[DES103-LAB08-Review]

LAB REVIEW & EXERCISES {Event-Driven Programming 1}

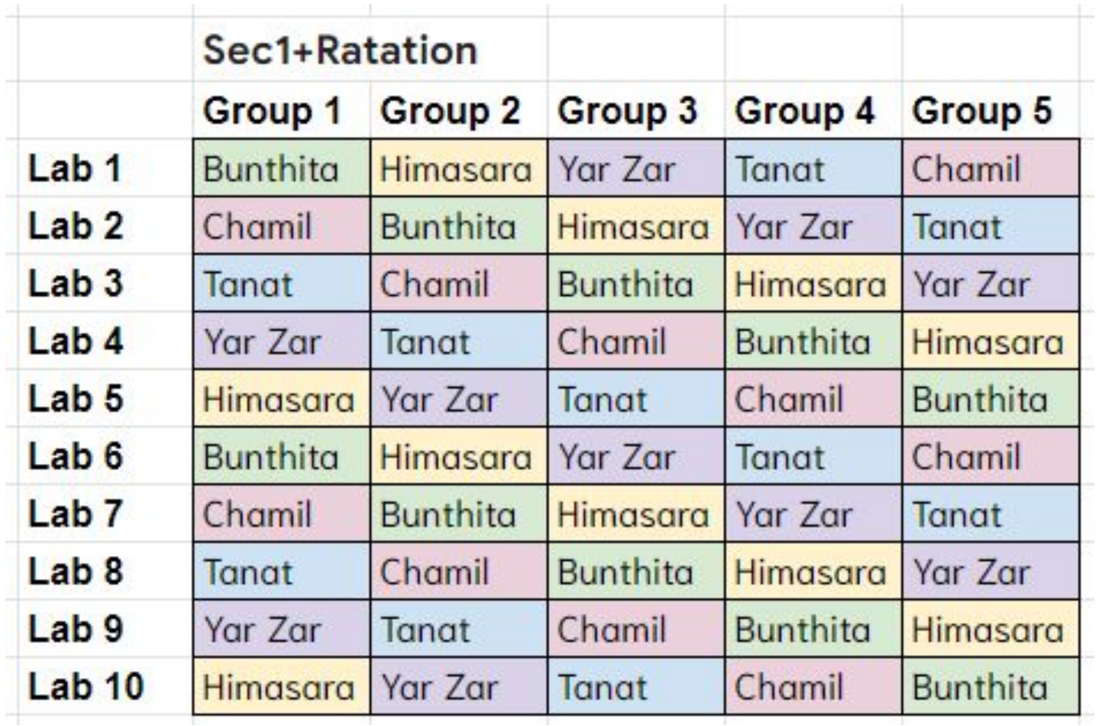
DES103: Object-Oriented Programming Laboratory (Java Lab)
Asst. Prof. Dr. Sasiporn Usanavasin, Dr. Jessada Karnjana, Dr. Kasorn Galajit and Dr. Akkarawoot Takhom
School of Information, Computer, and Communication Technology,
Sirindhorn International Institute of Technology
{sasiporn.us, akkharawoot.aj, jessada.aj, kasorn.aj}@siit.tu.ac.th

Calendar 2022/2

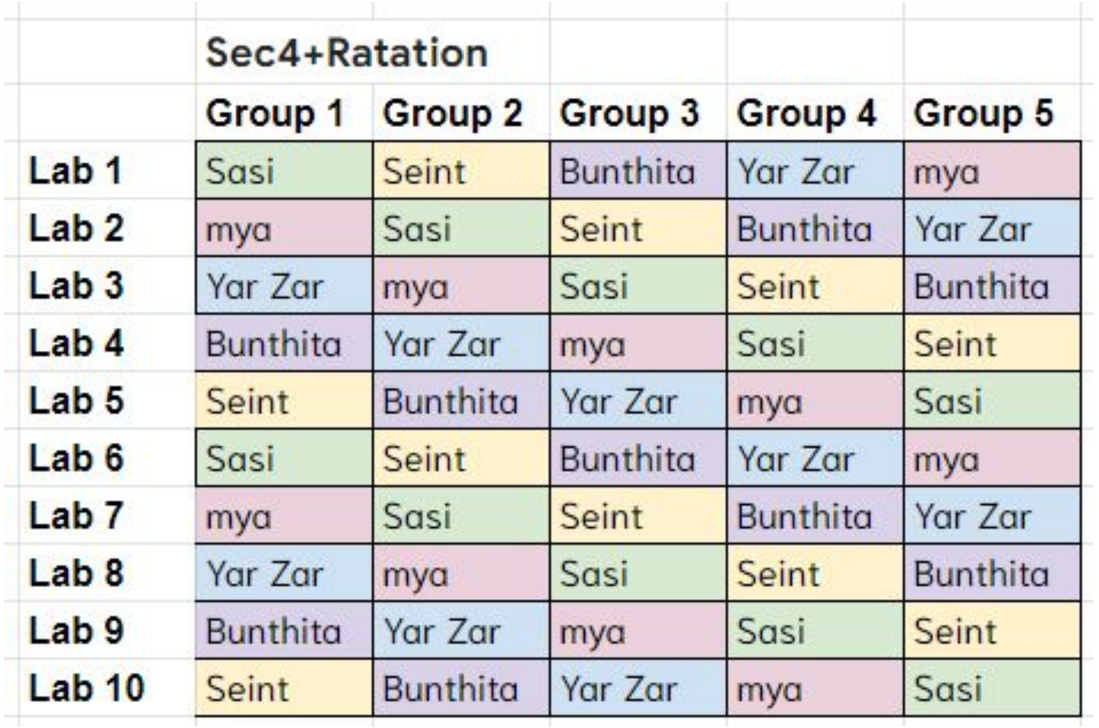
Lab Mon	Lab Tue	DES103
23–Jan	24–Jan	LAB01–Class Component, basic printout statement, the dot operator, the new operator
30–Jan	31–Jan	LAB02–Class components in more details, the this keyword, instance
6–Feb	7–Feb	LAB03–Inheritance, super class, constructor chaining
13–Feb	14–Feb	LAB04–Polymorphism, abstract, interface
20–Feb	21–Feb	LAB05–Array of Objects, and Visibility modifiers
27–Feb	28–Feb	No class due to Midterm
6–Mar	7–Mar	No class due to Makha Bucha day
13–Mar	14–Mar	LAB06–JContainer, JComponents, and Layout Managers
20–Mar	21–Mar	LAB07-Graphics
27–Mar	**28–Mar**	**LAB08 -Event Driven Programming I**
3–Apr	4–Apr	LAB09–Event Driven Programming II
10–Apr	11–Apr	Lecture Break
17–Apr	18–Apr	Lecture Break
24–Apr	25–Apr	LAB10–Timer
1–May	2–May	Final–presentation Exam

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

TAs Rotation



	Sec1+Ratation				
	Group 1	Group 2	Group 3	Group 4	Group 5
Lab 1	Bunthita	Himasara	Yar Zar	Tanat	Chamil
Lab 2	Chamil	Bunthita	Himasara	Yar Zar	Tanat
Lab 3	Tanat	Chamil	Bunthita	Himasara	Yar Zar
Lab 4	Yar Zar	Tanat	Chamil	Bunthita	Himasara
Lab 5	Himasara	Yar Zar	Tanat	Chamil	Bunthita
Lab 6	Bunthita	Himasara	Yar Zar	Tanat	Chamil
Lab 7	Chamil	Bunthita	Himasara	Yar Zar	Tanat
Lab 8	Tanat	Chamil	Bunthita	Himasara	Yar Zar
Lab 9	Yar Zar	Tanat	Chamil	Bunthita	Himasara
Lab 10	Himasara	Yar Zar	Tanat	Chamil	Bunthita



	Sec4+Ratation				
	Group 1	Group 2	Group 3	Group 4	Group 5
Lab 1	Sasi	Seint	Bunthita	Yar Zar	mya
Lab 2	mya	Sasi	Seint	Bunthita	Yar Zar
Lab 3	Yar Zar	mya	Sasi	Seint	Bunthita
Lab 4	Bunthita	Yar Zar	mya	Sasi	Seint
Lab 5	Seint	Bunthita	Yar Zar	mya	Sasi
Lab 6	Sasi	Seint	Bunthita	Yar Zar	mya
Lab 7	mya	Sasi	Seint	Bunthita	Yar Zar
Lab 8	Yar Zar	mya	Sasi	Seint	Bunthita
Lab 9	Bunthita	Yar Zar	mya	Sasi	Seint
Lab 10	Seint	Bunthita	Yar Zar	mya	Sasi

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

Regulations

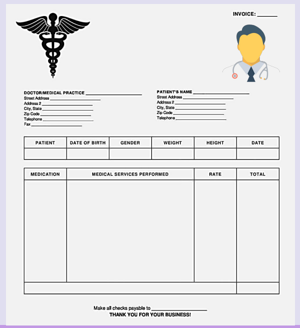
Classroom is expected:

1. **Punctuality**
2. **Responsibility**
3. **Availability with Evidence,**
   IF students have accident and emergency cases, missing class, Student must prepare **Evidence**
   e.g., Screen Capture, Receipt, Sending message.
   *Turning-in overdue time hands-on practices
   will be checked by modified date in file properties.

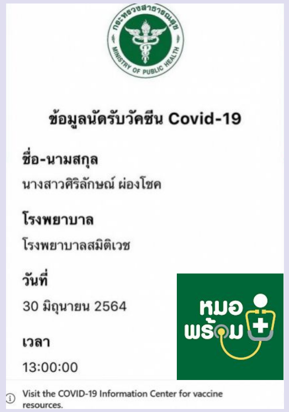
ACADEMIC REGULATIONS

<u>This course applied for conduct Score and disciplinary Actions :</u>
Warning Probation Status
https://www.siit.tu.ac.th/academics.php?sid=33&ssid=17

Evidence-Based Diagnosis


INVOICE:
DOCTOR/MEDICAL PRACTICE
PATIENT'S NAME
PATIENT
DATE OF BIRTH
GENDER
WEIGHT
HEIGHT
DATE
MEDICATION
MEDICAL SERVICES PERFORMED
RATE
TOTAL
THANK YOU FOR YOUR BUSINESS!
Medical Receipt

Timestamp


ข้อมูลนัดรับวัคซีน Covid-19

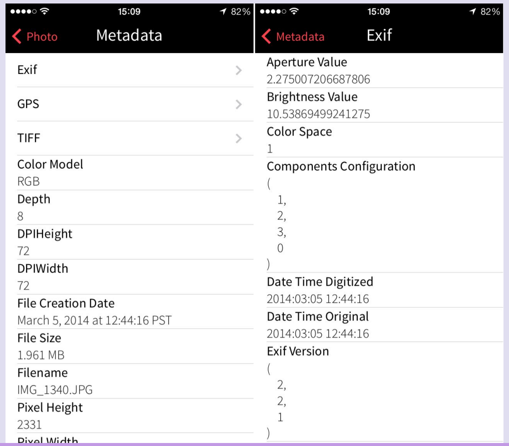
ชื่อ-นามสกุล
นางสาวศิริลักษณ์ ผ่องโชค

โรงพยาบาล
โรงพยาบาลสมิติเวช

วันที่
30 มิถุนายน 2564

เวลา
13:00:00

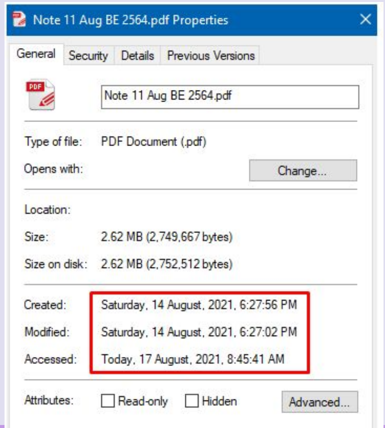
Visit the COVID-19 Information Center for vaccine resources.


