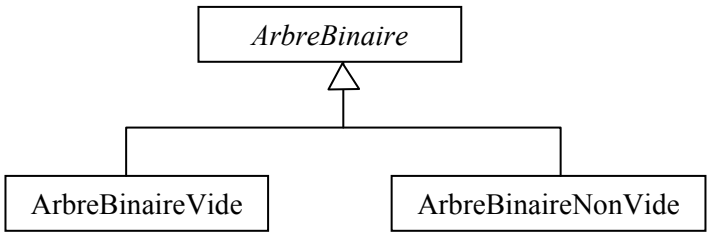
Programmation Examen

Nous nous proposons de représenter un arbre binaire ordonné à l'aide des classes *ArbreBinaireVide* et *ArbreBinaireNonVide*.



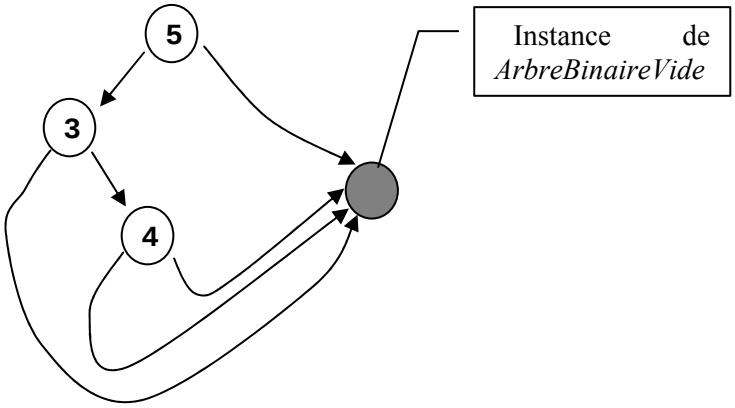
Question 1. (4 points)
Définir le constructeur de la classe *ArbreBinaireNonVide* et de la classe *ArbreBinaireVide*. Définir la méthode *getInstance()* de la classe *ArbreBinaireVide*.

Question 2. (4 points)
Définir la méthode *ajout(int a)* de la classe *ArbreBinaireNonVide* et *ArbreBinaireVide*.

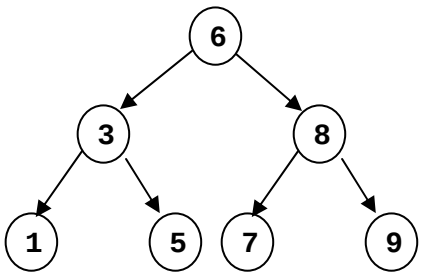
La séquence suivante :

```
ArbreBinaire ab = ArbreBinaireVide.getInstance() ;
ab.ajout(5) ;
ab.ajout(3) ;
ab.ajout(4) ;
```

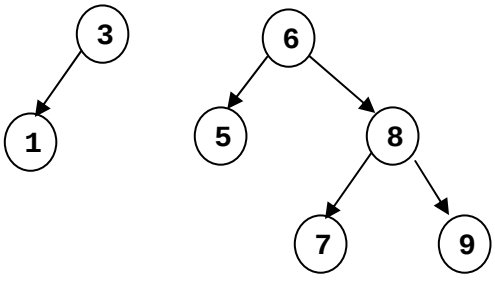
produit l'arbre ci-contre :

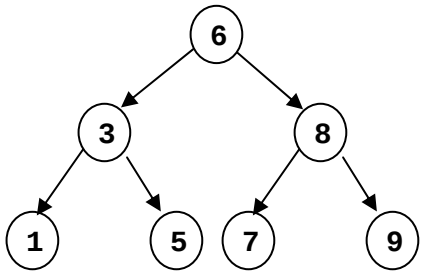


Question 3. (4 points)
Définir la méthode *coupe(int a)* de la classe *ArbreBinaireNonVide* et *ArbreBinaireVide*. Cette méthode retourne une paire d'arbres : un arbre qui contient toutes les valeurs inférieures à *a* et un arbre qui contient toutes les valeurs supérieures à *a*. La construction de ces deux arbres devra se faire directement, sans faire appel à la méthode *ajout* définie précédemment. L'arbre initial ne conserve pas forcément sa valeur après l'appel de la méthode.

