High School Of Technology Essaouira

MARRAKECH

جامعة القاضي عياض

UNIVERSITÉ CADI AYYAD

Report
Exceptions - Graphic interface - Collections.

Realized by :

Mr. Bella Abdelouahab

Supervised By :

Pr. Fahd Karami

27 février 2025

Table des matières

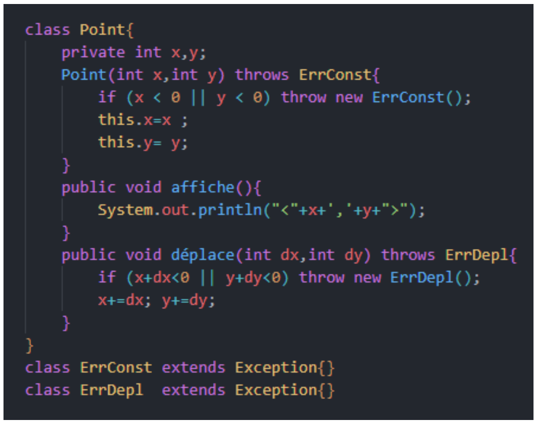
Table des figures

0.1 Course Examples

Creating a point using a Java class will be the first example we give.

— If the coordinates supplied have negative values, the class raises an error during initialization.

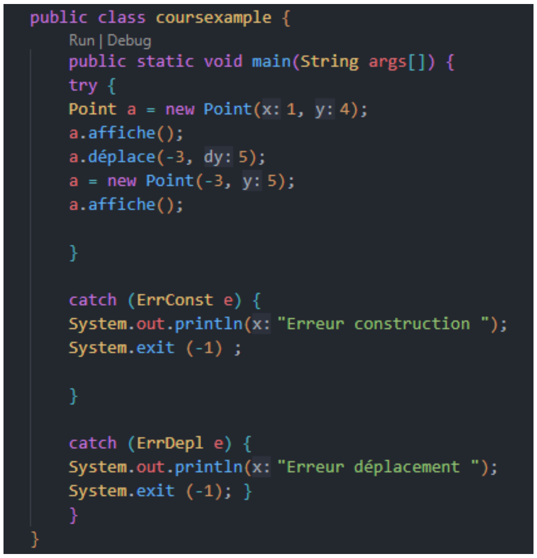
— If we attempt to relocate the point to an improper location, the class will likewise produce an error.



```java
class Point{
    private int x,y;
    Point(int x,int y) throws ErrConst{
        if (x < 0 || y < 0) throw new ErrConst();
        this.x=x ;
        this.y= y;
    }
    public void affiche(){
        System.out.println("<"+x+','+y+">");
    }
    public void déplace(int dx,int dy) throws ErrDepl{
        if (x+dx<0 || y+dy<0) throw new ErrDepl();
        x+=dx; y+=dy;
    }
}
class ErrConst extends Exception{}
class ErrDepl  extends Exception{}
```

FIGURE 1 – Class Point with construction exception

The next step is to begin a point using the class point and test its functionality (affiche,deplace).



```java
public class coursexample {
    public static void main(String args[]) {
    try {
    Point a = new Point(x: 1, y: 4);
    a.affiche();
    a.déplace(-3, dy: 5);
    a = new Point(-3, y: 5);
    a.affiche();

    }

    catch (ErrConst e) {
    System.out.println(x: "Erreur construction ");
    System.exit (-1) ;

    }

    catch (ErrDepl e) {
    System.out.println(x: "Erreur déplacement ");
    System.exit (-1); }
    }
}
```

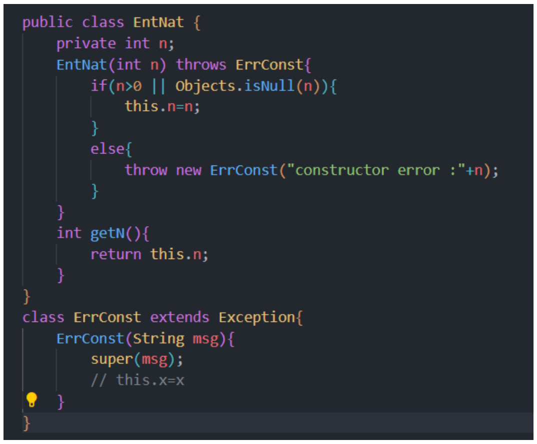
FIGURE 2 – Test class (Point) functions

0.2 Exercise on exception

0.2.1 Creation of an exception class

An exception is a circumstance that arises while a program is being executed that prevents the instructions from proceeding normally.

here is an example



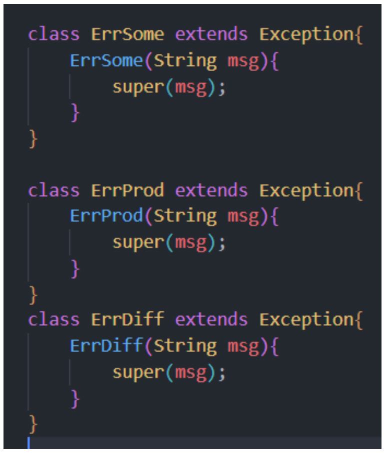
```java
public class EntNat {
    private int n;
    EntNat(int n) throws ErrConst{
        if(n>0 || Objects.isNull(n)){
            this.n=n;
        }
        else{
            throw new ErrConst("constructor error :"+n);
        }
    }
    int getN(){
        return this.n;
    }
}
class ErrConst extends Exception{
    ErrConst(String msg){
        super(msg);
        // this.x=x
    }
}
```

FIGURE 3 – Class to instantiate positive number

0.2.2 Using custom exception class to manage exceptions

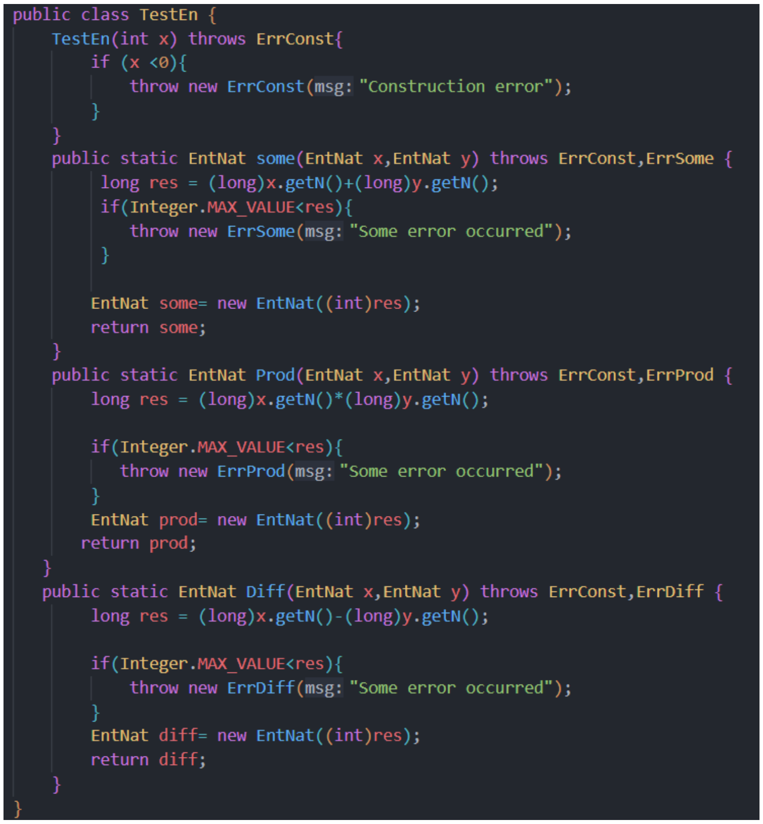
Using the custom exception, we may have our own exception and message. Here, the constructor of the superclass, the Exception class, has been given a string that can be obtained using the getMessage() function on the object we produced.

