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)-LAB06-REVIEW]

JContainer, JComponents, and Layout Managers

Learning Objectives

1. To use the previous knowledge to create a simple graphical user interface, or GUI
2. To learn how to create simple containers: JFrame and JPanel
3. To learn how to create JComponents and put them in the Container
4. To learn how to use Layout Managers to arrange the components in the Container

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

6.1 Frames

❖ **A frame** is a window that is not contained inside another window.

❖ Frame is the basis to contain other user interface components in Java GUI applications.

❖ Frame in `java.awt` and JFrame in `javax.swing` can be used to create windows.

A frame in Java GUI programming refers to a standalone window that isn't nested within another window. It serves as the fundamental container for organizing and presenting various user interface elements.

Frames play a pivotal role in structuring GUI applications by providing a canvas for hosting and arranging JComponents, such as buttons, text fields, and labels. They facilitate user interaction and application functionality.

In Java, **frames** are implemented through classes like `java.awt.Frame` and `javax.swing.JFrame`. These classes provide the necessary functionality to create and manage windows within a graphical user interface.

Frames serve as the starting point for incorporating layout managers, such as `BorderLayout`, `FlowLayout`, or `GridBagLayout`, which enable precise control over the positioning and sizing of UI components within the frame.

This flexibility in layout management empowers developers to craft visually appealing and responsive GUIs tailored to their specific application requirements.

javax.swing.JFrame	Description
`+JFrame()`	+ Creates a default frame with no title.
`+JFrame(title: String)`	+ Creates a frame with the specified title.
`+getSize(width: int, height: int): void`	+ Specifies the size of the frame.
`+setLocation(x: int, y: int): void`	+ Specifies the upper-left corner location of the frame.
`+setVisible(visible: boolean): void`	+ Sets true to display the frame.
`+setDefaultCloseOperation(mode: int):void`	+ Specifies the operation when the frame is closed.
`+setLocationRelativeTo (c:Component): void`	+Sets the location of the frame relative to the specified component. If the component is null, the frame is centered on the screen.

6.2 Components

❖ Things that are put in the container are called Components

java.awt.Component	Description
- font: java.awt.Font - background: java.awt.Color - foreground: java.awt.Color - preferredSize: Dimension - visible: boolean	The get and set methods for these data fields are provided in the class, but omitted in the UML diagram for brevity. – The font of this component. – The background color of this component. – The foreground color of this component. – The preferred size of this component. – Indicates whether this component is visible.
+ getWidth(): int + getHeight(): int + getX(): int + getY(): int	+ Returns the width of this component. + Returns the height of this component. + getX() and getY() return the coordinate of the component's upper-left corner within its parent component.

java.awt.Container	Description
+ add(comp: Component): Component +add(comp: Component, index: int): Component + remove(comp: Component): void + getLayout(): LayoutManager + setLayout(l: LayoutManager): void + paintComponents(g: Graphics): void	+ Adds a component to the container. + Adds a component to the container with the specified index. + Removes the component from the container. +Returns the layout manager for this container. + Sets the layout manager for this container. + Paints each of the components in this container.

java.awt.Container	Description
-toolTipText: String -border: javax.swing.border.Border	The get and set methods for these data fields are provided in the class, but omitted in the UML diagram for brevity. – The tool tip text for this component. Tooltip text is displayed when the mouse points on the component without clicking. – The border for this component.

Four types of Layout Managers

1. **FlowLayout**: Align components automatically.
2. **GridLayout**: Align components in table (row by column) fashion
3. **BorderLayout** : Align component according to 5 directions:

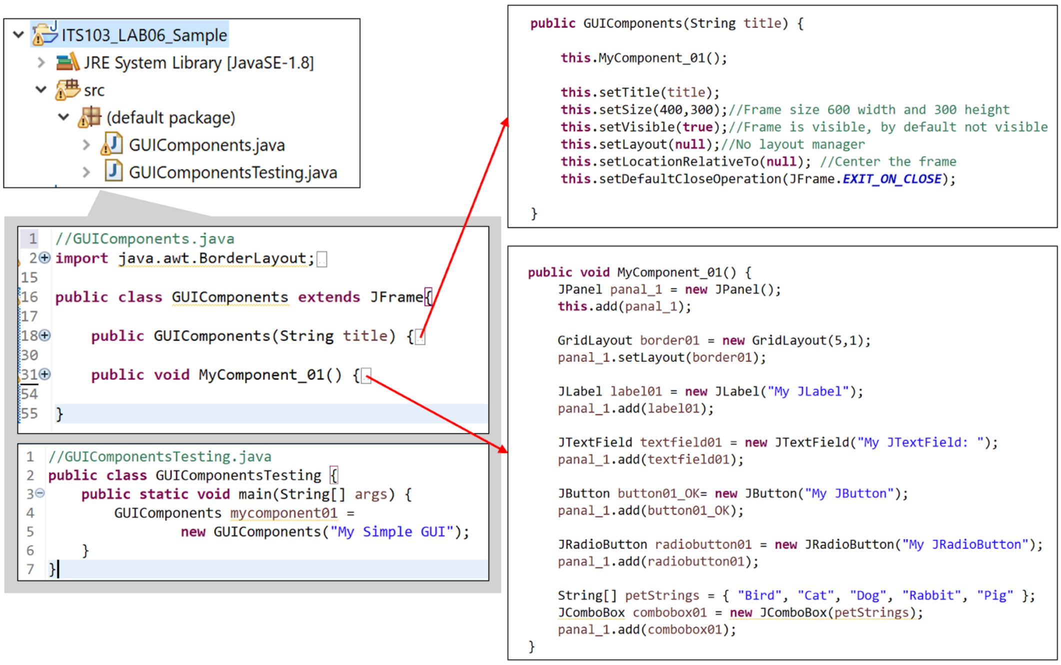
   North, South, East, West, and Center

* The default layout manager for the `JFrame` is BorderLayout at **Center** position

6.3 Examples of GUI

👉 To answer your TA questions, student SHOULD try to understand the following example:

Java project:



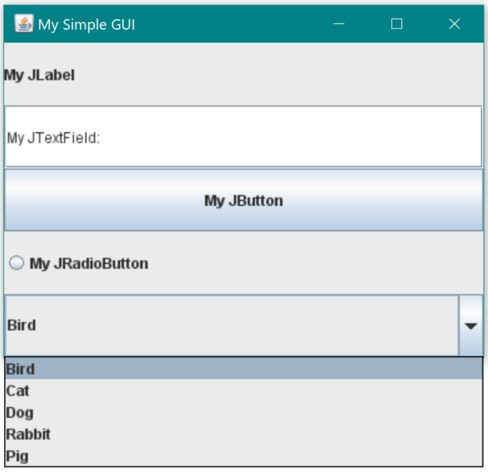
```java
//GUIComponents.java
import java.awt.BorderLayout;

public class GUIComponents extends JFrame{

    public GUIComponents(String title) {

    public void MyComponent_01() {

}
```

```java
//GUIComponentsTesting.java
public class GUIComponentsTesting {
    public static void main(String[] args) {
        GUIComponents mycomponent01 =
                new GUIComponents("My Simple GUI");
    }
}
```

```java
public GUIComponents(String title) {

    this.MyComponent_01();

    this.setTitle(title);
    this.setSize(400,300);//Frame size 600 width and 300 height
    this.setVisible(true);//Frame is visible, by default not visible
    this.setLayout(null);//No layout manager
    this.setLocationRelativeTo(null); //Center the frame
    this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

}
```

```java
public void MyComponent_01() {
    JPanel panal_1 = new JPanel();
    this.add(panal_1);

    GridLayout border01 = new GridLayout(5,1);
    panal_1.setLayout(border01);

    JLabel label01 = new JLabel("My JLabel");
    panal_1.add(label01);

    JTextField textfield01 = new JTextField("My JTextField: ");
    panal_1.add(textfield01);

    JButton button01_OK= new JButton("My JButton");
    panal_1.add(button01_OK);

    JRadioButton radiobutton01 = new JRadioButton("My JRadioButton");
    panal_1.add(radiobutton01);

    String[] petStrings = { "Bird", "Cat", "Dog", "Rabbit", "Pig" };
    JComboBox combobox01 = new JComboBox(petStrings);
    panal_1.add(combobox01);
}
```

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, Thammasat University.

[DES103(Y23S2)-LAB05-EXERCISES]

Students MUST adhere to the lab instructions and regulations provided below.

