Asst. Prof. Dr. Sasiporn Usanavasin, Dr. Jessada Karnjana, Dr. Kasorn Galajit and Dr. Akkarawoot Takhom {sasiporn.us, akkharawoot.aj, jessada.aj, kasorn.aj}@siit.tu.ac.th

School of Information, Computer, and Communication Technology, Sirindhorn International Institute of Technology, Thammasat University.

DES103(Y23S2)-LAB09-REVIEW]

Event-driven Programming EP2

Learning Objectives

1. To learn how to define a listener
2. To learn how to register an appropriate listener to the source
3. To learn how to implement appropriate methods and their details for the specified listener to perform the assigned task.

****Remark**** A ***pointer finger (👉)*** refers to an explanation between students and their TA.

9.1 Event-Driven Programming:

Event-driven programming is a programming paradigm where code execution is triggered by the occurrence of ***events***.

9.2 Events

❖ **Definition:** An event serves as a signal to the program indicating that a particular action or occurrence has taken place.

❖ **Sources:** Events are typically generated by:

- ➢ **External User Actions**: These include interactions such as:
  - ■ Mouse Movements: When the user moves the mouse cursor.
  - ■ Mouse Clicks: When the user clicks a mouse button.
  - ■ Keystrokes: When the user presses keys on the keyboard.
- ➢ **Operating System**: Events can also be triggered by the operating system, for example:
  - ■ Timer Events: The operating system sends a signal to the program after a specific duration has elapsed, allowing for time-sensitive functionalities or periodic tasks.
  - ■ System Notifications: Such as notifications about changes in system state or availability of resources.
  - ■ Network Events: Signals related to network activity, such as incoming data packets or connection status changes.

❖ **Role**: Events serve as triggers that drive the flow of execution within event-driven programs. They enable developers to create dynamic and responsive software systems that can

Asst. Prof. Dr. Sasiporn Usanavasin, Dr. Jessada Karnjana, Dr. Kasorn Galajit and Dr. Akkarawoot Takhom {sasiporn.us, akkharawoot.aj, jessada.aj, kasorn.aj}@siit.tu.ac.th

School of Information, Computer, and Communication Technology, Sirindhorn International Institute of Technology, Thammasat University.

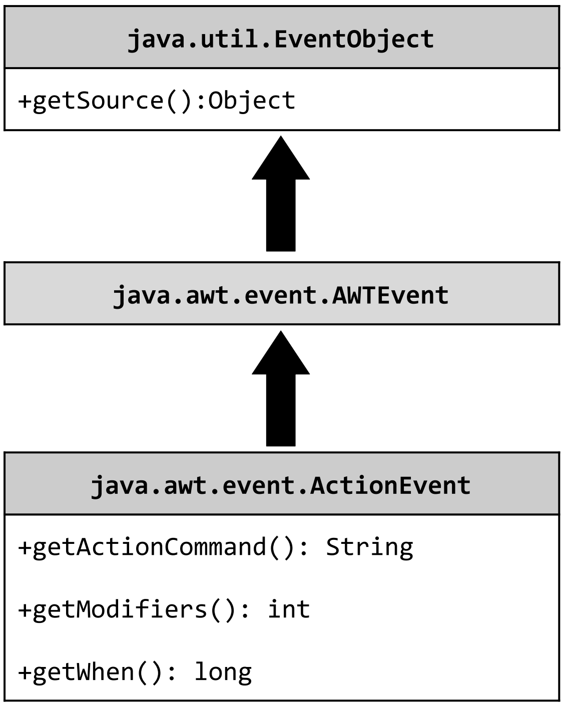
effectively interact with users and adapt to changing environments. By responding to events, programs can perform specific actions or update their state in real-time, enhancing user experience and overall functionality.

