EPF – Lausanne *COURS DE MATHEMATIQUES SPECIALES*

12 janvier 2011

Contrôle d'informatique no 2

Durée : 1 heure 45'

Nom : …… Groupe : ☐

Prénom : ……

No	1	2
Total points	90 points (50 + 40)	90 points (70 + 20)

Remarque générale : toutes les questions qui suivent se réfèrent au langage de programmation Java (à partir du JDK 5.0) et les réponses doivent être rédigées à l'encre et d'une manière propre sur ces feuilles agrafées.

Sujet no 1.

1.1 Préciser les messages qui seront affichés à l'écran suite à l'exécution du projet contenant les deux classes suivantes (qui appartiennent à un même package).

Remarque : la réponse au point 1.1 doit être donnée à la page 4.

```
package cms_ctr2;

public class Etudiant
{
	String nom;
	long noCarte = 1000;
	static int i = 0;

	Etudiant( )
	{
		if(i < 3)
		{
			System.out.println("Constructeur sans argument !");
		}
		nom = "Gigi";
		i++;
	}

	Etudiant(String unNom, long unNumero)
	{
		if(i < 3)
		{
			System.out.println("Constructeur 2 arguments !");
		}
		nom = unNom;
		noCarte = unNumero;
		i++;
	}

	void brouiller(Etudiant et)
	{
		System.out.println("On commence à mélanger !");
		this.nom = this.nom + et.nom;
		et.noCarte = et.noCarte + this.noCarte;
		System.out.println("On finit de mélanger !");
	}

	static Etudiant modifier(Etudiant et1, Etudiant et2)
	{
		System.out.println("On commence à modifier !");
		et1.brouiller(et2);
		System.out.println("On finit de modifier !");
		return et2;
	}

	void afficher()
	{
		System.out.println("L'étudiant(e) " + nom +
				" a le numéro CAMIPRO " + noCarte + ".");
	}
}//fin de la classe Etudiant
```

```
package cms_ctr2;

public class CP_Ctr2Exo1
{
	public static void main(String[] args)
	{
		System.out.println("Première partie");
		Etudiant stud_1 = new Etudiant ("Toto", 1111);
		Etudiant stud_2 = new Etudiant("Dupont", 2222);
		Etudiant stud_3 = new Etudiant( );

		stud_3.afficher( );

		System.out.println("********************");
		System.out.println("Deuxième partie");
		stud_2 = stud_1;
		stud_1 = stud_3;
		stud_3 = stud_2;

		stud_1.afficher( );
		stud_2.afficher( );
		stud_3.afficher( );

		System.out.println("********************");
		System.out.println("Troisième partie");
		Etudiant stud_4 = new Etudiant("Blanc", 4444);
		Etudiant stud_5 = new Etudiant("Bonnet", 5555);

		stud_4.brouiller(stud_5);
		stud_4.afficher( );
		stud_5.afficher( );

		System.out.println("********************");
		System.out.println("Quatrième partie");
		Etudiant stud_7 = new Etudiant("Petite", 7000);
		Etudiant stud_8 = new Etudiant("Blanche", 8000);
		Etudiant stud_9 = new Etudiant("Neige", 9000);

		stud_7 = Etudiant.modifier(stud_8, stud_9);
		stud_7.afficher( );
		stud_8.afficher( );
		stud_9.afficher();

	}//fin de la méthode main

}//fin de la classe principale
```

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

1.2 Préciser les messages qui seront affichés à l'écran suite à l'exécution du projet contenant la classe suivante.

```
package cms_ctr2;

public class CP_Ctr2Exo2
{   public static void main(String[] args)
    { boolean cc = true;
      //le code UNICODE du caractère 'a' vaut 97
      char car = 'd';
      int j, k;

      monOeil : while(cc)
      { //l'opérateur % est l'opérateur modulo
        j = car % 3;
        switch(j)
        { case 0 : System.out.println("Le cas 0 !");
                   car--;
                   System.out.println("La variable car : " + car + ".");
                   //pas de break entre le bloc case 0 et le bloc case 1
          case 1 : System.out.println("Le cas 1 !");
                   car += 3;
                   System.out.println("La variable car : " + car + ".");
                   break;
          case 2 : System.out.println("Le cas 2 !");
                   car -= 4;
                   System.out.println("La variable car : " + car + ".");
                   break;
          default : cc = false;
                   System.out.println("Le cas par défaut !");
        }//fin du choix switch

        --car;

        for(k = car; k > 0; k = k - 2)
        {
          if(k == 101)
          { car++;
            System.out.println("La variable car : " + car + ".");
            break;
          }
          if(k == 99)
          { car++;
            System.out.println("La variable car : " + car + ".");
            break monOeil;
          }
          if(k == 96)
          { System.out.println("La variable car : " + car + ".");
            break ;
          }
          System.out.println("Petite boucle avec k = " + k + ".");
        }//fin de la boucle for

        System.out.println("Après la petite boucle !");
        System.out.println("************************************");
      }//fin de la boucle while
      System.out.println("Après la grande boucle ");
    }//fin de la méthode main
}//fin de la classe principale
```

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Sujet no 2.