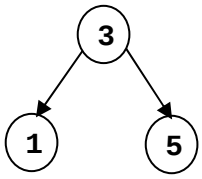


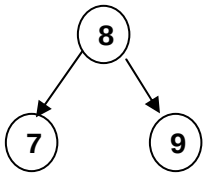
La coupe de l'arbre de gauche pour a = 4 donne les deux arbres :



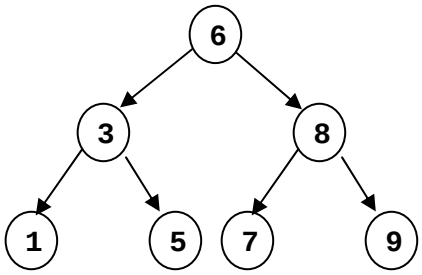


La coupe de l'arbre de gauche pour a = 6 donne les deux arbres :

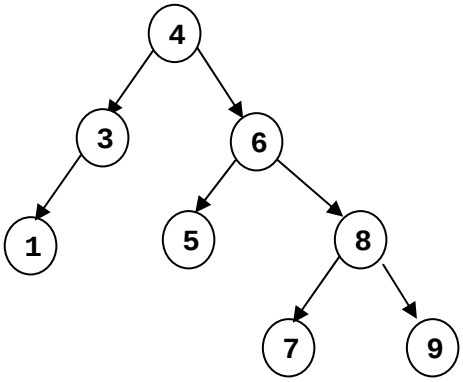




Question 4. (3 points)
Définir la méthode *ajoutCoupe(int a)* de la classe *ArbreBinaireNonVide* et *ArbreBinaireVide*. Cette méthode ajoute un nouvel élément comme racine de l'arbre :



L'ajout de 4 dans l'arbre de gauche donne l'arbre de droite :



Question 5. (5 points)
Définir les méthodes *nbNoeuds()* des classes *ArbreBinaireNonVide* et *ArbreBinaireVide*.
Définir les méthodes *toArray()* des classes *ArbreBinaireNonVide* et *ArbreBinaireVide*. Ces méthodes créent un tableau d'entiers, qui contient tous les entiers de l'arbre binaire, ordonnés par ordre croissant.

```
abstract public class ArbreBinaire {
  abstract public ArbreBinaire ajout( int a);
  abstract public Paire<ArbreBinaire> coupe(int a) ;
  abstract public ArbreBinaire ajoutCoupe(int a);
  abstract public int nbNoeud();
  public int[] toArray(){…}
  abstract protected int toArray(int [] t, int i);
}
```

```
public class ArbreBinaireVide extends ArbreBinaire{
  private static instance = new ArbreBinaireVide();
  protected ArbreBinaireVide(){…}
  public static ArbreBinaireVide getInstance(){…}
  public ArbreBinaire ajout( int a);
  public Paire<ArbreBinaire> coupe(int a) ;
  public ArbreBinaire ajoutCoupe(int a);
  public int nbNoeud();
  public int[] toArray(){…}
  protected int toArray(int [] t, int i);
}
```

```
public class ArbreBinaireNonVide extends ArbreBinaire {
  private ArbreBinaire gauche, droite ;
  private int valeur ;
  public ArbreBinaireNonVide(ArbreBinaire g, ArbreBinaire d, int v) {…}
  public ArbreBinaire ajout( int a) {…}
  public Paire<ArbreBinaire> coupe(int a) {…}
  public ArbreBinaire ajoutCoupe(int a) {…}
  public int nbNoeud();
  public int[] toArray(){…}
  protected int toArray(int [] t, int i) {…}
}
```

```
public class Paire<E> {
  E premier;
  E second;
  public Paire(E un, E deux){
        premier = un;
        second = deux;
  }
}
```