- We will utilize the EntNat class from before to develop a class that does positive number calculations.
- Additionally, if one of the calculations exceeds the maximum value for an integer, we may add a new exception.



```java
class ErrSome extends Exception{
    ErrSome(String msg){
        super(msg);
    }
}

class ErrProd extends Exception{
    ErrProd(String msg){
        super(msg);
    }
}
class ErrDiff extends Exception{
    ErrDiff(String msg){
        super(msg);
    }
}
```

FIGURE 4 – Custom exceptions that may occur



```
public class TestEn {
    TestEn(int x) throws ErrConst{
        if (x <0){
            throw new ErrConst(msg: "Construction error");
        }
    }
    public static EntNat some(EntNat x,EntNat y) throws ErrConst,ErrSome {
         long res = (long)x.getN()+(long)y.getN();
         if(Integer.MAX_VALUE<res){
            throw new ErrSome(msg: "Some error occurred");
         }

        EntNat some= new EntNat((int)res);
        return some;
    }
    public static EntNat Prod(EntNat x,EntNat y) throws ErrConst,ErrProd {
        long res = (long)x.getN()*(long)y.getN();

        if(Integer.MAX_VALUE<res){
           throw new ErrProd(msg: "Some error occurred");
        }
        EntNat prod= new EntNat((int)res);
       return prod;
   }
   public static EntNat Diff(EntNat x,EntNat y) throws ErrConst,ErrDiff {
        long res = (long)x.getN()-(long)y.getN();

        if(Integer.MAX_VALUE<res){
            throw new ErrDiff(msg: "Some error occurred");
        }
        EntNat diff= new EntNat((int)res);
        return diff;
    }
}
```

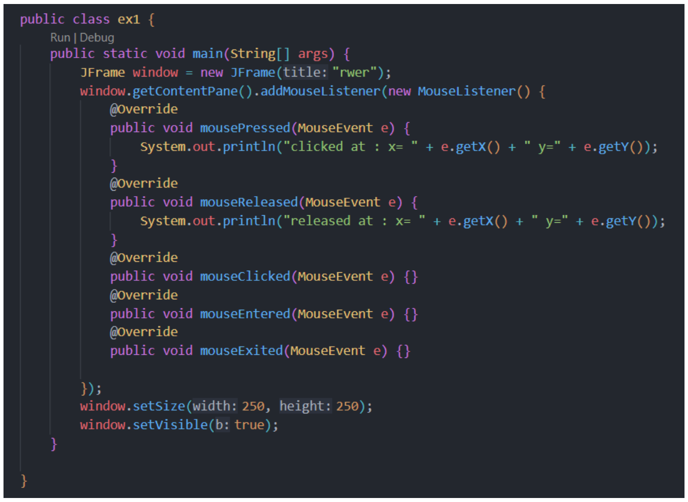
Figure 5 – Testing different Methods

0.3 Visual graphics with swing

The parts that follow will provide a variety of examples of visual graphics created using the Swing package.

0.3.1 A window with a mouse press and release detector

I decided to utilize the anonymous class Method here, which required me to override every other Method for the MouseListner class.



```java
public class ex1 {
    Run | Debug
    public static void main(String[] args) {
        JFrame window = new JFrame(title: "rwer");
        window.getContentPane().addMouseListener(new MouseListener() {
            @Override
            public void mousePressed(MouseEvent e) {
                System.out.println("clicked at : x= " + e.getX() + " y=" + e.getY());
            }
            @Override
            public void mouseReleased(MouseEvent e) {
                System.out.println("released at : x= " + e.getX() + " y=" + e.getY());
            }
            @Override
            public void mouseClicked(MouseEvent e) {}
            @Override
            public void mouseEntered(MouseEvent e) {}
            @Override
            public void mouseExited(MouseEvent e) {}

        });
        window.setSize(width: 250, height: 250);
        window.setVisible(b: true);
    }

}
```

Figure 6 – Mouse press and release detector

0.3.2 Generating buttons based on user input

We can create buttons in Java interfaces using the JButton class. As button content, you may use either text, an icon, or text with an icon next to it. One must first give the interface a name and import the javax.swing package in order to construct an interface in Java.

— we will use user input to generate the buttons
— the below figure represent the graphic result.

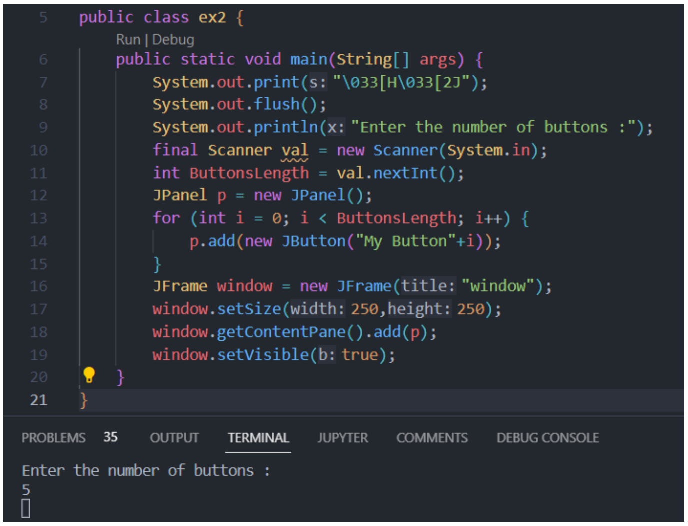
.

0.3.3 Presenting window layout

The BorderLayout class arranges the elements such that they may be placed in the east, west, north, south, and center areas. The linked constants NORTH, SOUTH, EAST, WEST, and CENTER are used to identify each component inside each area. Only one component can be stored in each zone at once.

0.3.4 Adding and deleting buttons based on user choise

In this example, we'll try to add and remove buttons in accordance with user preferences, thus we'll utilize two buttons : one to add buttons and the other to remove the



```
public class ex2 {
    Run | Debug
    public static void main(String[] args) {
        System.out.print(s: "\033[H\033[2J");
        System.out.flush();
        System.out.println(x: "Enter the number of buttons :");
        final Scanner val = new Scanner(System.in);
        int ButtonsLength = val.nextInt();
        JPanel p = new JPanel();
        for (int i = 0; i < ButtonsLength; i++) {
            p.add(new JButton("My Button"+i));
        }
        JFrame window = new JFrame(title: "window");
        window.setSize(width: 250,height: 250);
        window.getContentPane().add(p);
        window.setVisible(b: true);
    }
}
```

```
Enter the number of buttons :
5
```

FIGURE 7 – Generating button base on user input

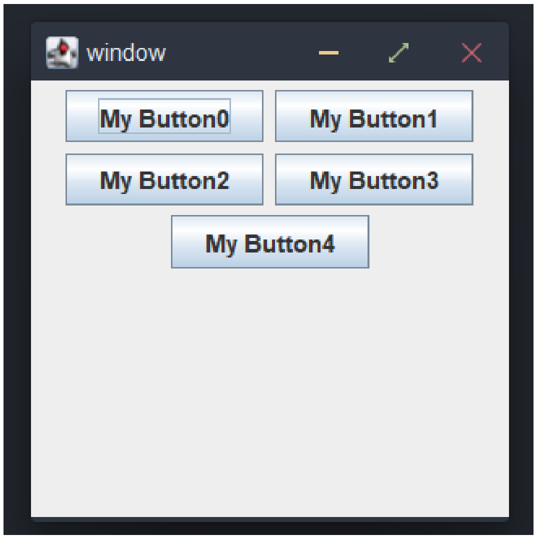
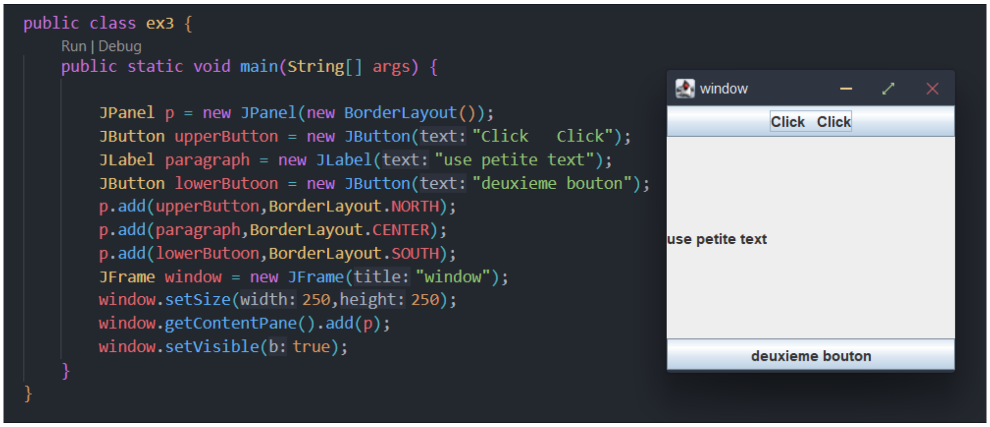


FIGURE 8 – Generated buttons UI

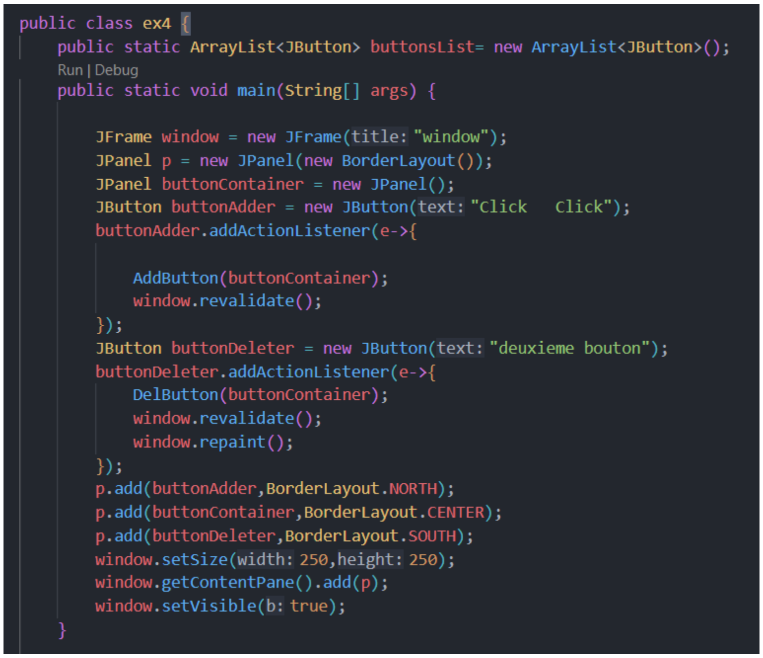


```
public class ex3 {
    Run | Debug
    public static void main(String[] args) {

        JPanel p = new JPanel(new BorderLayout());
        JButton upperButton = new JButton(text: "Click   Click");
        JLabel paragraph = new JLabel(text: "use petite text");
        JButton lowerButoon = new JButton(text: "deuxieme bouton");
        p.add(upperButton,BorderLayout.NORTH);
        p.add(paragraph,BorderLayout.CENTER);
        p.add(lowerButoon,BorderLayout.SOUTH);
        JFrame window = new JFrame(title: "window");
        window.setSize(width: 250,height: 250);
        window.getContentPane().add(p);
        window.setVisible(b: true);
    }
}
```

FIGURE 9 – Window layout

currently chosen button.

