

# GUI programming

Graphical User Interface (GUI)-based programming

# Windchill



- Windchill
  - There are several formulas for calculating the windchill temperature  $t_{wc}$
  - The one provided by U.S. National Weather Service and is applicable for a windspeed greater than four miles per hour

$$t_{wc} = 0.081(t - 91.4)(3.71\sqrt{v} + 5.81 - 0.25v) + 91.4$$

- Where
  - Variable  $t$  is the Fahrenheit temperature
  - Variable  $v$  is the windspeed in miles per hour

# Console-based programming

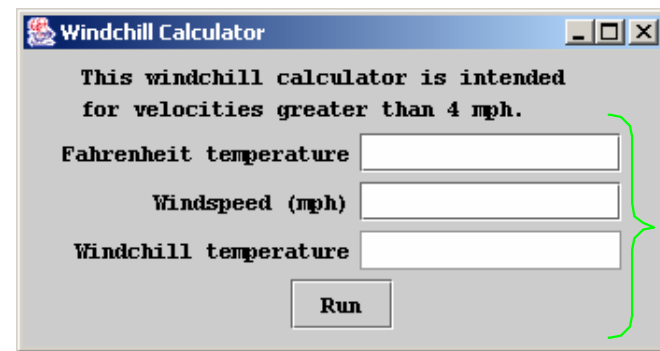
## Console program

```
Method main() {  
    statement;  
    statement;  
    ...  
    statement;  
}
```

Console programs begin and end in method main()

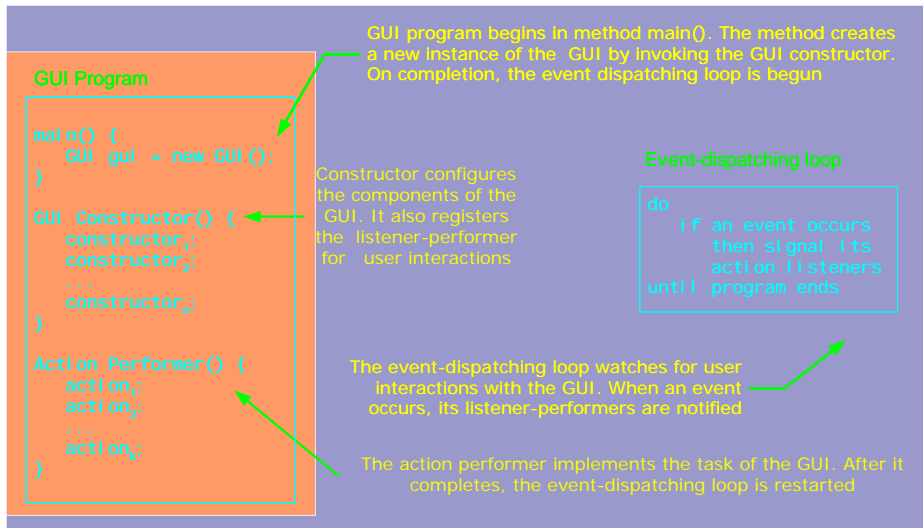
# In use

Program needs to respond *whenever* the run button is clicked



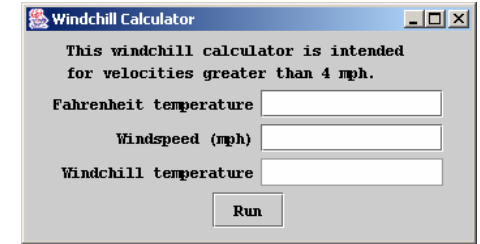
There needs to be an event loop that is looking for user interface events

## GUI-based programming



## Java support

- JFrame
  - Represents a titled, bordered window
- JTextArea
  - Represents an editable multiline text entry component
- JLabel
  - Represents a display area suitable for one or both of a single-line text or image.
- JTextField
  - Represents an editable single-line text entry component
- JButton
  - Represents a push button