15:09 82%
Photo | Metadata

Exif
GPS
TIFF
Color Model
RGB
Depth
8
DPIHeight
72
DPIWidth
72
File Creation Date
March 5, 2014 at 12:44:16 PST
File Size
1.961 MB
Filename
IMG_1340.JPG
Pixel Height
2331
Pixel Width

15:09 82%
Metadata | Exif

Aperture Value
2.275007206687806
Brightness Value
10.53869499241275
Color Space
1
Components Configuration
```
(
  1,
  2,
  3,
  0
)
```
Date Time Digitized
2014:03:05 12:44:16
Date Time Original
2014:03:05 12:44:16
Exif Version
```
(
  2,
  2,
  1
)
```


Note 11 Aug BE 2564.pdf Properties

General | Security | Details | Previous Versions

Note 11 Aug BE 2564.pdf

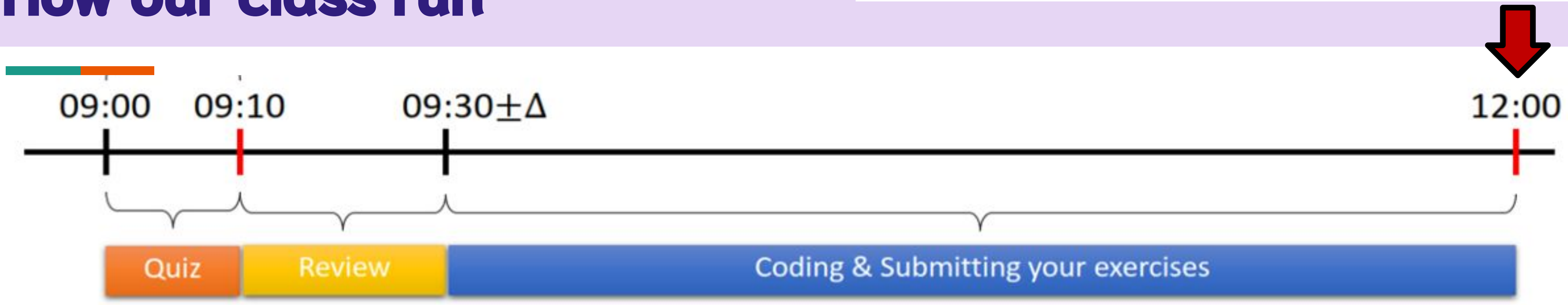
Type of file:	PDF Document (.pdf)
Opens with:	Change...
Location:	
Size:	2.62 MB (2,749,667 bytes)
Size on disk:	2.62 MB (2,752,512 bytes)
Created:	Saturday, 14 August, 2021, 6:27:56 PM
Modified:	Saturday, 14 August, 2021, 6:27:02 PM
Accessed:	Today, 17 August, 2021, 8:45:41 AM
Attributes:	☐ Read-only ☐ Hidden Advanced...

File properties (Metadata) of Student's Turning-in Files

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

How our class run

No scores for any late submission.



- [9.00–9.10] Students log in their Google classroom for doing lab quiz
- [9.10~9.30] Instructor reviews lab objectives and exercises
- [9.30~12.00] Students ask TA to check their code
    - TA wiil ask <u>5 questions</u> for evaluating student’s understanding.
    - Students submits your code into the Google classroom.

LAB REVIEW

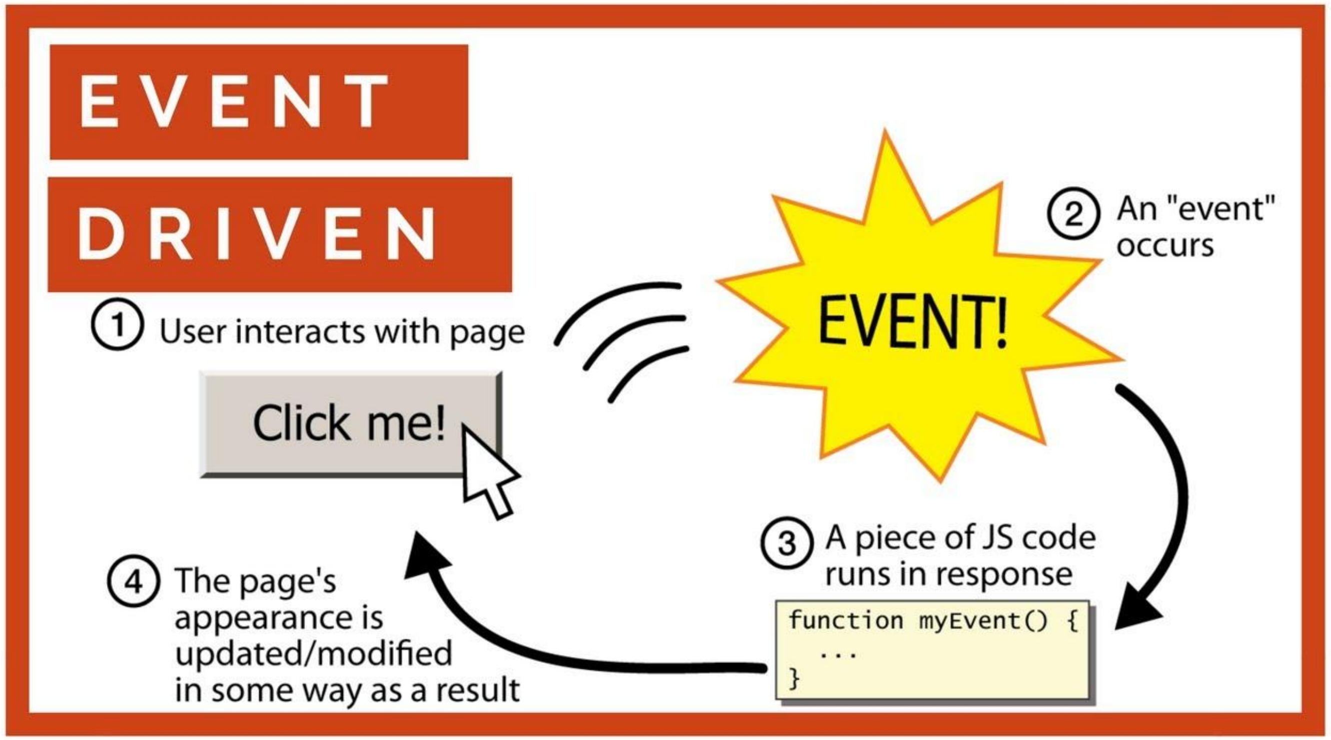
Learning Objectives

1. To understand the concept of event-driven programming
2. To be able to design and build usable Graphical User Interfaces (GUI)
3. To recall Java Graphics concept
4. To learn how to define a listener
5. To learn how to register an appropriate listener to the source
6. To learn how to implement appropriate methods and their details for the specified listener to perform the assigned task.
7.

Remark: A *pointer finger* (☞) refers to an explanation between students and their teaching assistants (TA).

9.1 Event-Driven Programming

- **Event-Driven Programming:**
  A programming of which the code is executed upon activation of events.


EVENT
DRIVEN
① User interacts with page
Click me!
EVENT!
② An "event" occurs
③ A piece of JS code runs in response
function myEvent() {
...
}
④ The page's appearance is updated/modified in some way as a result

Reference: *What is Event Driven Architecture, https://www.youtube.com/watch?v=SnO9dTbF6hM*

9.1 Event-Driven Programming

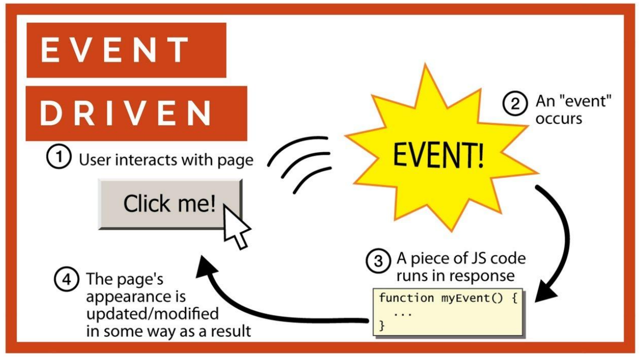
- In this lab, we will learn the basic usage of an event-driven programming.
- We start with learning the Java interface `ActionListener`.

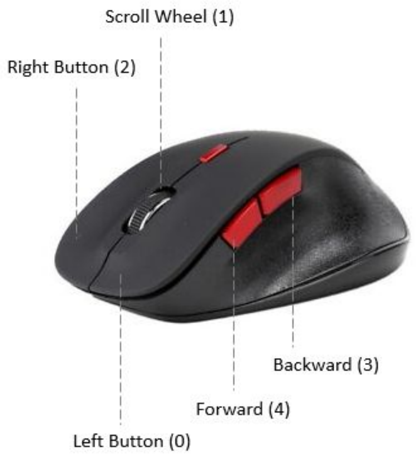
- This interface declares one method, i.e. `actionPerformed()` as follows:

```
public interface ActionListener extends...{
    void actionPerformed(ActionEvent e);
}
```

8.2 Events

- **An event** can be defined as
  a type of signal to the program telling that something has happened.


EVENT
DRIVEN
1 User interacts with page
Click me!
EVENT!
2 An "event" occurs
3 A piece of JS code runs in response
function myEvent() {
...
}
4 The page's appearance is updated/modified in some way as a result



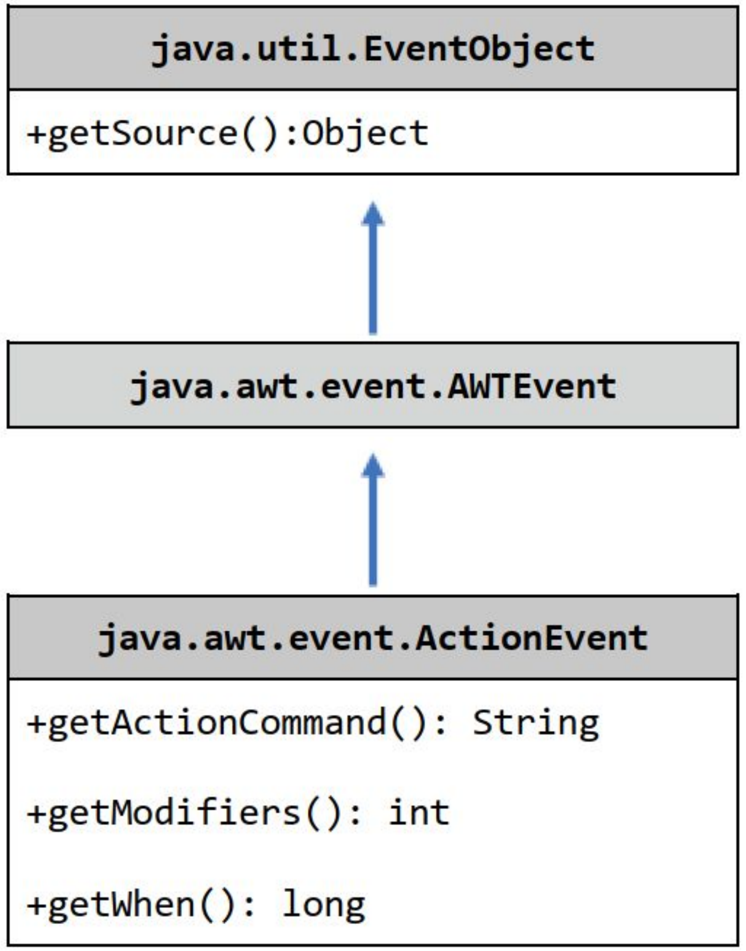
- **The event** is generated
  1. by **external user actions**
     such as mouse movements, mouse clicks, and keystrokes, or
  2. by **the operating system**, such as a timer.

