Prénom :
Nom :

Pour chaque programme, vous devez indiquer l'affichage produit en répondant sur l'énoncé, à droite du programme. Dans certains exercices vous devez écrire des programmes ou des morceaux de programmes, ce que vous pouvez faire sur des feuilles annexes. **Soyez très précis dans vos réponses.**

Affichage produit

```
public class Exo11 {
  private boolean flip;

  private int foo;

  public Exo11(int foo) {
    this.foo = foo;
    flip = foo > 0;
  }

  public Exo11(int foo, boolean flip) {
    this.foo = foo;
    this.flip = flip;
  }

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

  public Exo11 flop() {
    return new Exo11(-foo);
  }

  public Exo11 comb(Exo11 that) {
    // && désigne le et logique
    return new Exo11(foo - that.foo, flip && that.flip);
  }
}

public class TestExo11 {

  public static void main(String[] args) {
    Exo11 x = new Exo11(2);
    System.out.println(x);
    Exo11 y = new Exo11(-2, true);
    System.out.println(y.toString());
    System.out.println(x.flop());
    Exo11 z = y;
    y = z.comb(y);
    System.out.println(z);
    System.out.println(y);
    Exo11 w = new Exo11(3, false);
    System.out.println(w.flop().flop());
    System.out.println((new Exo11(3)).comb(x).flop());
  }

}
```

```
public class Exo21 {
  private String texte;

  public Exo21() {
    texte = "";
  }

  public void bla() {
    if (texte.length() > 0) {
      // s.substring(1) renvoie une nouvelle chaîne
      // obtenue en supprimant le premier caractère de s
      // par ex. "Toto".substring(1) donne "oto"
      texte = texte.substring(1);
    }
  }

  public void foo() {
    texte = texte + String.valueOf(2 * texte.length());
  }

  public String toString() {
    return "<- " + texte + " ->";
  }
}

public class TestExo21 {

  public static void main(String[] args) {
    Exo21 a = new Exo21();
    System.out.println(a);
    a.foo();
    System.out.println(a);
    a.foo();
    System.out.println(a);
    Exo21 b = a;
    String s = b.toString();
    a.bla();
    System.out.println(a);
    System.out.println(b);
    System.out.println(s);
    Exo21 c = new Exo21();
    for (int i = 0; i < 6; i++) {
      c.foo();
    }
    System.out.println(c);
  }

}
```

Affichage produit

Modifiez la classe Exo21 pour rendre ses instances immuables en gardant le même comportement général. Vous pouvez vous contenter d'écrire les parties modifiées, sans recopier ce qui ne change pas.

```
public class Exo31 {
  private boolean[] tab;

  public Exo31(int k) {
    // les cases du tableau contiendront toutes false
    tab = new boolean[k];
  }

  public void foo(int l) {
    // ! est la négation logique
    tab[l] = !tab[l];
  }

  public int bar() {
    int n = 0;
    for (boolean b : tab) {
      if (b) {
        n++;
      }
    }
    return n;
  }

  public String toString() {
    StringBuilder sb = new StringBuilder(tab.length + 2);
    sb.append('[');
    for (int i = 0; i < tab.length; i++) {
      if (tab[i]) {
        sb.append("X");
      } else {
        sb.append(".");
      }
    }
    sb.append("]");
    return sb.toString();
  }
}

public class TestExo31 {

  public static void main(String[] args) {
    Exo31 x = new Exo31(4);
    System.out.println(x);
    x.foo(2);
    System.out.println(x);
    System.out.println(x.bar());
    Exo31 y = x;
    for (int i = 0; i < 4; i++) {
      x.foo(i);
      System.out.println(y.bar());
      if (i % 2 == 0) {
        y.foo(i);
      }
    }
    System.out.println(x);
    System.out.println(y);
  }

}
```

Affichage produit

Quelle(s) méthode(s) doit-on modifier pour rendre immuables les instances de la classe Exo31 ? (Donnez une justification brève.)

```
public class A1 {
  public int f(int x) {
    return 2 * x;
  }
}

public class B1 extends A1 {
  public String toString() {
    return "[B]";
  }
}

public class C1 extends B1 {
  @Override
  public int f(int x) {
    return x + 3;
  }
}

public class TestABC1 {

  public static void main(String[] args) {
    A1 a = new A1();
    B1 b = new B1();
    C1 c = new C1();
    System.out.println(a + " -- " + b + " -- " + c);
    System.out.println(a.f(2) + " -- " + b.f(4));
    System.out.println(c.f(6));
    Object oa = a;
    Object ob = b;
    Object oc = c;
    System.out.println(oa + " -- " + ob + " -- " + oc);
    a = b;
    b = c;
    System.out.println(a + " -- " + b);
    System.out.println(a.f(1) + " -- " + b.f(3));
    // System.out.println(oa.f(2));
  }

}
```

Affichage produit

La dernière ligne du programme ne compile pas. Expliquez brièvement pourquoi.

Un objet de la classe `Suite` ci-dessous représente des suites $(u_n)_{n\geq 0}$ définies par récurrence par la relation $u_n = au_{n-1} + b$, pour tout $n > 0$. L'objectif de l'exercice est de compléter la classe de la façon suivante :

1. Les coefficients a et b de la relation sont fixés par le constructeur et stockés dans les variables `a` et `b` de l'objet.
2. La méthode `valeur` doit renvoyer la valeur de numéro `n` de la suite définie par l'objet appelant ($u_{\texttt{n}}$) appliquant la relation de récurrence et en prenant $u_0 =$`u0`. On n'utilisera pas la méthode `valeurs` pour programmer `valeur` (attention au « s » !).
3. La méthode `pseudoProduit` doit renvoyer un nouvel objet `Suite` dont les coefficients sont le produit par le paramètre `x` des coefficients de l'objet appelant. Par exemple si la suite appelante est définie par $a = 2$ et $b = 1$, un appel à `pseudoProduit` avec comme paramètre 2 doit produire une suite de paramètres $a = 4$ et $b = 2$.
4. La méthode `pseudoSomme` doit renvoyer un nouvel objet `Suite` dont les coefficients a et b sont la somme des coefficients correspondants de la suite appelant et de la suite paramètres `that`.
5. La méthode `valeurs` doit renvoyer l'ensemble des `n+1` premiers termes de la suite représentée par l'objet appelant de $u_0 =$`u0` à $u_{\texttt{n}}$. On n'utilisera pas la méthode `valeur` pour programmer `valeurs` (attention au « s » !).

```
public class Suite {
    private double a;
    private double b;

    public Suite(double a, double b) {
        // À COMPLÉTER



    }

    public double valeur(double u0, int n) {
        // À COMPLÉTER









    }

    public Suite pseudoProduit(double x) {
        // À COMPLÉTER



    }

    public Suite pseudoSomme(Suite that) {
        // À COMPLÉTER



    }

    public double[] valeurs(double u0, int n) {
        // À COMPLÉTER









    }
}
```