Master M1
Programmation Scientifique Orientée Objet
Partiel (1h30)

Exercice 1

Vous devez répondre à chacune des questions ci-dessous.

1) Qu'indique le mot clé "final" ?

que le composant est constant.

2) Qu'est ce qu'une classe ?

Une classe définit le type et le comportement communs à des instances. Elle est composé d'attributs définissant un état, et de méthodes définissant les comportements possibles.

3) Que signifie l'instanciation ?

Il s'agit du mécanisme de création d'un objet.

4) Quelle "méthode" particulière est appelée lorsqu'on instancie un objet?

le constructeur.

5) Quel mot clé permet d'appeler cette "méthode"?

new

6) Quelle méthode est appelée à la destruction d'un objet?

void finalize()

7) Quel mot clé désigne l'objet courant?

this

8) Que signifie l'utilisation du mot clé "static" lorsqu'on définit une méthode?

Cette méthode n'est pas rattaché à un objet en particulier, mais elle est commune à toutes les instances.

9) Quelle est la différence entre "double" et "Double" ?

"double" est un type primitif alors que "Double" est un objet.

10) Soit :

```
1 int a, b;
2 a = 10;
3 b = a;
4 a = a + 1;
```

Que vaut b ?

b = 10

11) Soit :

```
Point a, b;
a = new Point(10, 50);
b = a;
System.out.println("x = " + a.x + "; y = " + a.y + ";");
a.x = 11;
System.out.println("x = " + b.x + "; y = " + b.y);
```

Que va afficher ce code ?

x = 10; y = 50;

x = 11; y = 50;

12) Que signifie cette notation :

```
int[] entiers;
```

On déclare une variable "entiers" qui est un tableau d'entier.

13) Expliquez à quoi servent les trois blocs "try", "catch", et "finally" qui sont utilisés pour gérer des exceptions?

Le bloc "try" correspond au code susceptible de générer une exception.

Le bloc "catch" permet de définir le traitement qui doit être effectuer si l'exception est effectivement lever.

Le bloc "finally" permet de définir un traitement qui sera exécuter avec ou sans déclenchement de l'exception.

14) Que permet l'héritage? Et comment utilise-t-on ce mécanisme?

Il permet de définir une classe fille à partir d'une classe mère. La classe fille sera composée des mêmes attributs et méthodes que la classe mère. On peut rajouter des attributs et méthodes à la classe fille. On utilise le mot clé "extends" pour hériter. On ne peut hériter que d'une seule classe mère à la fois.

15) Que se passe-t-il si on définit une classe comme "final"? Qu'en est-il d'une méthode "final"?

Lorsqu'une classe est définit "final", on ne peut hériter de cette classe. Lorsqu'une méthode est définit "final", on ne peut la redéfinir.

16) Qu'est ce qu'une classe abstraite?

C'est une classe qui possède au moins une méthode abstraite. On dit qu'une méthode est abstraite lorsqu'on ne la définit pas. Cette classe ne peut être instanciée, elle ne sert donc que dans le cadre de l'héritage.

17) Quelles sont les différences entre une classe abstraite et une interface?

Une interface ne possède que des méthodes abstraites; elle ne peut avoir que des attributs constants; ses méthodes sont obligatoirement publiques. De plus, on n'hérite pas d'une interface, mais on l'implémente (en utilisant le mot clé "implement"). On peut implémenter plusieurs interfaces à la fois.

18) Soit une classe "ClasseA" qui possède la méthode "uneMethode()", et une classe "ClasseB" qui hérite de "ClasseA". "uneMethode()" a été redéfinie dans "ClasseB".

```
ClasseA a = new ClasseB();
a.uneMethode();
```

A-t-on le droit d'écrire ce code? Et si oui, quelle méthode sera appelée?

