Aix*Marseille université
Socialement engagée

Année 2021-22

Domaine Sciences et Technologies
Portail René Descartes

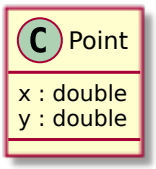
Programmation 1 : TP 2
Code UE : SPO2U07L

TP 2 : dessins

1 Classes `Point` et `Circle`

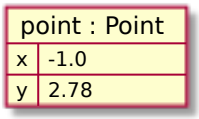
Tâche 1 : Créer un nouveau projet "TP2" sur repl.it. Ajouter des classes `Point` et `Circle`.

Tâche 2 : Ajouter les déclarations des attributs de la classe `Point` pour qu'elle corresponde au diagramme suivant :



Tâche 3 : Dans la méthode `main` de la classe `Main`, écrire des instructions pour réaliser les opérations suivantes :

- *Déclarer* une variable nommée `point` qui contiendra une référence vers un `Point`.
- *Initialiser* cette variable avec la référence d'un nouvel objet (appeler le constructeur par défaut sans arguments).
- *Affecter* aux attributs du nouvel objet des valeurs numériques de sorte qu'il corresponde au diagramme suivant :



- *Afficher* les valeurs des attributs de l'objet dans la console, sous le format `(abscisse, ordonnée)`.

Vérifier que votre programme fonctionne correctement avant de passer à la suite.

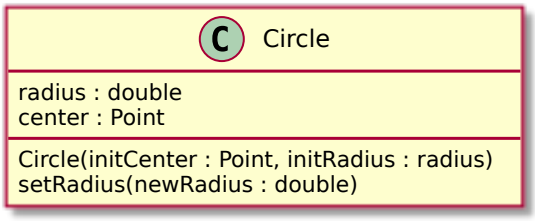
Tâche 4 : Ajouter le constructeur suivant à la classe `Point`.

```
// à ajouter dans la classe Point
public Point(double initX, double initY) {
  this.x = initX;
  this.y = initY;
}
```

Tâche 5 : Modifier les instructions déclarant et modifiant la variable `point`, afin de n'avoir que deux instructions : la création/initialisation de la variable `point`, et l'affichage des valeurs de ses attributs.

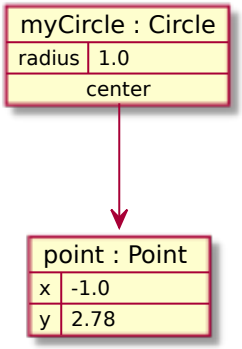
Utiliser le constructeur récemment ajouté pour cela.

Tâche 6 : Compléter la classe `Circle` pour qu'elle corresponde au diagramme suivant :



Tâche 7 : Ajouter des instructions pour créer une variable `myCircle` contenant la référence d'un cercle, l'initialiser, puis afficher ses propriétés.

La variable `myCircle` doit être égale à l'objet décrit dans le diagramme d'objet suivant :



1.1 Périmètre

Tâche 8 : Ajouter une méthode `double perimeter()` dans la classe `Circle` qui retourne la longueur du périmètre du cercle (la propriété `Math.PI` donne la valeur de π).

Tâche 9 : Ajouter des instructions pour afficher le périmètre de `myCircle`.

Tâche 10 : Avant de passer à l'exercice suivant, et pour éviter que la méthode main ne deviennent trop compliquée, créer dans la classe Main une méthode `exo123` sans paramètre.

Y déplacer toutes les instructions actuellement dans la méthode main. Ajouter dans main l'unique instruction exo123() ; et vérifier que le comportement du programme n'a pas changé. Finalement mettre en commentaire cette dernière instruction.