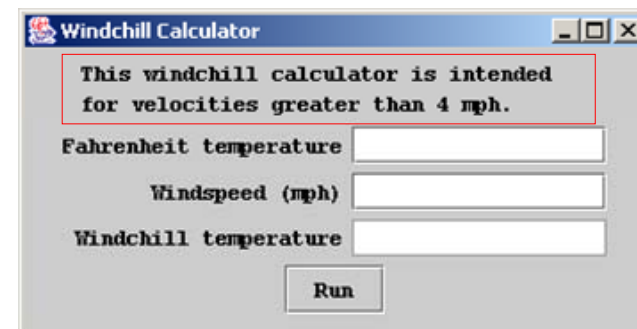
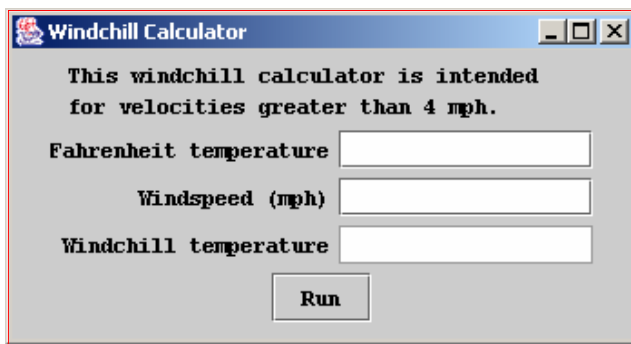


## Instance variables

- private JFrame window
  - References the window containing the other components of the GUI

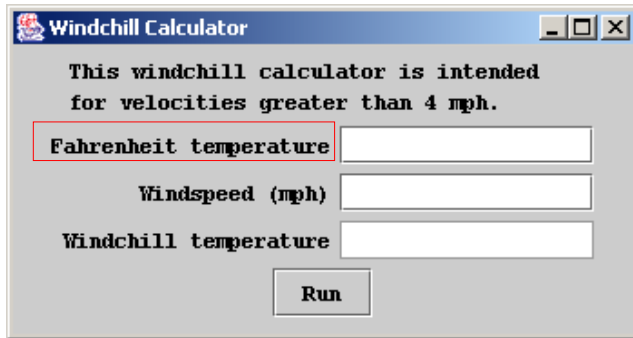
## Instance variables

- private JTextArea legendArea
  - References the text display for the multiline program legend



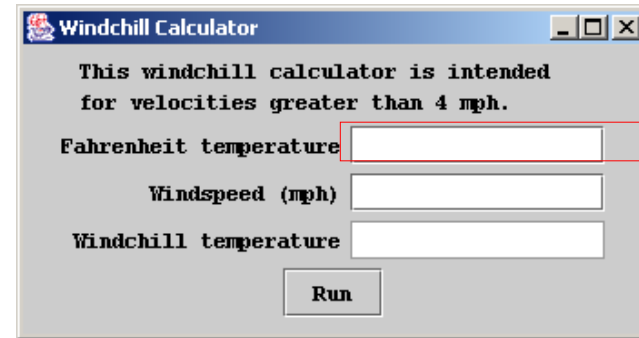
## Instance variables

- private JLabel fahrTag
  - References the label for the data entry area supplying the temperature



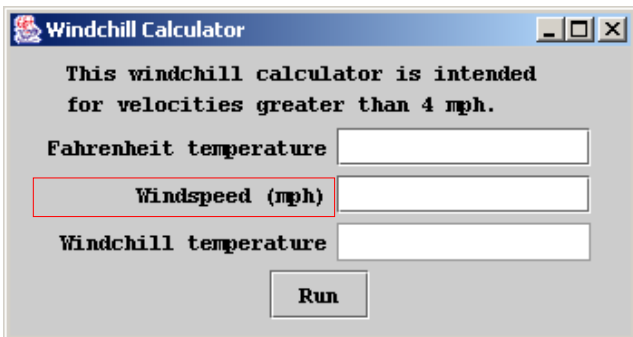
## Instance variables

- private JTextField fahrText
  - References the data area supplying the temperature



