Pour les exercices d'affichage, vous devez impérativement répondre sur l'énoncé et le rendre en fin d'examen (dans une copie cachetée permettant de vous identifier après correction). Exceptée cette partie cachetée, rien ne doit pouvoir identifier votre copie et votre énoncé.

Exercice 1

Indiquer l'affichage produit par le programme de gauche

```
public class Exo11 {

  public static void main(String[] args) {
    int i = 2;
    int j = 4;
    int k = 0;
    for (; k < j + 1; k++) {
      System.out.println(k + " " + i + " " + j);
      i = i + 1;
      if (i % 2 == 0) {
        j = j - 1;
      }
    }
    System.out.println(k + " " + i + " " + j);
  }

}
```

Exercice 2

Indiquer l'affichage produit par le programme de gauche

```
public class Exo21 {

  public static void main(String[] args) {
    int u = 1;
    int v = 2;
    while (u < v) {
      System.out.println("(" + u + ", " + v + ")");
      u = u + 1;
      if (u % 2 == 0) {
        v = v + 1;
      } else {
        if (u % 3 == 0) {
          v = v + 2;
        }
      }
    }
    System.out.println("(" + u + ", " + v + ")");
  }

}
```

Exercice 3

Indiquer l'affichage produit par le programme de gauche

```
public class Exo31 {

  public static void main(String[] args) {
    int[] x = { 5, 6, 7 };
    int[] y = new int[x.length + 1];
    System.out.println(y);
    int k = 0;
    for (int z : x) {
      y[k] = z;
      k = k + 1;
    }
    y[k] = k;
    System.out.println(Arrays.toString(y));
    int j = x.length - 1;
    while (j > 0) {
      x[j] = x[j] + j + 1;
      j = j - 1;
      System.out.println(j + " <-> " + x[j]);
    }
    System.out.println(Arrays.toString(x));
  }

}
```

Exercice 4

Indice : $30 = 2 + 4 + 8 + 16$

```
import java.util.Arrays;

public class Exo41 {

    public static void main(String[] args) {
        int[] x = new int[À COMPLÉTER];
        int j = À COMPLÉTER;
        for (int i = 0; i < x.length; i++) {
            j = À COMPLÉTER;
            x[i] = j;
        }
        System.out.println(Arrays.toString(x));
        int z = À COMPLÉTER;
        for (int y : x) {
            z = À COMPLÉTER;
        }
        System.out.println(z);
    }

}
```

Compléter le programme ci dessus pour qu'il affiche :

```
[2, 4, 8, 16]
30
```

Exercice 5

```
public class Exo51 {
  private StringBuilder x;

  public Exo51() {
    x = new StringBuilder(); // chaîne vide
  }

  public Exo51 plop(char c) {
    x.append(c); // ajout à la fin
    return this;
  }

  public Exo51 foo(char c) {
    x.insert(0, c); // insertion au début
    return this;
  }

  public String toString() {
    return "[" + x + "]";
  }
}

public class TestExo51 {

  public static void main(String[] args) {
    String s = "ABCD";
    Exo51 arf = new Exo51();
    System.out.println(arf);
    for (int i = 0; i < s.length(); i++) {
      System.out.println(arf.foo(s.charAt(i)));
    }
    Exo51 bla = new Exo51();
    for (int i = 0; i < s.length(); i++) {
      if (i % 2 == 0) {
        bla.foo(s.charAt(i));
        bla.plop(s.charAt(s.length() - i - 1));
      } else {
        bla.plop(s.charAt(i));
        bla.foo(s.charAt(s.length() - i - 1));
      }
      System.out.println(bla);
    }
  }

}
```

Indiquer l'affichage produit par le programme de gauche (méthode main)

Modifier la classe Exo51 pour rendre ses instances immuables

Donner les modifications minimales de la méthode main pour obtenir le même affichage avec les objets immuables

Exercice 6

Indiquer l'affichage produit par le programme de gauche (méthode main)

```
public class Exo61 {
  private int n;

  private double[] x;

  public Exo61(int p) {
    x = new double[p];
    n = 0;
  }

  public double f() {
    if (n > 0) {
      n = n - 1;
      return x[n];
    } else {
      // valeur spéciale qui s'affiche NaN
      return Double.NaN;
    }
  }

  public boolean g(double y) {
    if (n < x.length) {
      x[n] = y;
      n = n + 1;
      return true;
    } else {
      return false;
    }
  }
}

public class TestExo61 {

  public static void main(String[] args) {
    Exo61 t = new Exo61(3);
    for (int i = 0; i < 4; i++) {
      System.out.println(t.g(i * 2.0));
    }
    for (int i = 0; i < 4; i++) {
      System.out.println(t.f());
    }
    Exo61 u = new Exo61(2);
    Exo61 v = new Exo61(2);
    double[] x = { 2.0, -2.5, 4.0, 1.5 };
    for (double y : x) {
      if (!u.g(y)) {
        v.g(-y);
      }
    }
    for (int i = 0; i < 3; i++) {
      System.out.println(u.f() + " " + v.f());
    }
  }

}
```

