondant (exprimée en cl). En outre, la classe ***Verre*** précise la ***contenance*** figée et commune pour tous les verres (exprimée en cl) ainsi que des méthodes permettant de manipuler les objets de type ***Verre***.

Plus précisément, la classe ***Verre*** doit définir :

- **a)** un champ sans modificateur d'accès, de type numérique réel, nommé ***contenance*** et initialisé explicitement avec la valeur ***50*** ; la valeur de ce champ doit être figée et commune pour tous les objets de type ***Verre*** ; en fait, ce champ sert à stocker (au niveau de la classe) la contenance fixe et commune à tous les verres modélisés par des objets de type ***Verre*** ;
- **b)** un champ (d'instance) privé, de type numérique réel, nommé ***contenu*** et sans valeur initiale explicite ; en fait, ce champ sert à stocké le volume du contenu d'un certain verre modélisé par un objet de type ***Verre*** ;
- **c)** une méthode publique (d'instance) appelée ***getContenu*** qui permet de "lire" la valeur du champ privé ***contenu*** ; plus précisément, cette méthode n'a pas d'argument et retourne la valeur du champ ***contenu*** (de l'objet qui appelle la méthode) ;
- **d)** une méthode publique (d'instance) appelée ***setContenu*** qui permet de modifier la valeur du champ privé ***contenu*** ; ainsi, cette méthode a un seul argument (qui correspond à la nouvelle valeur proposée pour le champ ***contenu***) et ne retourne pas de résultat ; plus précisément :
  - **i.** si la valeur de l'argument de la méthode est strictement plus petite que ***10*** ou strictement plus grande que la valeur du champ ***contenance***, on affiche dans la fenêtre console le message ***Argument invalide !*** ;
  - **ii.** autrement, on copie la valeur de l'argument de la méthode dans le champ ***contenu*** (de l'objet qui appelle la méthode) ;

e) un constructeur public sans argument et qui stocke dans le champ ***contenu*** (du nouvel objet créé) la valeur réelle ***10*** ;

f) un constructeur public (surchargé) qui a un seul argument de type numérique réel dont la valeur doit être copiée (si elle est valide) dans le champ ***contenu*** (du nouvel objet créé) ; plus précisément, ce constructeur doit simplement appeler de manière appropriée la méthode ***setContenu*** définie plus haut ;

g) une méthode publique (d'instance) nommée ***ajouter*** qui a un seul argument de type numérique entier nommé ***delta*** et qui ne retourne pas de résultat ; plus précisément :

- **i.** si la somme entre la valeur de l'argument de la méthode et la valeur du champ ***contenu*** (de l'objet appelant) est strictement plus grande que la valeur du champ ***contenance***, on copie la valeur du champ ***contenance*** dans le champ ***contenu*** ;
- **ii.** autrement, on stocke dans le champ ***contenu*** (de l'objet appelant) la somme entre l'ancienne valeur du champ ***contenu*** et la valeur de son l'argument ;

h) une méthode publique (d'instance) nommée ***vider*** qui n'a pas d'argument et qui retourne une valeur numérique réelle ; plus précisément, cette méthode :

- **i.** copie la valeur du champ ***contenu*** (de l'objet appelant) dans une variable locale créée à cet effet ;
- **ii.** stocke la valeur réelle ***0*** dans le champ ***contenu*** ;
- **iii.** retourne l'ancienne valeur du champ ***contenu*** (grâce à la variable locale définie au point **i.** ci-dessus) ;

i) une méthode publique et statique nommée ***cloner*** dont le seul argument nommé ***original*** correspond à un objet de type ***Verre*** ; en fait, cette méthode doit créer un clone de l'objet argument et retourner l'adresse de ce clone ; plus précisément, cette méthode doit :

- **i.** créer un nouvel objet de type ***Verre*** en assurant que la valeur de son champ ***contenu*** soit égale à la valeur du champ ***contenu*** de l'objet argument ;
- **ii.** retourner l'adresse du nouvel objet créé ;

j) une méthode publique (d'instance) nommée ***versString*** qui n'a pas d'argument et qui retourne une chaîne de caractères qui donne des informations concernant l'objet qui appelle la méthode, à savoir :

Verre de contenance XXX cl et de contenu YYY cl.

où **XXX** représente la valeur du champ ***contenance*** et **YYY** représente la valeur du champ ***contenu*** (de l'objet appelant).

Écrivez ci-dessous le code complet de la classe publique ***Verre*** qui appartient au package ***cms_ctr1*** (c'est-à-dire la déclaration du package, l'en-tête de la classe et le corps de la classe).

```java
package cms_ctr1;

public class Verre
{
     final static double contenance = 50;
     private double contenu;

     public double getContenu()
     {
          return contenu;
     }

     public void setContenu (double nouveauContenu)
     {
          if(nouveauContenu < 10 ||
                    nouveauContenu > contenance)
          {
               System.out.println("Argument invalide !");
          }else
          {
               contenu = nouveauContenu;
          }
     }

     public Verre()
     {
          //this(10);
          contenu = 10;
     }

     public Verre(double contenuInitial)
     {
          setContenu(contenuInitial);
     }
```

```
    public void ajouter(double delta)
    {
        if(contenu + delta > contenance)
        {
            contenu = contenance;
        }else
        {
            contenu = contenu + delta;
        }
    }

    public double vider( )
    {
        double res = contenu;
        contenu = 0;
        return res;
    }

    public static Verre cloner(Verre original)
    {
        //return new Verre(original.contenu);
        Verre clone = new Verre();
        clone.contenu = original.contenu;
        return clone;
    }

    public String versString( )
    {
        return "Verre de contenance=" + contenance +
                  " cl et de contenu=" + contenu + " cl.";
    }
}
```

