EPF – Lausanne *COURS DE MATHEMATIQUES SPECIALES*

11 janvier 2012

Contrôle d'informatique no 2

Durée : 1 heure 45'

Nom : …… Groupe : ☐

Prénom : ……

No	1	2
Total points	85 points (55 + 30)	75 points (43 + 32)

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.

Le but de cet exercice est de réaliser une application autonome interactive qui aide l'utilisateur à traiter des problèmes de mécanique faisant intervenir des plans inclinés. 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 **PlanIncline** et, respectivement, **CP_Ctr2Exo1**. Par la suite, aux points **1.1** et **1.2**, on vous demande d'écrire le code complet de ces deux classes, en respectant les consignes précisées. Vous pouvez éventuellement répondre au point **1.2** en supposant le point **1.1** résolu correctement.

Indications :

- afin de calculer la tangente d'un angle mesuré en radians, vous pouvez appeler la méthode publique et statique ***tan*** de la classe prédéfinie ***Math*** qui fait partie du package ***java.lang*** ; plus précisément, la méthode ***tan*** retourne un résultat (de type numérique réel) qui représente la tangente de l'angle mesuré en radians qui est passé comme l'argument (de type numérique réel) de cette méthode ;
- afin de transformer la mesure d'un angle de degrés en radians, vous pouvez appeler la méthode publique et statique ***toRadians*** de la même classe prédéfinie ***Math*** ; plus précisément, la méthode ***toRadians*** retourne un résultat (de type numérique réel) qui représente la mesure en radians de l'angle mesuré en degrés qui est passé comme l'argument (de type numérique réel) de cette méthode.

1.1 Une instance de la classe **PlanIncline** est un objet de type **PlanIncline** qui regroupe (ou encapsule) des informations concernant un plan incliné d'angle **alpha** et sur lequel un point matériel est posé et étudié sous l'effet d'un champ gravitationnel d'accélération **g** et en présence d'un frottement statique de coefficient **mu**. De plus, la classe **PlanIncline** est munie des méthodes permettant de manipuler les objets de type **PlanIncline**.
Écrire le code complet de la classe **publique PlanIncline** qui appartient au package **cms_ctr2** et qui définit :

a) un champ (d'instance) nommé **alpha** de type réel, déclaré **privé** et sans valeur initiale explicite et qui sert à stocker la valeur de l'angle du plan incliné exprimé en degrés ;

b) un champ (d'instance) nommé **mu** de type réel, déclaré **privé** et sans valeur initiale explicite et qui sert à stocker la valeur (maximale) du coefficient de frottement statique (qui est adimensionnel) ;

c) un champ nommé **g** de type réel, déclaré sans modificateur d'accès et avec la valeur initiale ***9.81*** et qui sert à stocker la valeur de l'accélération gravitationnelle ; de plus, la valeur initiale de ce champ doit rester figée (non modifiable) par la suite et elle doit être commune à toutes les instances de la classe **PlanIncline** ;

d) une méthode publique (d'instance) "**getter**" nommée **convenablement** et qui retourne (sans aucune vérification) la valeur du champ privé **alpha** ;

e) une méthode publique (d'instance) "**setter**" nommée **convenablement** et qui permet la modification de la valeur du champ privé **alpha** ; cette méthode a un seul argument de type réel et doit respecter les consignes suivantes :

- **i.** si la valeur de son argument appartient à l'intervalle ouvert **]0; 90[**, on copie dans le champ **alpha** la valeur de cet argument ;
- **ii.** autrement, on stocke dans le champ **alpha** la valeur **45** ;

f) une méthode publique (d'instance) "**getter**" nommée **convenablement** et qui retourne (sans aucune vérification) la valeur du champ privé **mu** ;

g) une méthode publique (d'instance) "**setter**" nommée **convenablement** et qui permet la modification de la valeur du champ privé **mu** ; cette méthode a un seul argument de type réel et doit respecter les consignes suivantes :

- **i.** si la valeur de son argument appartient à l'intervalle ouvert **]0; 5[**, on copie dans le champ **mu** la valeur de cet argument ;
- **ii.** autrement, on stocke dans le champ **mu** la valeur **2** ;

h) un constructeur **public** (surchargé) avec deux arguments de type réel et qui doit respecter les consignes suivantes :