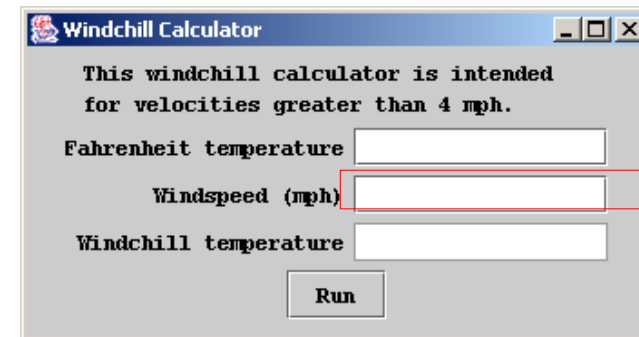
## Instance variables

- private JLabel windTag
  - References the label for the data entry area supplying the windspeed



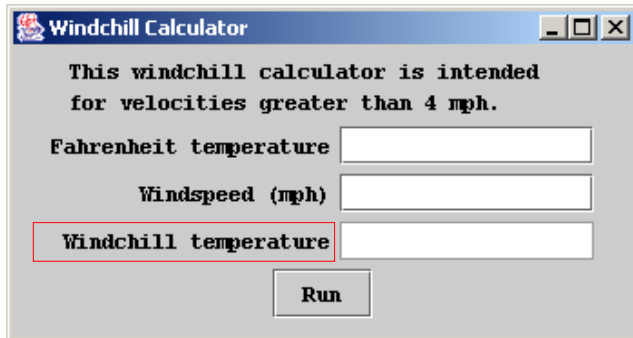
## Instance variables

- private JTextField windText
  - References the data area supplying the windspeed



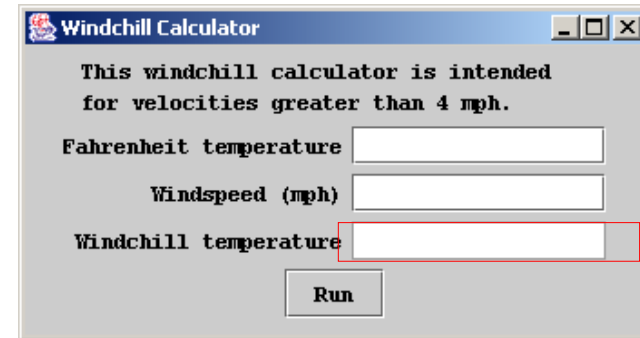
## Instance variables

- private JLabel chillTag
  - References the label for the data area giving the windchill



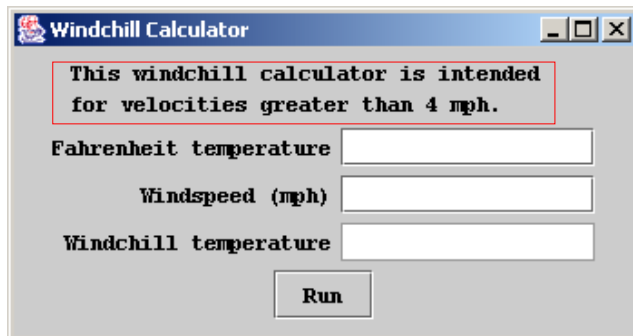
## Instance variables

- private JTextField chillText
  - References the data area giving the windchill



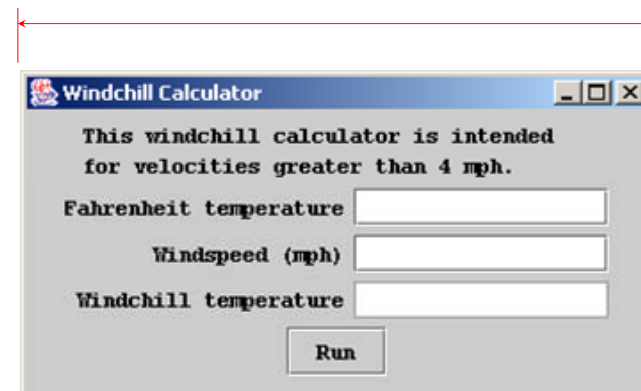
## Class constants

- private **static** final String LEGEND = "This windchill calculator" + "is intended for velocities greater than 4 mph."
  - Program legend text



