G.0. FrameFrame

Alăturat aveți codul sursă al unui program Java proiectat conform arhitecturii (pattern) Model View Controller (MVC).

MVC Pattern is used to separate application's concerns.

- **Model** - Model represents an object carrying data. It can also have logic to update data changes.
- **View** - View represents the visualization of the data that model contains.
- **Controller** - Controller acts on both model and view. It controls the data flow into model object and updates the view whenever data changes. It keeps view and model separate.

Se cere să completați acest program astfel încât la acționarea butonului "Actiune" să fie apelată metoda increment() pentru obiectul Model md din Controller c iar noua valoare a atributului md.x să fie afișată în campul TextField tf al interfeței View.

```
// file Model.java
public class Model{
        private int x=0;
        public Model(){};
        public void increment(){x++;}
        public int get_x(){return x;}
}
```

```
// file MVC.java
public class MVC{
        public static void main(String[] args){
                View v=new View();
                Model m= new Model();
                Controller c= new Controller(v,m);
        }
}
```

```
// file View.java

import java.awt.*;
public class View extends Frame{
        Button b;
    TextField tf;
        public View(){
                setTitle("Exemplu Model-View-Controller");

                b= new Button("Actiune");
                add("North",b);

                tf=new TextField(10);
                add("Center",tf);

                setSize(100,250);
                setVisible(true);
        }
}
```

```
// file Controller.java
import java.awt.event.*;
public class Controller implements ActionListener{

	public Controller(View v, Model m){
		vw=v;md=m;
		v.b.addActionListener(this);
	}
	public void actionPerformed(ActionEvent e){
		// ???
		vw.tf.setText(String.valueOf(md.get_x()));
	}
	private View vw;
	private Model md;
}
```