8.3 Examples of sources and events

User actions	**Source objects**	**Type of fired events**
Click a button	`JButton`	`ActionEvent`
Click a checkbox	`JCheckBox`	`ItemEvent, ActionEvent`
Click a radio button	`JRadioButton`	`ItemEvent, ActionEvent`
Press return on a text field	`JTextField`	`ActionEvent`
Select a new item	`JComboBox`	`ItemEvent, ActionEvent`
Windows opened, closed, etc.	`Window`	`WindowEvent`
Mouse pressed, released, dragged etc.	`Mouse`	`MouseEvent`
Key released, pressed, etc.	`Keyboard`	`KeyEvent`

8.4 EventObject

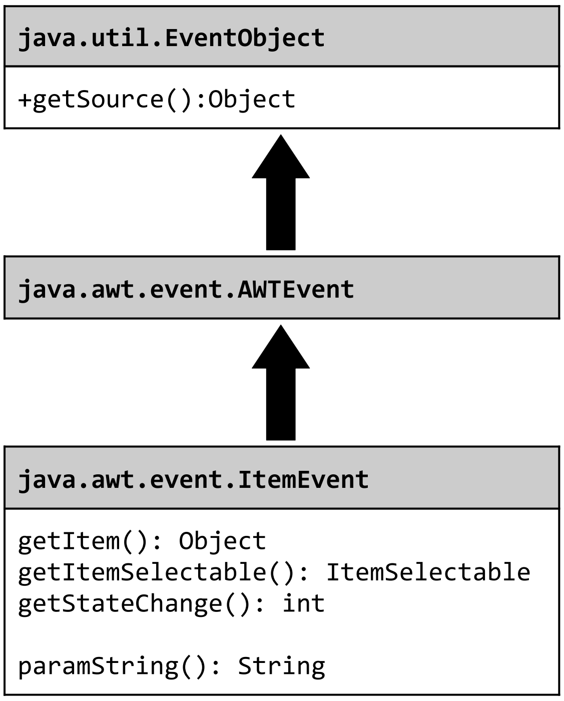
8.4.1 ActionEvent



- ❖ Returns the object on which the event initially occurred.

- ❖ Returns the *command* string associated with this action. For a button, its text is the command string.
- ❖ Returns the *modifier keys* held down during this action event.
- ❖ Returns the *timestamp* when this event occurred. The time is the number of milliseconds since January 1, 1970, 00:00:00 GMT.

8.4.2 ItemEvent



- ❖ Returns the object on which the event initially occurred.

- ❖ Returns the item affected by the event
- ❖ Returns the originator of the event
- ❖ Returns the type of state change (selected or deselected).
- ❖ Returns a parameter string identifying this item event.

```
+getKeyModifiersText(int modifiers) :
String
```

8.4.3 MouseEvent

```
java.awt.event.InputEvent
-----------------------------
+getWhen(): long

+isAltDown(): boolean

+isControlDown(): boolean

+isMetaDown(): boolean

+isShiftDown(): boolean
```

- ❖ Returns the timestamp when this event occurred.
- ❖ Returns whether or not the Alt modifier is down on this event.
- ❖ Returns whether or not the Control modifier is down on this event.
- ❖ Returns whether or not the Meta modifier is down on this event
- ❖ Returns whether or not the Shift modifier is down on this event.

⬆

```
java.awt.event.MouseEvent
-----------------------------
+getButton(): int
+getClickCount(): int
+getPoint(): java.awt.Point
+getX(): int
+getY(): int
```

- ❖ Indicates which mouse button has been clicked.
- ❖ Returns the number of mouse clicks associated with this event.
- ❖ Returns a Point object containing the x and y coordinates.
- ❖ Returns the x-coordinate of the mouse point. Returns the y-coordinate of the mouse point.

8.4.3 MouseEvent

```
java.awt.event.KeyEvent
```

⬆

```
java.awt.event.MouseEvent
-----------------------------
+getKeyChar(): char

+getKeyCode(): int

+getKeyLocation():int

+getKeyText(int keyCode) :String
```

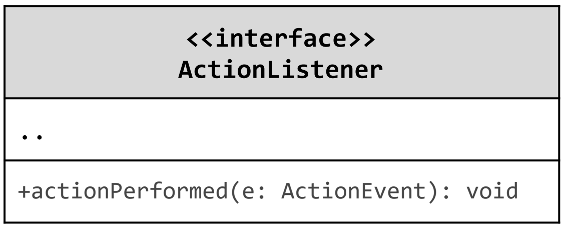
- ❖ Returns the character associated with the key in this event.
- ❖ Returns the integer keyCode associated with the key in this event.
- ❖ Returns the location of the key that originated this key event.

- ❖ Returns a String describing the keyCode, Ex., "HOME", "F1" or "A".
- ❖ Returns a String describing the modifier key(s), such as "Shift", or "Ctrl+Shift".

