TD1 et TD2

TD1

Exercice 1 :

1. What happens when you type sysout and press Ctrl + space in a main method ?

Le code se complète avec :

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

2. Same question with toStr then Ctrl + space inside a class ?

Le code se complète avec :

```
@Override
public String toString() {
  // TODO Auto-generated method stub
  return super.toString();
  }
```

3. Same question with main then Ctrl + space inside a class ?
   le code se complète avec :

```
public static void main(String[] args) {}
```

4. Create a new int field called foo inside the class. What happens if you type :
   Ctrl + space inside the class, what if you now type set then press Ctrl + space

5. Select the class name. Que What happens if you type Alt+Shift+R ? Same question with the int field foo ?

La classe est renommer dans tous les endroits ou elle apparait.
La méthode est renommer dans tous les endroit ou elle apparait.

Exercice 2 :

1. Create a new class Point with two private fields x and y. Why does it work ?

Les attribut sont priver mais on peut les utiliser dans pour écrire une méthode de la classe.

2. Create a class TestPoint with a main and the same code as before. What happens ? How can we fix it ?

Les attributs X et Y ne sont pas visible.
Pour le code marche on peut rendre les attributs visibles de la manière suivante :

```
public int x;
public int y;
```

Cela est une solution qui n’est pas bonne, les attributs doivent être privés pour un code de qualité. Le mieux est d’ajouter une méthode pour avoir accès aux valeurs de x et y.

3. Why do you need to set all fields visibility to private ?

Pour qu'il ne soit pas modifier par inadvertance.

4. What is an accessor(getter) ? Do we have to do it here ?

Permet de récupérer la valeur d'un attribut. Pour réaliser le code suivant : System.out.println(p.x+" "+p.y). Nous avons besoin d’ajouter un accessor.

5. Create a constructor with two arguments (called px and py). What is the problem ?

6. Modify the parameters of the constructor to call them x and y. What happens ?

7. We would like to keep track of the number of Points that have been created so far. How to proceed ?

On peut ajouter un compteur dans le constructeur de la manière suivante :

```
public class Point {
    private int x;
    private int y;
    private int cpt= 0;
public Point(int x, int y) {
    this.x=x;
    this.y=y;
    this.cpt +=1;
  }
}
```

8. Write a second contructor with a single Point p2 argument that copies the coordinates from p2 into the current Point. How does the compiler know whichconstructor to call ?

```
public Point(Point p2) {
    this.x = p2.x;
    this.y = p2.y;
    this.cpt +=1;
  }
```

Le constructeur sait quel constructeur appeler parce que la signature des deux constructeurs n'est pas la même. (Ils n'ont pas le même nombre d'attribut)

9. Update the class so that a call to System.out.println(point); will print the point coordinates as follows : (x, y).

```
public static void main(String[] args) {
    Point p=new Point(3,4);
    Point p3 = new Point(p);
    System.out.println("("+p3.x+","+p3.y+")");
  }
```

Exercice 3. Equality

1. What does this code print ? Why ?

```
true
false
```

L'opérateur "==" compare les réferences d'un object. C'est pour ça que "p1==p3" affiche "false".

2. Write a method isSameAs(Point) that will return true if the two points have the same coordinates.

```
public boolean isSameAs(Point p4) {
    if (p4.x == this.x && p4.y==this.y)
      return true;
    return false;
}
```

3. What is the problem with this code ? Read the documentation of indexOf and check which method is called. Modify the Point class to fix this problem

L'affichage est incohérent, les points p3 et p2 n'ont pas été ajouté dans la list. Deplus "indexOf" renvoi la place d'un caractère dans une string.

Exercice 4. Polyline

1. You will use an array to store the Points of the polyline. Write the constructor for PolyLine.

```
package fr.dauphine.javaavance.td1;
import java.util.ArrayList;
import java.util.List;

public class PolyLine {
  private static int MAXCAPACITY;
  private ArrayList<Point> pointList;

  public PolyLine(int mxcapt){
    this.MAXCAPACITY =mxcapt;
    pointList = new ArrayList<Point>();
    pointList.ensureCapacity(MAXCAPACITY);
  }
```

2. Write a method add that can be used to add a new point to the line. What happens if we add more points that the maximum capacity of the array ? What to do about it ?

```
public void add(Point p) {
    pointList.add(p);
  }
```

Ther will be an error. We can it make impossible to add an point past the max capacity of the array.

3. Write a method pointCapacity() and nbPoints() that will return the maximum capacity of the polyline and the number points currently in the polyline.

```
public int pointCapacity() {
    return this.MAXCAPACITY;
  }

public int nbPoints() {
    return pointList.size();
  }
```

4. Write a method contains which will return true if a given point is in the polyline. Use a for each loop to do this (instead of a classical index based loop).

```
public boolean contains (Point p1) {
        for(Point p :pointList){
            if (p == p)
                return true;
        }
        return false;
    }
```

5. What happens if null is given instead of an actual Point object ? What if you do add(null) before ? Read about Objects.requireNonNull(o).

L’object null est ajouté dans la liste il est compté comme un point.

6. Update the class and use a LinkedList instead of an array (and remove the maximum capacity limit). How to update pointCapacity, nbPoints and contains ?

Exercice 5. Mutability and circle

1. Add a method translate(dx, dy) in Point. What are the different options to write this method ?

```
public void translate(int dx, int dy) {
    this.x= dx;
    this.y=dy;
  }
```

2. A circle can be represented with a center and a radius. Write a new Circle class. Don’t forget the constructor.

```
package fr.dauphine.javaavance.td1;

public class Cercle {
  private int raduis;
  private Point p;

  public Cercle( Point p1,int r) {
    this.raduis = r;
    this.p = p1;
  }
```

3. Write the toString method.

```
public String toString() {
        return "Cercle{" +
                "raduis=" + raduis +
                ", point=" + p +
                '}';
    }
```

4. Write the translate(dx, dy) that translate the circle.

```
public void translate(int r2, Point p2) {
    this.raduis = r2;
    this.p = p2;
  }
```

translate(dx, dy) aggrandit et déplace et le cerlcle on veut juste le déplacer.

