(10) 1. **Identifier 10 erreurs dans le code suivant.** Pour chaque erreur, indiquez le numéro de ligne et fournir un court description. Vous recevrez un demi-point pour identifier chaque erreur et un demi-point la corriger.

```
1  class European {
2      private String name;
3      European(String name) {
4          this.name=name;
5      }
6  }
7
8  class Swiss extends European {
9      int goldInKg;
10     int bankAccounts=0;
11     Swiss(String name, double goldInKg) {
12
13         //1: this.name est privé.
14         //solution: Utiliser ``super(name)''
15         //ou changer ``String name'' d'être ``protected''
16         this.name = "Swiss citizen:" + name;
17
18         //2: On ne peut pas traiter un int comme un double
19         //solution: this.goldInKg = (int)goldInKg
20         this.goldInKg=goldInKg;
21
22     }
23
24     void openSwissBankAccount(int gold)
25           throws NotEnoughGoldException {
26         if (gold>10) {
27             bankAccounts++;
28             goldInKg+=gold;
29         } else {
30
31             //3:  ``throw'' pas ``throws''
32             //solution: throw (new NotEnoughGoldException());
33             throws(new NotEnoughGoldException());
34         }
35     }
36 }
```

Voir la page suivante.

```
37
38 class NotEnoughGoldException extends Exception {
39     NotEnoughGoldException() {
40         super("You need more gold to open a Swiss bank account!");
41     }
42 }
43
44
45 class TestEuropeans {
46
47     //4: erreur de syntaxe
48     //solution:  ``String [] args'' pas ``String [] args''
49
50     public static void main(Strings args [] ) {
51
52         // 5: la classe ``Swiss'' est plus specifique que European
53         // solution:   Swiss s = new Swiss(``Norman'')
54         //    ou European s = new Swiss(``Norman'')
55         //    ou European e = new European(``Norman'')
56         Swiss s = new European("Norman");
57
58         Swiss s2 = new Swiss("William Tell", 18.5);
59         European e = (European) s2;
60         s2.goldInKg++;
61
62         //6: goldInKg n'est pas dans European
63         //solution:  ((Swiss)e).goldInKg++;
64         e.goldInKg++;
65
66         //7: le ``throw'' n'est pas attrapé
67         //solution:
68         //try {
69         //       s2.openSwissBankAccount(50);
70         //   } catch (NotEnoughGoldException e) {
71         //        System.out.println(e);
72         //   }
73         s2.openSwissBankAccount(50);
74
75     }
76 }
```

Voir la page suivante.

```
77
78 class MathFunctions {
79     //8:  retourner ``int'', pas ``void''
80     //solution:    public static int factorial(x)
81     public static void factorial(int x)
82
83
84         //9: ``assignment operator''
85         //vs ``logical operator'' (= vs ==)
86         //solution:  if (x==0) {
87         if (x=0) {
88
89             return 1;
90         } else {
91             int val=1;
92
93             //10: l'implementation de factorial toujours calcule 0
94             //solution:   for (int i=x;i>0;i++)
95             for (int i=x;i>=0;i--) {
96                 //11: erreur de syntaxe: multiplication avec =*
97                 //solution: val*=i;
98                 val=*i;
99
100                //12) erreur de syntax: on manque un ``}''
101                //solution:   }
102
103            return val;
104        }
105    }
106 }
```