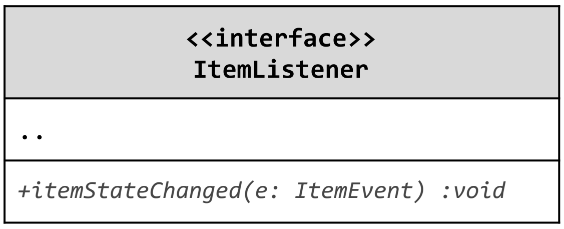
8.5 Interaction between Source and Listener

8.5.1 UML of Listener's Class

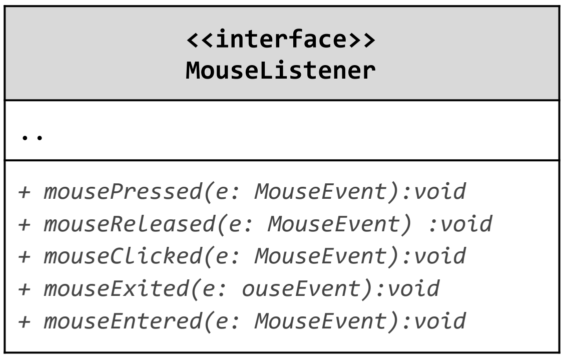
Listeners are defined as <<interface>>



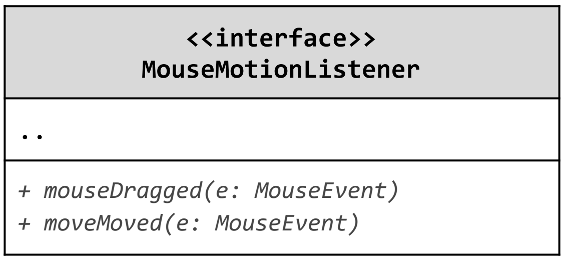
```
<<interface>>
ActionListener
..
+actionPerformed(e: ActionEvent): void
```



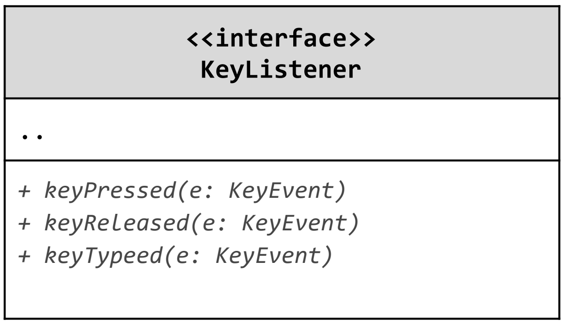
```
<<interface>>
ItemListener
..
+itemStateChanged(e: ItemEvent) :void
```



```
<<interface>>
MouseListener
..
+ mousePressed(e: MouseEvent):void
+ mouseReleased(e: MouseEvent) :void
+ mouseClicked(e: MouseEvent):void
+ mouseExited(e: ouseEvent):void
+ mouseEntered(e: MouseEvent):void
```



```
<<interface>>
MouseMotionListener
..
+ mouseDragged(e: MouseEvent)
+ moveMoved(e: MouseEvent)
```



```
<<interface>>
KeyListener
..
+ keyPressed(e: KeyEvent)
+ keyReleased(e: KeyEvent)
+ keyTypeed(e: KeyEvent)
```

Asst. Prof. Dr. Sasiporn Usanavasin, Dr. Jessada Karnjana, Dr. Kasorn Galajit and Dr. Akkarawoot Takhom {sasiporn.us, akkharawoot.aj, jessada.aj, kasorn.aj}@siit.tu.ac.th
School of Information, Computer, and Communication Technology, Sirindhorn International Institute of Technology, Thammasat University.