8.3 Examples of Sources and Events

- To learn how to register an appropriate listener to the source

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
Window opened, closed, etc.	Window	WindowEvent
Mouse pressed, released,dragged etc.	Mouse	MouseEvent
Key released, pressed, etc.	Keyboard	KeyEvent

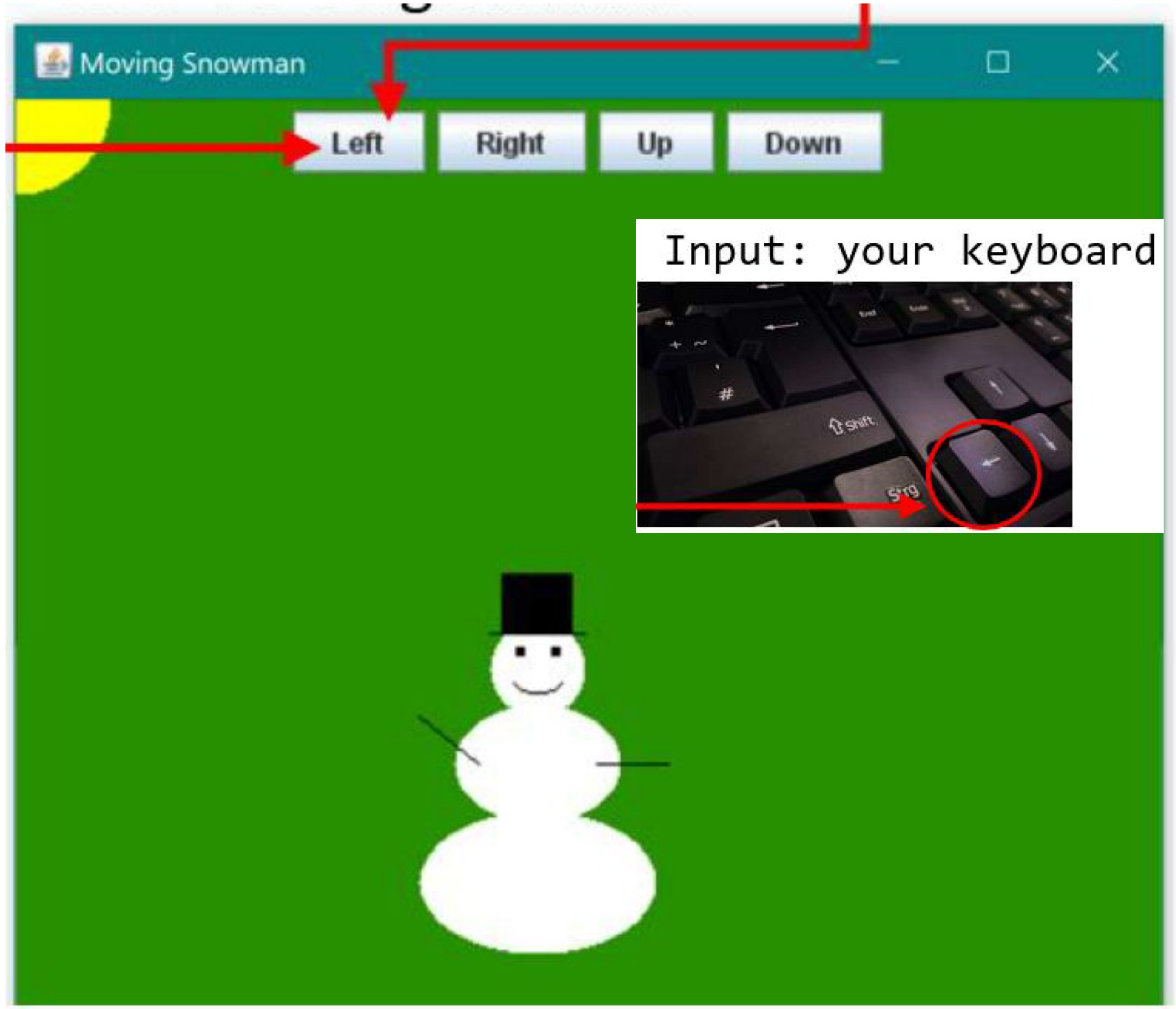
8.4 EventObject: Type-1 ActionEvent



java.util.EventObject
+getSource():Object

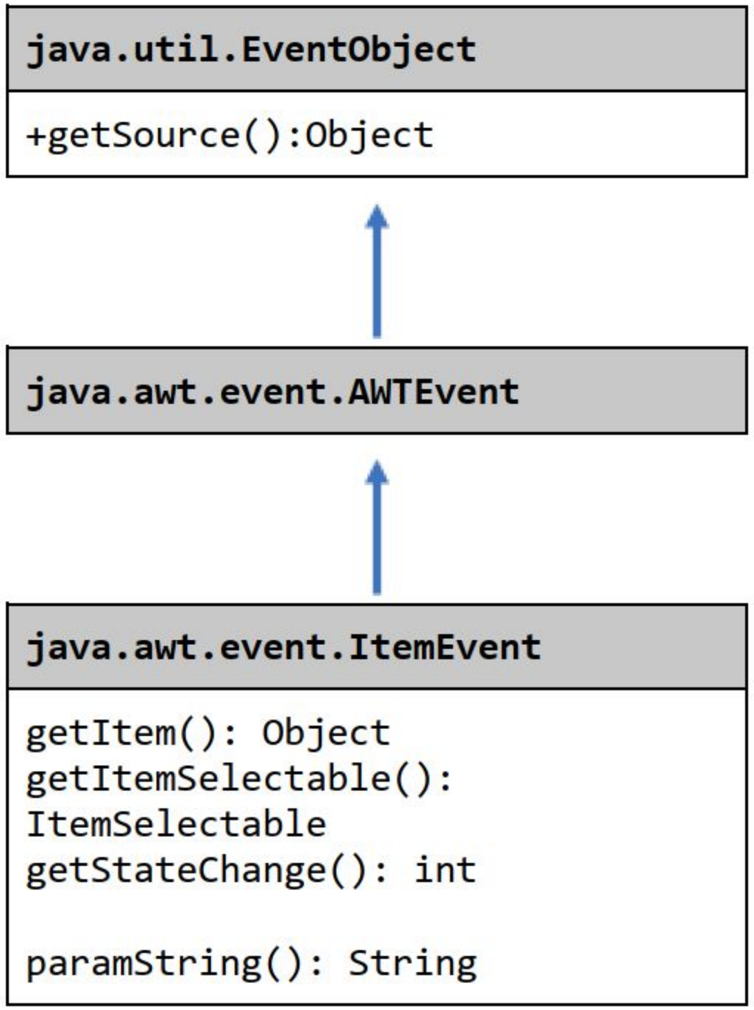
java.awt.event.AWTEvent

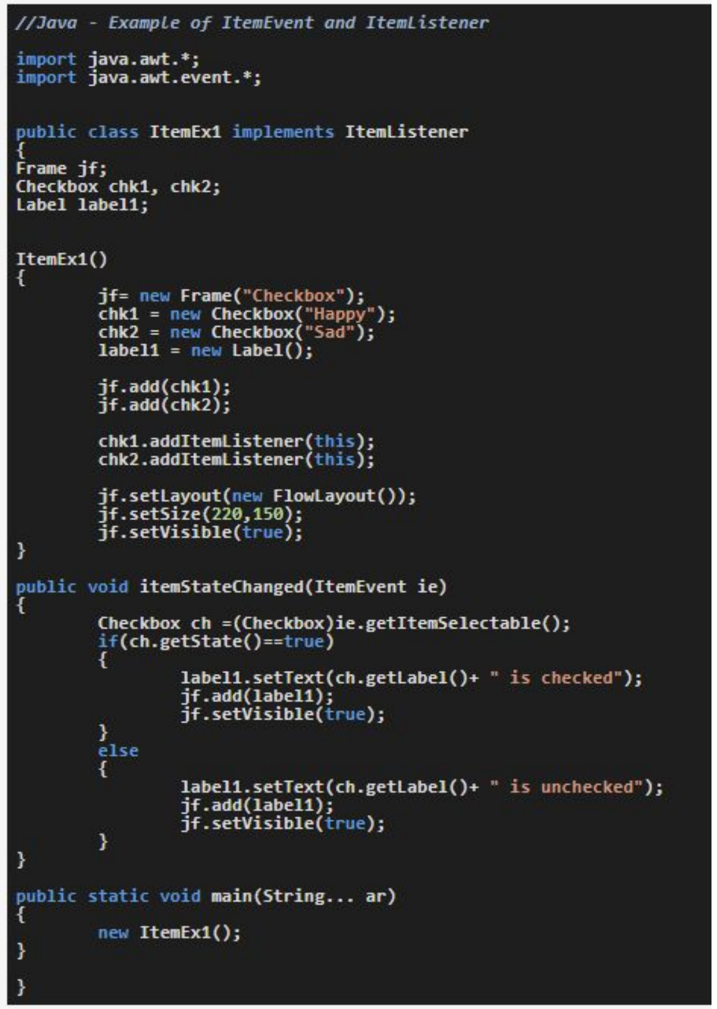
java.awt.event.ActionEvent
+getActionCommand(): String
+getModifiers(): int
+getWhen(): long



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

8.4 EventObject: Type-2 ItemEvent





```
//Java - Example of ItemEvent and ItemListener

import java.awt.*;
import java.awt.event.*;


public class ItemEx1 implements ItemListener
{
Frame jf;
Checkbox chk1, chk2;
Label label1;


ItemEx1()
{
	jf= new Frame("Checkbox");
	chk1 = new Checkbox("Happy");
	chk2 = new Checkbox("Sad");
	label1 = new Label();

	jf.add(chk1);
	jf.add(chk2);

	chk1.addItemListener(this);
	chk2.addItemListener(this);

	jf.setLayout(new FlowLayout());
	jf.setSize(220,150);
	jf.setVisible(true);
}

public void itemStateChanged(ItemEvent ie)
{
	Checkbox ch =(Checkbox)ie.getItemSelectable();
	if(ch.getState()==true)
	{
		label1.setText(ch.getLabel()+ " is checked");
		jf.add(label1);
		jf.setVisible(true);
	}
	else
	{
		label1.setText(ch.getLabel()+ " is unchecked");
		jf.add(label1);
		jf.setVisible(true);
	}
}

public static void main(String... ar)
{
	new ItemEx1();
}

}
```