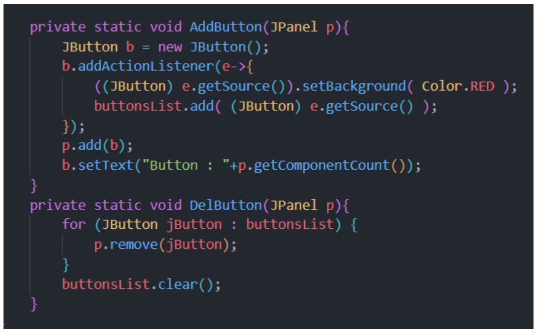
- The button to be deleted are the one with color red
- To select a button the user just need to press on it



```java
public class ex4 {
    public static ArrayList<JButton> buttonsList= new ArrayList<JButton>();
    public static void main(String[] args) {

        JFrame window = new JFrame(title: "window");
        JPanel p = new JPanel(new BorderLayout());
        JPanel buttonContainer = new JPanel();
        JButton buttonAdder = new JButton(text: "Click   Click");
        buttonAdder.addActionListener(e->{

            AddButton(buttonContainer);
            window.revalidate();
        });
        JButton buttonDeleter = new JButton(text: "deuxieme bouton");
        buttonDeleter.addActionListener(e->{
            DelButton(buttonContainer);
            window.revalidate();
            window.repaint();
        });
        p.add(buttonAdder,BorderLayout.NORTH);
        p.add(buttonContainer,BorderLayout.CENTER);
        p.add(buttonDeleter,BorderLayout.SOUTH);
        window.setSize(width: 250,height: 250);
        window.getContentPane().add(p);
        window.setVisible(b: true);
    }
```

FIGURE 10 – Add and remove buttons in accordance with user

- the function add and delete written as follow



```
private static void AddButton(JPanel p){
    JButton b = new JButton();
    b.addActionListener(e->{
        ((JButton) e.getSource()).setBackground( Color.RED );
        buttonsList.add( (JButton) e.getSource() );
    });
    p.add(b);
    b.setText("Button : "+p.getComponentCount());
}
private static void DelButton(JPanel p){
    for (JButton jButton : buttonsList) {
        p.remove(jButton);
    }
    buttonsList.clear();
}
```

FIGURE 11 – Add and delete functions

- the result of this program in the GUI presented in the below label.

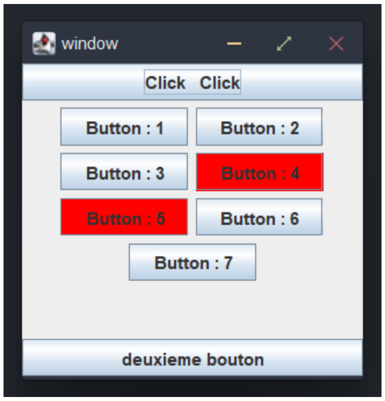
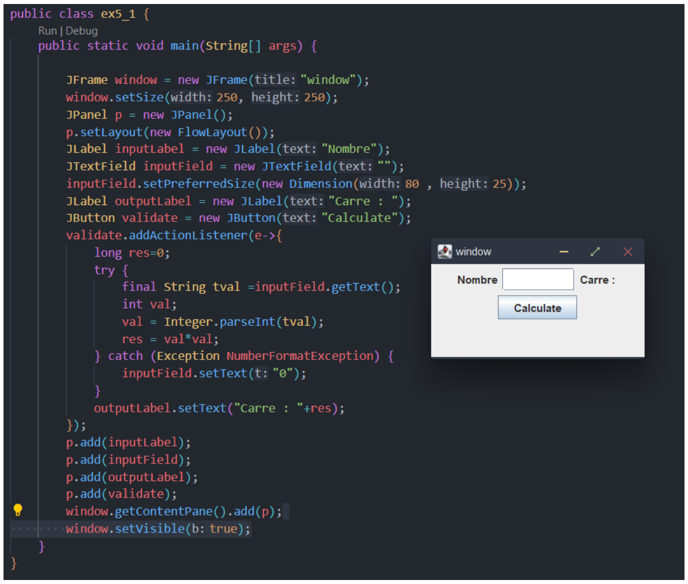


FIGURE 12 – the UI for add and delete functions

0.3.5 Calculating the square root of a number

To construct an input field for the user to enter the member, we will use the Jtext-Field component. We will compute that using two different techniques.

Method 1 : using calculate button

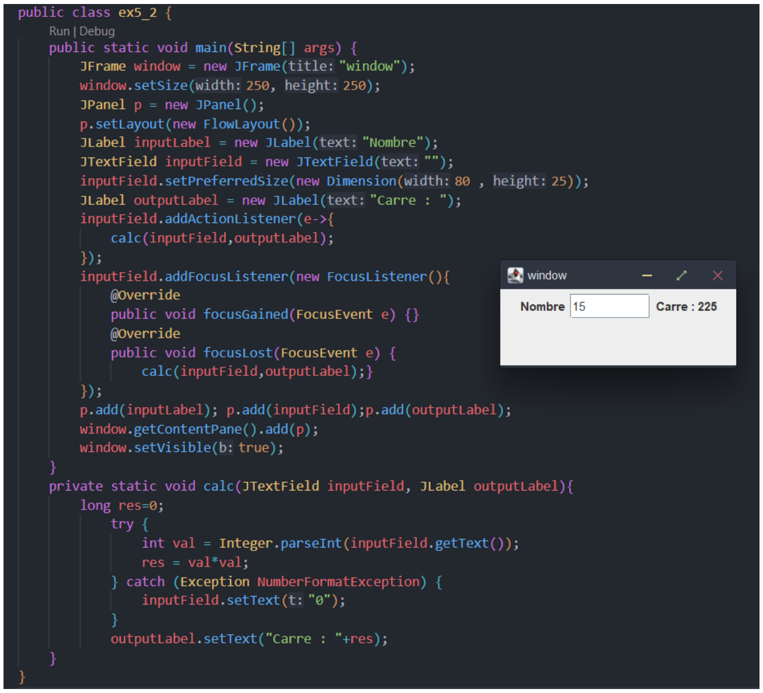


```java
public class ex5_1 {
    public static void main(String[] args) {

        JFrame window = new JFrame(title: "window");
        window.setSize(width: 250, height: 250);
        JPanel p = new JPanel();
        p.setLayout(new FlowLayout());
        JLabel inputLabel = new JLabel(text: "Nombre");
        JTextField inputField = new JTextField(text: "");
        inputField.setPreferredSize(new Dimension(width: 80 , height: 25));
        JLabel outputLabel = new JLabel(text: "Carre : ");
        JButton validate = new JButton(text: "Calculate");
        validate.addActionListener(e->{
            long res=0;
            try {
                final String tval =inputField.getText();
                int val;
                val = Integer.parseInt(tval);
                res = val*val;
            } catch (Exception NumberFormatException) {
                inputField.setText(t: "0");
            }
            outputLabel.setText("Carre : "+res);
        });
        p.add(inputLabel);
        p.add(inputField);
        p.add(outputLabel);
        p.add(validate);
        window.getContentPane().add(p);
        window.setVisible(b: true);
    }
}
```

Figure 13 – Calculating the square root

Method 2 : using key input and focus lost

Now that the calculate button has been removed, we'll utilize two functions : one for when the user touches the keyboard's enter key and another for when they lose focus on the input field.