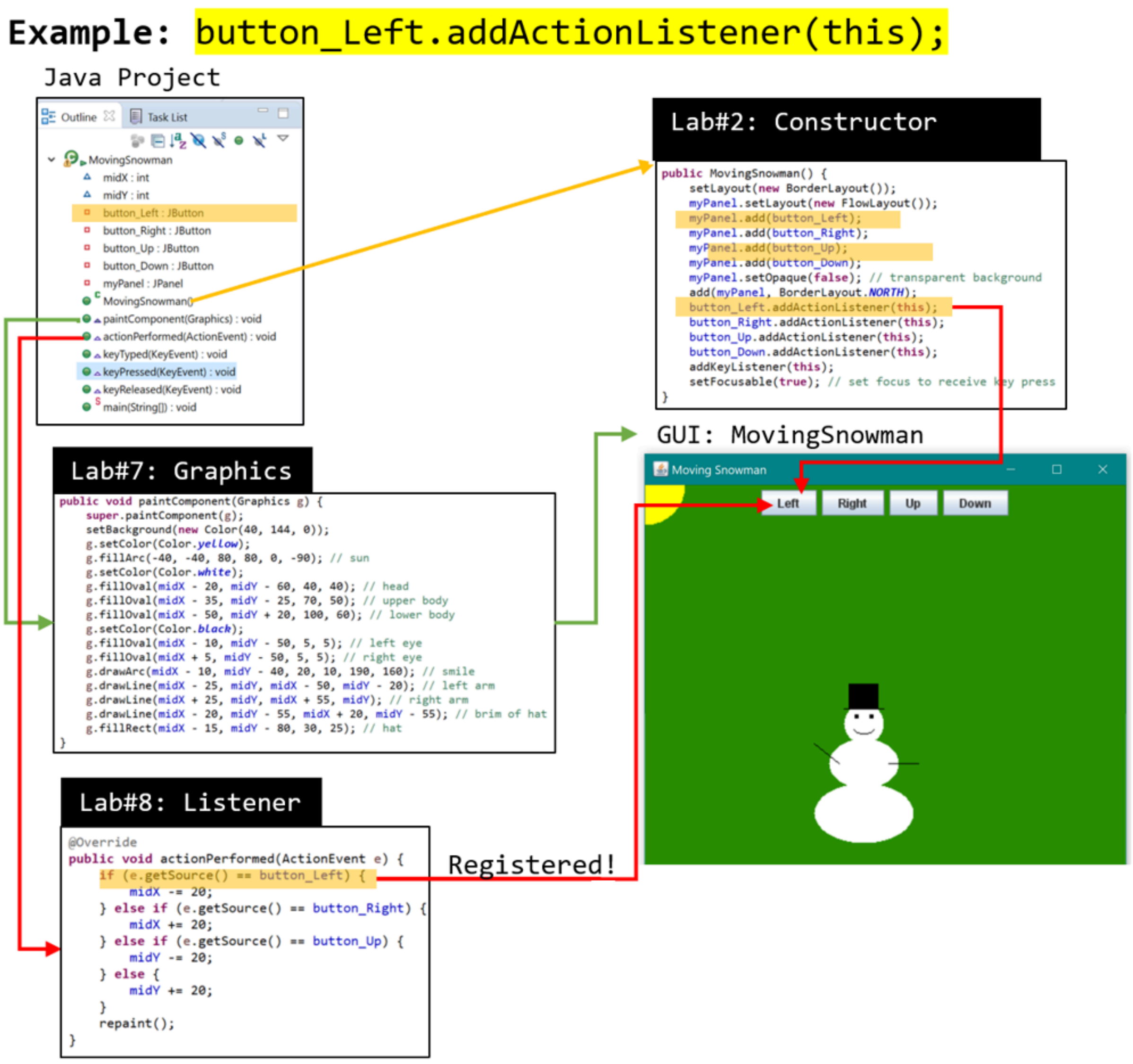
8.5.2 Example: <object>.add<Listener>(this);

☞ To remember type of your object and name, TA suggests to define your variables name as below format:

`<object>_<name>`

For example, `public JButton button_Left = new JButton("Left");`

☞ To explain your TA, student SHOULD try to understand the following example:


Example: button_Left.addActionListener(this);

Java Project

```
MovingSnowman
    midX : int
    midY : int
    button_Left : JButton
    button_Right : JButton
    button_Up : JButton
    button_Down : JButton
    myPanel : JPanel
    MovingSnowman()
    paintComponent(Graphics) : void
    actionPerformed(ActionEvent) : void
    keyTyped(KeyEvent) : void
    keyPressed(KeyEvent) : void
    keyReleased(KeyEvent) : void
    main(String[]) : void
```