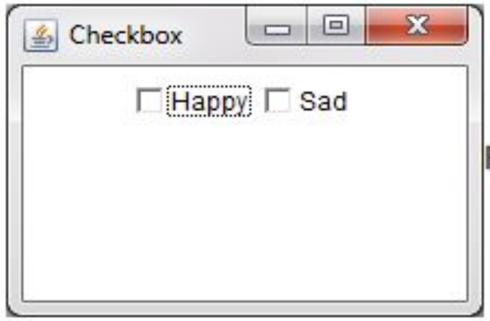
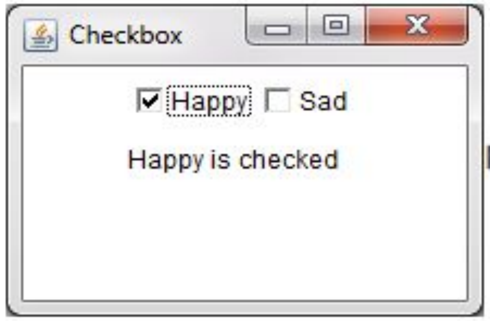
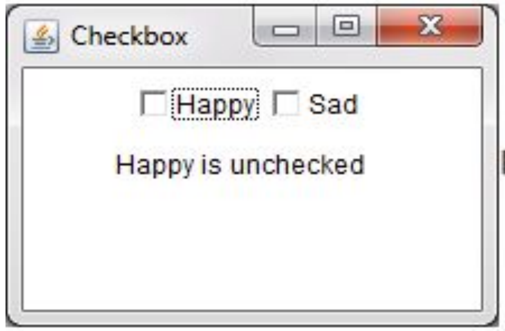


Figure 1



Happy is checked

Figure 2



Happy is unchecked

Figure 3

Reference: *ItemEvent and ItemListener, https://www.decodejava.com/java-itemlistener.htm*

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

8.4 EventObject: Type-2 MouseEvent

java.awt.event.InputEvent
+getWhen(): long
+isAltDown(): boolean
+isControlDown(): boolean
+isMetaDown(): boolean
+isShiftDown(): boolean

↑

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

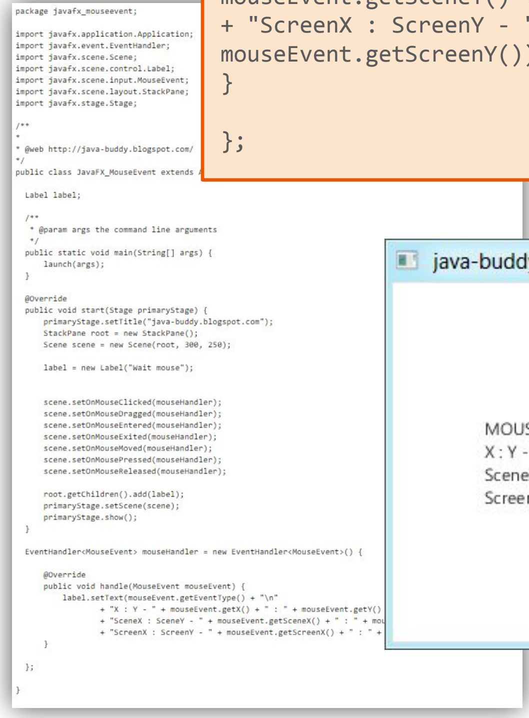
```
scene.setOnMouseClicked(mouseHandler);
scene.setOnMouseDragged(mouseHandler);
scene.setOnMouseEntered(mouseHandler);
scene.setOnMouseExited(mouseHandler);
scene.setOnMouseMoved(mouseHandler);
scene.setOnMousePressed(mouseHandler);
scene.setOnMouseReleased(mouseHandler);
```

java.awt.event.KeyEvent

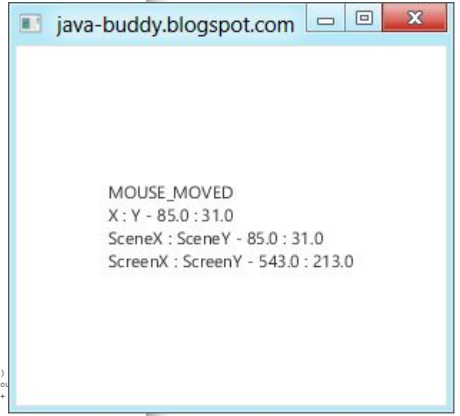
↑

java.awt.event.MouseEvent
+getKeyChar(): char
+getKeyCode(): int
+getKeyLocation():int
<u>+getKeyText(int keyCode) :String</u>
<u>+getKeyModifiersText(int modifiers) : String</u>

```
EventHandler<MouseEvent> mouseHandler = new
EventHandler<MouseEvent>() {

@Override
public void handle(MouseEvent mouseEvent) {
label.setText(mouseEvent.getEventType() + "\n"
+ "X : Y - " + mouseEvent.getX() + " : " + mouseEvent.getY() + "\n"
+ "SceneX : SceneY - " + mouseEvent.getSceneX() + " : " +
mouseEvent.getSceneY() + "\n"
+ "ScreenX : ScreenY - " + mouseEvent.getScreenX() + " : " +
mouseEvent.getScreenY());
}

};
```

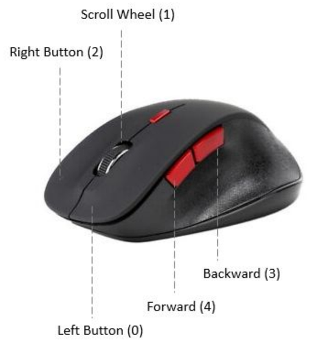


```
package javafx_mouseevent;

import javafx.application.Application;
import javafx.event.EventHandler;
import javafx.scene.Scene;
import javafx.scene.control.Label;
import javafx.scene.input.MouseEvent;
import javafx.scene.layout.StackPane;
import javafx.stage.Stage;

/**
*
* @web http://java-buddy.blogspot.com/
*/
public class JavaFX_MouseEvent extends A

  Label label;

  /**
   * @param args the command line arguments
   */
  public static void main(String[] args) {
      launch(args);
  }

  @Override
  public void start(Stage primaryStage) {
      primaryStage.setTitle("java-buddy.blogspot.com");
      StackPane root = new StackPane();
      Scene scene = new Scene(root, 300, 250);

      label = new Label("Wait mouse");


      scene.setOnMouseClicked(mouseHandler);
      scene.setOnMouseDragged(mouseHandler);
      scene.setOnMouseEntered(mouseHandler);
      scene.setOnMouseExited(mouseHandler);
      scene.setOnMouseMoved(mouseHandler);
      scene.setOnMousePressed(mouseHandler);
      scene.setOnMouseReleased(mouseHandler);

      root.getChildren().add(label);
      primaryStage.setScene(scene);
      primaryStage.show();
  }

  EventHandler<MouseEvent> mouseHandler = new EventHandler<MouseEvent>() {

      @Override
      public void handle(MouseEvent mouseEvent) {
          label.setText(mouseEvent.getEventType() + "\n"
                  + "X : Y - " + mouseEvent.getX() + " : " + mouseEvent.getY()
                  + "SceneX : SceneY - " + mouseEvent.getSceneX() + " : " + mou
                  + "ScreenX : ScreenY - " + mouseEvent.getScreenX() + " : " +
      }

  };

}
```



java-buddy.blogspot.com

```
MOUSE_MOVED
X : Y - 85.0 : 31.0
SceneX : SceneY - 85.0 : 31.0
ScreenX : ScreenY - 543.0 : 213.0
```

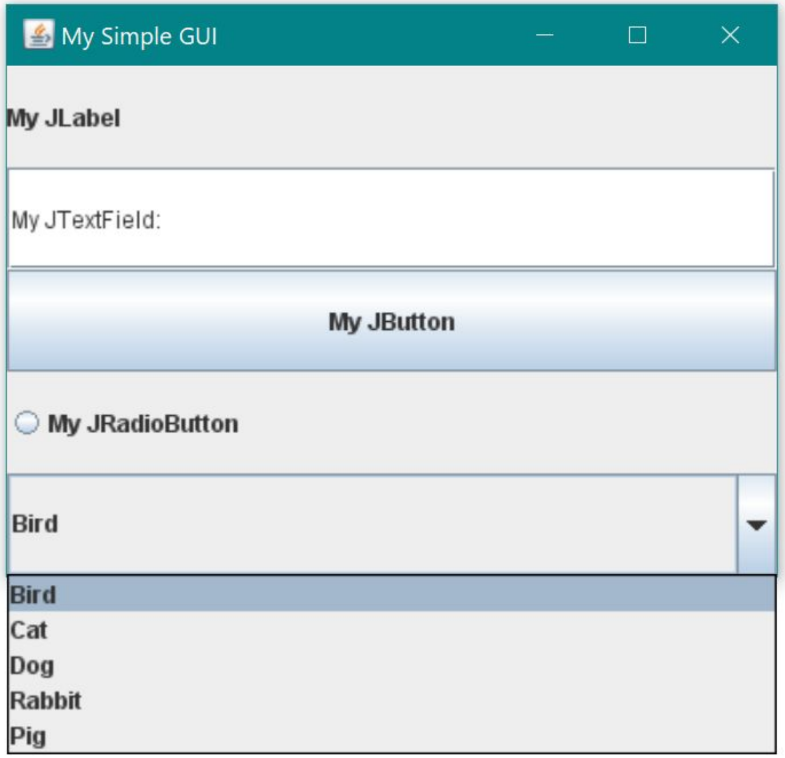


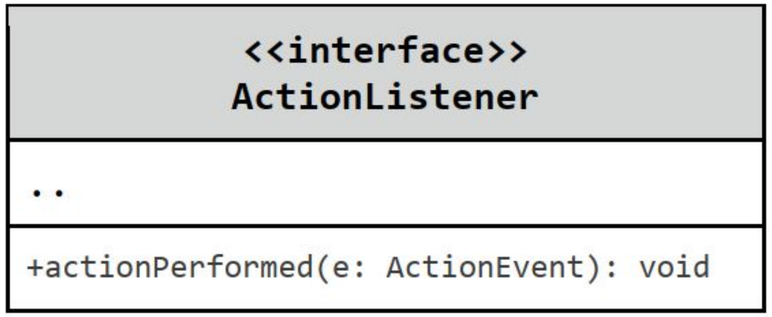
Reference: *Detect mouse event, http://java-buddy.blogspot.com/2012/03/detect-mouse-event.html*

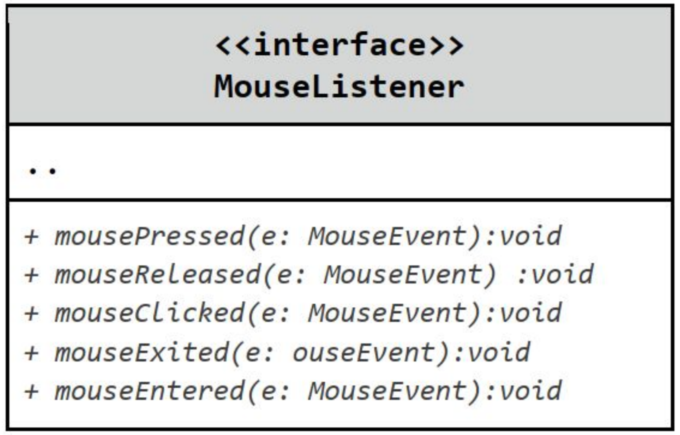
Asst.Prof.Dr.Sasiporn Usanavasin, Dr.Jessada Karnjana, Dr.Kasorn Galajit and Dr.Akkarawoot Takhom
{sasiporn.us, akkharawoot.aj, jessada.aj, kasorn.aj}@siit.tu.ac.th
School of Information, Computer, and Communication Technology, Sirindhorn International Institute of Technology