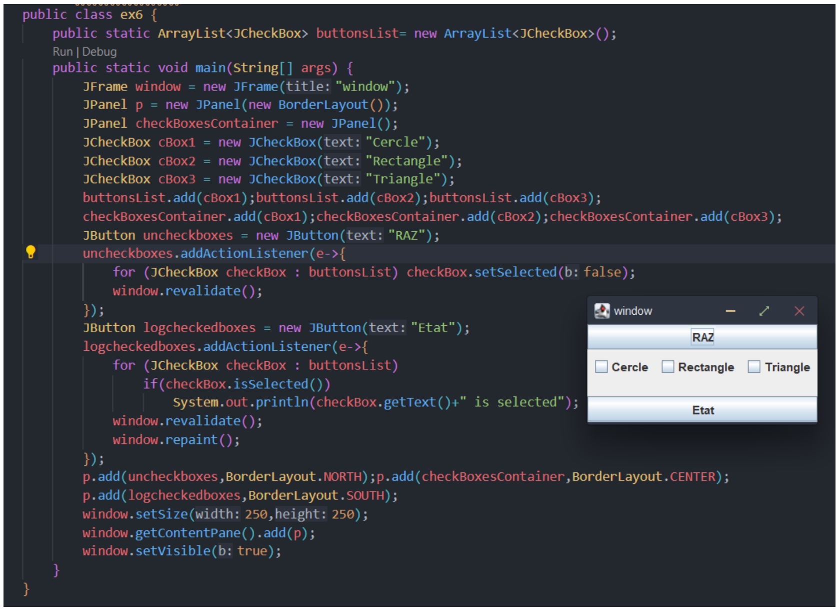
Nombre 15 Carre : 225

```java
public class ex5_2 {
    public static void main(String[] args) {
        JFrame window = new JFrame(title: "window");
        window.setSize(width: 250, height: 250);
        JPanel p = new JPanel();
        p.setLayout(new FlowLayout());
        JLabel inputLabel = new JLabel(text: "Nombre");
        JTextField inputField = new JTextField(text: "");
        inputField.setPreferredSize(new Dimension(width: 80 , height: 25));
        JLabel outputLabel = new JLabel(text: "Carre : ");
        inputField.addActionListener(e->{
            calc(inputField,outputLabel);
        });
        inputField.addFocusListener(new FocusListener(){
            @Override
            public void focusGained(FocusEvent e) {}
            @Override
            public void focusLost(FocusEvent e) {
                calc(inputField,outputLabel);}
        });
        p.add(inputLabel); p.add(inputField);p.add(outputLabel);
        window.getContentPane().add(p);
        window.setVisible(b: true);
    }
    private static void calc(JTextField inputField, JLabel outputLabel){
        long res=0;
            try {
                int val = Integer.parseInt(inputField.getText());
                res = val*val;
            } catch (Exception NumberFormatException) {
                inputField.setText(t: "0");
            }
            outputLabel.setText("Carre : "+res);
    }
}
```

Figure 14 – Calculating the square root / automated

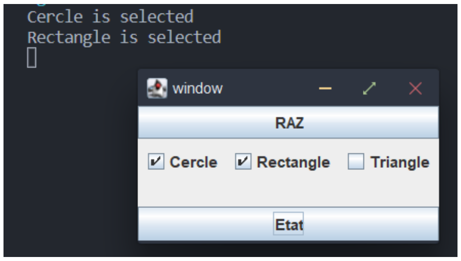
0.3.6 Implementing the JCheckBox component

now we will explore the checkbox



```java
public class ex6 {
    public static ArrayList<JCheckBox> buttonsList= new ArrayList<JCheckBox>();
    Run | Debug
    public static void main(String[] args) {
        JFrame window = new JFrame(title: "window");
        JPanel p = new JPanel(new BorderLayout());
        JPanel checkBoxesContainer = new JPanel();
        JCheckBox cBox1 = new JCheckBox(text: "Cercle");
        JCheckBox cBox2 = new JCheckBox(text: "Rectangle");
        JCheckBox cBox3 = new JCheckBox(text: "Triangle");
        buttonsList.add(cBox1);buttonsList.add(cBox2);buttonsList.add(cBox3);
        checkBoxesContainer.add(cBox1);checkBoxesContainer.add(cBox2);checkBoxesContainer.add(cBox3);
        JButton uncheckboxes = new JButton(text: "RAZ");
        uncheckboxes.addActionListener(e->{
            for (JCheckBox checkBox : buttonsList) checkBox.setSelected(b: false);
            window.revalidate();
        });
        JButton logcheckedboxes = new JButton(text: "Etat");
        logcheckedboxes.addActionListener(e->{
            for (JCheckBox checkBox : buttonsList)
                if(checkBox.isSelected())
                    System.out.println(checkBox.getText()+" is selected");
            window.revalidate();
            window.repaint();
        });
        p.add(uncheckboxes,BorderLayout.NORTH);p.add(checkBoxesContainer,BorderLayout.CENTER);
        p.add(logcheckedboxes,BorderLayout.SOUTH);
        window.setSize(width: 250,height: 250);
        window.getContentPane().add(p);
        window.setVisible(b: true);
    }
}
```

FIGURE 15 – Code for the checkbox

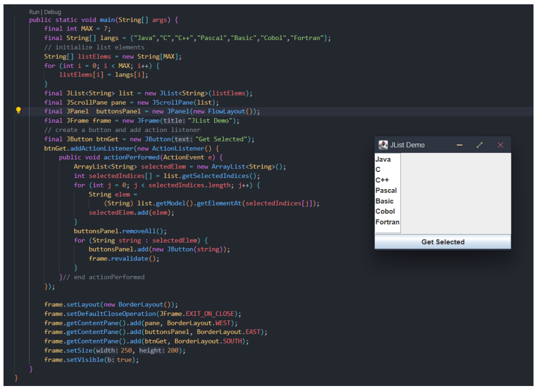


```
Cercle is selected
Rectangle is selected
```

FIGURE 16 – Results of figure 12

0.3.7 Implementing the JList component

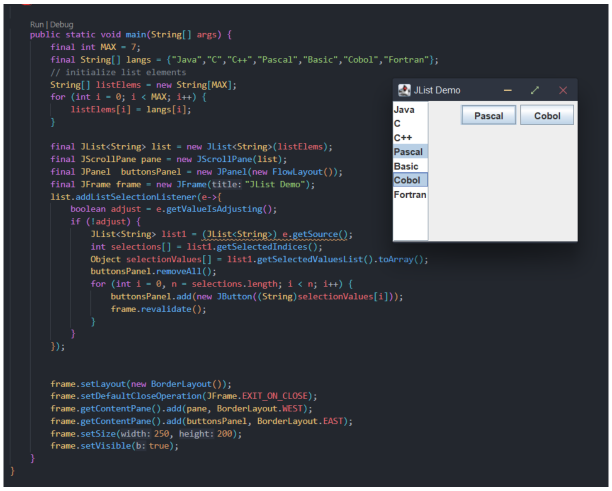
It's time to learn more about Jlist, a component that is similar to VBox in our list of elements. Two techniques or tools were applied to this example, the first of which was pressing the "OK" button.



```
Run | Debug
public static void main(String[] args) {
    final int MAX = 7;
    final String[] langs = {"Java","C","C++","Pascal","Basic","Cobol","Fortran"};
    // initialize list elements
    String[] listElems = new String[MAX];
    for (int i = 0; i < MAX; i++) {
        listElems[i] = langs[i];
    }
    final JList<String> list = new JList<String>(listElems);
    final JScrollPane pane = new JScrollPane(list);
    final JPanel  buttonsPanel = new JPanel(new FlowLayout());
    final JFrame frame = new JFrame(title: "JList Demo");
    // create a button and add action listener
    final JButton btnGet = new JButton(text: "Get Selected");
    btnGet.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            ArrayList<String> selectedElem = new ArrayList<String>();
            int selectedIndices[] = list.getSelectedIndices();
            for (int j = 0; j < selectedIndices.length; j++) {
                String elem =
                    (String) list.getModel().getElementAt(selectedIndices[j]);
                selectedElem.add(elem);
            }
            buttonsPanel.removeAll();
            for (String string : selectedElem) {
                buttonsPanel.add(new JButton(string));
                frame.revalidate();
            }
        }// end actionPerformed
    });

    frame.setLayout(new BorderLayout());
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.getContentPane().add(pane, BorderLayout.WEST);
    frame.getContentPane().add(buttonsPanel, BorderLayout.EAST);
    frame.getContentPane().add(btnGet, BorderLayout.SOUTH);
    frame.setSize(width: 250, height: 200);
    frame.setVisible(b: true);
}
}
```

Figure 17 – Code for JList component

simply by clicking on the element we wish to be added in the second one :



```
Run | Debug
public static void main(String[] args) {
    final int MAX = 7;
    final String[] langs = {"Java","C","C++","Pascal","Basic","Cobol","Fortran"};
    // initialize list elements
    String[] listElems = new String[MAX];
    for (int i = 0; i < MAX; i++) {
        listElems[i] = langs[i];
    }

    final JList<String> list = new JList<String>(listElems);
    final JScrollPane pane = new JScrollPane(list);
    final JPanel  buttonsPanel = new JPanel(new FlowLayout());
    final JFrame frame = new JFrame(title: "JList Demo");
    list.addListSelectionListener(e->{
        boolean adjust = e.getValueIsAdjusting();
        if (!adjust) {
            JList<String> list1 = (JList<String>) e.getSource();
            int selections[] = list1.getSelectedIndices();
            Object selectionValues[] = list1.getSelectedValuesList().toArray();
            buttonsPanel.removeAll();
            for (int i = 0, n = selections.length; i < n; i++) {
                buttonsPanel.add(new JButton((String)selectionValues[i]));
                frame.revalidate();
            }
        }
    });


    frame.setLayout(new BorderLayout());
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.getContentPane().add(pane, BorderLayout.WEST);
    frame.getContentPane().add(buttonsPanel, BorderLayout.EAST);
    frame.setSize(width: 250, height: 200);
    frame.setVisible(b: true);
}
}
```