Lab#2: Constructor

```java
public MovingSnowman() {
    setLayout(new BorderLayout());
    myPanel.setLayout(new FlowLayout());
    myPanel.add(button_Left);
    myPanel.add(button_Right);
    myPanel.add(button_Up);
    myPanel.add(button_Down);
    myPanel.setOpaque(false); // transparent background
    add(myPanel, BorderLayout.NORTH);
    button_Left.addActionListener(this);
    button_Right.addActionListener(this);
    button_Up.addActionListener(this);
    button_Down.addActionListener(this);
    addKeyListener(this);
    setFocusable(true); // set focus to receive key press
}
```

GUI: MovingSnowman

Lab#7: Graphics

```java
public void paintComponent(Graphics g) {
    super.paintComponent(g);
    setBackground(new Color(40, 144, 0));
    g.setColor(Color.yellow);
    g.fillArc(-40, -40, 80, 80, 0, -90); // sun
    g.setColor(Color.white);
    g.fillOval(midX - 20, midY - 60, 40, 40); // head
    g.fillOval(midX - 35, midY - 25, 70, 50); // upper body
    g.fillOval(midX - 50, midY + 20, 100, 60); // lower body
    g.setColor(Color.black);
    g.fillOval(midX - 10, midY - 50, 5, 5); // left eye
    g.fillOval(midX + 5, midY - 50, 5, 5); // right eye
    g.drawArc(midX - 10, midY - 40, 20, 10, 190, 160); // smile
    g.drawLine(midX - 25, midY, midX - 50, midY - 20); // left arm
    g.drawLine(midX + 25, midY, midX + 55, midY); // right arm
    g.drawLine(midX - 20, midY - 55, midX + 20, midY - 55); // brim of hat
    g.fillRect(midX - 15, midY - 80, 30, 25); // hat
}
```

Lab#8: Listener

```java
@Override
public void actionPerformed(ActionEvent e) {
    if (e.getSource() == button_Left) {
        midX -= 20;
    } else if (e.getSource() == button_Right) {
        midX += 20;
    } else if (e.getSource() == button_Up) {
        midY -= 20;
    } else {
        midY += 20;
    }
    repaint();
}
```

Asst. Prof. Dr. Sasiporn Usanavasin, Dr. Jessada Karnjana, Dr. Kasorn Galajit and Dr. Akkarawoot Takhom {sasiporn.us, akkharawoot.aj, jessada.aj, kasorn.aj}@siit.tu.ac.th

School of Information, Computer, and Communication Technology, Sirindhorn International Institute of Technology, Thammasat University.

[DES103(Y23S2)-LAB09-EXERCISES]

Students MUST adhere to the lab instructions and regulations provided below.
Please consult your TA to review your completed exercises and submit them on Google Classroom.

Be noticed that for all lab exercises, you need to define your *Java* project as the following name format:

<StudentID>_<Lab number>_<Exercise name>

If your student's ID is 6722300208, the name format of your java project should be:

"6422300208_LAB09_BMICalculator" for exercise 1 and 2

"6422300208_LAB09_KeyboardEvent" for exercise 3

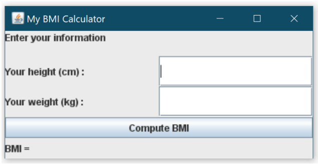
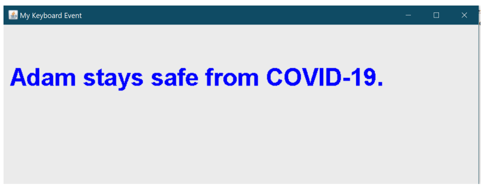
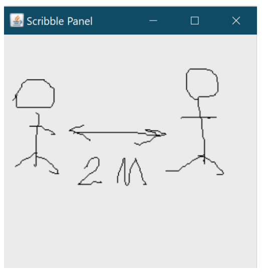
"6422300208_LAB09_ScribblePanel" for exercise 4 and 5

1. Students MUST explain your understanding to your TA before submitting your finished exercise,
2. Students can attach your finished exercise into Google Class, and click 'Turn-In' when your TA allows you to submit,

For today's exercises, students are going to:

- Draw a panel of moving message by using `paintComponent` and methods of a graphics object learned in class,
- Register appropriate listeners for moving messages
- Implement appropriate methods and their details for the specified listener to perform the assigned task.