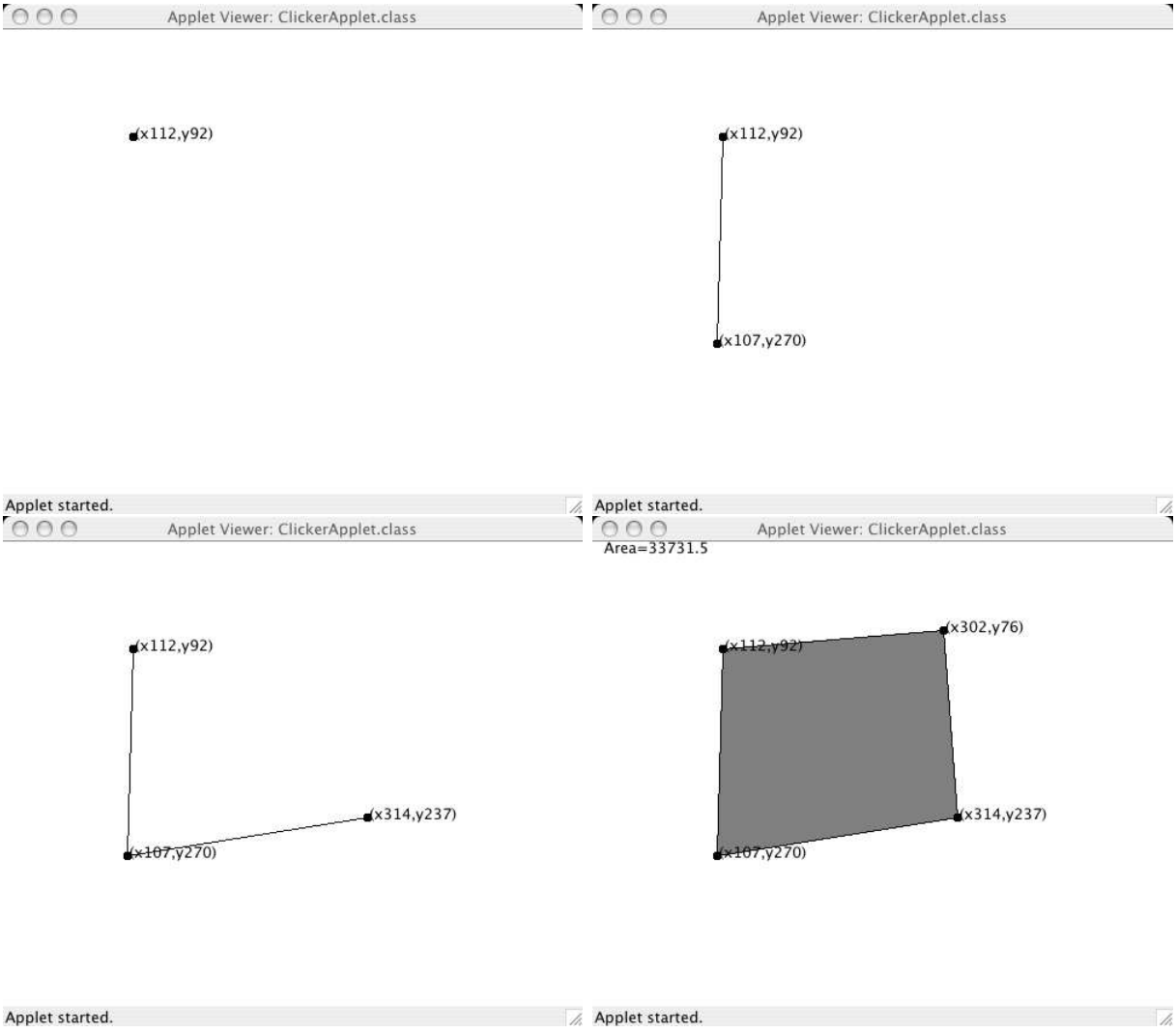

Figure 1: L'applet `ClickerApplet` après 1, 2, 3 et 4 cliques.

