# MOOC Init Prog Java
# Tutoriels semaine 2

Les tutoriels sont des exercices qui reprennent des exemples du cours et dont le corrigé est donné progressivement au fur et à mesure de la donnée de l'exercice lui-même.

Ils sont conseillés comme un premier exercice sur un sujet que l'étudiant ne pense pas encore assez maîtriser pour aborder par lui-même un exercice «classique».

---

## Semaine 2 : Résolution de polynômes de degré 2

Nous voulons écrire un programme permettant de trouver les zéros de polynômes de degré 2 de la forme $ax^2+bx+c$, où les coefficients sont des nombres réels et a est non nul (sans quoi ce ne serait plus un polynôme de degré 2...).

1. Comme d'habitude, commencez par ouvrir un fichier (vide) `Degre2.java` dans Eclipse ou votre éditeur favori et si nécessaire préparez la «coquille vide» de base accueillant votre programme :

   ```
   class Degre2 {
       public static void main(String[] args) {
       }
   }
   ```

2. Commençons notre programme en prévoyant la place pour les trois coefficients de notre polynôme : a, b et c. Il faut pour cela déclarer des variables.

   Comme nous souhaitons que ces coefficients soient des nombres réels, nous **déclarons des variables** de type `double` :

   ```
   class Degre2 {
       public static void main(String[] args) {
           double a;
           double b;
           double c;
       }
   }
   ```

   Note : on aurait aussi pu écrire en une ligne

   ```
   double a,b,c;
   ```

3. Maintenant, le réflexe du bon programmeur : il faut penser à initialiser ces variables. Il nous semble ici naturel de les initialiser à zéro :

   ```
   class Degre2 {
       public static void main(String[] args) {
           double a = 0.0;
           double b = 0.0;
           double c = 0.0;
       }
   ```

```
}
```

4. Il faut ensuite récupérer les valeurs des paramètres entrés par l'utilisateur. Étant donné que `a` ne doit pas être nul, nous allons utiliser une boucle dans laquelle nous vérifierons cette condition. Nous pouvons ensuite acquérir les paramètres `b` et `c`.

```
import java.util.Scanner;
class Degre2 {
    private static Scanner scanner = new Scanner(System.in);
    public static void main(String[] args) {
        double a = 0.0;
        double b = 0.0;
        double c = 0.0;

        // tant que a est nul, demander une valeur a l'utilisateur
        while (a == 0.0) {
            System.out.print("Entrez une valeur non nulle pour a :");
            a = scanner.nextDouble();
        }

        System.out.print("Entrez une valeur pour b:");
        b = scanner.nextDouble();
        System.out.print("Entrez une valeur pour c:");
        c = scanner.nextDouble();
    }
}
```

5. Maintenant que nous connaissons tous les paramètres de l'équation, nous pouvons la résoudre. Il nous faut tout d'abord calculer le discriminant `delta = b*b - 4*a*c`.

   Il faut donc prévoir de la place pour le stocker : une nouvelle variable de type `double`.

   On peut ici choisir de la déclarer sans initialisation puis de lui affecter la bonne valeur à l'aide d'une autre instruction ou plus simplement de l'initialiser directement avec `b*b - 4*a*c` puisque toutes ces valeurs sont connues à ce stade là. C'est cette seconde solution qui est choisie ici :

```
import java.util.Scanner;
class Degre2 {
    private static Scanner scanner = new Scanner(System.in);
    public static void main(String[] args) {
        double a = 0.0;
        double b = 0.0;
        double c = 0.0;
        // tant que a est nul, demander une valeur a l'utilisateur
        while (a == 0.0) {
            System.out.print("Entrez une valeur non nulle pour a :");
            a = scanner.nextDouble();
        }

        System.out.print("Entrez une valeur pour b:");
        b = scanner.nextDouble();
        System.out.print("Entrez une valeur pour c:");
        c = scanner.nextDouble();
        double delta = b * b - 4.0 * a * c;
    }
```

```
}
```

6. Pour le calcul de la solution, différents cas peuvent se produire en fonction de la valeur prise par le discriminant, que nous distinguons dans le programme à l'aide de blocs `if-else` :
    - Si `delta` est négatif, il n'y a pas de solution réelle.
    - Si `delta` est nul, la solution est unique et vaut `-b/2a`.
    - Autrement, les solutions sont `(-b ± sqrt(delta))/2a`.

    La méthode Java permettant de calculer des racines carrées s'appelle `sqrt()`, qui vient de l'anglais "square root".
    Il est possible d'utiliser cette méthode (ainsi que d'autres fonctions mathématiques) en précisant la classe (`Math`) à laquelle elle appartient. Ceci se fait au moyen de la notation `Math.sqrt()` que nous comprendrons lorsque nous aurons abordé la programmation orientée-objet.

    Le programme complet devient donc :

```
import java.util.Scanner;
class Degre2 {
    private static Scanner scanner = new Scanner(System.in);
    public static void main(String[] args) {
        double a = 0.0;
        double b = 0.0;
        double c = 0.0;

        // tant que a est nul, demander une valeur a l'utilisateur
        while (a == 0.0) {
            System.out.print("Entrez une valeur non nulle pour a:");
            a = scanner.nextDouble();
        }

        System.out.print("Entrez une valeur pour b:");
        b = scanner.nextDouble();
        System.out.print("Entrez une valeur pour c:");
        c = scanner.nextDouble();
        double delta = b * b - 4.0 * a * c;
        if (delta < 0.0) {
            System.out.println("Pas de solutions reelles");
        } else if (delta > 0.0) {
                System.out.println("Deux solutions : "
                        + (-b - Math.sqrt(delta)) / (2.0 * a)
                        +  " et " + (-b + Math.sqrt(delta)) / (2.0 * a));

        } else {
                System.out.println("Une solution unique : "
                                + -b / (2.0 * a));

        }
    }
}
```