Bases du langage Java

Exercice 1 : Racines d'un polynôme

Ecrire le corps d'une méthode qui permet de résoudre une équation de degré 2 maximum. Le résultat sera affiché, qu'il soit réel ou complexe.

Remarque : la méthode System.out.println permet d'afficher un message (chaine de caractère, variable ou littéral de type primitif) à l'écran.

```
public class Exercice4 {

    public static void main(String[] args) {
      // x2, x1, x0
      double[] polynome = {-1, 1, -1};

      double delta = polynome[1] * polynome[1] - 4 * polynome[0] *
polynome[2];

      double moinsBSur2A = -polynome[1] / (2 * polynome[0]);
      double racineDeltaSur2A = Math.sqrt(Math.abs(delta)) / (2 *
polynome[0]);

      if (delta > 0) {
          System.out.println(" (Discrimant positif)");
          System.out.println("Racine réelle 1 : " + (moinsBSur2A +
racineDeltaSur2A));
          System.out.println("Racine réelle 2 : " + (moinsBSur2A -
racineDeltaSur2A));

      } else if (delta == 0) {
          System.out.println(" (Discrimant nul)");
          System.out.println("Racine double : " + moinsBSur2A);
      } else {
          System.out.println(" (Discrimant négatif)");
          System.out.println("Racine complexe 1 : " + moinsBSur2A + " + i"
+ racineDeltaSur2A);
          System.out.println("Racine complexe 2 : " + moinsBSur2A + " - i"
+ racineDeltaSur2A);
      }
    }
}
```

Exercice 2 : Manipulation de tableaux

On désire écrire le corps d'une méthode qui effectue le produit matriciel de deux tableaux d'entiers. Cette méthode devra donc effectuer les opérations suivantes :

1. Vérifier que les dimensions des matrices sont compatibles ;
2. Effectuer le produit matriciel
3. Afficher le résultat formaté sur la sortie standard.

```
public class Exercice2 {

    public static void main(String[] args) {

      int [][] tab1 = {{1, 2, 3}, {2, 0, 1}, {3, 2, 1}};
      int [][] tab2 = {{2, 1, 3}, {0, 0, 1}, {1, 2, 0}};
```

```
      if (tab1[1].length != tab2.length) {
          System.out.println("Dimensions incompatibles (col1 = " +
tab1[1].length + " and lig2 = " + tab2.length + ")");
      } else {
          int[][] resultat = new int[tab1.length][tab2[1].length];
          for (int ligne = 0; ligne < tab1.length; ligne++) {
            System.out.print("[\t");
            for (int colonne = 0; colonne < tab2[1].length; colonne++) {
                int produit = 0;
                for (int position = 0; position < tab1[ligne].length;
position++) {
//                System.out.println();
                  produit += tab1[ligne][position] *
tab2[position][colonne];
                }
                resultat[ligne][colonne] = produit;
                System.out.print(produit + "\t");
            }
            System.out.print("]\n");
          }
      }
    }

}
```

Exercice 3 : Changement de base

Ecrire le corps d'une méthode qui permette d'effectuer le changement de base d'un entier donné en base 10 vers une base quelconque (au maximum hexadécimale). La base et le nombre à convertir seront paramétriques. Le résultat de la conversion sera stocké dans un tableau où chaque élément contiendra un digit. On effectuera ensuite l'affichage du résultat.

```
public class Exercice3 {

    public static void main(String[] args) {
      // TODO Auto-generated method stub

      int [] resultat = new int[16];
      int nombre = 45628;
      byte base = 2;
      int reste = 0;

      // Conversion dans la base cible
      int indice = 0;

      do {
          reste = nombre % base;
          nombre = nombre / base;
          resultat[resultat.length - indice - 1] = reste;
          indice++;
      } while (nombre != 0);

      // Affichage du résultat
      boolean entete = true;

      for (int i = 0; i < resultat.length; i++) {
          int chiffre = resultat[i];

          if (chiffre != 0 && entete == true) {
            entete = false;
```

```
            }

            switch (chiffre) {
            case 0:
              // On supprime les 0 en tête du résultat
              if (entete == false) {
                  System.out.print(chiffre);
              }
              break;
            case 10:
              System.out.print("A");
              break;
            case 11:
              System.out.print("B");
              break;
            case 12:
              System.out.print("C");
              break;
            case 13:
              System.out.print("D");
              break;
            case 14:
              System.out.print("E");
              break;
            case 15:
              System.out.print("F");
              break;
            default:
              System.out.print(chiffre);
            }
        }
    }
}
```

Exercices à faire en temps hors-encadrement (THE)

Exercice 4 : Opérations binaires

On désire écrire un programme qui manipule des adresses IP. Une adresse est composée de 32 bits dont la première partie désigne le réseau sur lequel est situé une machine et la seconde partie la machine. Afin de savoir parmi ces 32 bits où ce situe la séparation, on utilise un masque qui est une suite de 32 bits de 1 suivis par des 0.

Ecrire le corps d'une méthode qui permette :

1. d'afficher la partie réseau et la partie machine d'une adresse IP.
2. de déterminer la classe d'une adresse. La classe d'une adresse est déterminée par la valeur des bits de poids fort du premier octet :

Classe	Bits du premier octet
A	0xxxxxxx
B	10xxxxxx
C	110xxxxx
D	1110xxxx

```
public class Exercice4 {

    public static void main(String[] args) {

      byte[][] adresse1 = {{(byte)192, (byte)168, 2, 1}, {(byte)255,
(byte)255, (byte)255, 0}};

      System.out.print("Adresse du réseau : ");

      for (int octet = 0; octet < adresse1[0].length; octet++) {
          System.out.print((adresse1[0][octet] & adresse1[1][octet]) +
"\t");
      }

      System.out.print("\n");

      for (int octet = 0; octet < 4; octet ++) {
          int valeur = 0;
          byte masque = 0x01;
          for (int i = 0; i < 8; i++) {
            valeur += ((int) adresse1[0][octet]) & masque;
            masque*= 2; // ou operateur <<
          }
          System.out.print(valeur + ".");
      }

      System.out.print("\n");

      int premierOctet = adresse1[0][0];
      byte position = 1;
      while ((premierOctet & 0x80) != 0) {
          premierOctet = premierOctet << 1;
          position++;
      }

      switch(position) {
      case 1:
          System.out.println("Adresse de classe A");
```

```
            break;
        case 2:
            System.out.println("Adresse de classe B");
            break;
        case 3:
            System.out.println("Adresse de classe C");
            break;
        case 4:
            System.out.println("Adresse de classe D");
            break;
        default:
            System.out.println("Adresse de classe inconnue");
        }
    }

}
```