- **i.** la valeur du premier argument "est stockée" dans le champ **alpha** par un appel à la méthode **"setter"** correspondante ;
- **ii.** la valeur du deuxième argument "est stockée" dans le champ **mu** par un appel à la méthode **"setter"** correspondante ;

i) un constructeur **public** (surchargé) sans argument et qui doit respecter les consignes suivantes :

- **i.** la valeur réelle **45** est stockée dans le champ **alpha** ;
- **ii.** la valeur réelle **2** est stockée dans le champ **mu** ;

j) une méthode **publique d'instance** nommée **estAuRepos** qui renseigne si un point matériel posé sur un plan incliné reste en repos ou descend sous l'effet du champ gravitationnel et en présence du frottement (statique) ; par conséquent, cette méthode :

- **i.** n'a pas d'argument ;
- **ii.** retourne un résultat de **type logique**, à savoir :
  - **vrai** si la tangente de l'angle **alpha** du plan incliné correspondant à l'objet appelant la méthode est plus petite ou égale au coefficient de frottement (statique) de ce même plan incliné ;

- **false** dans le cas contraire;

k) une méthode **publique** et **statique** nommée **comparer** qui compare les raideurs de deux plans inclinées ; par conséquent, cette méthode :

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

ii. retourne comme résultat un **entier**, à savoir :

- **-1** si l'angle **alpha** du plan incliné correspondant au premier argument est strictement plus petit que l'angle **alpha** correspondant au deuxième argument ;
- **1** si l'angle **alpha** du plan incliné correspondant au premier argument est strictement plus grand que l'angle **alpha** correspondant au deuxième argument;
- **0** dans les autres cas, c'est-à-dire si l'angle **alpha** du plan incliné correspondant au premier argument est égal à l'angle **alpha** du plan incliné correspondant au deuxième argument ;

l) une méthode publique d'instance nommée **afficher** qui renseigne sur un plan incliné ; plus précisément, cette méthode :

i. n'a pas d'argument ;

ii. ne retourne pas de résultat ;

iii. affiche, sur une ligne, les valeurs de l'angle **alpha** et du coefficient de frottement (statique) **mu** du plan incliné correspondant à l'objet appelant cette méthode (conformément à l'exemple d'affichage donné au point **1.2**).

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

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

1.2 En fonction des indications données sous forme de commentaires, compléter le code de la classe principale **publique CP_Ctr2Exo1** qui appartient au package **cms_ctr2** et qui contient la méthode **main**. Cette classe utilise la classe **PlanIncline** disponible dans le même package **cms_ctr2**. À l'exécution, la sortie du projet (contenant les deux classes évoquées ci-dessus) doit respecter la mise en page donnée dans l'exemple ci-dessous.

```
Bonjour !
Introduisez l'angle alpha (en degrés) :
60
Introduisez le coefficient mu :
1.5
Plan incliné : alpha = 60.0 degrés et mu = 1.5.
false
1
```

```
//déclarez le package
.....
.....
.....
//importez une classe nécessaire pour la lecture au clavier
.....
.....
.....
//précisez l'en-tête de la classe principale
.....
.....
.....
{
     //précisez l'en-tête de la méthode main
.....
.....
.....
     {
          //créez un objet vous permettant la lecture au clavier et
          //stockez son adresse dans une variable locale adéquate
.....
.....
.....
```

//affichez à l'écran (dans la fenêtre console), sur une
//ligne, le message ***Bonjour !***

……

……

……

//demandez à l'utilisateur d'introduire l'angle alpha
//(en degrés)

……

……

……

//récupérez la réponse introduite par l'utilisateur au
//clavier et stockez la dans une variable locale adéquate

……

……

……

//demandez à l'utilisateur d'introduire le coefficient
//de frottement mu

……

……

……

//récupérez la réponse introduite par l'utilisateur au
//clavier et stockez la dans une variable locale adéquate

……

……

……

//créez un nouvel objet de type **PlanIncline** qui correspond
//à un plan incliné ayant l'angle et le coefficient de
//frottement indiqués par l'utilisateur et stockez son
//adresse dans une variable locale déclarée à cet effet
//et nommée **p1**