Oui on a le droit, c'est le principe du polymorphisme. Les méthodes dans "ClasseA" et "ClasseB" n'ont pas les mêmes signatures. La méthode appelé dépend du type réel de l'objet. En l'occurrence, ici la variable "a" est de type "ClasseB". C'est donc la méthode redéfinie dans "ClasseB" qui est appelée.

19) Quelles sont les trois grandes étapes dans la gestion d'un flux d'entrée ou de sortie?

Étape 1 : ouverture du flux

Étape 2 : lecture ou écriture du flux

Étape 3 : fermeture du flux

20) Que désigne "System.in" ?

Il s'agit du flux d'entrée standard, en l'occurrence le clavier.

Exercice 2

Dans cet exercice vous allez devoir développer une classe Vecteur contenant des réels. Vous devrez notamment développer les méthodes suivantes :

- public static Vecteur litVecteur() : cette première méthode demande à un utilisateur de saisir la taille du vecteur qu'il souhaite créer, ainsi que la valeur de chacun de ses éléments.
- public Vecteur getOpposite() : cette méthode crée un nouveau vecteur correspondant au vecteur opposé du vecteur courant tel que : si $\vec{v}$ est le vecteur opposé de $\vec{u}$ alors : $\vec{u} + \vec{v} = \vec{0}$
- public Vecteur subtraction(Vecteur v) : cette dernière méthode calcul la soustraction entre deux vecteurs tel que : $\vec{u} - \vec{v} = \vec{u} + (-\vec{v})$ (où $(-\vec{v})$ est le vecteur opposé de $\vec{v}$)

Vous pourrez développer toute méthode ou tout constructeur dont vous pourriez avoir besoin.

Une attention particulière devra être portée sur l'indentation de votre code que vous devrez également commenter.

Vous pourrez vous aider d'une méthode "main" afin de tester votre code.

Enfin, vous enverrez par mail (à l'adresse thibaut.demare@univ-lehavre.fr) une archive ZIP du package contenant votre classe Vecteur. Le nom du package devra être de la forme "nom.prenom". Le sujet du mail devra comporter l'intitulé "[PartielPSOO]".

```java
package psoo.math;

import java.util.Scanner;

public class Vecteur2 {
  double[] elmts;

  public Vecteur2(int dim){
    elmts = new double[dim];
  }

  public static Vecteur2 litVecteur(){
    Vecteur2 vec;
    System.out.print("Taille du vecteur : ");
    Scanner entree = new Scanner(System.in);
    vec = new Vecteur2(entree.nextInt());
    for (int i = 0; i < vec.elmts.length; i++) {
      System.out.print("Valeur " + i + " : ");
      vec.elmts[i] = entree.nextDouble();
```

```
    }
    entree.close();
    return vec;
  }

  public Vecteur2 getOpposite(){
    Vecteur2 vec = new Vecteur2(elmts.length);
    for(int i = 0; i<elmts.length; i++){
      vec.elmts[i] = elmts[i] * -1;
    }
    return vec;
  }

  public Vecteur2 subtraction(Vecteur2 vec){
    if(vec.elmts.length == elmts.length){
      Vecteur2 opp = vec.getOpposite();
      Vecteur2 res = new Vecteur2(elmts.length);
      for(int i = 0; i < elmts.length; i++){
        res.elmts[i] = elmts[i] + opp.elmts[i];
      }
      return res;
    }
    else
      return null;
  }

  public String toString(){
    String s = "(";
    for(int i = 0; i<elmts.length; i++){
      s += elmts[i];
      if(i != elmts.length - 1)
        s += "; ";
    }
    return s + ")";
  }

  public static void main(String[] args) {
    Vecteur2 v1 = Vecteur2.litVecteur();
    System.out.println("v1 = "+v1);
    Vecteur2 v2 = v1.getOpposite();
    System.out.println("v2 = "+v2);
    Vecteur2 v3 = v1.subtraction(v2);
    System.out.println("v3 = "+v3);
  }
}
```