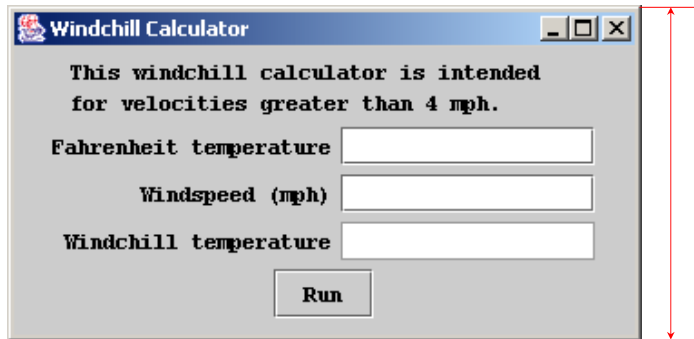
## Class constants

- private static final int WINDOW\_WIDTH = 400
  - Initial width of the GUI



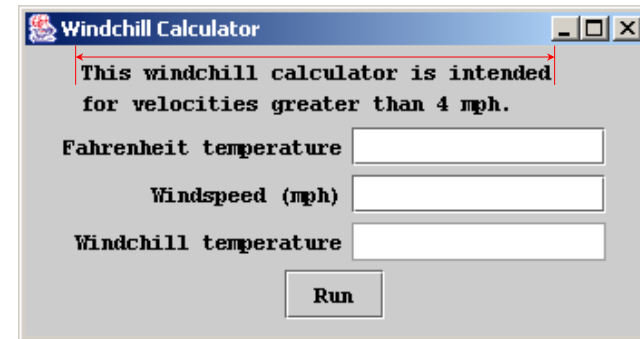
## Class constants

- private static final int WINDOW\_HEIGHT = 200
  - Initial height of the GUI



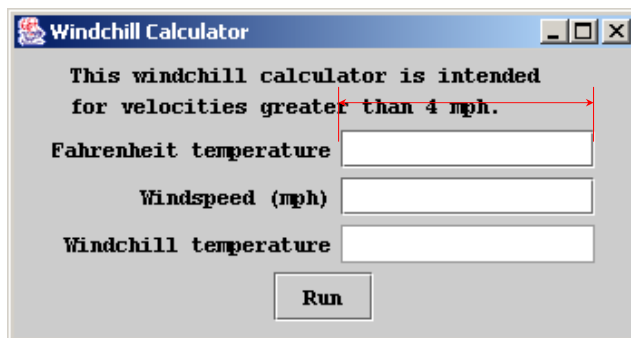
## Class constants

- private static final int AREA\_WIDTH = 40
  - Width of the program legend in characters



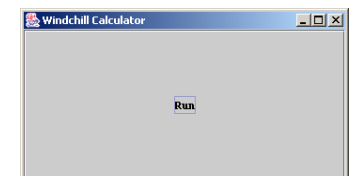
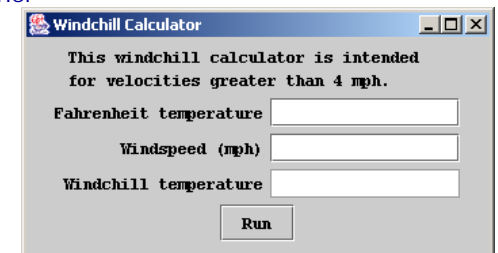
## Class constants

- private static final int FIELD\_WIDTH = 40
  - Number of characters per data entry area



## Class constants

- private static final `FlowLayout LAYOUT_STYLE = new FlowLaout()`
  - References manager that lays out GUI components in a top-to-bottom, left-to-right manner
  - Different GUI layout
    - [BorderLayout](#)
    - [BoxLayout](#)
    - [CardLayout](#)
    - [FlowLayout](#)
    - [GridBagLayout](#)
    - [GridLayout](#)
    - [SpringLayout](#)
  - If no layout is specified, then the last component added to the window occupies the entire window