Figure 18 – JList component - automate rendering

0.3.8 Testing mouse event to change window color

We've implemented a simple mouse listener where the background color changes whenever the mouse enters the component (a JPanel)

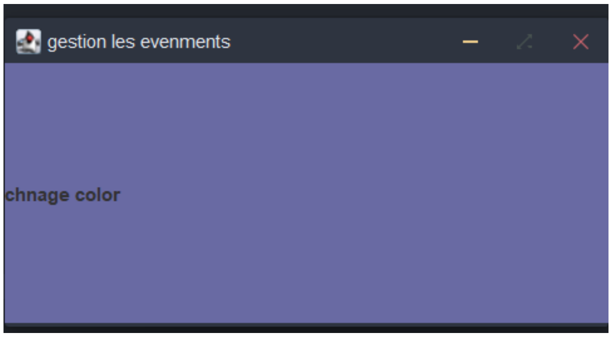
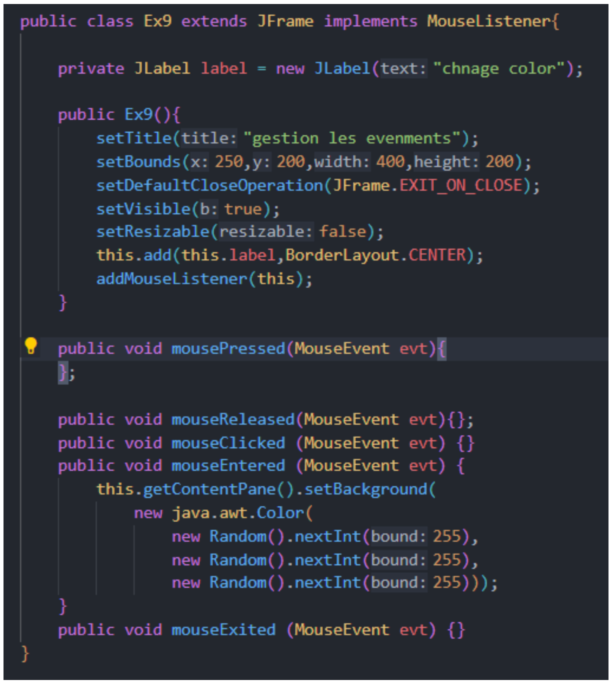


FIGURE 19 – Window changing color

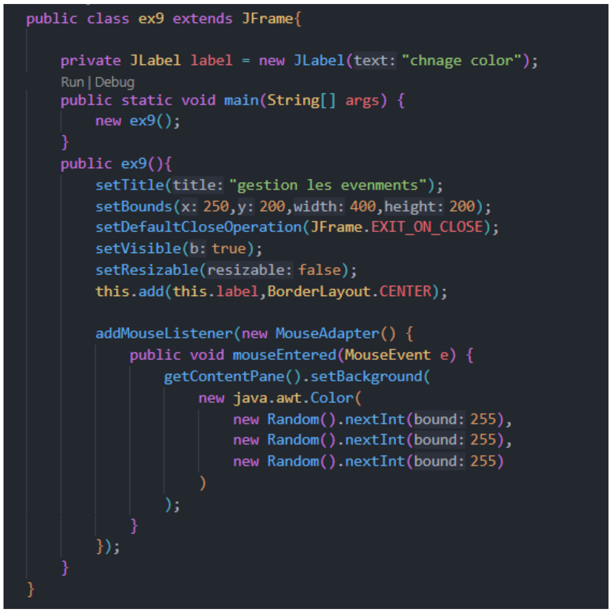
Method 1 : Integrated class



```
public class Ex9 extends JFrame implements MouseListener{

    private JLabel label = new JLabel(text: "chnage color");

    public Ex9(){
        setTitle(title: "gestion les evenments");
        setBounds(x: 250,y: 200,width: 400,height: 200);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setVisible(b: true);
        setResizable(resizable: false);
        this.add(this.label,BorderLayout.CENTER);
        addMouseListener(this);
    }

    public void mousePressed(MouseEvent evt){
    };

    public void mouseReleased(MouseEvent evt){};
    public void mouseClicked (MouseEvent evt) {}
    public void mouseEntered (MouseEvent evt) {
        this.getContentPane().setBackground(
            new java.awt.Color(
                new Random().nextInt(bound: 255),
                new Random().nextInt(bound: 255),
                new Random().nextInt(bound: 255)));
    }
    public void mouseExited (MouseEvent evt) {}
}
```

FIGURE 20 – Window changing color - integrated class

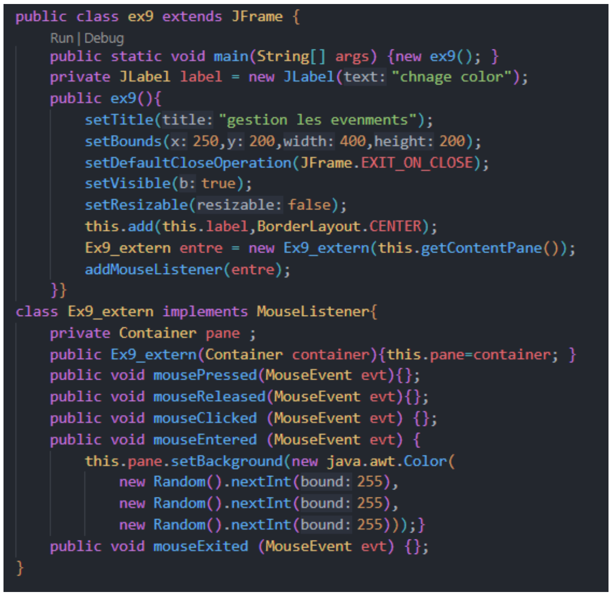
Method 2 : Anonymous class



```java
public class ex9 extends JFrame{

    private JLabel label = new JLabel(text: "chnage color");
    Run | Debug
    public static void main(String[] args) {
        new ex9();
    }
    public ex9(){
        setTitle(title: "gestion les evenments");
        setBounds(x: 250,y: 200,width: 400,height: 200);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setVisible(b: true);
        setResizable(resizable: false);
        this.add(this.label,BorderLayout.CENTER);

        addMouseListener(new MouseAdapter() {
            public void mouseEntered(MouseEvent e) {
                getContentPane().setBackground(
                    new java.awt.Color(
                        new Random().nextInt(bound: 255),
                        new Random().nextInt(bound: 255),
                        new Random().nextInt(bound: 255)
                    )
                );
            }
        });
    }
}
```

Figure 21 – Window changing color - anonymous class

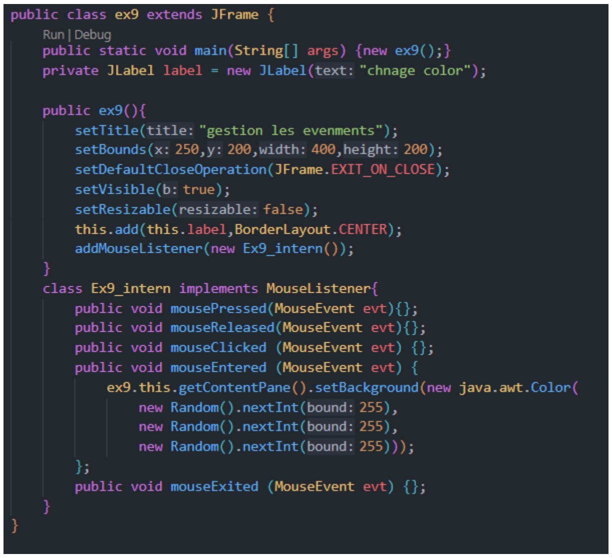
Method 3 : Outer class



```java
public class ex9 extends JFrame {
    Run | Debug
    public static void main(String[] args) {new ex9(); }
    private JLabel label = new JLabel(text: "chnage color");
    public ex9(){
        setTitle(title: "gestion les evenments");
        setBounds(x: 250,y: 200,width: 400,height: 200);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setVisible(b: true);
        setResizable(resizable: false);
        this.add(this.label,BorderLayout.CENTER);
        Ex9_extern entre = new Ex9_extern(this.getContentPane());
        addMouseListener(entre);
    }}
class Ex9_extern implements MouseListener{
    private Container pane ;
    public Ex9_extern(Container container){this.pane=container; }
    public void mousePressed(MouseEvent evt){};
    public void mouseReleased(MouseEvent evt){};
    public void mouseClicked (MouseEvent evt) {};
    public void mouseEntered (MouseEvent evt) {
        this.pane.setBackground(new java.awt.Color(
            new Random().nextInt(bound: 255),
            new Random().nextInt(bound: 255),
            new Random().nextInt(bound: 255)));}
    public void mouseExited (MouseEvent evt) {};
}
```