Le but de cet exercice est de réaliser une application autonome interactive qui aide l'utilisateur à travailler avec des droites situées dans un plan muni d'un repère cartésien. Cette application doit correspondre à un projet doté d'un package appelé **cms_ctr2** et qui contient deux classes **publiques**, définies dans deux fichiers distincts et appelées **Droite** et, respectivement, **CP_Ctr2Exo3**. 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 au point **2.2** en supposant le point **2.1** résolu correctement.

Indications :

- la valeur spéciale "Plus Infini" est disponible dans la classe enveloppe (wrapper class) **Double** sous la forme d'un champ statique (et final) appelé **POSITIVE_INFINITY** ;
- la valeur spéciale "Moins Infini" est disponible dans la classe enveloppe (wrapper class) **Double** sous la forme d'un champ statique (et final) appelé **NEGATIVE_INFINITY** ;
- la valeur spéciale "Not A Number" est disponible dans la classe enveloppe (wrapper class) **Double** sous la forme d'un champ statique (et final) appelé **NaN** ;
- afin de tester si une variable réelle a la valeur spéciale "Plus Infini" ou "Moins Infini", on utilise la (**même**) méthode statique **isInfinite** de la classe enveloppe **Double**, qui retourne un résultat de type booléen et qui a un seul argument de type réel ; plus précisément :
    - **i.** si la variable réelle passée comme argument à la méthode **isInfinite** a bien la valeur spéciale "Plus Infini" ou "Moins Infini", le résultat retourné est **true** ;
    - **ii.** autrement (c'est-à-dire si la variable réelle passée comme argument à la méthode **isInfinite** n'a pas une valeur infinie), le résultat retourné est **false** ;
- afin de tester si une variable réelle a la valeur spéciale "Not A Number", on utilise la méthode statique **isNaN** de la classe enveloppe **Double**, qui retourne un résultat de type booléen et qui a un seul argument de type réel ; plus précisément :
    - **i.** si la variable réelle passée comme argument à la méthode **isNaN** a bien la valeur spéciale "Not A Number", le résultat retourné est **true** ;
    - **ii.** autrement (c'est-à-dire si la variable réelle passée comme argument à la méthode **isNaN** a une valeur réelle "normale"), le résultat retourné est **false**.

2.1 Une instance de la classe **Droite** est un objet de type **Droite** qui regroupe (ou encapsule) des informations sur une droite située dans un plan donné et précisée par sa pente et son ordonnée à l'origine. De plus, la classe **Droite** est munie des méthodes permettant de manipuler les objets de type **Droite**.

Ecrire le code complet de la classe **publique Droite** qui appartient au package **cms_ctr2** et qui définit :

- **a)** un champ nommé **pente** de type réel, déclaré privé et sans valeur initiale explicite et qui sert à stocker, soit la valeur finie de la pente de la droite, soit la valeur spéciale "Plus Infini" ou "Moins Infini" si la droite est (presque) verticale ;
- **b)** un champ nommé **ordOrig** de type réel, déclaré sans modificateur d'accès et sans valeur initiale explicite et qui sert à stocker, soit la valeur de l'ordonnée à l'origine de la droite, soit la valeur spéciale "Not A Number" si la droite est (presque) verticale ;
- **c)** une méthode publique (d'instance) "getter" nommée **getPente** qui retourne (sans aucune vérification) la valeur du champ privé **pente** ;
- **d)** une méthode publique (d'instance) "setter" nommée **setPente** qui permet la modification de la valeur du champ privé **pente** ; cette méthode a un seul argument de type réel et doit respecter les consignes suivantes :
  - **i.** si la valeur de son argument est strictement plus grande que 115, on stocke dans le champ **pente** la valeur spéciale "Plus Infini" et dans le champ **ordOrig** la valeur spéciale " Not A Number " ;
  - **ii.** autrement, si la valeur de son argument est strictement plus petite que -115, on stocke dans le champ **pente** la valeur spéciale "Moins Infini" et dans le champ **ordOrig** la valeur spéciale " Not A Number " ;
  - **iii.** autrement, on copie dans le champ **pente** la valeur de l'argument de la méthode ;
- **e)** un constructeur **public** (surchargé) avec deux arguments de type réel et qui doit respecter les consignes suivantes :
  - **i.** la valeur du deuxième argument est stockée dans le champ **ordOrig** ;
  - **ii.** la valeur du premier argument "est stockée" dans le champ **pente** par un appel à la méthode **setPente** ;
- **f)** un constructeur **public** (surchargé) sans argument et qui doit respecter les consignes suivantes :

i. la valeur réelle 1 est stockée dans le champ **pente** ;

ii. la valeur réelle 0 est stockée dans le champ **ordOrig** ;

g) une méthode **publique d'instance** nommée **calculerY** qui doit calculer l'ordonnée d'un point dont l'abscisse est précisée par l'argument de la méthode et qui appartient à la droite représentée par l'objet appelant la méthode; par conséquent, cette méthode :

i. a un seul argument de type réel ;

ii. retourne un résultat de type réel, à savoir :

