Exemple 01 : Penser Objet

Dans ce premier exemple, on va réaliser un simple programme java qui permet de résoudre l'équation $ax+b = 0$. On proposera deux solutions :

– La première en utilisant la programmation direct (algorithmique ou structurelle : structure de données, variables et procédures / fonctions). Pas de classes, d'instances ni d'envoi de messages (invocation de méthodes sur des instances)

– La deuxième en utilisant la programmation orientée objets : On définit la classe *Equation1D* pour représenter les équations mathématiques de premier degré avec un seul inconnu. Et dans la méthode main, on crée un instance de *Equation1D*. Pour trouver le résultat, il suffit d'envoyer le message *e.getSolution()*

La première solution :

Un programme très simple (qui ressemble énormément aux programme C ou Pascal : simple démarche algorithmique). Le programme Java est composé d'une seule classe : la classe mère (contenant la méthode main). Voici le listing du code source.

// Le fichier : MainExemple.java

```
import java.util.Scanner;


/**
 *
 * @author OUZEGGANE Redouane
 *    <br/><br/>
 *          La rÃ©solution de l'Ã©quation ax + b = 0 avec la programmation impÃ©rative (structurelle)
 *
 *    <br/><br/>
 *          Dans cet exemple, on montre qu'on peut programmer JAVA sans orientÃ© objet (pensÃ© algorithmique)
 *          EntrÃ©e + Traitement (affectation et calcul) + Sortie
 *
 *    <br/><br/>
 *          ==> Ce qu'il faut Ã©viter dans vos programmes JAVA<br/><br/>
 *
 *          Il faut concevoir avant d'implÃ©menter<br/>
 *          Concevoir : dÃ©terminer les classes et leurs inter-relations nÃ©cessaires pour rÃ©soudre le problÃ¨me<br/>
 *          Quels sont les classes nÃ©cessaire poru rÃ©soudre l'Ã©quation ax + b = 0 ?<br/>
 */
public class MainExemple{
      public static void main(String args[]){
            /**
             * L'instance kbScanner nous permet de rÃ©aliser des entrÃ©es Ã travers le clavier : System.in (entrÃ©e standard)
             */
```

```
            Scanner kbScanner = new Scanner(System.in);

            /**
             * L'introduction des deux nombres a et b
             */
            System.out.println ("Donnez la valeur de a : ");
            float a = kbScanner.nextFloat();

            System.out.println ("Donnez la valeur de b : ");
            float b = kbScanner.nextFloat();


            /**
              * Le teste des cas - 3 cas (InfinitÃ© de solution, Solution
impossible et Solution unique)
             */
            if (a == 0){
                  if (b == 0){
                        System.out.println ("InfinitÃ© de solution !!!");
                  }
                  else{
                        System.out.println     ("Impossible     de     rÃ©oudre
l'Ã©quaiton");
                  }
            }
            else{
                  float x = -b / a;
                  System.out.println ("Il y a une solution x = "+x);
            }
      }
}
```

La première solution :

Dans la deuxième solution, on aura deux classes : la classe Equation1D qui prends la responsabilité de représenter les équations du premier degré avec un seul inconnu, et cherche le résultat de la solution. Et la deuxième est la classe principale.