Exercice 7

```
public class Exo71 {
    private À COMPLÉTER;

    private À COMPLÉTER;

    public Exo71(À COMPLÉTER) {
        À COMPLÉTER
    }

    @Override
    public String toString() {
        À COMPLÉTER
    }

    public double insert(double x) {
        double res = x;

        À COMPLÉTER

        return res;
    }
}

public class TestExo71 {

  public static void main(String[] args) {
    Exo71 u = new Exo71(0, 1);
    System.out.println(u);
    System.out.println(u.insert(0.5));
    System.out.println(u);
    System.out.println(u.insert(1.5));
    System.out.println(u);
    System.out.println(u.insert(-0.5));
    System.out.println(u);
    System.out.println(u.insert(1.0));
    System.out.println(u);
  }

}
```

Compléter la classe Exo71 afin que le programme (méthode main) affiche le texte suivant :

```
[0.0, 1.0]
0.5
[0.0, 1.0]
1.0
[0.0, 1.5]
0.0
[-0.5, 1.5]
1.0
[-0.5, 1.5]
```

Question difficile à traiter en dernier : modifier la classe Exo71 pour que ses instances soient immuables, sans changer significativement son comportement. Attention, il faut obligatoirement changer le type du résultat de la méthode `insert` et ajouter une autre méthode.

Exercice 8

Indiquer l'affichage produit par le programme de gauche (méthode main).

```
public class A81 {
  private int x;

  public A81() {
    x = 1;
  }

  public int f() {
    return x;
  }
}

public class B81 extends A81 {
  @Override
  public String toString() {
    return String.valueOf(f());
  }

  public int g() {
    return 3;
  }
}

public class C81 extends B81 {
  @Override
  public int f() {
    return 2;
  }
}

public class Test81 {

  public static void main(String[] args) {
    A81 x = new A81();
    System.out.println(x + " || " + x.f());
    B81 y = new B81();
    System.out.println(y + " || " + y.f() + " || " + y.g());
    C81 z = new C81();
    System.out.println(z + " || " + z.f() + " || " + z.g());
    x = z;
    System.out.println(x + " || " + x.f());
    y = z;
    System.out.println(y + " || " + y.f() + " || " + y.g());
    Object o = z;
    System.out.println(o);
  }

}
```

Si on ajoute la ligne suivante
`System.out.println(x.g());`
juste après la dernière ligne du programme (après `System.out.println(o);`**), le programme ne compile plus. Expliquer brièvement pourquoi.**

Exercice 9

Indiquer l'affichage produit par le programme de gauche (méthode main)

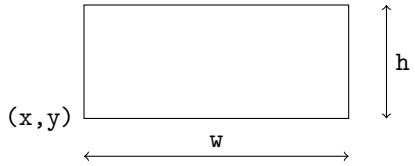
```
public class Exo101 {
  private int x;
  private int y;
  private int w;

  public Exo101(int w) {
    this.w = w;
    x = 0;
    y = 0;
  }

  public int foo() {
    x = x + 1;
    if (x > w) {
      x = 0;
      y = y + 1;
    }
    return x;
  }

  public int bar() {
    x = x - 1;
    int res = x;
    if (x < 0) {
      x = 0;
      y = y - 1;
    }
    return res;
  }

  @Override
  public String toString() {
    return x + " [" + y + "]";
  }
}

public class TestExo101 {

  public static void main(String[] args) {
    Exo101 robert = new Exo101(3);
    System.out.println(robert);
    System.out.println(robert.foo());
    System.out.println(robert);
    System.out.println(robert.bar());
    System.out.println(robert);

    Exo101 hank = new Exo101(2);
    Exo101 charlie = hank;
    for (int i = 0; i < 5; i++) {
      System.out.println(hank.foo());
      System.out.println(hank);
    }
    System.out.println(charlie);
  }

}
```

Exercice 10

Un objet de la classe `Rectangle` représente un rectangle (fermé) du plan. Le rectangle est décrit par son coin inférieur gauche (de coordonnées `x` et `y`), sa largeur `w` et sa hauteur `h` (cf la figure).

L'objectif de l'exercice est de compléter la classe ci-dessous de la façon suivante :

- la méthode `largeur` renvoie la largeur du rectangle ;
- la méthode `hauteur` renvoie la hauteur du rectangle ;
- la méthode `xMax` renvoie l'abscisse du coin supérieur droit du rectangle ;
- la méthode `yMax` renvoie l'ordonnée du coin supérieur droit du rectangle ;
- la méthode `contient` renvoie `true` si le point de coordonnées `x,y` est contenu dans le rectangle appelant (et `false` sinon) ;
- la méthode `toString` représente le rectangle sous la forme d'un produit d'intervalles `[a,b]x[c,d]` ;
- la méthode `intersecte` renvoie `true` si le rectangle appelant et le rectangle `that` ont une intersection non vide (et `false` sinon) ;
- la méthode `intersection` renvoie un nouveau rectangle représentant l'intersection entre le rectangle appelant et un rectangle paramètre (vous devez écrire cette méthode complètement).




```
public class Rectangle {
    private double x;
    private double y;
    private double w; // largeur
    private double h; // hauteur

    public Rectangle(double x, double y, double w, double h) {
        this.x = x;
        this.y = y;
        this.w = w;
        this.h = h;
    }

    public double largeur() {
        À COMPLÉTER
    }

    public double hauteur() {
        À COMPLÉTER
    }

    public double xMax() {
        À COMPLÉTER
    }

    public double yMax() {
        À COMPLÉTER
    }

    @Override
    public String toString() {
        À COMPLÉTER
    }

    public boolean contient(double u, double v) {
        À COMPLÉTER
    }

    public boolean intersecte(Rectangle that) {
        À COMPLÉTER
    }
}
```