Découvrir l'environnement JAVA Introduction	Diagramme de classes Classe & objet Encapsulation	Héritage Classes abstraites Interfaces	Collections Gestion des exceptions

Exercice 1 : Environnement, de C à java

1. Quel sont les rôles des éléments suivants :

   JVM –JRE – JDK – javac – class - main
   Quelle est relation entre ces éléments
2. Quel est le code source minimal pour :
   a. Définir une classe
   b. Définir une classe exécutable
3. Ecrire un programme JAVA qui permet d'initialiser et d'afficher un tableau de chaine de caractère
4. Ecrire un programme JAVA qui permet d'établir une liste chainée de produits. Un produit est défini par un code à barre, une désignation, un prix d'achat et un prix de vente.
   Ajouter une fonction qui permet d'afficher le contenu de ce tableau.

Exercice 2 : classe, objet, constructeur, attributs, accès

1. Définir les concepts package, class, constructeur, objet, attribut, état d'un objet
   Soit le code source des deux classes suivantes :

Program.java

```
1  package app.java.ateliers.step2;
2  import java.util.Scanner;
3  class Program {
4      public static void main(String[] args) {
5          int a;
6          Scanner sc = new Scanner(System.in);
7          a = sc.nextInt();
8          System.out.println("a=" + a);
9
10 }
11 }
```

Point.java

```
1 package app.java.ateliers.step2;
2 class Point {
3 int abs;
4 int ord;
5 }
6
```

2. Quel est le rôle de la ligne 1 et le rôle de la ligne 2 du fichier Program.java
3. Quel est le rôle de la méthode main de la classe Program
4. Interpréter le code de la classe Program et donner le résultat de son exécution
5. Pourquoi la classe Point ne peut pas être exécutable ? à quoi va servir cette classe ?
6. A partir de ce qui précède, quel est le rôle d'une classe JAVA ?
7. Ajouter le code suivant au code source de la classe Program après la ligne 8

Point.java | Point.java | Program.java

```
1  package app.java.ateliers.step2;
2  import java.util.Scanner;
3  public class Program {
4      public static void main(String[] args) {
5          int a;
6          Scanner sc = new Scanner(System.in);
7          a = sc.nextInt();
8          System.out.println("a=" + a);
9
10         Point p1;
11         p1=new Point();
12         p1.abs=1;
13         p1.ord=1;
14         System.out.println("("+p1.abs+","+p1.ord+")");
15
16     }
17 }
```

8. Interpréter le code source de la méthode main à partir de la ligne 10 et donner son résultat
9. Modifier les instructions des lignes 12 et 13 pour lire au clavier les coordonnées du point p1.

Exercice 3 : gestion d'une boutique

L'objectif de l'application est la gestion des utilisateurs.
Un utilisateur est défini par un nom, un prenom, un login et un mot de passe.

1. Définir le diagramme de classes
2. Ecrire le code JAVA de la classe User
3. Ecrire une classe pour tester l'application
    a. Créer un utilisateur, le stocker dans la mémoire
    b. Demander à l'utilisateur, à l'aide de la classe Scanner, de saisir son login et son mot de passe
    c. Si les informations correspondent à celles mémorisées, afficher bienvenue avec un menu de gestion
    d. Si non, afficher un message d'erreur.

Exercice 4

L'objectif de l'application à réaliser est de déplacer un cercle ayant une couleur dans un panneau.
Dans un premier temps, on va essayer juste d'afficher les différentes positions de son centre dans une application console.
Le cercle est défini par un Point(x,y) qui représente son centre, un rayon et une couleur.
Le panneau est défini par deux points p1(x,y) et p2(x,y) comme montré dans la figure (figure3).

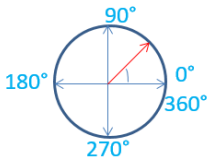

Figure 1

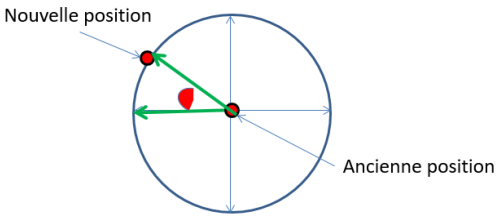

Figure 2

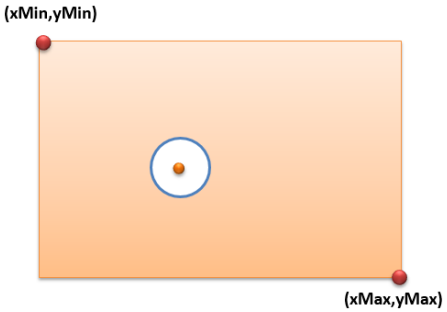

Figure 3

La position prochaine est un nouveau Point selon une direction. La position prochaine ne doit pas déborder du panneau (Figure 3).
La direction peut être représentée par un angle par rapport à l'axe horizontal variant de 0° à 360° (Figure1)
Le cercle ne peut dévier qu'avec un angle appartenant à l'intervale [-45°, +45°]

Exemple

Au début, On suppose que le centre dans son ancienne position a l'angle 180° (voir figure 2)
Calculer le nouveau centre du cercle centreNew(x2,y2) en fonction de la direction choisie.
x2 et y2 sont calculés à l'aide des deux formules suivantes :
x2 = xAncien + r * *cos(Math.PI * angleChoisie / 180);*
y2 = yAncien + r * *sin(Math.PI * angleChoisie / 180);*
AngleChoisie est trouvée aléatoirement à l'aide de l'instruction suivante :
AngleChoisie =-45.0 + Math.random() * 90 ;
// Math.random() permet de retourner une valeur entre 0 et 1
Vérifier si centreNew ne dépasse pas les limites ; x2 est entre xMin et xMax, et y2 est entre yMin et yMax (Figure3)
Une fois la nouvelle position calculée, faire le déplacement.
À chaque déplacement, mémoriser le nouvel angle pour connaitre sa direction

1. Etablir le diagramme de cas d'utilisation
2. Etablir le diagramme de classes relatif à ce problème
3. Etablir le diagramme de séquences relatif à ce problème
4. Ecrire le corps de la méthode ***deplacer*** pour déplacer un cercle.
   Comment déplacer un cercle ? le problème se résume au déplacement de son centre.(voir l'exemple).
   A son tour, la méthode depacer, peut faire appel à une autre méthode **trouverNouvellePositionValide**
5. Ajouter une autre fonction qui vérifie si le cercle a entré dans une zone bien définie. On peut supposer par exemple que cette zone est définie par un rectangle.
   Compléter le diagramme de classes
   Implémenter la méthode ***verifier***
   Une fois le cercle entre dans la zone en question, afficher JEU TERMINE
6. Ecrit une classe Program qui teste l'application
   Compléter le diagramme de classes
   Implémenter la classe Program