## Program Windchill.java

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class Windchill implements ActionListener {
    // class constants

    // instance variables with initialization

    // Windchill(): default constructor

    // actionPerformed(): run button action event handler

    // main(): application entry point
}
```

## Program Windchill.java – class constants

```
private static final int WINDOW_WIDTH = 400; // pixels
private static final int WINDOW_HEIGHT = 200; // pixels
private static final int FIELD_WIDTH = 20; // characters
private static final int AREA_WIDTH = 40; // characters

private static final FlowLayout LAYOUT_STYLE =
    new FlowLayout();

private static final String LEGEND = "This windchill "
    + "calculator is intended for velocities greater than 4 mph.";
```

## Program Windchill.java – instance variables

```
// window for GUI
private JFrame window =
    new JFrame("Windchill Calculator");

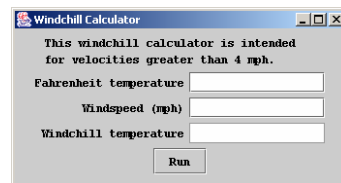
// legend
private JTextArea legendArea = new JTextArea(LEGEND, 2,
    AREA_WIDTH);

// user entry area for temperature
private JLabel FahrTag = new JLabel("Fahrenheit temperature");
private JTextField FahrText = new JTextField(FIELD_WIDTH);
// user entry area for windspeed
private JLabel WindTag = new JLabel("Windspeed (mph)");
private JTextField WindText = new JTextField(FIELD_WIDTH);

// entry area for windchill result
private JLabel ChillTag =
    new JLabel("Windchill temperature");

private JTextField ChillText = new JTextField(FIELD_WIDTH);

// run button
private JButton runButton = new JButton("Run");
```



## Program Windchill.java – constructor

```
public Windchill() {
    // configure GUI

    // register event listener

    // add components to container

    // display GUI
}
```

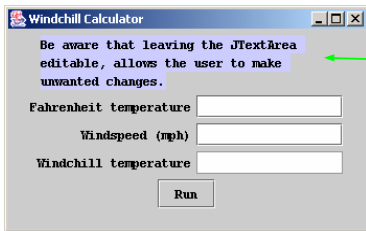
## Program Windchill.java – constructor

```
public Windchill () {
    // configure GUI
    window.setSize(WINDOW_WIDTH, WINDOW_HEIGHT);
    window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

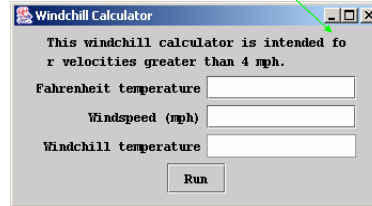
    legendArea.setEditable(false);
    legendArea.setLineWrap(true);
    legendArea.setWrapStyleWord(true);
    legendArea.setBackground(window.getBackground());

    chillText.setEditable(false);
    chillText.setBackground(Color.WHITE);
}
```

Otherwise line wrapping in the middle of a word



It is important to make program legends uneditable



## Program Windchill.java – constructor

```
public Windchill () {
    // configure GUI
    window.setSize(WINDOW_WIDTH, WINDOW_HEIGHT);
    window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    legendArea.setEditable(false);
    legendArea.setLineWrap(true);
    legendArea.setWrapStyleWord(true);
    legendArea.setBackground(window.getBackground());

    chillText.setEditable(false);
    chillText.setBackground(Color.WHITE);
}
```



A JLabel is noneditable by the user

By default the text field of a JTextField is editable by the user

## Program Windchill.java – constructor

```
public Windchill () {
    // configure GUI ...