……

……

……

//créez un nouvel objet de type **PlanIncline** par un appel
//au constructeur sans argument et stockez son adresse
//dans une variable locale déclarée à cet effet
//et nommée **p2**

……

……

……

```
            //affichez à l'écran l'angle alpha et le coefficient de
            //frottement mu du plan incliné correspondant aux
            //données introduites par l'utilisateur (par un appel
            //de la méthode adéquate de la classe PlanIncliné)
……
……
……
            //affichez à l'écran le résultat retourné suite à l'appel
            //de la méthode estAuRepos pour le plan incliné
            //correspondant à la variable locale p1
……
……
……
            //affichez à l'écran le résultat retourné suite à l'appel
            //de la méthode comparer pour les plans inclinés
            //correspondant aux variables locales p1 et p2
……
……
……


      }//fin de la méthode main
}//fin de la classe principale
```

Sujet no 2.

2.1 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).

```
package cms_ctr2;

public class CP_Ctr2Exo2_1
{
	public static void main(String[] args)
	{
		Bidon b1 = new Bidon();
		System.out.println("-----------------------");
		Bidon b2 = new Bidon(3.5, "d'essence");
		System.out.println("-----------------------");
		Bidon b3 = new Bidon(5, "d'huile");
		System.out.println("-----------------------");
		b1.afficher();

	System.out.println("****************************************");

		b1 = b2;
		b2 = b3;
		b3 = b1;
		b1.afficher();
		b2.afficher();
		b3.afficher();

	System.out.println("****************************************");

		Bidon b4 = new Bidon(5, "de diesel");
		Bidon b5 = new Bidon(-3, "de gazoline");
		b4.mixer(b5);
		b4.afficher();
		b5.afficher();

	System.out.println("****************************************");

		Bidon b6 = new Bidon(2, "de vin");
		Bidon b7 = new Bidon(4, "d'eau");
		Bidon b8 = Bidon.fusionner(b6, b7);
		b8.afficher();

	}//fin de la méthode main
}//fin de la classe principale
```

```
class Bidon
{
	double volume;
	String contenu;
	static int nb = 0;

	Bidon()
	{
		this(1.5, "de vin");
		if(nb < 4)
		{
			System.out.println("Un bidon \"vide\".");
		}
	}

	Bidon(double arg1, String arg2)
	{
		if(++nb < 4)
		{
			System.out.println("Un bidon plein !");
		}
		volume = arg1;
		contenu = arg2;
	}

	void mixer(Bidon bid)
	{
		bid.volume = bid.volume + this.volume;
		this.contenu = this.contenu + " et " + bid.contenu;
	}

	static Bidon fusionner(Bidon b1, Bidon b2)
	{
		return new Bidon(b1.volume + b2.volume, "de cocktail");
	}

	void afficher()
	{
		System.out.println("Un bidon de " + volume
				+ " litres plein " + contenu + " !");
	}
}//fin de la classe Bidon
```

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

2.2 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_2
{
	public static void main(String[] args)
	{
		String tab1[] = {"Je", "Ich", "Io", "I"};
		String tab2[] = {"dis:", "sage:", "dico:", "say:"};
		String tab3[] = {"\"Bonne chance!\"", "\"Viel Glück!\"",
				"\"Buona fortuna!\"", "\"Good luck!\""};

		for(int i=0; i<4; i++)
		{
			System.out.println("Le tour " + i+1 + " :");
			System.out.println(methode1(tab1, tab2, tab3, i));
			System.out.println("----------------------------");
		}
	}//fin de la méthode main

	static String methode1(String a[], String b[],
				String c[], int i)
	{
		i++;
		if(i > 3)
			i = 0;
		String s = methode2(b, c, i);
		return a[i] + "  " + s;
	}//fin de la methode1

	static String methode2(String b[], String c[], int j)
	{
		++j;
		if(j > 3)
			j = 1;
		String s = methode3(c, j);
		return b[j] + "  " + s;
	}//fin de la methode2

	static String methode3(String c[], int k)
	{
		k -= 2;
		if(k < 0)
			k = 2;
		return c[k];
	}//fin de la methode3

}//fin de la classe principale
```

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....