(30) 2. **Implémenter l'applet `PolygonApplet` qui répond aux cliques de la souris.** Pour la premier clique, l'applet montre un point comme un cercle avec coordonnées ou le clique est fait sur le panneau. Pour le deuxième et troisième cliques, l'applet montre les points et aussi les lignes qui connectent les points. Pour le quatrième clique, l'applet montre complètement le polygone avec deux lignes et le rempli avec le couleur gris. Finalement, l'applet montre l'aire du polygone à $x = 10, y = 10$. Le cinquième clique efface le polygone qui existe déjà et montre un point comme le premier clique.

```
//Solution:
import java.util.ArrayList;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.util.ListIterator;
public class PolygonApplet extends JApplet {
    public void init() {
        getContentPane().add(new PolygonPanel());
        setBackground(Color.white);
    }
}

class PolygonPanel extends JPanel implements MouseListener {
    //A  Variables d'instance et constructeur.
    int clickCounter;
    int [] xPoints;
    int [] yPoints;

    PolygonPanel() {
       clickCounter=0;
       xPoints = new int [4];
       yPoints = new int [4];
       addMouseListener(this);
       setBackground(Color.white);
    }
    protected void paintComponent(Graphics g) {
         super.paintComponent(g);

        //B  Montrer les points et lignes en couleur noi
        g.setColor(Color.black);
        for (int i=1;i<clickCounter;i++) {
            g.drawLine(xPoints[i-1],yPoints[i-1],xPoints[i],yPoints[i]);
        }
        for (int i=0;i<clickCounter;i++) {
            g.fillOval(xPoints[i]-3,yPoints[i]-3,6,6);
        }

        //C  Montrer les coordinees en couleur noir
        for (int i=0;i<clickCounter;i++) {
            g.fillOval(xPoints[i]-3,yPoints[i]-3,6,6);
            g.drawString("(x"+xPoints[i] +",y"+ yPoints[i]+")",
                         xPoints[i]+2,yPoints[i]+2);
        }
```

```
        //D  Au quatrième clique, montrer le polygon en couleur
             noir et le remplir en couleur gris
        if (clickCounter==4) {
           g.setColor(Color.gray);
           g.fillPolygon(xPoints, yPoints, 4);
           g.setColor(Color.black);
           g.drawPolygon(xPoints, yPoints, 4);
        }

        //E  Au quatrième clique, montrer l'aire à x=10,y=10
        if (clickCounter==4) {
           double area=0;
           for (int i=0;i<4;i++) {
               area+=xPoints[i]*yPoints[(i+1)%4]
                     - xPoints[(i+1)%4] * yPoints[i];
           }
           area/=2.0;
           g.drawString("Area="+Math.abs(area),10,10);
        }
     }

    //interface MouseListener
     public void mouseClicked(MouseEvent e) {

        //F  Processer les cliques
        if (clickCounter>=4) {
            clickCounter=0;
        }
        xPoints[clickCounter]=e.getX();
        yPoints[clickCounter]=e.getY();
        clickCounter++;


        repaint();
     }

     public void mouseEntered(MouseEvent e) {}
     public void mouseExited(MouseEvent e) {}
     public void mousePressed(MouseEvent e) {}
     public void mouseReleased(MouseEvent e) {}
}
```