    // register event listener
    runButton.addActionListener(this);
}
```

Action events are sent to registered action listeners

Run

Action Event

When the run button is clicked, it dispatches an action event

An ActionListener has an actionPerformed() method that handles the class-specific activity

GUI : Action Listener

actionPerformed() Method

- Get data entries from temperature and windspeed data areas
- Compute windchill according to the Weather Service formula
- Display result to windchill data area

## Program Windchill.java – constructor

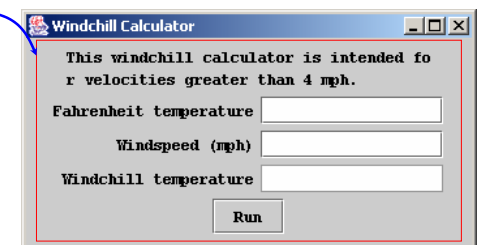
```
public Windchill () {
    // configure GUI ...

    // register event listener ...

    // add components to container

    Container c =
    window.getContentPane();
    c.setLayout(LAYOUT_STYLE);

    c.add(legendArea);
    c.add(fahrTag);
    c.add(fahrText);
    c.add(windTag);
    c.add(windText);
    c.add(chillTag);
    c.add(chillText);
    c.add(runButton);
}
```



## Program Windchill.java – constructor

```
public Windchill() {  
    // configure GUI ...  
  
    // register event listener ...  
  
    // add components to container ...  
  
    // make GUI visible  
    window.setVisible(true);  
}
```

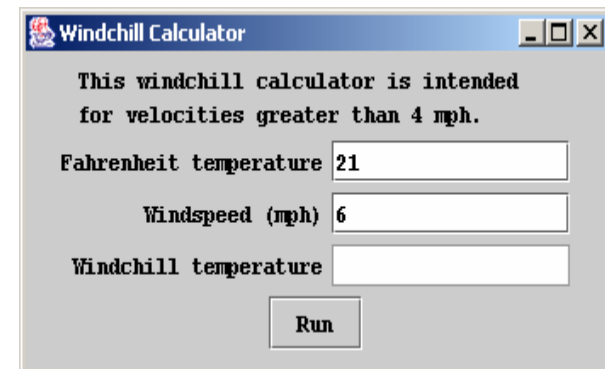
## Program Windchill.java – action performer

```
public void actionPerformed(ActionEvent e) {  
    // get user's responses  
  
    // compute windchill  
  
    // display windchill  
}
```

## Program Windchill.java – action performer

```
public void actionPerformed(ActionEvent e) {  
    // get user's responses  
    String response1 = fahrText.getText();  
    double t = Double.parseDouble(response1);  
    String response2 = windText.getText();  
    double v = Double.parseDouble(response2);  
  
    // compute windchill  
  
    // display windchill  
}
```

## Program Windchill.java – action performer



## Program Windchill.java – action performer

```
public void actionPerformed(ActionEvent e) {
    // get user's responses
    String response1 = fahrText.getText();
    double t = Double.parseDouble(response1);
    String response2 = windText.getText();
    double v = Double.parseDouble(response2);

    // compute windchill
    double windchillTemperature = 0.081 * (t - 91.4)
        * (3.71*Math.sqrt(v) + 5.81 - 0.25*v) + 91.4;

    int perceivedTemperature =
        (int) Math.round(windchillTemperature);

    // display windchill
}
```

## Program Windchill.java – action performer

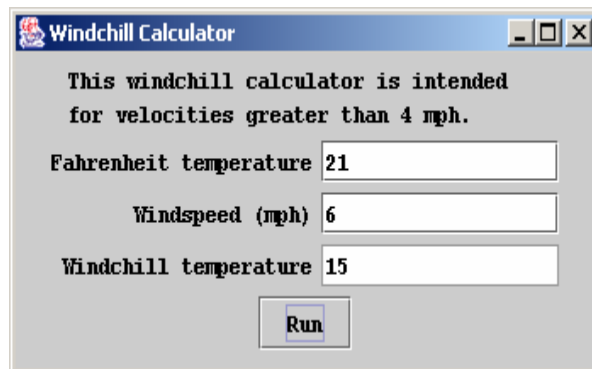
```
public void actionPerformed(ActionEvent e) {
    // get user's responses
    String response1 = fahrText.getText();
    double t = Double.parseDouble(response1);
    String response2 = windText.getText();
    double v = Double.parseDouble(response2);