1.2 Carré

On considère la classe `Square` définie par le code suivant :

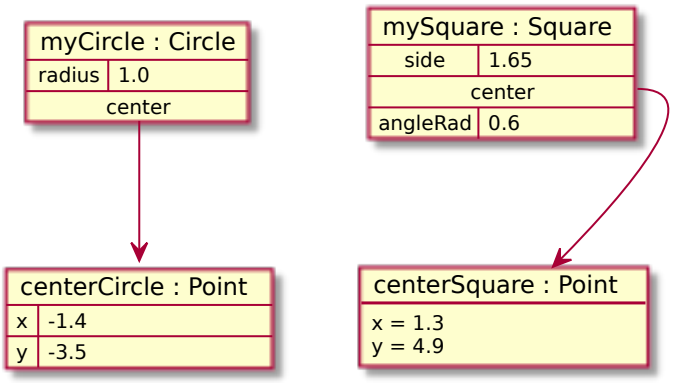
```
public class Square {
  double side;
  Point center;
  double angleRad; // angle of rotation around its center

  Square(Point initCenter, double initSide, double initAngleRad) {
    this.center = initCenter;
    this.angleRad = initAngleRad;
    this.side = initSide;
  }

  void rotateClockwise(double radians) {
    this.angleRad = this.angleRad + radians;
  }
}
```

On se place dans la situation où aucun objet n'a encore été créé.

Tâche 11 : Écrire dans une méthode `public static void exo4()` les instructions permettant d'obtenir le diagramme suivant. Faire un appel à la méthode `exo4()` depuis la méthode `main` (de la même façon que cela a été fait pour `exo123`).



Tâche 11 : Ajouter dans la méthode `exo4` des instructions d'affichage, pour vérifier que les objets ont bien été initialisés.

2 Définir une nouvelle classe : `LineSegment`

Nous voulons définir une nouvelle classe `LineSegment` permettant de représenter les segments du plan :

Tâche 12 : Choisir les attributs définissant un segment.

Tâche 13 : Déterminer le type de chacun des attributs.

Tâche 14 : Créer la classe en `LineSegment` qui devra contenir un constructeurs dont vous devrez déterminer le(s) type(s) des paramètre(s).

Tâche 15 : Ajouter une méthode retournant la longueur du segment.

Rechercher les fonctions mathématiques nécessaires dans la documentation de la classe java.lang.Math, disponible sur le

3 Produire un dessin en `SVG` (bonus)

Le format de fichier `SVG` permet de représenter des dessins. Nous allons ajouter des méthodes à nos classes afin de générer un fichier `SVG` permettant de visualiser les objets graphiques instanciés dans le `main`.

Tâche 16 : Ajouter dans les classes `Circle` et `Square` la méthode `void printSvg()` qui permet d'afficher dans la console la ligne représentant l'objet dans le format `SVG`.

Par exemple, les instructions suivantes :

```
System.out.println("<svg");
System.out.println("  viewBox='0 0 100 100' style='background: white'");
System.out.println("  version='1.1' xmlns='http://www.w3.org/2000/svg'");
System.out.println(">");

Circle circle1 = new Circle(new Point(20, 30), 10);
circle1.printSvgLine();

Square square1 = new Square(new Point(50, 40));
```

```java
square1.side = 20;
square1.rotateClockwise(Math.PI/4);
square1.printSvgLine();

System.out.println("</svg>");
```

doivent produire l'affichage suivant en console :

```
<svg
  viewBox='0 0 100 100' style='background: white'
  version='1.1' xmlns='http://www.w3.org/2000/svg'
>
  <circle cx='20.0' cy='30.0' r='10.0' />
  <rect x='40.0' y='30.0' width='20.0' height='20.0' transform='rotate(45.0 50.0 40.0)' />
</svg>
```

Les coordonnées x et y du rectangle en `svg` sont celles d'un coin et sont donc égales à celle du centre moins la moitié de la valeur du côté. La rotation `'rotate(45.0 50.0 40.0)'` décrit une rotation de `45°` avec comme centre le point $(50, 40)$.

Une fois que votre programme produit bien cet affichage, vous pouvez visualiser le résultat en tapant la commande `java Main > example1.svg`. Cela va créer le fichier `example1.svg` que vous pouvez visualiser en cliquant dessus.