- si le champ **pente** de l'objet appelant a la valeur spéciale "Plus Infini" ou "Moins Infini", le résultat retourné est la valeur spéciale "Not A Number" ;
- autrement, le résultat retourné est la somme entre la valeur du champ **ordOrig** (de l'objet appelant) et le produit entre la valeur du champ **pente** (de l'objet appelant) et la valeur de l'argument de la méthode ;

h) une méthode **publique d'instance** nommée **sontPerpendiculaires** qui vérifie si la droite correspondant à l'objet appelant la méthode et la droite correspondant à l'objet argument de la méthode sont perpendiculaires ou non ; par conséquent, cette méthode :

i. a un seul argument de type **Droite** ;

ii. retourne un résultat de type logique, à savoir :

- si le champ **pente** de l'objet appelant a la valeur spéciale "Plus Infini" ou "Moins Infini" et le champ **pente** de l'objet argument à la valeur 0, le résultat retourné correspond à VRAI ;
- si le champ **pente** de l'objet appelant a la valeur 0 et le champ **pente** de l'objet argument a la valeur spéciale "Plus Infini" ou "Moins Infini", le résultat retourné correspond à VRAI ;
- si le produit entre la valeur du champ **pente** de l'objet appelant et la valeur du champ **pente** de l'objet argument vaut -1, le résultat retourné correspond à VRAI ;
- dans tous les autres cas, le résultat retourné correspond à FAUX ;

i) une méthode **publique** et **statique** nommée **sontParalleles** qui vérifie si la droite correspondant au premier argument de la méthode et la droite correspondant au deuxième argument de la méthode sont parallèles ou non ; par conséquent, cette méthode :

i. a deux arguments de type **Droite** ;

ii. retourne un résultat de type logique, à savoir :

- si les champs **pente** des deux arguments ont la valeur spéciale "Plus Infini" ou "Moins Infini", le résultat retourné correspond à VRAI ;
- si les valeurs des champs **pente** des deux arguments sont égales, le résultat retourné correspond à VRAI ;
- dans tous les autres cas, le résultat retourné correspond à FAUX.

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

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

2.2 En fonction des indications données sous forme de commentaires, compléter le code de la classe principale **publique CP_Ctr2Exo3** qui appartient au package **cms_ctr2** et qui contient la méthode **main**. Cette classe utilise la classe **Droite** disponible dans le même package **cms_ctr2**.

```
//déclarez le package
.....
.....
.....

//précisez l'en-tête de la classe principale
.....
.....
.....
{
     //précisez l'en-tête de la méthode main
.....
.....
.....
     {
          //créez un nouvel objet de type Droite qui correspond à
          //la première bissectrice (la droite de pente 1 passant
          //par l'origine) et stockez son adresse dans une variable
          //locale déclarée à cet effet et nommée d1
.....
.....
.....
          //créez un nouvel objet de type Droite qui correspond à
          //la deuxième bissectrice (la droite de pente -1 passant
          //par l'origine) et stockez son adresse dans une variable
          //locale déclarée à cet effet et nommée d2
.....
.....
.....
          //déclarez une variable locale x de type réel et dont la
          //valeur initiale vaut 2.5
.....
.....
.....
```

```
            //déclarez une variable locale y de type réel et dont la
            //valeur initiale est l'ordonnée du point d'abscisse x
            //qui appartient à la première bissectrice
            //(par un appel à la méthode calculerY)
……
……
……
            //déclarez deux variables locales de type logique
            //appelées bool_1 et bool_2
……
……
……
            //appelez la méthode sontPerpendiculaires afin de savoir
            //si les deux droites correspondant aux objets créés
            //ci-dessus (c'est-à-dire la première et la deuxième
            //bissectrice) sont perpendiculaires et stockez le
            //résultat retourné par cette méthode dans la variable
            //bool_1
……
……
……
            //appelez la méthode sontParalleles afin de savoir
            //si les deux droites correspondant aux objets créés
            //ci-dessus (c'est-à-dire la première et la deuxième
            //bissectrice) sont parallèles et stockez le résultat
            //retourné par cette méthode dans la variable bool_2
……
……
……
      }//fin de la méthode main
}//fin de la classe principale
```