Figure 22 – Window changing color - outer class

Method 4 : Internal class

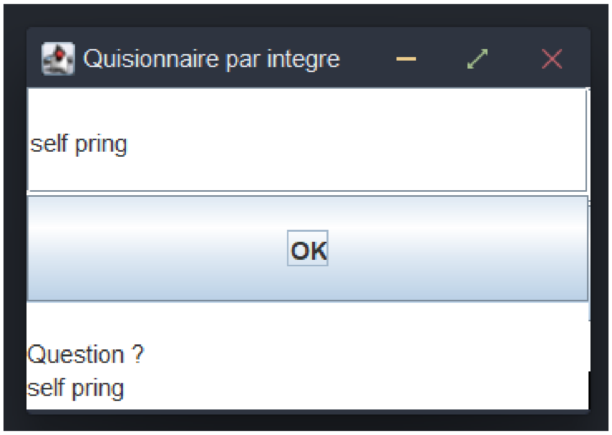


```java
public class ex9 extends JFrame {
    Run | Debug
    public static void main(String[] args) {new ex9();}
    private JLabel label = new JLabel(text: "chnage color");

    public ex9(){
        setTitle(title: "gestion les evenments");
        setBounds(x: 250,y: 200,width: 400,height: 200);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setVisible(b: true);
        setResizable(resizable: false);
        this.add(this.label,BorderLayout.CENTER);
        addMouseListener(new Ex9_intern());
    }
    class Ex9_intern implements MouseListener{
        public void mousePressed(MouseEvent evt){};
        public void mouseReleased(MouseEvent evt){};
        public void mouseClicked (MouseEvent evt) {};
        public void mouseEntered (MouseEvent evt) {
            ex9.this.getContentPane().setBackground(new java.awt.Color(
                new Random().nextInt(bound: 255),
                new Random().nextInt(bound: 255),
                new Random().nextInt(bound: 255)));
        };
        public void mouseExited (MouseEvent evt) {};
    }
}
```

FIGURE 23 – Window changing color - internal class

0.3.9 Displaying user questions using 4 different Methods

The difference is that in this instance, in order to compose a question and put it to a panel, we must develop an interface.

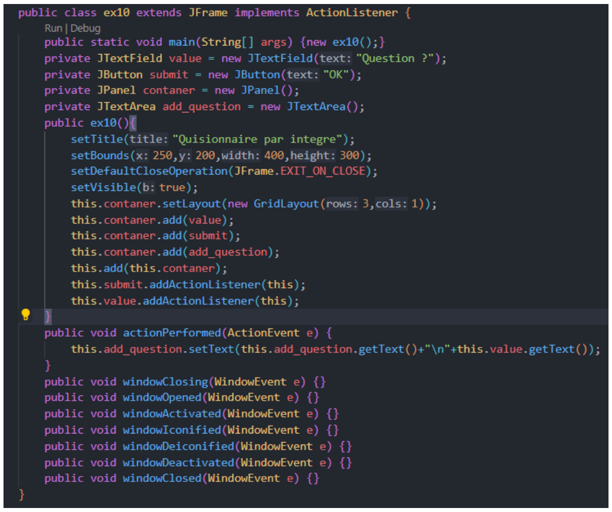


self pring

Question ?
self pring

FIGURE 24 – User questions UI

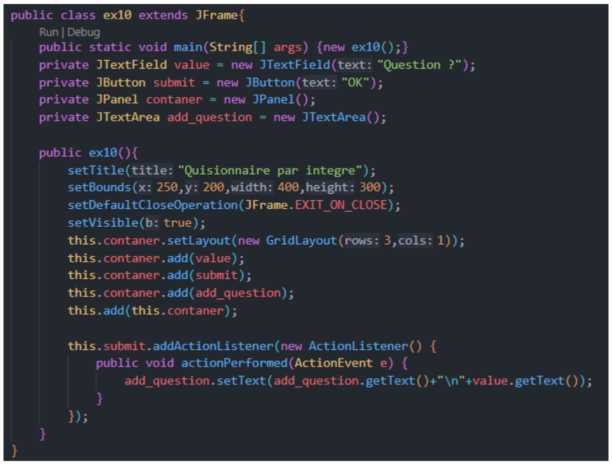
Method 1 : Integrated class



```java
public class ex10 extends JFrame implements ActionListener {
    Run | Debug
    public static void main(String[] args) {new ex10();}
    private JTextField value = new JTextField(text: "Question ?");
    private JButton submit = new JButton(text: "OK");
    private JPanel contaner = new JPanel();
    private JTextArea add_question = new JTextArea();
    public ex10(){
        setTitle(title: "Quisionnaire par integre");
        setBounds(x: 250,y: 200,width: 400,height: 300);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setVisible(b: true);
        this.contaner.setLayout(new GridLayout(rows: 3,cols: 1));
        this.contaner.add(value);
        this.contaner.add(submit);
        this.contaner.add(add_question);
        this.add(this.contaner);
        this.submit.addActionListener(this);
        this.value.addActionListener(this);
    }
    public void actionPerformed(ActionEvent e) {
        this.add_question.setText(this.add_question.getText()+"\n"+this.value.getText());
    }
    public void windowClosing(WindowEvent e) {}
    public void windowOpened(WindowEvent e) {}
    public void windowActivated(WindowEvent e) {}
    public void windowIconified(WindowEvent e) {}
    public void windowDeiconified(WindowEvent e) {}
    public void windowDeactivated(WindowEvent e) {}
    public void windowClosed(WindowEvent e) {}
}
```

Figure 25 – User questions - Integrated class

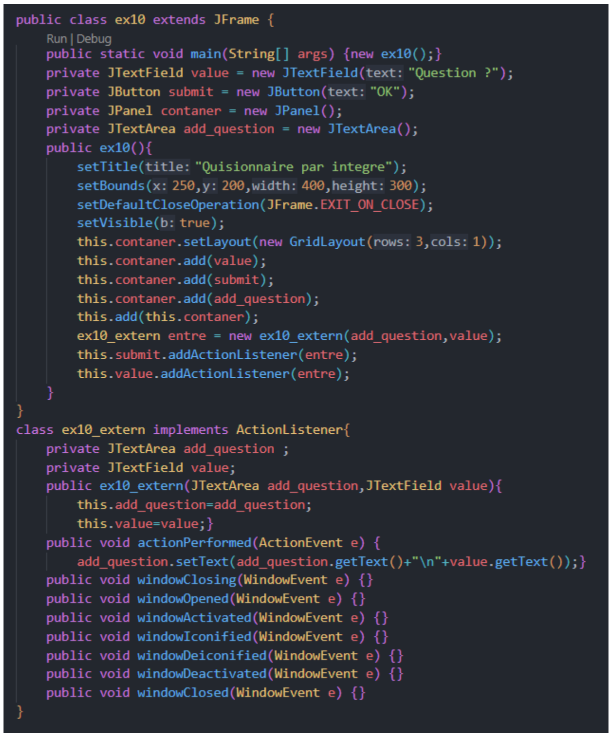
Method 2 : Anonymous class



```java
public class ex10 extends JFrame{
    Run | Debug
    public static void main(String[] args) {new ex10();}
    private JTextField value = new JTextField(text: "Question ?");
    private JButton submit = new JButton(text: "OK");
    private JPanel contaner = new JPanel();
    private JTextArea add_question = new JTextArea();

    public ex10(){
        setTitle(title: "Quisionnaire par integre");
        setBounds(x: 250,y: 200,width: 400,height: 300);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setVisible(b: true);
        this.contaner.setLayout(new GridLayout(rows: 3,cols: 1));
        this.contaner.add(value);
        this.contaner.add(submit);
        this.contaner.add(add_question);
        this.add(this.contaner);

        this.submit.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                add_question.setText(add_question.getText()+"\n"+value.getText());
            }
        });
    }
}
```

