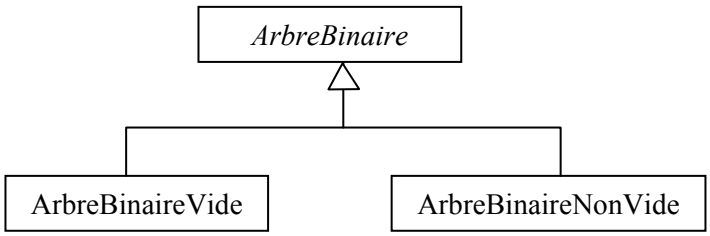
Programmation 2
Examen

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*.

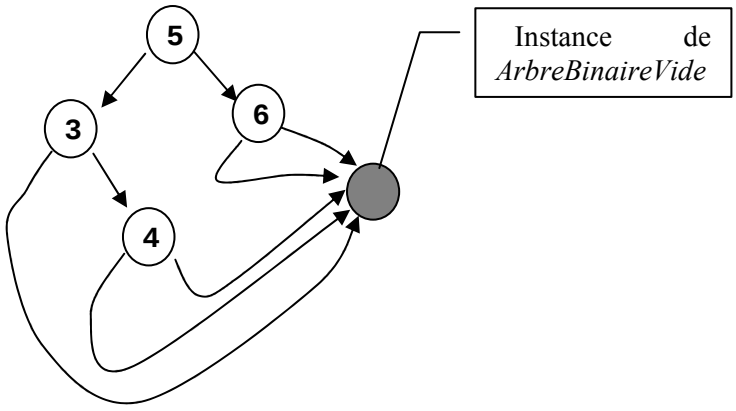
Question 2. (4 points)

Définir la méthode *premier()* de la classe *ArbreBinaireNonVide* et *ArbreBinaireVide*.
Définir la méthode *dernier()* de la classe *ArbreBinaireNonVide* et *ArbreBinaireVide*.

La séquence suivante :

```
ArbreBinaire ab =
ArbreBinaireVide.getI
nstance() ;
ab.ajout(5) ;
ab.ajout(3) ;
ab.ajout(4) ;
ab.ajout(6) ;
```

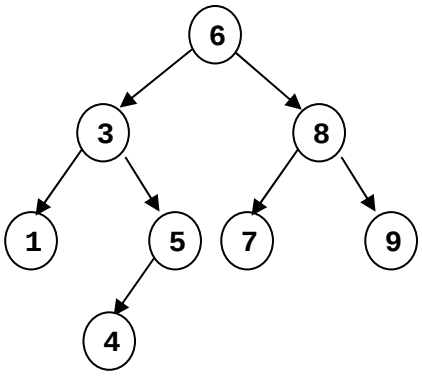
produit l'arbre ci-contre :



La méthode *premier()* retourne 3 *et dernier()* retourne 6.
Si l'arbre est vide, on lève une *RuntimeException*, avec le message "arbre vide"

Question 3. (4 points)
Définir la méthode *int suivant(int a)* de la classe *ArbreBinaireNonVide* et *ArbreBinaireVide*. Cette méthode retourne la valeur qui suit la valeur *a* dans l'arbre binaire. La valeur qui suit *a*, est, si elle existe, la plus petite valeur de l'arbre qui est plus grande que *a*.

- Si *a* n'est pas dans l'arbre, on lève une *RuntimeException*, avec le message "a n'est pas dans l'arbre"
- Si *a* n'a pas de suivant dans l'arbre, on lève une *RuntimeException*, avec le message "pas de suivant"



suivant (3) = 4
suivant (4) = 5
suivant (5) = 6

Question 4. (4 points)
Définir la méthode *Object clone()* de la classe *ArbreBinaireNonVide* et *ArbreBinaireVide*. Cette méthode crée une copie de l'arbre binaire.

Question 5. (4 points)
Définir la méthode *boolean equals (Object o)* des classes *ArbreBinaireNonVide* et *ArbreBinaireVide*.

```
abstract public class ArbreBinaire {
  abstract public int premier();
  abstract public int dernier();
  abstract public int suivant(int a) ;
  abstract public Object clone();
  abstract public boolean equals(Object o);
}
```

```
public class ArbreBinaireVide extends ArbreBinaire{
  private static instance = new ArbreBinaireVide();
  protected ArbreBinaireVide(){…}
  public static ArbreBinaireVide getInstance(){…}
  public int premier(){…}
  public int dernier(){…}
  public int suivant(int a) {…}
  public Object clone(){…}
  public boolean equals(Object o) {…}
}
```

```
public class ArbreBinaireNonVide extends ArbreBinaire {
  private ArbreBinaire gauche, droite ;
  private int valeur ;
  public ArbreBinaireNonVide(ArbreBinaire g, ArbreBinaire d, int v) {…}
  public int premier(){…}
  public int dernier(){…}
  public int suivant(int a) {…}
  public Object clone(){…}
  public boolean equals(Object o) {…}
}
```