// Le fichier : Equation1D.java

```
/**
 *
 * @author OUZEGGANE Redouane
 *    <br/><br/>Cette classe modÃ©lise (reprÃ©sente) les Ã©quations du premier
degrÃ© avec un seul inconnu ax + b = 0
 */
public class Equation1D{
      /**
       * Codification de l'Ã©tat de l'Ã©quation : trois Ã©tats
       */

      /**
       * Etat : Solution impossible
       */
      public static final int ETAT_IMPOSSIBLE              = 1;
```

```java
/**
 * Etat : InfinitÃ© de solution
 */
public static final int ETAT_INIFINITE_SOLUTION      = 2;

/**
 * Etat : Une seule solution
 */
public static final int ETAT_UNE_SOLUTION                  = 3;


/**
 * Les attributs sont toujours inivisbles
 */
private float a;
private float b;

/**
 * Le premier constructeur par dÃ©faut
 */
public Equation1D() {
}

/**
 * Le deuxiÃ¨me constructeur
 * @param a
 * @param b
 */
public Equation1D(float a, float b){
      this.a = a;
      this.b = b;
}

/*
 * Les accesseurs et les modificateurs
 *
 */

public float getA() {
      return a;
}

public void setA(float a) {
      this.a = a;
}

public float getB() {
      return b;
}

public void setB(float b) {
      this.b = b;
}

/**
 * Obtenir l'Ã©tat de l'Ã©quation (3 Ã©tats)
 * @return
 */
public int getEtat(){
      if (a != 0) return ETAT_UNE_SOLUTION;
      if (a == 0 && b == 0) return ETAT_INIFINITE_SOLUTION;
```

```
            return ETAT_IMPOSSIBLE;
      }


      /**
       * Obtenir la solution de l'Ã©quation (si elle possÃ¨de une solution)
       * @return Soit la solution de l'Ã©quation (une instance de Float)
       * Ou bien null (si pas de solution ou inifnitÃ© de solution)
       */
      public Float getSolution(){
            if (getEtat() == ETAT_UNE_SOLUTION){
                  return -b / a;
            }

            return null;
      }

      /**
       * Obtenir la solution sous forme chaÃ®ne de caractÃ¨re.
       * @return la solution accompagnÃ© de chaÃ®ne de caractÃ¨re pour une
meilleure reprÃ©sentation
       */
      public String getSolutionAsString(){
            String reponse = "Solution de l'Ã©quaiton : "+a+"x + "+b+" = 0\n";
            int etat = getEtat();

            if (etat == ETAT_UNE_SOLUTION){
                  reponse += "Il y a une seul solution x = "+(-b / a);
            }
            else if (etat == ETAT_INIFINITE_SOLUTION){
                  reponse += "InfinitÃ© de solution ...";
            }
            else if (etat == ETAT_IMPOSSIBLE){
                  reponse += "Soution Impossible";
            }

            return reponse;
      }


      /**
       * toString() : une mÃ©thode de la classe Object (surcharge de mÃ©thode)
       * @return une chaÃ®ne de caractÃ¨res
       */

      @Override
      public String toString() {
            return this.getSolutionAsString();
      }

}
```

// Le fichier : MainExample.java

```
import java.util.Scanner;

/**
 *
 * @author OUZEGGANE Redouane
 *    <br/>
 *          La rÃ©solution de l'Ã©quation ax + b = 0 avec la programmation
OrientÃ©e Objets
 *
 *    <br/><br/>
 *          La solution de cette Ã©quation passe par la crÃ©ation d'une instance
e de la classe Equation
 *          <br/> Il suffit d'envoyer le message getSolutionAsString() pour
obtenir la rÃ©ponse.
 *          <br/><br/>la mÃ©thode principale main est plus simplifiÃ©e :
entrÃ©es + instanciation + envoi de message + sortie
 *          <br/><br/>Ici, c'est une solution orientÃ© objets
 *          <br/> C'est quoi l'intÃ©rÃªt / c'est quoi l'avantage par rapport Ã
l'exemple prÃ©cÃ©dent
 */
public class MainExample{
      public static void main(String args[]){
            Scanner scanner = new Scanner(System.in);

            System.out.println ("Donnez la valeur de a : ");
            float a = scanner.nextFloat();

            System.out.println ("Donnez la valeur de b : ");
            float b = scanner.nextFloat();

            Equation1D e = new Equation1D(a, b);
            System.out.println (e.getSolutionAsString());
      }
}
```

Récapitulatif

Ce qu'il faut retenir de ces deux exemples est de prendre le soin et le temps nécessaires pour modéliser un problème. Cette modélisation prend en considération les concepts du paradigme orienté objets. Il faut analyser le problème, le bien comprendre et faire extraire les concepts de base du problème. Ces concepts sont candidats à devenir des classes d'objets.