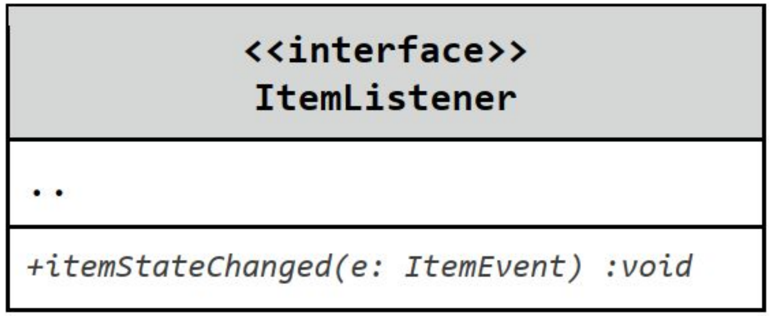
8.5 Interaction between Source and Listener

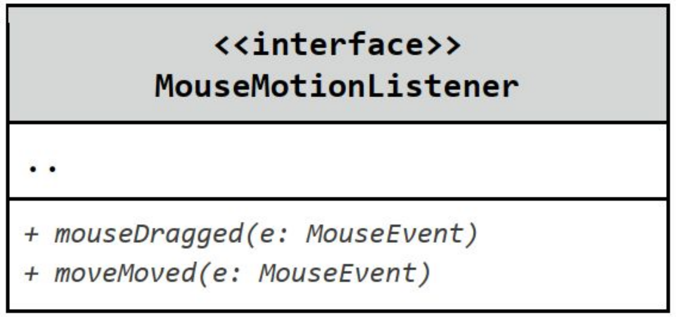
8.5.1 UML of Listener's Class Listeners are defined as <<interface>>




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


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


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

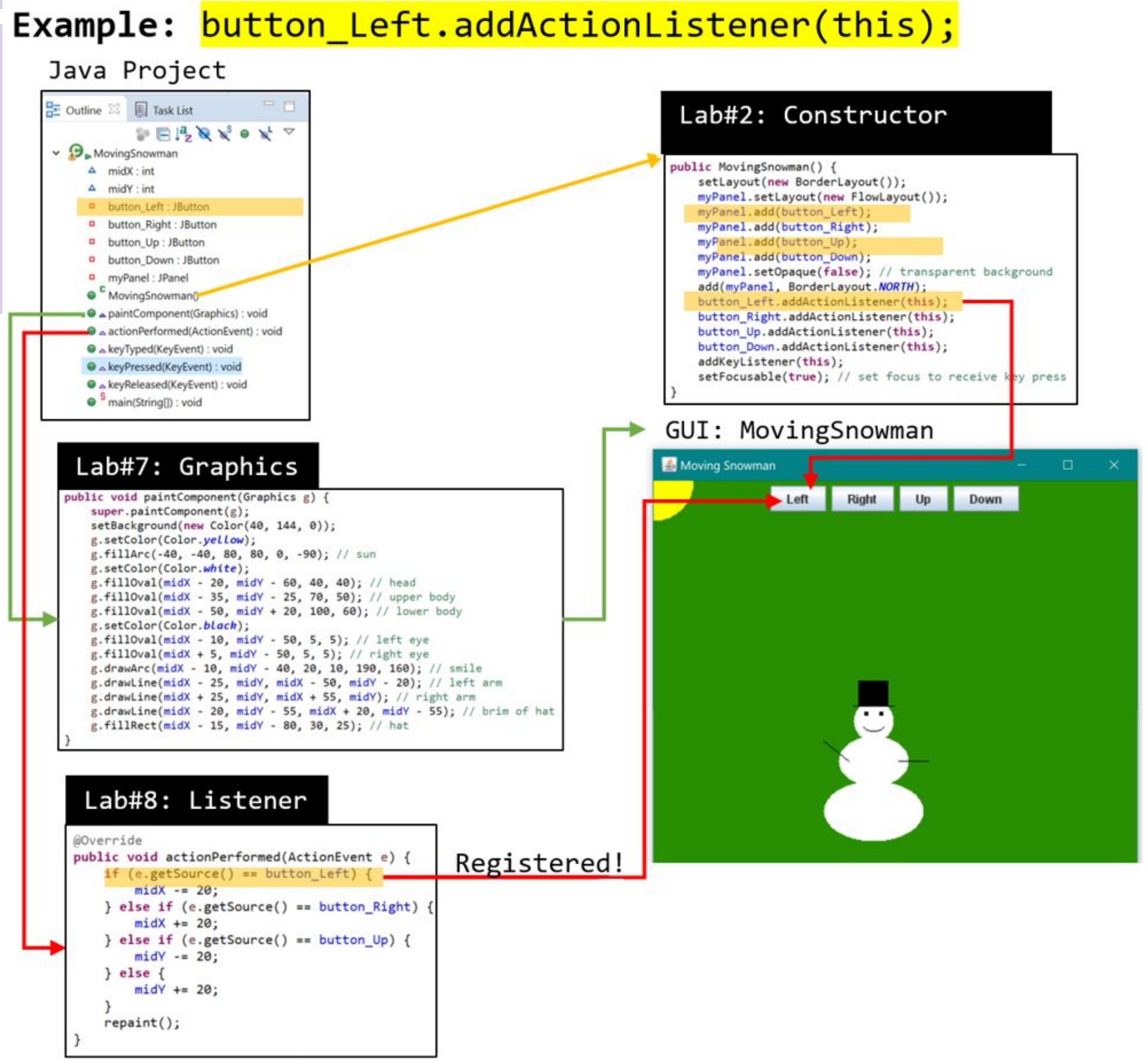

```
<<interface>>
MouseMotionListener
..
+ mouseDragged(e: MouseEvent)
+ moveMoved(e: MouseEvent)
```

8.5 Interaction between Source and Listener

8.5.2 Example:

`<object>.add<Listener>(this);`

Example: `button_Left.addActionListener(this);`



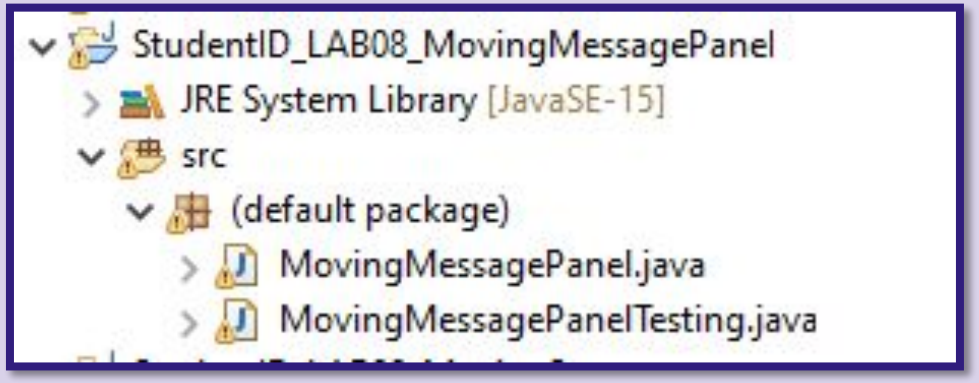
```
MovingSnowman
    midX : int
    midY : int
    button_Left : JButton
    button_Right : JButton
    button_Up : JButton
    button_Down : JButton
    myPanel : JPanel
    MovingSnowman()
    paintComponent(Graphics) : void
    actionPerformed(ActionEvent) : void
    keyTyped(KeyEvent) : void
  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

```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
}
```

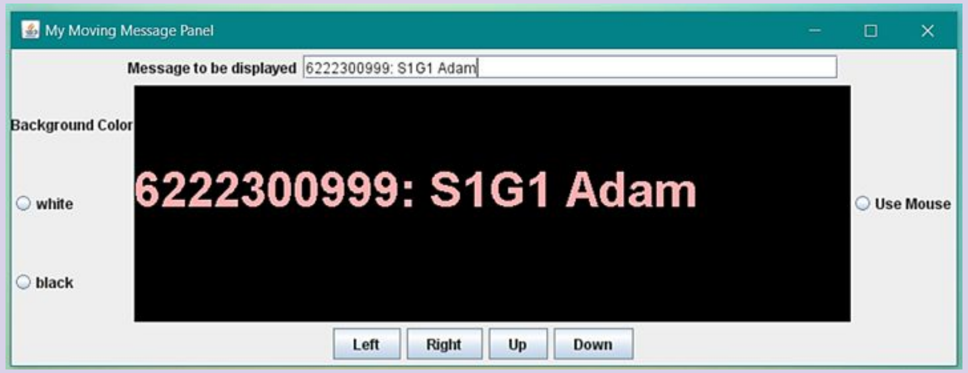
```java
public void paintComponent(Graphics g) {
    super.paintComponent(g);
    setBackground(new Color(40, 144, 0));
    g.setColor(Color.yellow);
    g.fillArc(-40, -40, 80, 80, 0, -90); // sun
    g.setColor(Color.white);
    g.fillOval(midX - 20, midY - 60, 40, 40); // head
    g.fillOval(midX - 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
}
```

