Aix*Marseille université
Initiative d'excellence
Année 2018-19

Domaine Sciences et Technologies
Licence informatique

Programmation 2 : Corrigé du TD 8
Code UE : SIN3U02

Sujet blanc d'examen

1 Questions

1. La remarque la plus importante est que les variables sont mal nommées : `w`, `h` et `r` ont de noms pas assez explicites. Il devraient être `width`, `height` et `radius`.

2.
```java
public interface ClipInterface {
    boolean contains(int x, int y);
}
```

3.
```java
public class Rectangle implements ClipInterface {
    int x, y, width, height;

    public Rectangle(int x, int y, int width, int height) {
        this.x = x;
        this.y = y;
        this.width = width;
        this.height = height;
    }

    @Override
    public boolean contains(int x, int y) {
        return this.x < x && x <= this.x+width
                && this.y < y && y <= this.y+height;
    }
}
```

4.
```java
public class Disk implements ClipInterface{
    private int x;
    private int y;
    private int radius;

    public Disk(int x, int y, int radius) {
        this.x = x;
        this.y = y;
        this.radius = radius;
    }

    @Override
    public boolean contains(int x, int y) {
        return squareOfDistanceToCenter(x, y)
                <= squareOfRadius();
    }

    private int squareOfRadius() {
        return radius*radius;
    }
```

```
    private int squareOfDistanceToCenter(int x, int y) {
        return (this.x-x)*(this.x-x)+(this.y-y)*(this.y-y);
    }

    public boolean hasTheSameCenterAs(Disk disk){
        return this.x == disk.x && this.y == disk.y;
    }
}
```

5.
```
public class Annulus implements ClipInterface{
    private Disk innerDisk;
    private Disk outerDisk;

    public Annulus(Disk innerDisk, Disk outerDisk) {
        if(!innerDisk.hasTheSameCenterAs(outerDisk))
            throw new IllegalArgumentException(
                    "The disks must have the same center.");
        this.innerDisk = innerDisk;
        this.outerDisk = outerDisk;
    }

    @Override
    public boolean contains(int x, int y) {
        return !innerDisk.contains(x,y)
                && outerDisk.contains(x,y);
    }
}
```

6. L'exception est déjà levé dans le code donné ci-dessus. Il suffit de rajouter un test et de lever l'exception.

7.
```
public class BitMapClip implements ClipInterface{
    boolean[][] isPresent;

    public BitMapClip(boolean[][] isPresent) {
        this.isPresent = isPresent;
    }

    boolean contains(int x, int y){
        return isPresent[x][y];
    }
}
```

8.
```
public class IntersectionClips {
    Clip[] clips;

    public IntersectionClips(Clip[] clips) {
        this.clips = clips;
    }

    @Override
    public boolean contains(int x, int y) {
        for(Clip clip : clips){
```

```
            if(!clips.contains(x,y)){
                return false;
            }
        }
        return true;
    }
}
```

9. Il y plusieurs façon de mutualiser le code entre intersection et union. Une manière de faire est de faire une classe contenant une liste de `Clip` avec des méthodes permettant de connaître le nombre de `Clip` contenant un point et le nombre de `Clip` total. Cela permet de faire l'intersection et l'union facilement.

```
public abstract class AbstractClipCollection
        implements ClipInterface {
    private Clip[] clips;

    public AbstractClipCollection(Clip[] clips) {
        this.clips = clips;
    }

    protected int numberOfClipsContaining(int x, int y){
        int numberOfClips=0;
        for (Clip clip : clips){
            if(clip.contains(x,y))
                numberOfClips++;
        }
        return numberOfClips;
    }

    protected int numberOfClips(){
        return clips.length;
    }
}

public class IntersectionClips extends AbstractClipCollection {
    public IntersectionClips(Clip[] clips) {
        super(clips);
    }

    @Override
    public boolean contains(int x, int y) {
        return numberOfClipsContaining(x,y)
                == numberOfClips();
    }
}

public class UnionClips extends AbstractClipCollection {
    public UnionClips(Clip[] clips) {
        super(clips);
    }

    @Override
```

```
    public boolean contains(int x, int y) {
        return numberOfClipsContaining(x,y)>=1;
    }
}
```

10. On pourrait améliorer le code en créant par exemple une classe `Point` avec une méthode `distanceTo` permettant de calculer la distance entre deux points.