Figure 26 – User questions - Anonymous class

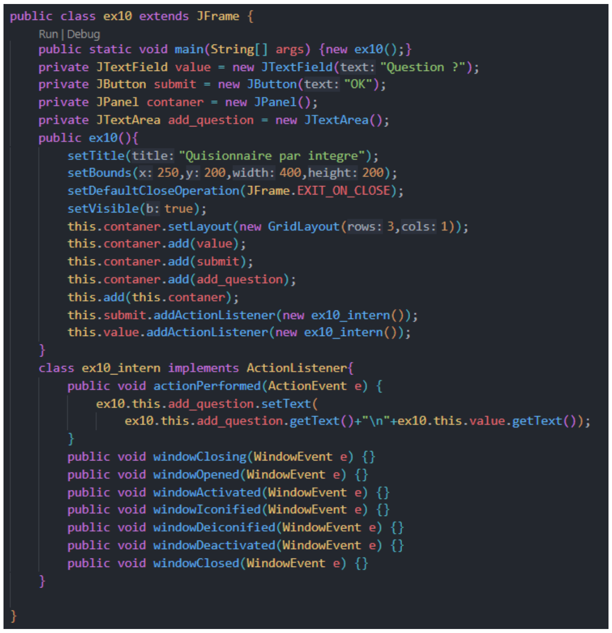
Method 3 : Outer class



```java
public class ex10 extends JFrame {
    public static void main(String[] args) {new ex10();}
    private JTextField value = new JTextField(text: "Question ?");
    private JButton submit = new JButton(text: "OK");
    private JPanel contaner = new JPanel();
    private JTextArea add_question = new JTextArea();
    public ex10(){
        setTitle(title: "Quisionnaire par integre");
        setBounds(x: 250,y: 200,width: 400,height: 300);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setVisible(b: true);
        this.contaner.setLayout(new GridLayout(rows: 3,cols: 1));
        this.contaner.add(value);
        this.contaner.add(submit);
        this.contaner.add(add_question);
        this.add(this.contaner);
        ex10_extern entre = new ex10_extern(add_question,value);
        this.submit.addActionListener(entre);
        this.value.addActionListener(entre);
    }
}
class ex10_extern implements ActionListener{
    private JTextArea add_question ;
    private JTextField value;
    public ex10_extern(JTextArea add_question,JTextField value){
        this.add_question=add_question;
        this.value=value;}
    public void actionPerformed(ActionEvent e) {
        add_question.setText(add_question.getText()+"\n"+value.getText());}
    public void windowClosing(WindowEvent e) {}
    public void windowOpened(WindowEvent e) {}
    public void windowActivated(WindowEvent e) {}
    public void windowIconified(WindowEvent e) {}
    public void windowDeiconified(WindowEvent e) {}
    public void windowDeactivated(WindowEvent e) {}
    public void windowClosed(WindowEvent e) {}
}
```

FIGURE 27 – User questions - Outer class

Method 4 : Internal class



```java
public class ex10 extends JFrame {
    public static void main(String[] args) {new ex10();}
    private JTextField value = new JTextField(text: "Question ?");
    private JButton submit = new JButton(text: "OK");
    private JPanel contaner = new JPanel();
    private JTextArea add_question = new JTextArea();
    public ex10(){
        setTitle(title: "Quisionnaire par integre");
        setBounds(x: 250,y: 200,width: 400,height: 200);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setVisible(b: true);
        this.contaner.setLayout(new GridLayout(rows: 3,cols: 1));
        this.contaner.add(value);
        this.contaner.add(submit);
        this.contaner.add(add_question);
        this.add(this.contaner);
        this.submit.addActionListener(new ex10_intern());
        this.value.addActionListener(new ex10_intern());
    }
    class ex10_intern implements ActionListener{
        public void actionPerformed(ActionEvent e) {
            ex10.this.add_question.setText(
                ex10.this.add_question.getText()+"\n"+ex10.this.value.getText());
        }
        public void windowClosing(WindowEvent e) {}
        public void windowOpened(WindowEvent e) {}
        public void windowActivated(WindowEvent e) {}
        public void windowIconified(WindowEvent e) {}
        public void windowDeiconified(WindowEvent e) {}
        public void windowDeactivated(WindowEvent e) {}
        public void windowClosed(WindowEvent e) {}
    }

}
```

FIGURE 28 – user questions - Internal class

0.4 Collections

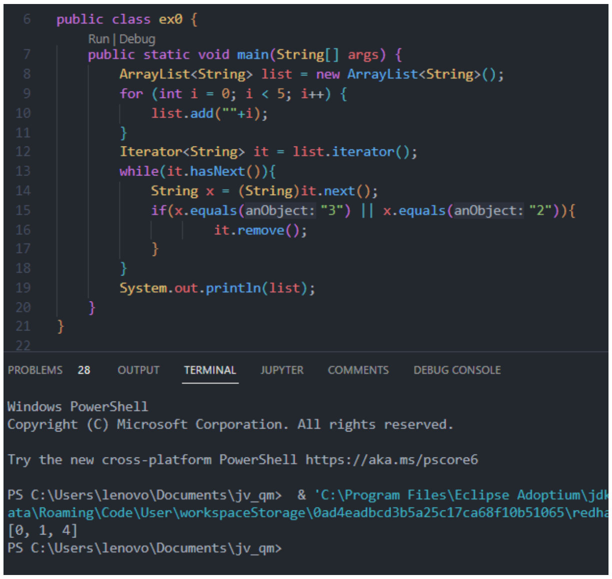
Java's Collection framework offers an architecture for storing and managing a collection of objects.

All data actions, including searching, sorting, insertion, modification, and deletion, are possible with Java Collections.

A collection in Java is a group of related objects. The Java Collection Framework has several classes and interfaces (Set, List, Queue, and Deque) (ArrayList, Vector, LinkedList, PriorityQueue, HashSet, LinkedHashSet, TreeSet).

0.4.1 Course examples

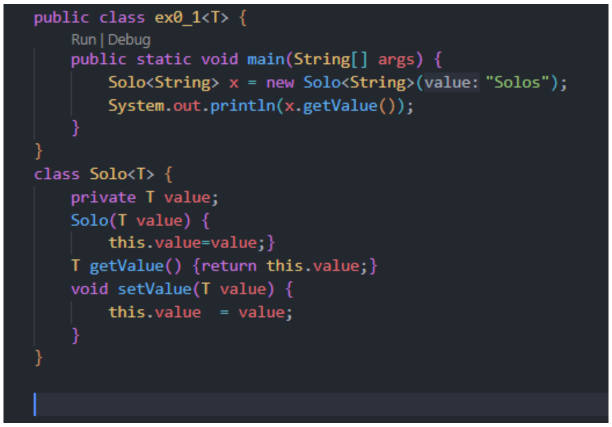
we will present some sample examples to demonstrate how collections works :



```
public class ex0 {
    public static void main(String[] args) {
        ArrayList<String> list = new ArrayList<String>();
        for (int i = 0; i < 5; i++) {
            list.add(""+i);
        }
        Iterator<String> it = list.iterator();
        while(it.hasNext()){
            String x = (String)it.next();
            if(x.equals(anObject: "3") || x.equals(anObject: "2")){
                    it.remove();
            }
        }
        System.out.println(list);
    }
}
```

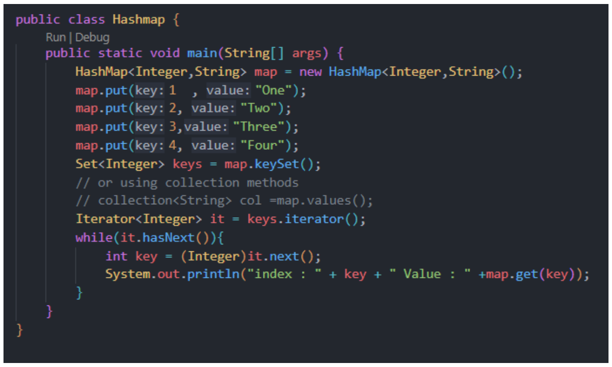
```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\lenovo\Documents\jv_qm>  & 'C:\Program Files\Eclipse Adoptium\jdk
ata\Roaming\Code\User\workspaceStorage\0ad4eadbcd3b5a25c17ca68f10b51065\redha
[0, 1, 4]
PS C:\Users\lenovo\Documents\jv_qm>
```

Figure 29 – Testing Collection ArrayList



```
public class ex0_1<T> {
    public static void main(String[] args) {
        Solo<String> x = new Solo<String>(value: "Solos");
        System.out.println(x.getValue());
    }
}
class Solo<T> {
    private T value;
    Solo(T value) {
        this.value=value;}
    T getValue() {return this.value;}
    void setValue(T value) {
        this.value  = value;
    }
}
```

Figure 30 – Testing custom collection Solo

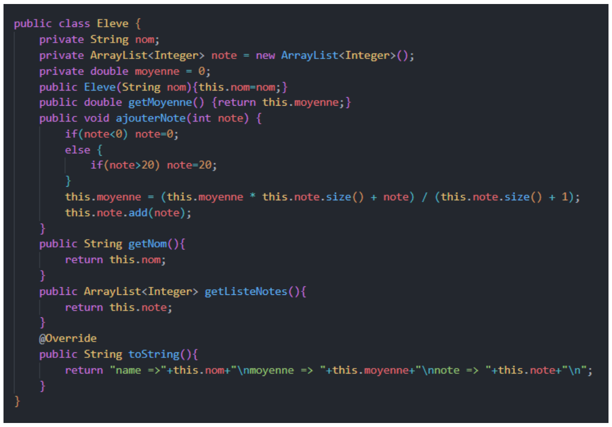


```
public class Hashmap {
    public static void main(String[] args) {
        HashMap<Integer,String> map = new HashMap<Integer,String>();
        map.put(key: 1  , value: "One");
        map.put(key: 2, value: "Two");
        map.put(key: 3,value: "Three");
        map.put(key: 4, value: "Four");
        Set<Integer> keys = map.keySet();
        // or using collection methods
        // collection<String> col =map.values();
        Iterator<Integer> it = keys.iterator();
        while(it.hasNext()){
            int key = (Integer)it.next();
            System.out.println("index : " + key + " Value : " +map.get(key));
        }
    }
}
```