```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();
}
```

LAB Exercises



```
StudentID_LAB08_MovingMessagePanel
  JRE System Library [JavaSE-15]
  src
    (default package)
      MovingMessagePanel.java
      MovingMessagePanelTesting.java
```



My Moving Message Panel

Message to be displayed 6222300999: S1G1 Adam

Background Color

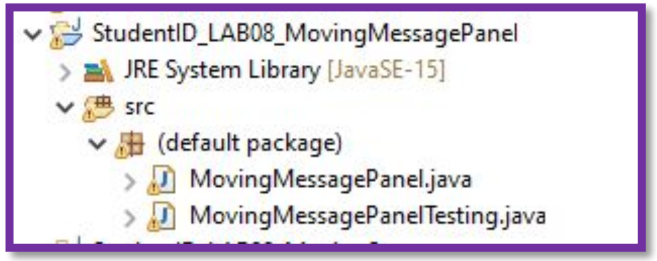
6222300999: S1G1 Adam

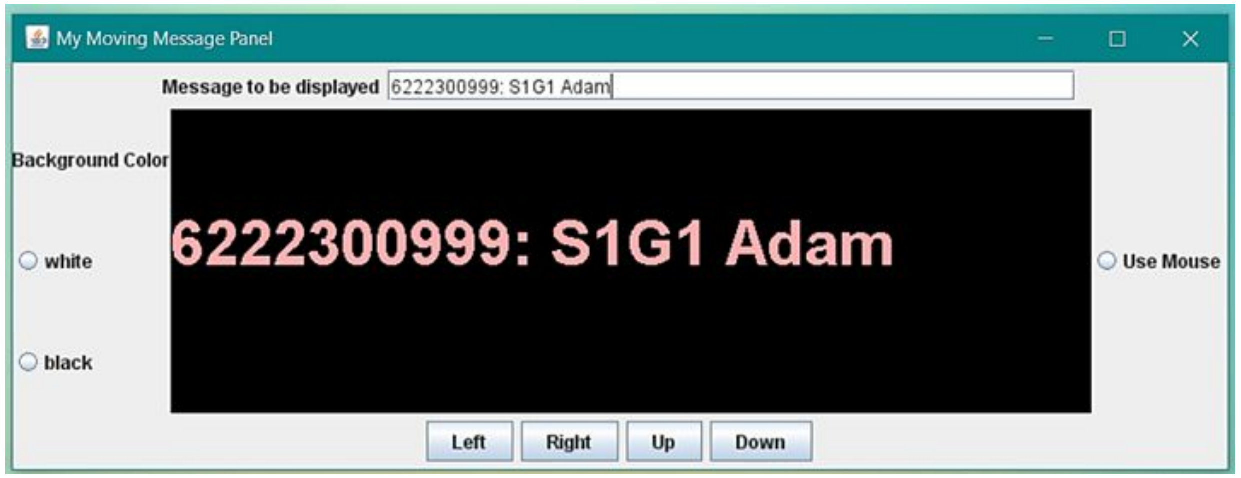
5 Exercises (10 points)

- ❖ Exercise 1 (2 points)
- ❖ Exercise 2 (2 points)
- ❖ Exercise 3 (2 points)
- ❖ Exercise 4 (2 points)
- ❖ Exercise 5 (2 points)

LAB EXERCISES

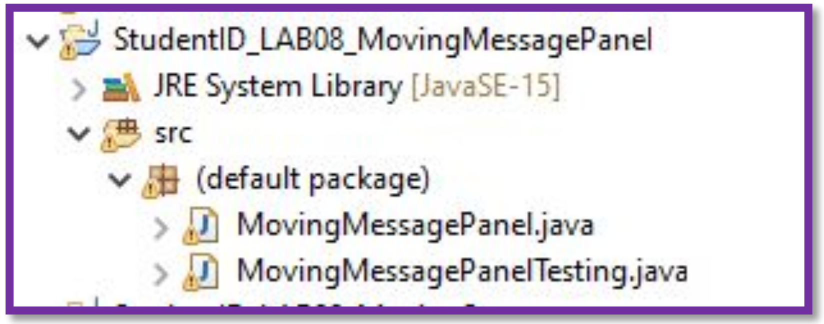
The name format of a java project:
"<StudentID>_LAB08_MovingMessagePanel" for exercise 1-5.


StudentID_LAB08_MovingMessagePanel
JRE System Library [JavaSE-15]
src
(default package)
MovingMessagePanel.java
MovingMessagePanelTesting.java

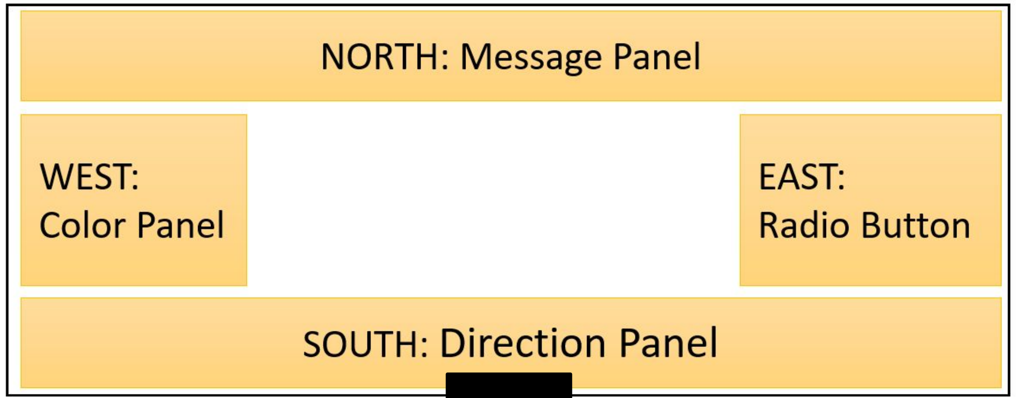


Exercise 1 (2 points)

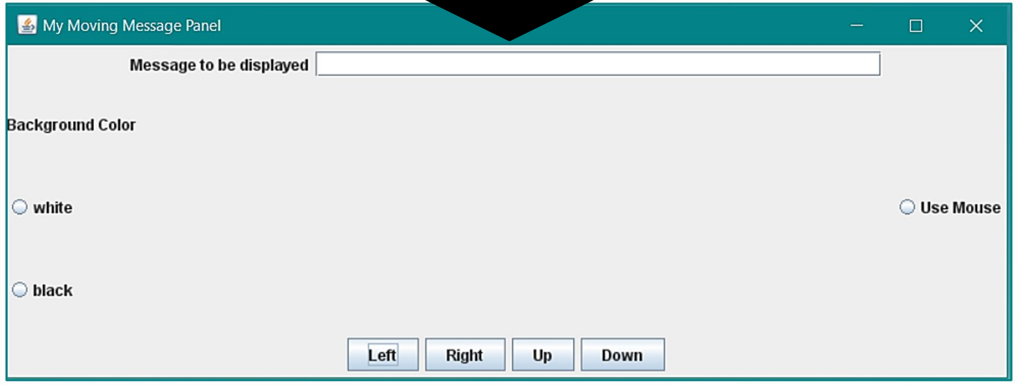
- **Project Name**: <Student_ID>_LAB08_MovingMessagePanel
- **Instruction**: Write code in the following tasks.


StudentID_LAB08_MovingMessagePanel
JRE System Library [JavaSE-15]
src
(default package)
MovingMessagePanel.java
MovingMessagePanelTesting.java

a) Add a new java class MovingMessagePanel that makes the following GUI design.

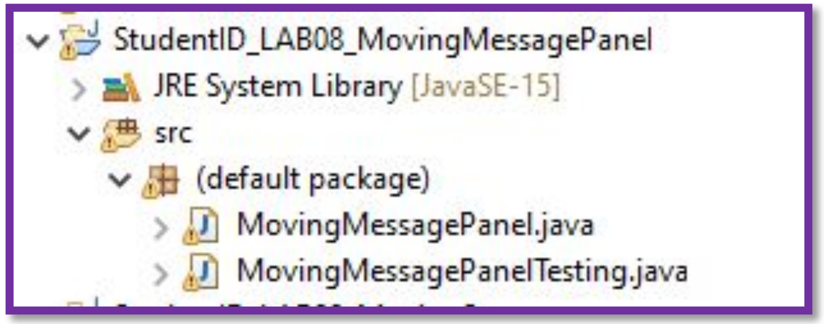


b) Add a new java class MovingMessagePanelTesting and write a main method for a running output of the MovingMessagePanel GUI.



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

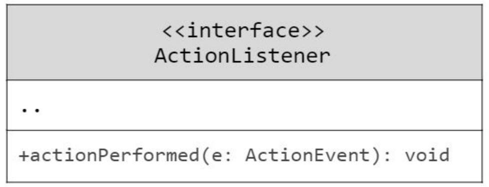
Exercise 2 (2 points)



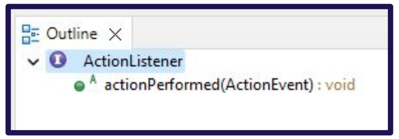
```
StudentID_LAB08_MovingMessagePanel
  JRE System Library [JavaSE-15]
  src
    (default package)
      MovingMessagePanel.java
      MovingMessagePanelTesting.java
```

- **Project Name**: <Student_ID>_LAB08_MovingMessagePanel
- **Instruction**: Write code in the following tasks.

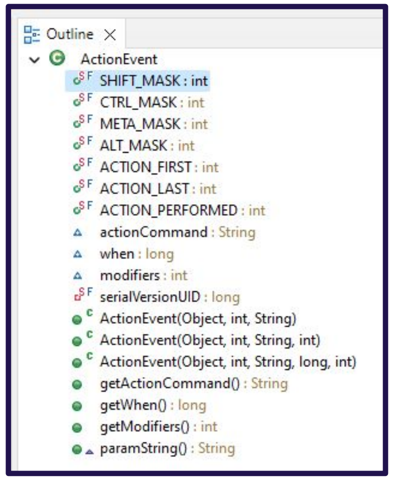
a) Make the MovingMessagePanel class to be a subclass of the interface ActionListener.



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



```
Outline
  ActionListener
    actionPerformed(ActionEvent) : void
```

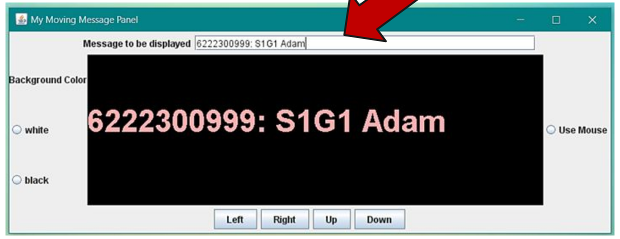


```
Outline
  ActionEvent
    SHIFT_MASK : int
    CTRL_MASK : int
    META_MASK : int
    ALT_MASK : int
    ACTION_FIRST : int
    ACTION_LAST : int
    ACTION_PERFORMED : int
    actionCommand : String
    when : long
    modifiers : int
    serialVersionUID : long
    ActionEvent(Object, int, String)
    ActionEvent(Object, int, String, int)
    ActionEvent(Object, int, String, long, int)
    getActionCommand() : String
    getWhen() : long
    getModifiers() : int
    paramString() : String
```

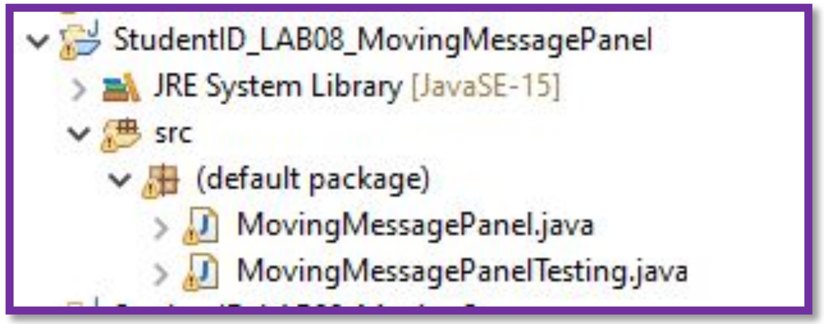
b) Register the textfield with itself which acts as the ActionListener using an appropriate method.

c) Override the implementation details of the overridden method of ActionListener.

Your program should get the text from the textfield when the user writes a text into the textfield box and hits enter.

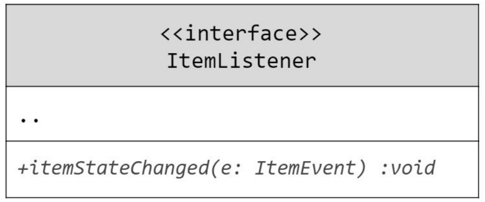


Exercise 3 (2 points)