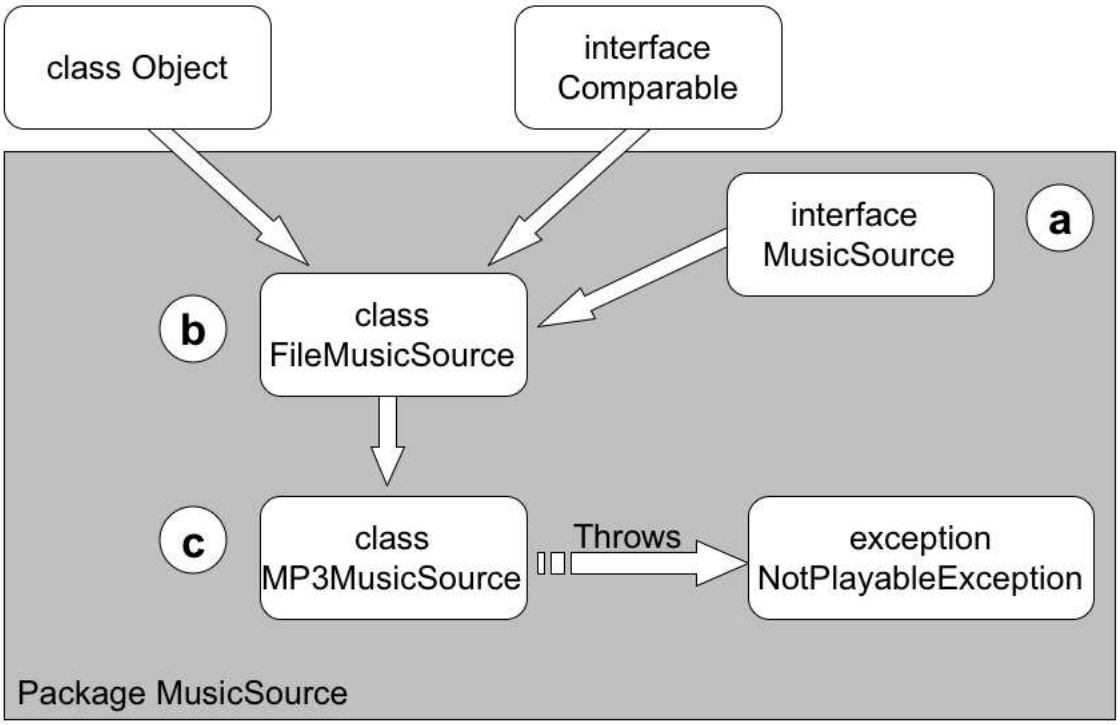

Figure 2: La hiérarchie pour `package MusicSource`.

(60) 3. **Implémenter quelques éléments de** `package MusicSource` (Figure 2). Il y a trois parties à cette question qui correspond au **a**, **b** et **c** dans le figure 2. On n'a pas besoin d'aucune connaissance de la musique ni de la technologie des fichiers de musique pour répondre à cette question.

(10) (a) Créer l'interface `MusicSource`, qui stock l'information pour une source de musique. Mettre l'interface dans le package MusicSource. Voici le constructeur et les méthodes que vous devez fournir dans l'interface:

Méthodes	
`getArtistName`	ne prend aucun argument; retourne le nom d'artiste
`getSongName`	ne prend aucun argument; retourne le nom de chanson
`getLength`	ne prend aucun argument; retourne la longueur de la chanson en secondes

```
//Solution
package MusicSource;
interface MusicSource {
    String getArtistName();
    String getSongName();
    int getSongLength();
    void setValues(String artist, String song, int len);
}
```

(30) (b) Créer la classe `FileMusicSource` qui implémente `interface MusicSource` et `interface Comparable`. Mettre la classe dans le package `MusicSource`. Vous aurez besoin quelques variables pour implémenter interface MusicSource. Toutes ces variables doit être accessible *seulement* par (1) toutes les classes dans `package MusicSource` et (2) toutes les classes qui `extend FileMusicSource`.

Voici l'interface `Comparable`:

```
interface Comparable {
    public int compareTo(Object o);
}
```

où `compareTo(Object o)` retourne:

- -1 si `this` > `otherObject`
- 1 si `this` < `otherObject`
- 0 si `this` == `otherObject`

Utiliser premièrement le nom d'artiste pour votre comparaison. Si les noms d'artiste sont égaux, utiliser le nom du chanson.

`class FileMusicSource` doit aussi offre les méthodes suivantes:

Méthods de la classe `FileMusicSource`	
`toString`	ne prend aucun argument; retourne le nom d'artiste, le nom de la chanson est la longueur de la chanson
`equals`	prend un `Object` comme argument; retourne `true` seulement si l'objet est de type `FileMusicSource` et les noms d'artiste et les noms de chansons sont égaux; autrement `false`.
`hashCode`	ne prend aucun argument; retourne un hashcode de type `int` qui est différent pour chaque instance unique de type `FileMusicSource`. Comme la méthode `equals`, nous ne considérons que le nom d'artiste et le nom de chanson pour générer le hashcode.

Rappel pour `class FileMusicSource`**:**

- La méthode `String Object.getClass()` retourne le nom de la classe d'un instance.
- La méthode `int Object.equals(Object obj)` est implémentée dans la classe `String()`.
- La méthode `int Object.hashCode()` est implémentée dans la classe `String()`.
- Pour mélanger plusieurs hashcodes c'est plus efficace de multiplier par un nombre premier. Par exemple si on a deux hashcodes, `int h1` et `int h2`:
  `int hash = h1 * 31 + h2;`

```
//Solution
package MusicSource;
class FileMusicSource extends Object implements
                      Comparable, MusicSource {
    protected String artistName;
    protected String songName;
    protected int songLength;

    FileMusicSource() {
        artistName=new String("");
        songName=new String("");
        songLength=0;
    }

    FileMusicSource(String artistName, String songName,
                    int songLength) {
        this.artistName=artistName;
        this.songName=songName;
        this.songLength=songLength;
    }

    //interface MusicSource
    public String getArtistName() {
        return artistName;
    }

    public String getSongName() {
        return songName;
    }

    public int getSongLength() {
        return songLength;
    }

    public void setValues(String artistName,
                          String songName, int songLength) {
        this.artistName=artistName;
        this.songName=songName;
        this.songLength=songLength;
    }
```