The final output should look like this:

LAB09_BMICalculator	LAB09_KeyboardEvent	LAB09_ScribblePanel
 My BMI Calculator Enter your information Your height (cm) : Your weight (kg) : Compute BMI BMI =	 My Keyboard Event Adam stays safe from COVID-19.	 Scribble Panel

Asst. Prof. Dr. Sasiporn Usanavasin, Dr. Jessada Karnjana, Dr. Kasorn Galajit and Dr. Akkarawoot Takhom {sasiporn.us, akkharawoot.aj, jessada.aj, kasorn.aj}@siit.tu.ac.th
School of Information, Computer, and Communication Technology, Sirindhorn International Institute of Technology, Thammasat University.

Exercise 1: (*2 points*)

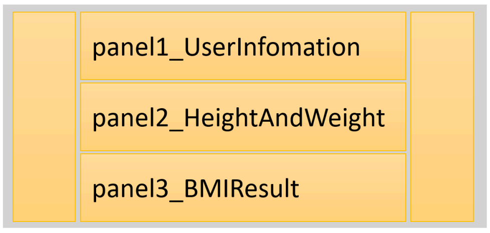
- **Java Project**: <Student_ID>_ LAB09_BMICalculator
- **Objective**: To learn how to use `LayoutManagers` to arrange the components in the Container
- **Suggestion**: To define your variables name as below format:

  `<object>_<name>`

  For example, `public JButton button_Left = new JButton("Left");`

- **Instruction**: Create a java project and write code in the following tasks.

a. Add a new java class `BMICalculator` that makes the following GUI design:

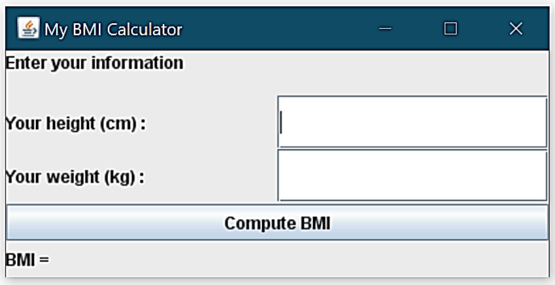


b. Add a new java class `BMICalculatorTesting` and write a `main` method that has a frame to add an object `BMICaculator`.

Remark: a skeleton-code of the frame is as below:

```
JFrame frame = new JFrame("My BMI Calculator");

____________________________________ //create an object of BMICaculator
frame.add(_____________________________); //add JPanel into JFrame

frame.setSize(400, 200); //set a frame's resolution
frame.setLocationRelativeTo(null); //set a location at center the frame
frame.setVisible(true); //set visible
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); //set default Close
Operation
```

c. Your running output of the `BMICalculator` GUI, as shown below:



Asst. Prof. Dr. Sasiporn Usanavasin, Dr. Jessada Karnjana, Dr. Kasorn Galajit and Dr. Akkarawoot Takhom {sasiporn.us, akkharawoot.aj, jessada.aj, kasorn.aj}@siit.tu.ac.th
School of Information, Computer, and Communication Technology, Sirindhorn International Institute of Technology, Thammasat University.

Exercise 2: (*2 points*)

- **Java Project**: <Student_ID>_ LAB9_BMICalculator
- **Objective**: To learn how to register an appropriate listener to the source, and implement appropriate methods and their details for the specified listener to perform the assigned task.
- **Suggestion**: To define your variables name as below format:

  `<object>_<name>`

  For example, `public JButton` **`button_Left`** `= new JButton("Left");`

- **Instruction**: Continue from Exercise 1, and write code in the following tasks.

  a. Define a class `ActionInterpretor` which is a subclass of an interface `ActionListener` inside of the `BMICalculator` class.

     i. Remark: The `ActionInterpretor` class is an inner class of the `BMICalculator` class.

        It can access all properties and methods of the BMICalculator.

  b. Override the abstract method of `ActionListener` for the `ActionInterpretor` class. Your program should compute the BMI when the user clicks the `button_BMIcalculation` and display the result at a panel of BMI result at the bottom of your GUI.

     i. Remark: The *Body Mass Index* (*BMI*) measures body fat based on `height` (m) and `weight` (kg).