```
StudentID_LAB08_MovingMessagePanel
  JRE System Library [JavaSE-15]
  src
    (default package)
      MovingMessagePanel.java
      MovingMessagePanelTesting.java
```

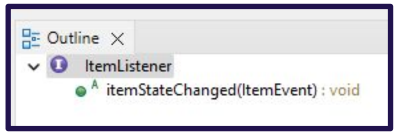
- **Project Name**: <Student_ID>_LAB08_MovingMessagePanel
- **Instruction**: Write code in the following tasks.

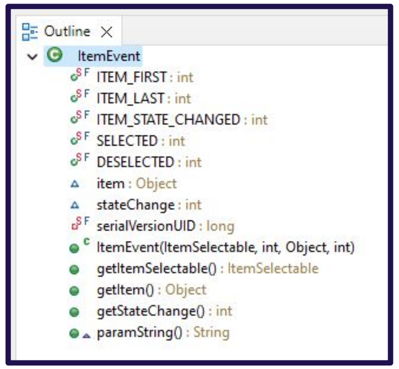
a) Make the MovingMessagePanel class also a subclass of the interface ItemListener

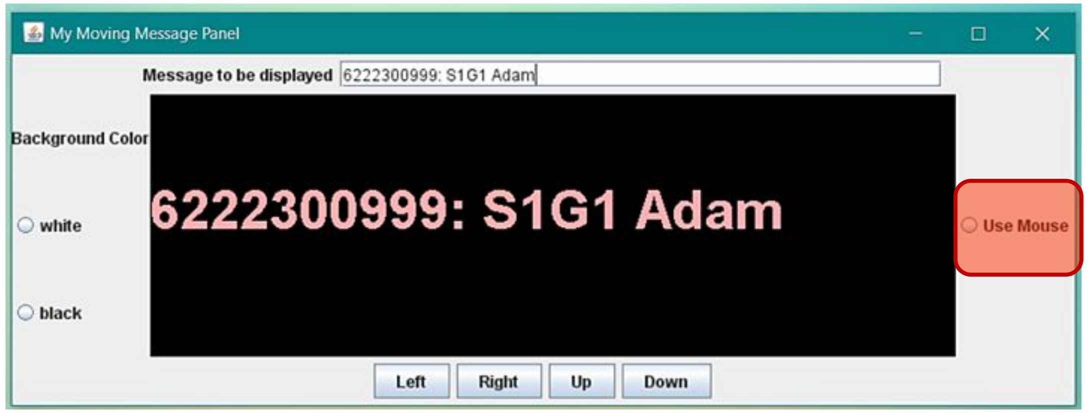

```
<<interface>>
ItemListener
..
+itemStateChanged(e: ItemEvent) :void
```

b) Register the black and white radio buttons with itself which acts as the ItemListener using an appropriate method.

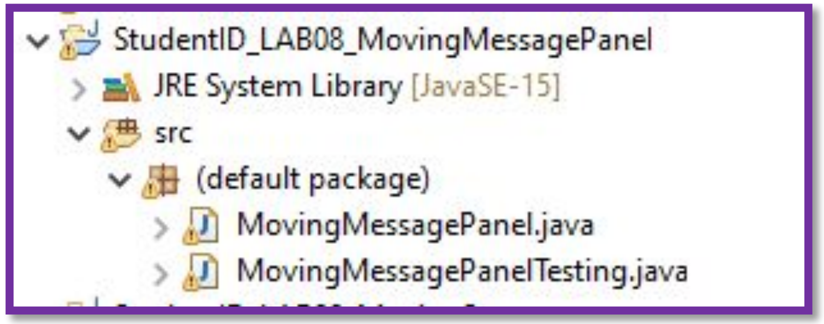
c) Add in the implementation details of the overridden method of ItemListener.


```
Outline
  ItemListener
    itemStateChanged(ItemEvent) : void
```


```
Outline
  ItemEvent
    ITEM_FIRST : int
    ITEM_LAST : int
    ITEM_STATE_CHANGED : int
    SELECTED : int
    DESELECTED : int
    item : Object
    stateChange : int
    serialVersionUID : long
    ItemEvent(ItemSelectable, int, Object, int)
    getItemSelectable() : ItemSelectable
    getItem() : Object
    getStateChange() : int
    paramString() : String
```

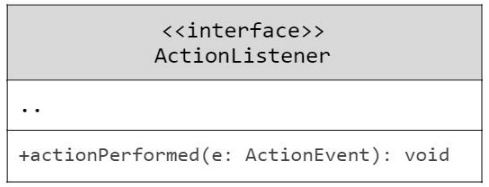


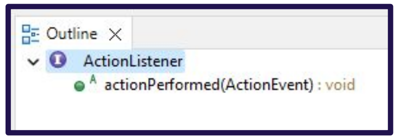
Exercise 4 (2 points)

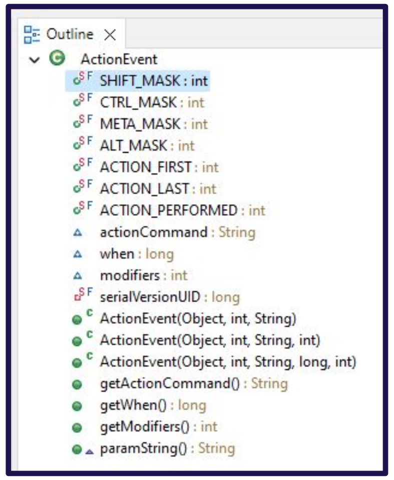

StudentID_LAB08_MovingMessagePanel
JRE System Library [JavaSE-15]
src
(default package)
MovingMessagePanel.java
MovingMessagePanelTesting.java

- **Project Name**: <Student_ID>_LAB08_MovingMessagePanel
- **Instruction**: Write code in the following tasks.

a) Register the four buttons: Left, Right, Up, and Down with itself which acts as the ActionListener using an appropriate method.

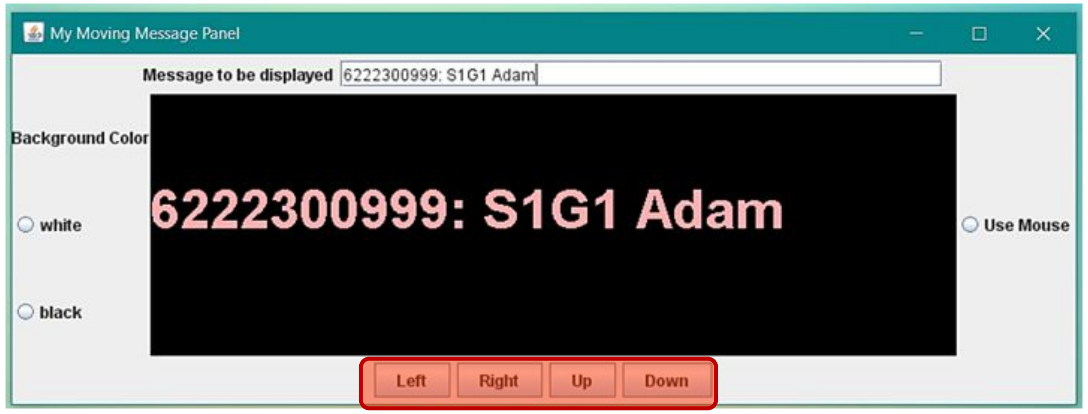

```
<<interface>>
ActionListener
..
+actionPerformed(e: ActionEvent): void
```


Outline
ActionListener
actionPerformed(ActionEvent) : void


Outline
ActionEvent
SHIFT_MASK : int
CTRL_MASK : int
META_MASK : int
ALT_MASK : int
ACTION_FIRST : int
ACTION_LAST : int
ACTION_PERFORMED : int
actionCommand : String
when : long
modifiers : int
serialVersionUID : long
ActionEvent(Object, int, String)
ActionEvent(Object, int, String, int)
ActionEvent(Object, int, String, long, int)
getActionCommand() : String
getWhen() : long
getModifiers() : int
paramString() : String

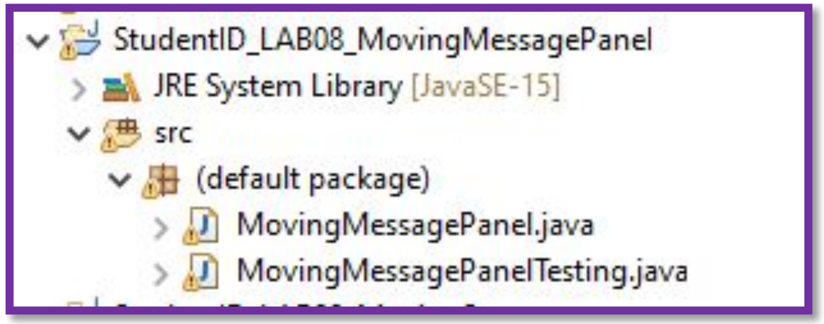
b) Add in the implementation details of the overridden method of ActionListener.

c) Your program should move the message to 4 directions according to correspond directions from 4 buttons: **Left, Right, Up, and Down.**

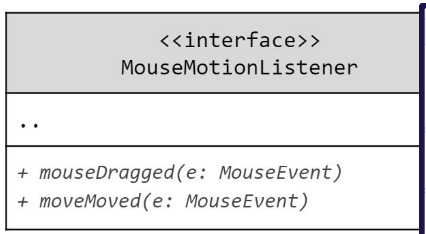


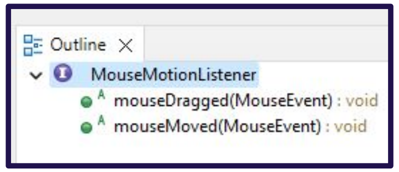
Asst.Prof.Dr.Sasiporn Usanavasin, Dr.Jessada Karnjana, Dr.Kasorn Galajit and Dr.Akkarawoot Takhom
{sasiporn.us, akkharawoot.aj, jessada.aj, kasorn.aj}@siit.tu.ac.th
School of Information, Computer, and Communication Technology, Sirindhorn International Institute of Technology

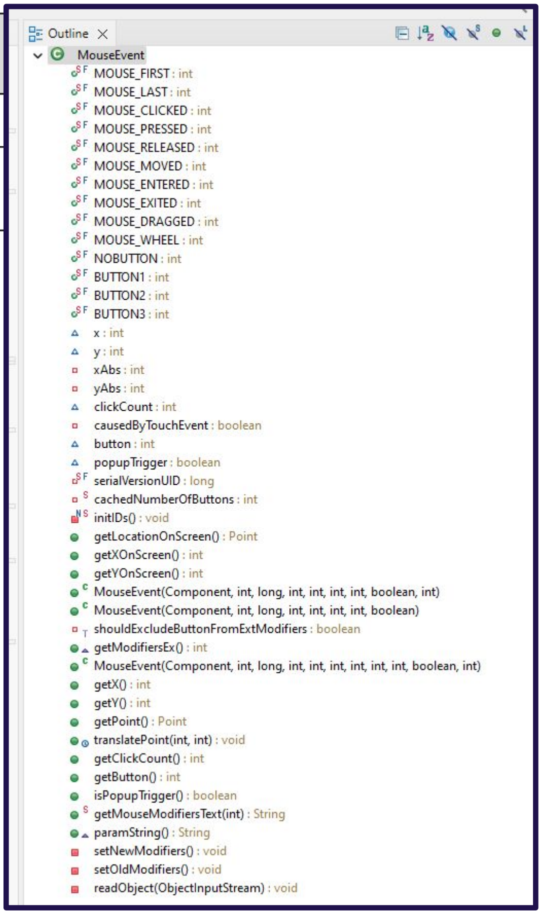
Exercise 5 (2 points)


StudentID_LAB08_MovingMessagePanel
JRE System Library [JavaSE-15]
src
(default package)
MovingMessagePanel.java
MovingMessagePanelTesting.java

- **Project Name**: <Student_ID>_LAB08_MovingMessagePanel
- **Instruction**: Write code in the following tasks.


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


```
Outline
MouseMotionListener
  mouseDragged(MouseEvent) : void
  mouseMoved(MouseEvent) : void
```

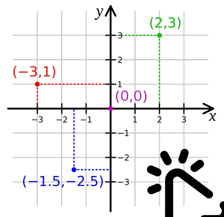