5. What is the problem with this code ? How to avoid it ?

La méthode translate est appelée pour un object de type cerle alors que avec cette signature elle devrait etre appelé pour un object de type Point. On peut changer les noms des méthodes.

6. What would be the problem with a getCenter() method that would return the center ? To find out, consider the following code ? Modifier pour que cela soit correct.

```
```

7. Add area() and update toString() to print the area as well.

```
public void  area() {
    double area = Math.PI*(this.raduis*this.raduis);
  }

public String toString() {
        return "Cercle{" +"raduis=" + raduis + ", point=" + p.getX() +","+p.getY() +", area=" + area+ "}";
    }
```

8. Add a contains(Point p) method to return true if p is inside the circle (hint :use Pythagoras theorem).

9. Add contains(Point p, Circle...circles) that will return true of the point is inside one of the circles ? What other change should you do about the method declaration ? Why ?

Exercice 6. Anneaux

1. Should you use inheritance ?
2. Write a new class Ring, with a center and two radius (beware, the inner radius must always be smaller than the outer one.
3. Write equals.
4. On veut afficher un anneau avec son centre, son rayon et son rayon interne. Quel est le probl`eme si on fait System.out.println(ring); sans code supplémentaire ? Le corriger.
5. Write contains(Point), avoid useless object construction.
6. Write contains(Point p, Ring...rings)

TD2

Exercice 1. Redefinition

1. Qu’affiche le main et pourquoi ?

```
42
42
24
```

```
24
24
24
```

Les deux premiers “42” s’affiche suite au fonction de meth et printMeth d’un object Mere.

Les deux premiers “24” s’affiche suite au fonction de meth et printMeth d’un object Fille.

Les deux second “24” s’affiche suite au fonction de meth et printMeth d’un object Mere instancié tel un object Fille.

2. S’il est dans Fille, à combien de méthodes meth() un objet de type Fille à accès (et comment il y accède) ? Et s’il est dans Main ?

Un object Fille aura accès à deux méthode dans Fille et dans Main

Un object Mere aura accès à une seule méthodes dans Fille et dans Main

3. Quel est le comportement si les méthodes meth() sont statiques ?

l’affichage devient le suivant :

```
42
42
24
42
42
42
```

Avec static la fonction printMeth() devient propre à la classe.

c’est pour ça que fille.printMeth(); affiche 42.

Dans System.out.println(mereFille.meth()); mereFille est de type Fille est peut donc accèder au deux méthodes meth() ici elle accède à la methode meth() de mère.

4. Et si meth sont maintenant des champs ? Pourquoi ?

Exercice 2. Redéfinition - surcharges

1. Quelles sont les erreurs de compilation et pourquoi ?

```
The method miage() is undefined for the type Mere
// la methode n'existe pas pour le type mere
Cannot reduce the visibility of the inherited method from Mere
// la visibilité de la méthode hérité est plus large dans la superClasse
The return type is incompatible with Mere.h()
// le type n'est pas le même dans la superClasse char VS int
The return type is incompatible with Mere.i()
// le type n'est pas le même dans la superClasse int VS void
Exception Exception is not compatible with throws clause in Mere.k()
// l'excetion dans la superClasse n'est pas la même Exception VS IOException
```

2. Rappeler ce qu’est une redéfinition et une surcharge, et indiquer où sont les surcharges et où sont les redéfinitions ici.

S**urcharge :** elle a lieu au sein d’une même classe, elle constiste à récricre le contenu d’une méthode en changeant les paramètres.

Redéfinition : elle a lieu lors de l’héritage elle constiste à récricre le contenu d’une méthode en gardant la meme signature.

```
//dans le classe Mere
void c() {System.out.println("Mere_c");}
void c(Mere mere) {System.out.println("Mere_c(Mere)"); } // surcharge

//dans la classe Fille
class Fille extends Mere{
  void miage() {System.out.println("Miage");}

  public void a() {System.out.println("Fille_a"); } //redefinition

  protected void b(Fille fille) {System.out.println("Fille_b(Fille)");} //redefinition

  public void c(Mere mere) {System.out.println("Fille_c(Mere)");} //redefinition
  void c(Fille b) {System.out.println("Fille_c(Fille)"); } //surcharge
  static void d() {System.out.println("static Fille_d");} //redefinition
  static void d(Mere mere) {System.out.println("Fille_d(Mere)");}//surcharge


  protected void f() {System.out.println("Fille_f");} //redefinition
  String g() {System.out.println("Fille_g"); return "c";} //surcharge


  void j() throws IOException {System.out.println("Fille_j"); } //redefinition

  void l() {System.out.println("Fille_l");} //redefinition
  void m() throws IOException, IllegalArgumentException {System.out.println("Fille_m"); }//redefinition
}
```

1. Expliquer chaque affichage

Exercice 3. Expressions arithmétiques

1. Ecrire les types Expr, Value, Add, les méthodes eval et une classe Main avec un Main de test dans un même package.
2. Implémenter l’affichage d’une expression arithmétique non ´évaluée.
3. Ajouter l’opération de multiplication.
4. Ajouter l’opération de racine carr´ee. Pour l’affichage, vous utiliserez le symbole unicode \u221a. Pour l’évaluation, vous utiliserez la m´ethode Math.sqrt().