FIGURE 31 – Testing map collection

0.4.2 Use collections to manipulate data

First we went to create a Package Called manageStudent, with their Come Methods below :



```
public class Eleve {
    private String nom;
    private ArrayList<Integer> note = new ArrayList<Integer>();
    private double moyenne = 0;
    public Eleve(String nom){this.nom=nom;}
    public double getMoyenne() {return this.moyenne;}
    public void ajouterNote(int note) {
        if(note<0) note=0;
        else {
            if(note>20) note=20;
        }
        this.moyenne = (this.moyenne * this.note.size() + note) / (this.note.size() + 1);
        this.note.add(note);
    }
    public String getNom(){
        return this.nom;
    }
    public ArrayList<Integer> getListeNotes(){
        return this.note;
    }
    @Override
    public String toString(){
        return "name =>"+this.nom+"\nmoyenne => "+this.moyenne+"\nnote => "+this.note+"\n";
    }
}
```

FIGURE 32 – Student class

After the creation of a student class we will create a modeling of the latter which appoints by GroupeStudent.

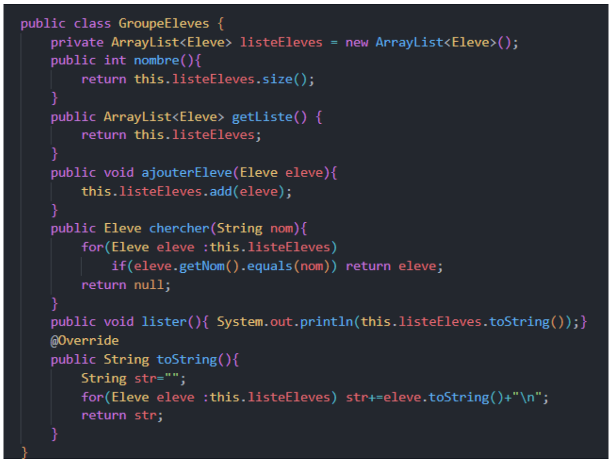
now we will move to another notion wish is compare collections data

but first we need to adapt the previous class by implementing the comparable interface

with the above class we can now compare to student .

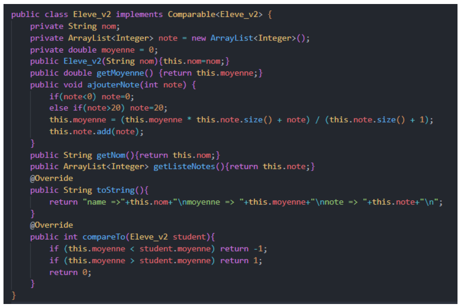
and here is an example

we can also compair a hole collection to see whos the best student the below figure presents how it can be achieved



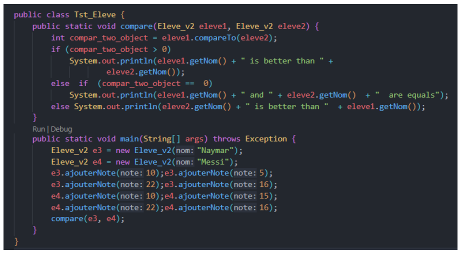
```java
public class GroupeEleves {
    private ArrayList<Eleve> listeEleves = new ArrayList<Eleve>();
    public int nombre(){
        return this.listeEleves.size();
    }
    public ArrayList<Eleve> getListe() {
        return this.listeEleves;
    }
    public void ajouterEleve(Eleve eleve){
        this.listeEleves.add(eleve);
    }
    public Eleve chercher(String nom){
        for(Eleve eleve :this.listeEleves)
            if(eleve.getNom().equals(nom)) return eleve;
        return null;
    }
    public void lister(){ System.out.println(this.listeEleves.toString());}
    @Override
    public String toString(){
        String str="";
        for(Eleve eleve :this.listeEleves) str+=eleve.toString()+"\n";
        return str;
    }
}
```

FIGURE 33 – Group of student class



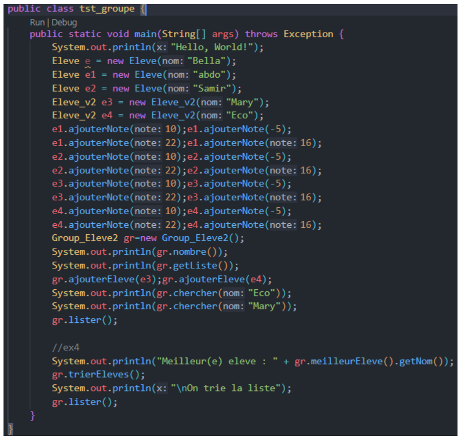
```java
public class Eleve_v2 implements Comparable<Eleve_v2> {
    private String nom;
    private ArrayList<Integer> note = new ArrayList<Integer>();
    private double moyenne = 0;
    public Eleve_v2(String nom){this.nom=nom;}
    public double getMoyenne() {return this.moyenne;}
    public void ajouterNote(int note) {
        if(note<0) note=0;
        else if(note>20) note=20;
        this.moyenne = (this.moyenne * this.note.size() + note) / (this.note.size() + 1);
        this.note.add(note);
    }
    public String getNom(){return this.nom;}
    public ArrayList<Integer> getListeNotes(){return this.note;}
    @Override
    public String toString(){
        return "name =>"+this.nom+"\nmoyenne => "+this.moyenne+"\nnote => "+this.note+"\n";
    }
    @Override
    public int compareTo(Eleve_v2 student){
        if (this.moyenne < student.moyenne) return -1;
        if (this.moyenne > student.moyenne) return 1;
        return 0;
    }
}
```

FIGURE 34 – Compare Students



```java
public class Tst_Eleve {
    public static void compare(Eleve_v2 eleve1, Eleve_v2 eleve2) {
        int compar_two_object = eleve1.compareTo(eleve2);
        if (compar_two_object > 0)
            System.out.println(eleve1.getNom() + " is better than " +
                    eleve2.getNom());
        else  if  (compar_two_object ==  0)
            System.out.println(eleve1.getNom() + " and " + eleve2.getNom()  + "  are equals");
        else System.out.println(eleve2.getNom() + " is better than "  + eleve1.getNom());
    }
    Run | Debug
    public static void main(String[] args) throws Exception {
        Eleve_v2 e3 = new Eleve_v2(nom: "Naymar");
        Eleve_v2 e4 = new Eleve_v2(nom: "Messi");
        e3.ajouterNote(note: 10);e3.ajouterNote(note: 5);
        e3.ajouterNote(note: 22);e3.ajouterNote(note: 16);
        e4.ajouterNote(note: 10);e4.ajouterNote(note: 15);
        e4.ajouterNote(note: 22);e4.ajouterNote(note: 16);
        compare(e3, e4);
    }
}
```

FIGURE 35 – Main Method to compare students



```java
public class tst_groupe {
    Run | Debug
    public static void main(String[] args) throws Exception {
        System.out.println(x: "Hello, World!");
        Eleve e = new Eleve(nom: "Bella");
        Eleve e1 = new Eleve(nom: "abdo");
        Eleve e2 = new Eleve(nom: "Samir");
        Eleve_v2 e3 = new Eleve_v2(nom: "Mary");
        Eleve_v2 e4 = new Eleve_v2(nom: "Eco");
        e1.ajouterNote(note: 10);e1.ajouterNote(-5);
        e1.ajouterNote(note: 22);e1.ajouterNote(note: 16);
        e2.ajouterNote(note: 10);e2.ajouterNote(-5);
        e2.ajouterNote(note: 22);e2.ajouterNote(note: 16);
        e3.ajouterNote(note: 10);e3.ajouterNote(-5);
        e3.ajouterNote(note: 22);e3.ajouterNote(note: 16);
        e4.ajouterNote(note: 10);e4.ajouterNote(-5);
        e4.ajouterNote(note: 22);e4.ajouterNote(note: 16);
        Group_Eleve2 gr=new Group_Eleve2();
        System.out.println(gr.nombre());
        System.out.println(gr.getListe());
        gr.ajouterEleve(e3);gr.ajouterEleve(e4);
        System.out.println(gr.chercher(nom: "Eco"));
        System.out.println(gr.chercher(nom: "Mary"));
        gr.lister();

        //ex4
        System.out.println("Meilleur(e) eleve : " + gr.meilleurEleve().getNom());
        gr.trierEleves();
        System.out.println(x: "\nOn trie la liste");
        gr.lister();
    }
}
```

FIGURE 36 – Getting the best student