    // compute windchill
    double windchillTemperature = 0.081 * (t - 91.4)
        * (3.71*Math.sqrt(v) + 5.81 - 0.25*v) + 91.4;

    int perceivedTemperature =
        (int) Math.round(windchillTemperature);

    // display windchill
    String output = String.valueOf(perceivedTemperature);
    chillText.setText(output);
}
```

## Program Windchill.java – action performer



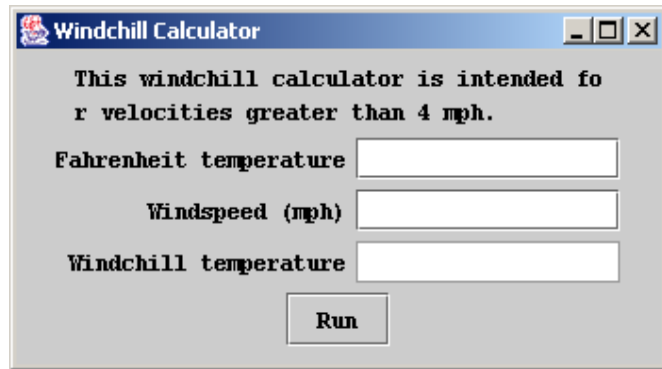
## How to handle exceptions in GUI

- We used to use *throws IOException* right after we defines a method which may encounter IOException problem. For example, needs user input, or, opens and reads from a file.
- But in GUI, inside of actionPerformed method, we have to use:

```
try {
    // whatever the actions, for example, opens a file and performs
    calculations, etc.
}
catch (IOException e){
    //whatever you want to output, for example:
    System.out.println("Cannot read file!!!");
}
```

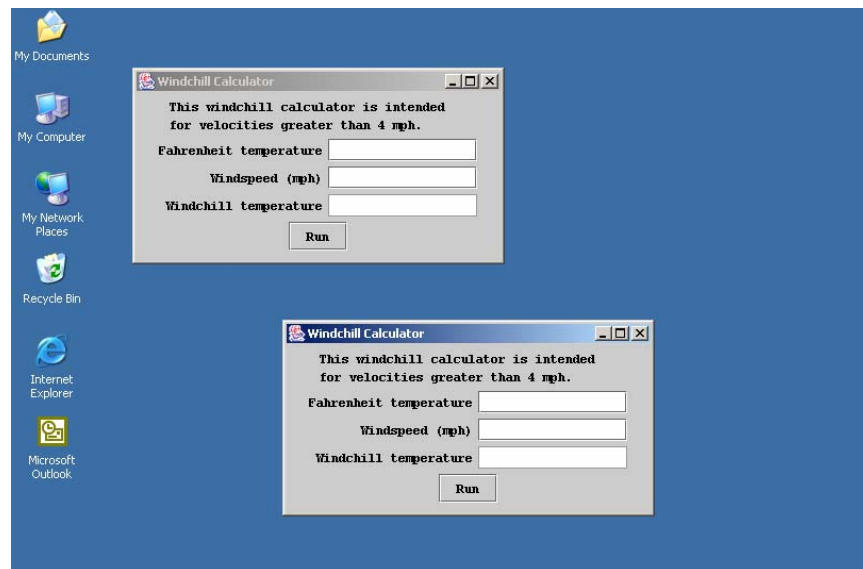
## Method main()

```
public static void main(String[] args) {  
    Windchill gui = new Windchill();  
}
```



## Another method main()

```
public static void main(String[] args) {  
    Windchill gui 1 = new Windchill();  
    Windchill gui 2 = new Windchill();  
}
```



## Assignment, Due March 13, Tuesday, before noon

- [Practice question, do not need to submit] Design and implement a GUI program that processes a user-specified filename and string, then the program displays the number of lines and characters in the file, and the number of occurrences of that string in the specified file (you can design your own interface).
- Design and implement a GUI program using the following interface as a reference, this program takes two numbers (doubles) and an operator (only '+', '-', '\*', and '/' are considered this time) as inputs and then outputs the calculation result.