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

2.2 Ecrire le code complet de la classe publique ***CP_Ctr1Exo2*** qui appartient au package ***cms_ctr1*** (c'est-à-dire la déclaration du package, les éventuelles instructions ***import***, l'en-tête et le corps de la classe). Cette classe "principale" contient notamment la méthode ***main*** qui réalise la partie interactive du projet et utilise la classe ***Verre***.

Ci-dessous se trouve un exemple d'affichage obtenu suite à l'exécution de la classe ***CP_Ctr1Exo2***.

```
Introduisez le contenu du 1er verre :
30
Verre de contenance=50.0 cl et de contenu=30.0 cl.
On a vidé le premier verre de ses 40.0 cl.
Verre de contenance=50.0 cl et de contenu=10.0 cl.
Introduisez le contenu du 2ème verre :
20
Le troisième verre contient 20.0 cl.
```

On présente ci-dessous le squelette de la classe publique ***CP_Ctr1Exo2*** et vous devez compléter ce canevas en fonction des indications données en commentaires.

```
//déclaration du package
package cms_ctr1;

//les éventuelles instructions import
import java.util.Scanner;

//l'en-tête de la classe principale
public class CP_Ctr1Exo2
{
    //l'en-tête de la méthode main
    public static void main(String[] args)
```

```
{
    //déclarez et initialisez une variable locale nécessaire pour la lecture des
    //réponses de l'utilisateur
    Scanner scan = new Scanner(System.in);

    //déclarez trois variables locales de type Verre nommées
    // verre1, verre2 et verre3 (sans valeurs initiales)
    Verre verre1, verre2, verre3;

    //déclarez quatre variables locales de type numérique réel nommées
    //contenu1, contenuVerse, contenu2 et contenu3 (sans valeurs initiales)
    double contenu1, contenuVerse, contenu2, contenu3;

    //affichez dans la console le message Introduire le contenu du 1er verre :
    System.out.println("Introduisez le contenu du 1er verre :");

    //récupérez la réponse de l'utilisateur et stockez la dans la variable contenu1
    contenu1 = scan.nextDouble( );

    //créez un nouvel objet de type Verre dont le champ contenu correspond à la
    //valeur de la variable contenu1 et stockez son adresse dans la variable verre1
    verre1 = new Verre(contenu1);

    //à l'aide d'un appel de la méthode d'instance versString, affichez dans la
    //console les informations pertinentes concernant l'objet d'adresse verre1
    System.out.println(verre1.versString());
```

//par un appel à la méthode d'instance ***ajouter***, augmentez de ***10*** la valeur du
//champ ***contenu*** de l'objet d'adresse ***verre1***

```
verre1.ajouter(10);
```

//appelez la méthode d'instance ***vider*** pour l'objet appelant d'adresse ***verre1*** et
//copiez le résultat retourné dans la variable ***contenuVerse***

```
contenuVerse = verre1.vider( );
```

//affichez le message
//***On a vidé le premier verre de ses ZZZ cl.***
//où ***ZZZ*** correspond à la valeur de la variable ***contenuVerse***

```
System.out.println("On a vidé le premier verre de ses " +
                                        contenuVerse + " cl.");
```

//à l'aide d'un appel au constructeur sans argument, créez un nouvel objet de
//type ***Verre*** et stockez son adresse dans la variable ***verre2***

```
verre2 = new Verre();
```

//par un appel de la méthode d'instance ***versString***, affichez dans la console les
//informations pertinentes concernant l'objet d'adresse ***verre2***

```
System.out.println(verre2.versString());
```

//affichez dans la console le message ***Introduire le contenu du 2ème verre :***

```
System.out.println("Introduit le contenu du 2ème verre :");
```

//récupérez la réponse de l'utilisateur et stockez la dans la variable ***contenu2***

```
contenu2 = scan.nextDouble( );
```

```
        //par un appel à la méthode d'instance "setter" appropriée, copiez la valeur de la
        //variable contenu2 dans le champ contenu de l'objet d'adresse verre2
        verre2.setContenu(contenu2);

        //par un appel à la méthode statique cloner, créez un clone de l'objet d'adresse
        //verre2 et stockez son adresse dans la variable verre3
        verre3 = Verre.cloner(verre2);

        //à l'aide d'un appel à la méthode d'instance "getter" appropriée, copiez la
        //valeur du champ contenu de l'objet d'adresse verre3 dans la variable
        //contenu3
        contenu3 = verre3.getContenu();

        //affichez le message
        //Le troisième verre contient WWW cl.
        //où WWW correspond à la valeur de la variable contenu3

        System.out.println("Le troisième verre contient "
                                    + contenu3 + " cl.");
    }//fin de la méthode main
}//fin de la classe principale
```