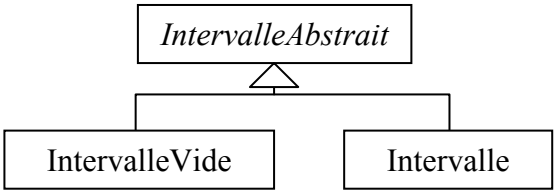
Algorithmique et programmation



Nous nous proposons de représenter des intervalles sur des entiers. Un intervalle est défini par un entier de début et un entier de fin. Un intervalle représente l'ensemble de tous les entiers qui sont supérieurs ou égaux à l'entier de début et inférieurs ou égaux à l'entier de fin. L'intervalle vide (qui ne contient aucun entier) est représenté par un seul objet qui est l'instance unique de la classe *IntervalleVide*.

Question 1 (3 points)

Ecrire la définition du constructeur de la classe *Intervalle*. Si *d* est supérieur à *f*, l'intervalle ne peut pas être construit, et une exception *RuntimeException* est levée.

Question 2 (3 points)

Ecrire les définitions des méthodes *appartient* des classes *IntervalleVide* et *Intervalle*. Cette méthode retourne *true* si le paramètre appartient à l'intervalle et *false* sinon.

Question 3 (4 points)

Ecrire les définitions des méthodes *toString* des classes *IntervalleVide* et *Intervalle*.

- Pour *IntervalleVide* la méthode *toString* retourne la chaîne de caractères "\u2205" (caractère ∅)
- Pour *Intervalle* la méthode *toString* retourne la chaîne de caractères `"["d"-"f"]"` si *d* est différent de *f* et la chaîne de caractères `"{"d"}"` sinon

Question 4 (5 points)

Ecrire les définitions des méthodes *intersection* des classes *IntervalleVide* et *Intervalle*. L'intersection de deux intervalles est l'intervalle qui contient tous les entiers communs aux deux intervalles.

Question 5 (5 points)

Ecrire les définitions des méthodes *union* des classes *IntervalleVide* et *Intervalle*. L'union de deux intervalles n'est pas toujours définie : [1-10] union [15-20] n'est pas un intervalle !

- Dans le cas où l'union n'est pas définie, il faudra lever une exception.
- Quand l'union des deux intervalles est définie, c'est un intervalle qui contient les entiers appartenant à l'un ou l'autre intervalle.

```
abstract public class IntervalleAbstrait {
    abstract public boolean appartient (int i);
    abstract public IntervalleAbstrait intersection(IntervalleAbstrait i);
    abstract public IntervalleAbstrait union(IntervalleAbstrait i)
        throws Exception;
}
```

```
public class IntervalleVide extends IntervalleAbstrait {
    private static IntervalleVide instance = new IntervalleVide();
    protected IntervalleVide(){}

    static public IntervalleVide getInstance(){
      return instance;
    }
    public boolean appartient(int i) {…}
    public IntervalleAbstrait intersection(IntervalleAbstrait i) {…}
    public IntervalleAbstrait union(IntervalleAbstrait i)
             throws Exception {…}
    public String toString(){…}
}
```

```
public class Intervalle extends IntervalleAbstrait {
    private int d, f;

   public Intervalle(int d, int f) {…}
   public boolean appartient(int i) {…}

   public IntervalleAbstrait intersection(IntervalleAbstrait i) {…}
   public IntervalleAbstrait union(IntervalleAbstrait i)
      throws Exception {…}
   public String toString(){…}
}
```