```
Outline
MouseEvent
  MOUSE_FIRST : int
  MOUSE_LAST : int
  MOUSE_CLICKED : int
  MOUSE_PRESSED : int
  MOUSE_RELEASED : int
  MOUSE_MOVED : int
  MOUSE_ENTERED : int
  MOUSE_EXITED : int
  MOUSE_DRAGGED : int
  MOUSE_WHEEL : int
  NOBUTTON : int
  BUTTON1 : int
  BUTTON2 : int
  BUTTON3 : int
  x : int
  y : int
  xAbs : int
  yAbs : int
  clickCount : int
  causedByTouchEvent : boolean
  button : int
  popupTrigger : boolean
  serialVersionUID : long
  cachedNumberOfButtons : int
  initIDs() : void
  getLocationOnScreen() : Point
  getXOnScreen() : int
  getYOnScreen() : int
  MouseEvent(Component, int, long, int, int, int, int, boolean, int)
  MouseEvent(Component, int, long, int, int, int, int, boolean)
  shouldExcludeButtonFromExtModifiers : boolean
  getModifiersEx() : int
  MouseEvent(Component, int, long, int, int, int, int, int, int, boolean, int)
  getX() : int
  getY() : int
  getPoint() : Point
  translatePoint(int, int) : void
  getClickCount() : int
  getButton() : int
  isPopupTrigger() : boolean
  getMouseModifiersText(int) : String
  paramString() : String
  setNewModifiers() : void
  setOldModifiers() : void
  readObject(ObjectInputStream) : void
```

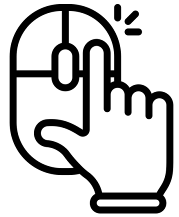
a) Make the MovingMessagePanel class also a subclass of the interface MouseMotionListener

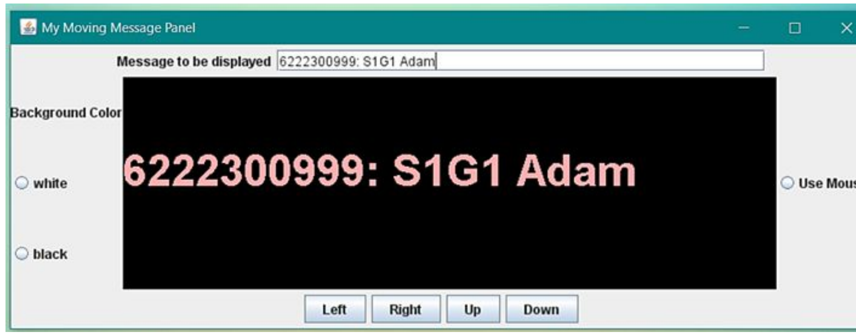
b) Register the display panel(itself) with itself which acts as the MouseMotionListener using an appropriate method.

c) Add in the implementation details of the overridden method of MouseMotionListener.

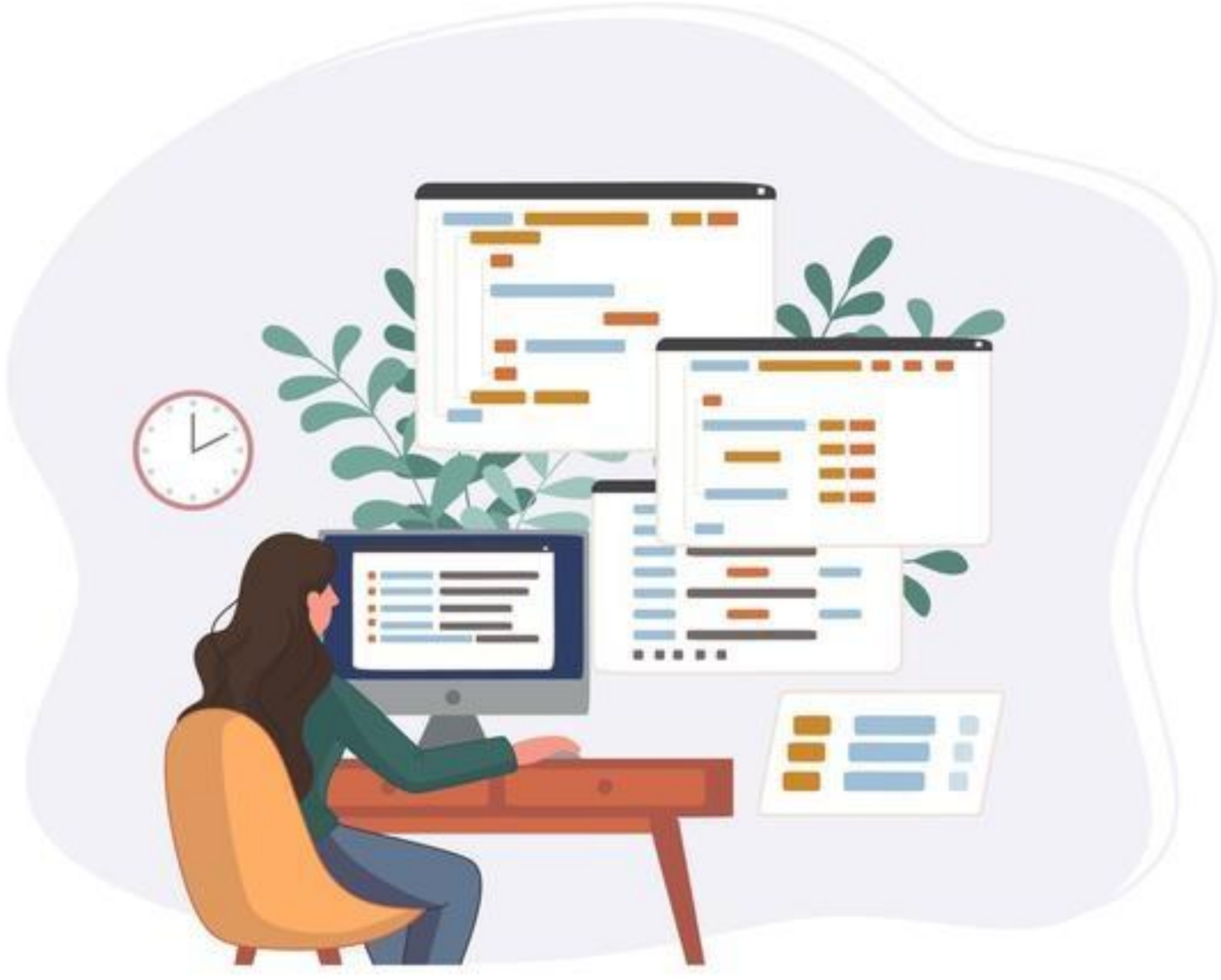
When the use-mouse radio button is selected and the user drags the mouse, your program should move the message at the location of the mouse.






My Moving Message Panel
Message to be displayed: 6222300999: S1G1 Adam
Background Color
○ white
○ black
6222300999: S1G1 Adam
○ Use Mouse
Left | Right | Up | Down

{Let's Code}



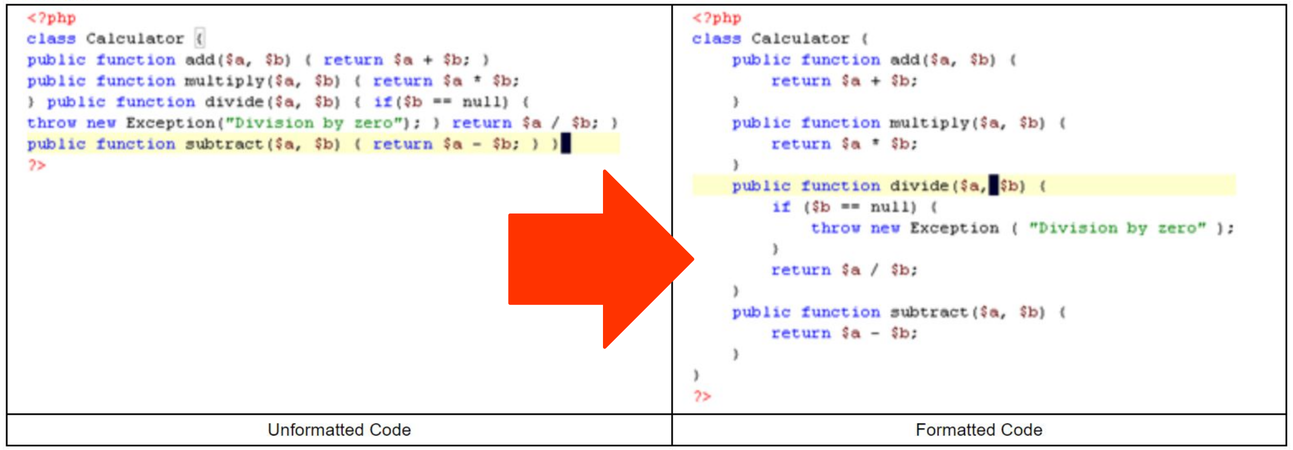
Asst.Prof.Dr.Sasiporn Usanavasin, Dr.Jessada Karnjana, Dr.Kasorn Galajit and Dr.Akkarawoot Takhom
{sasiporn.us, akkharawoot.aj, jessada.aj, kasorn.aj}@siit.tu.ac.th
School of Information, Computer, and Communication Technology, Sirindhorn International Institute of Technology

Suggestions: Formatting Code

- PDT can auto-format your code according to set standards in order to make it easily navigable and readable.
- To format your whole script:
  - Open the required file.
  - Go to Source | Format Document or press

Ctrl+Shift+F

Example:



```
<?php
class Calculator {
public function add($a, $b) { return $a + $b; }
public function multiply($a, $b) { return $a * $b;
} public function divide($a, $b) { if($b == null) {
throw new Exception("Division by zero"); } return $a / $b; }
public function subtract($a, $b) { return $a - $b; } }
?>
```

Unformatted Code

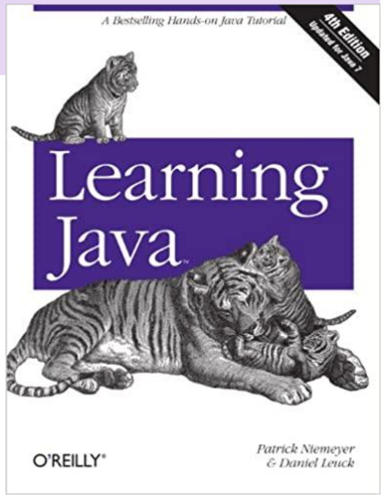
```
<?php
class Calculator {
    public function add($a, $b) {
        return $a + $b;
    }
    public function multiply($a, $b) {
        return $a * $b;
    }
    public function divide($a, $b) {
        if ($b == null) {
            throw new Exception ( "Division by zero" );
        }
        return $a / $b;
    }
    public function subtract($a, $b) {
        return $a - $b;
    }
}
?>
```

Formatted Code

- https://www.eclipse.org/pdt/help/html/formatting_code.htm

References

1. Icon made by Flat icon; www.flaticon.com.
2. Figure made by Freepik; https://www.freepik.com
3. Learning Java: A Bestselling Hands-On Java Tutorial Fourth Edition
4. การเขียนโปรแกรมด้วย Java สำหรับผู้เริ่มต้น, บัญชา ปะสีละเตสัง, se-ed


A Bestselling Hands-on Java Tutorial
4th Edition
Learning Java
O'REILLY
Patrick Niemeyer & Daniel Leuck