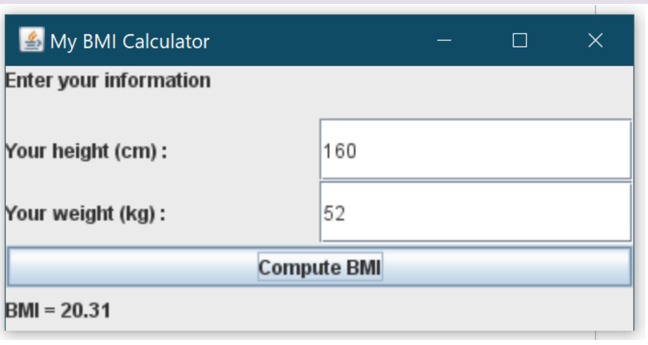
        The formula of BMI calculation: `kg/m2` .

     ii. For example, if your `weight` is 90 kg and your `height` is 1.77 m, your BMI is $90/1.77^2$ or 28.7

  c. Your running output of the `BMICalculator` program should display as shown as follows:

👉 To submit this exercise, students **MUST** present:

student's height and student's weight as shown as below:


My BMI Calculator
Enter your information
Your height (cm) : 160
Your weight (kg) : 52
Compute BMI
BMI = 20.31

Exercise 3: (*2 points*)

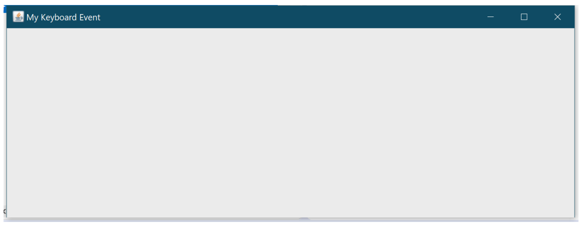
❖ **Java Project**: <Student_ID>_LAB9_KeyboardEvent

❖ **Objective**: To learn how to register an appropriate listener to the source, and implement appropriate methods and their details for the specified listener to perform the assigned task.

❖ **Hint**: Using `getKeychar()` method of the `KeyEvent` to get a character from the keyboard. Append this `char` to the `String message` defined as a property of this class and redisplay this message at the specified location.

❖ **Instruction**: Create a java project and write code in the following tasks.

a. Add a new java class `KeyboardEvent` that makes the following GUI design:



b. Complete an override method `KeyboardEvent` that repeatedly gets a character from the keyboard one at the time and displays it on the panel.

Remark: The panel displays a BLUE font and starts at location (`0,100`).

c. Add a new java class `KeyboardEventTesting` and write a `main` method that has a frame to add an object `KeyboardEvent`.

Remark: a skeleton-code of the frame is as below:

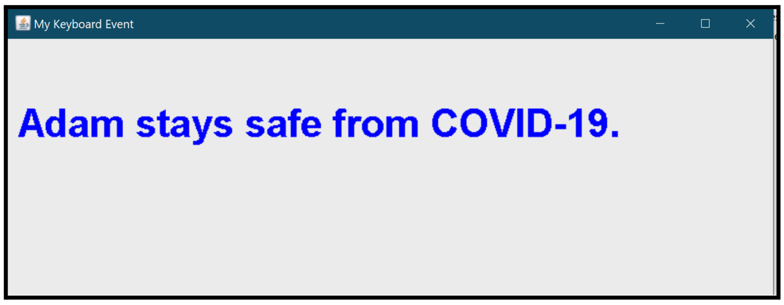
```
JFrame frame = new JFrame("My Keyboard Event");

____________________________________ //create an object of KeyboardEvent
frame.add(___________________________); //add add the object into the frame

frame.setSize(800, 300); //set a frame's resolution
frame.setLocationRelativeTo(null); //set a location at center the frame
frame.setVisible(true); //set visible
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); //set default Close
Operation
```

Asst. Prof. Dr. Sasiporn Usanavasin, Dr. Jessada Karnjana, Dr. Kasorn Galajit and Dr. Akkarawoot Takhom {sasiporn.us, akkharawoot.aj, jessada.aj, kasorn.aj}@siit.tu.ac.th
School of Information, Computer, and Communication Technology, Sirindhorn International Institute of Technology, Thammasat University.

👉 To submit this exercise 3, students **MUST** present:

your student's nickname with your situation during the COVID-19 outbreak.

`<Your FirstName> stays safe from COVID-19.`



Adam stays safe from COVID-19.