Voir la page suivante.

```
    //interface Comparable
    public int compareTo(Object o) {
        FileMusicSource f = ((FileMusicSource)o);
        int val = songName.compareTo(f.getSongName());
        if (val==0) {
            val =  artistName.compareTo(f.getArtistName());
        }
        return val;
    }

    //toString
    public String toString() {
        return artistName + " " + songName + " " + songLength;
    }

    //equals
    public boolean equals(Object o) {
        if (!(o instanceof FileMusicSource)) {
        //aussi possible!
        //if (!(o.getClass().equals("class FileMusicSource"))) {
            return false;
        }
        FileMusicSource f = ((FileMusicSource)o);
        return (f.getArtistName().equals(songName)
                && f.getSongName().equals(songName));
    }

    //hashCode
    public int hashCode() {
        return artistName.hashCode() * 31 + songName.hashCode();
    }
}
```

(20) (c) Implémenter `class MP3MusicSource` qui hérite de `class FileMusicSource`. On utilise cette classe pour manipuler des fichiers de type "MP3". MP3 est un format compact pour stocker la musique, mais les détails ne sont pas importants pour cette question.

On suppose l'existence d'une classe `MP3File` qui offre quelques méthodes. Vous n'avez pas besoin de fournir ni d'inclure (avec `import`) cette classe! Voici les méthodes offert par cette classe:

```
class MP3File {
    MP3File(String fileName) //constructeur
    String artist()          //nom d'artiste stocké dans le fichier
    String song()            //nom de la chanson stocké dans le fichier
    int timeInMS()           //la longueur du fichier en millisecondes
                             //(1000 milliseconds = 1 sec)
}
```

Le constructeur `MP3File(String fileName)` lance plusieur types d'exception. Toutes ces exceptions héritent de type `MP3FileException`. Vous n'avez besoin d'écrire `class MP3FileException` ou ses sous-classes, mais vous devez les attraper.

Voici la structure de `Class MP3MusicSource` que vous devez implémenter:

Variables de la classe `MP3MusicSource`	
fileName	un `String` qui contient le nom d'un fichier. Cette variable doit être accessible seulement de la classe `MP3MusicSource`
fileMp3	un `MP3File` qui est le fichier. Cette variable doit être accessible seulement de la classe `MP3MusicSource`
Constructeur de la classe `MP3MusicSource`	
MP3MusicSource	prends comme argument un `String` qui contient le nom d'un fichier. Dans le constructeur stocker le `String` dans `fileName` et l'utiliser pour ouvrir le `MP3File fileMp3`. Si votre initialisation de `fileMp3` génère une exception, vous devez lancer une exception de type `NotPlayableException`. Après vous créez `fileMp3`, utilisez les méthodes de `class MP3File` pour initialiser les variables dans votre super-classe.
Méthodes de la classe `MP3MusicSource`	
`toString`	ne prend aucun argument; retourne le nom de fichier et tout l'information fournit par `class FileMusicSource.toString()`.

Rappel pour `classe Mp3MusicSource`**:**
Voici l'implémentation de `NotPlayableException`:

```
public class NotPlayableException extends Exception {
    NotPlayableException(String fn) {
        super(fn + "n'est pas jouable");
    }
}
```

```
class MP3MusicSource extends FileMusicSource {
    private String fileName;
    private MP3File fileMp3;

    MP3MusicSource(String fileName) throws NotPlayableException {
        this.fileName=fileName;
        try {
            fileMp3 = new MP3File(fileName);
        } catch (MP3FileException e) {
            throw (new NotPlayableException(fileName));
        }
        this.artistName=fileMp3.artist();
        this.songName=fileMp3.song();
        this.songLength=fileMp3.timeInMs() / 1000;
    }

    public String toString() {
        return fileMp3 + " " + super.toString();
    }
}
```