Prénom :
Nom :

Quand cela est demandé, vous devez indiquer l'affichage produit en répondant sur l'énoncé, à droite du programme. **Soyez très précis dans vos réponses.**

Affichage produit

```
public class Exo11 {
  private int x;

  private int y;

  public Exo11(int x, int y) {
    this.x = x;
    this.y = y;
  }

  public Exo11(int t) {
    this(t - 1, t + 2);
  }

  public int delta() {
    return x - y;
  }

  public int sigma() {
    return x + y + 1;
  }

  public Exo11 combine(Exo11 that) {
    return new Exo11(this.x + that.y, this.y - that.x);
  }
}

public class TestExo11 {
  public static void main(String[] args) {
    Exo11 oub = new Exo11(1, 2);
    System.out.printf("%d %d%n", oub.delta(), oub.sigma());
    Exo11 bla = new Exo11(4);
    System.out.printf("%d %d%n", bla.delta(), bla.sigma());
    Exo11 pouic = oub.combine(bla);
    System.out.printf("%d %d%n", pouic.delta(), pouic.sigma());
  }
}
```

Programmation (1)

Ajouter à la classe `Exo11` :

1. un constructeur sans paramètre qui initialise les variables d'instance à 1 ;
2. écrire une méthode de la forme suivante :

   ```
   public Exo11 minMax(Exo11 that) {
     ...
   }
   ```

   qui renvoie un nouvel objet `Exo11` dont la variable `x` prend la plus grande valeur des variables `x` de l'objet appelant et de l'objet paramètre, et dont la variable `y` prend la plus petite valeur des variables `x` de l'objet appelant et de l'objet paramètre.

On se contentera d'écrire le constructeur et la méthode demandés, sans réécrire toute la classe.

```
import java.util.Arrays;

public class Exo21 {
  private int[] x;

  public Exo21(int n, int a) {
    x = new int[n];
    for (int i = 0; i < x.length; i++) {
      x[i] = a + i;
    }
  }

  public Exo21(int n) {
    this(n, 0);
  }

  public int sweep(int a) {
    int r = a;
    for (int t : x) {
      r += t;
    }
    return r;
  }

  public Exo21 flop(int a) {
    return new Exo21(x.length + 1, a);
  }

  public void blip(int y) {
    int[] z = new int[x.length + 2];
    z[x.length] = sweep(y);
    z[x.length + 1] = y;
    for (int i = 0; i < x.length; i++) {
      z[i] = x[i] - (i % 2);
    }
    x = z;
  }

  public String toString() {
    return Arrays.toString(x);
  }
}

public class TestExo21 {

  public static void main(String[] args) {
    Exo21 u = new Exo21(4, 2);
    System.out.println(u.toString());
    System.out.println(u.sweep(-3));
    Exo21 v = new Exo21(3);
    System.out.println(v.toString());
    v = v.flop(3);
    System.out.println(v.toString());
    Exo21 w = u;
    u.blip(2);
    System.out.println(u.toString());
    System.out.println(w.toString());
  }

}
```

Affichage produit

Programmation (2)

Un objet de la classe `Dice` ci-dessous permet de simuler le lancer d'un nombre quelconque de dés (à six faces). L'objectif de l'exercice est de compléter la classe (directement sur l'énoncé de préférence) selon les indications suivantes :

- il est permis d'ajouter des variables d'instance si besoin ;
- les variables d'instance ne doivent être initialisées que dans le constructeur ;
- le constructeur permet de préciser le nombre de dés grâce au paramètre `nb` ;
- la méthode `roll` doit renvoyer un tableau contenant un chiffre aléatoire compris entre 1 et 6 pour chaque dé que l'objet représente ;
- la méthode `rollSum` doit renvoyer la somme des valeurs obtenues en lançant autant de dés que l'objet en représente. Attention, cette méthode ne doit pas utiliser de tableau ;
- la méthode `addDice` permet d'augmenter le nombre de dés simulés par l'objet appelant. Le paramètre `n` indique le nombre dés ajoutés ;
- la méthode `rollUntil` renvoie le nombre de lancers nécessaires pour obtenir exactement `count` fois la valeur `val` parmi les valeurs d'un tirage. Par exemple, si `d` est un objet `Dice` représentant 3 dés, l'appel `d.rollUntil(6,2)` renvoie le nombre de lancers deux dés que le programme réalise pour obtenir deux 6 et une autre valeur (différente de 6). S'il est impossible d'obtenir les valeurs demandées, la méthode doit renvoyer -1. On pourra ici utiliser la méthode `roll`.

```
import java.util.Random;

public class Dice {
    private Random rng;


    public Dice(int nb) {



    }

    public int[] roll() {




    }

    public int rollSum() {






    }

    public void addDice(int n) {



    }

    public int rollUntil(int val, int count) {









    }
}
```