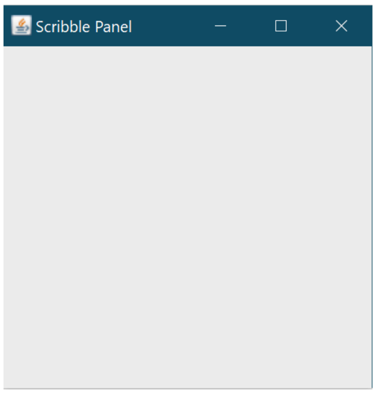
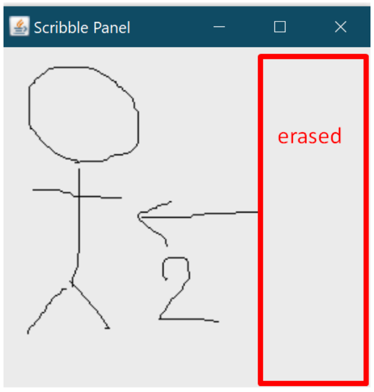
Exercise 4: (*2 points*)

- **Java Project**: <Student_ID>_ LAB9_ScribblePanel
- **Objective**: To learn how to register an appropriate listener to the source, and implement appropriate methods and their details for the specified listener to perform the assigned task.
- **Suggestion**: To define your variables name as below format:

  `<object>_<name>`

  For example, `public JButton button_Left = new JButton("Left");`

- **Instruction**: Create a java project and write code in the following tasks.

a. Add a new java class `ScribblePanel` for drawing using a mouse, as shown in the following GUI design: 	b. Draw by dragging with the left mouse button pressed; **erase** by dragging with the right button pressed. 

Asst. Prof. Dr. Sasiporn Usanavasin, Dr. Jessada Karnjana, Dr. Kasorn Galajit and Dr. Akkarawoot Takhom {sasiporn.us, akkharawoot.aj, jessada.aj, kasorn.aj}@siit.tu.ac.th
School of Information, Computer, and Communication Technology, Sirindhorn International Institute of Technology, Thammasat University.

Exercise 5: (*2 points*)

❖ **Java Project**: <Student_ID>_ LAB9_ScribblePanel

❖ **Objective**: To learn how to register an appropriate listener to the source, and implement appropriate methods and their details for the specified listener to perform the assigned task.

❖ **Suggestion**: To define your variables name as below format:

`<object>_<name>`

For example, `public JButton button_Left = new JButton("Left");`

❖ **Instruction**: Write code in the following tasks that continue from Exercise 4,

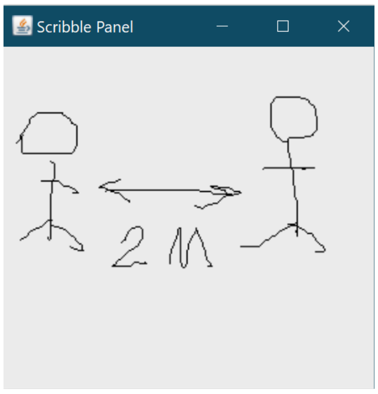
a. Add some code to the `ScribblePenel` so that it **erases** all drawing when the user *clicks* on the panel.

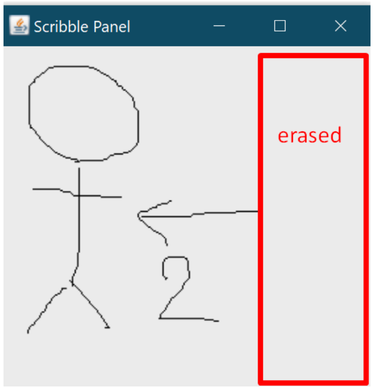
b. Add a new java class `ScribblePenelTesting` and write a `main` method that has a frame to add an object `ScribblePenel` at the CENTER of the frame's layout .

Remark: a skeleton-code of the frame is as below:

```
JFrame frame = new JFrame("Scribble Panel");

____________________________________; //create an object of ScribblePenel
frame.add(_________ , _________.CENTER); //add the object into CENTER of JFrame

frame.setSize(300, 300); //set a frame's resolution
frame.setLocationRelativeTo(null); //set a location at center the frame
frame.setVisible(true); //set visible
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); //set default Close Operation
```

👉 Students **MUST** present a GUI illustrating their interpretation of "Social Distancing."


Scribble Panel
2 m


Scribble Panel
2
erased