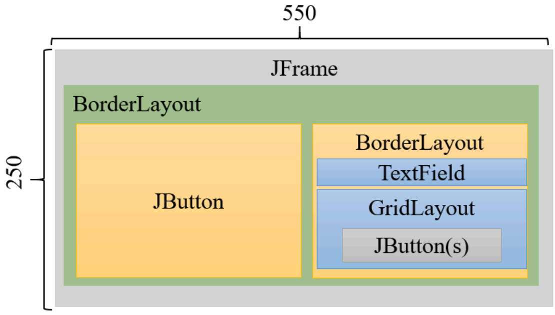
Please consult your TA to review your completed exercises and submit them on Google Classroom.

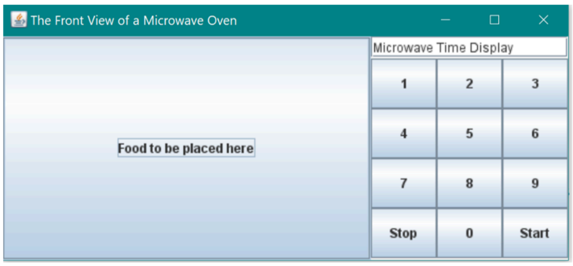
Be noticed that for all lab exercises, you need to define your *Java* project as the following name format:

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

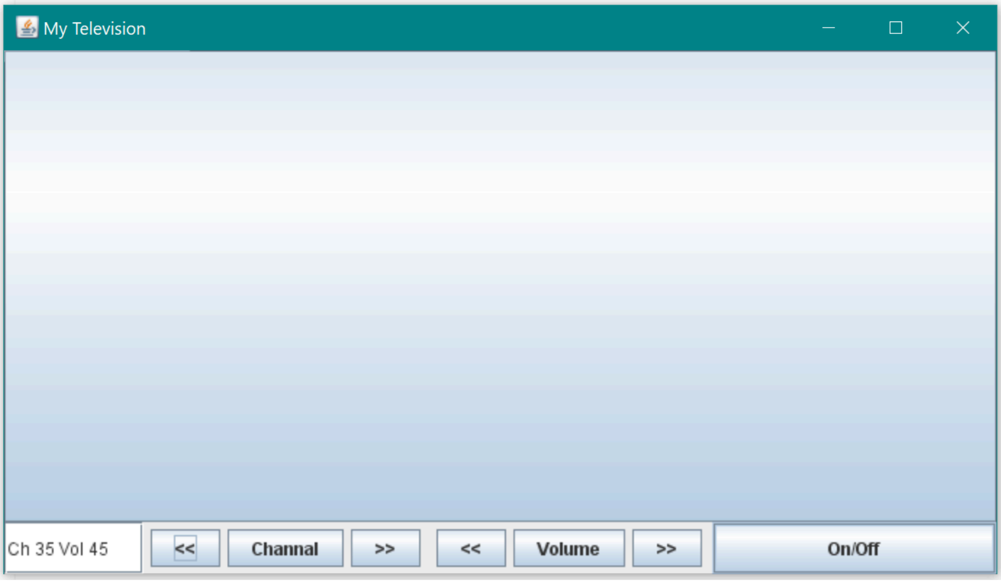
If your student's ID is 6722300208, the name format of your java project should be:

Project1: "6422300208_LAB06_Microwave" for exercise 1.



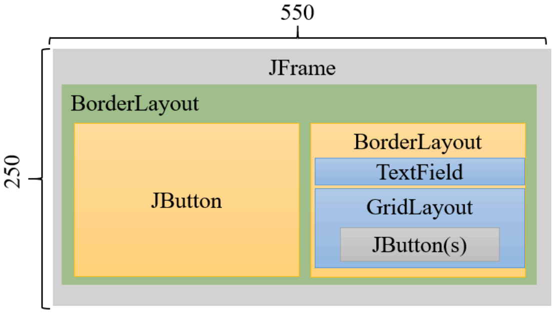


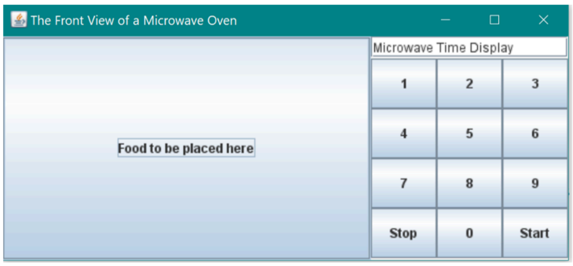
Project2: "6422300208_LAB06_Television" for exercise 2,3,4 and 5.



Exercise 1: (2 points)

- **Java Project**: <Student_ID>_LAB06_Microwave
- **Objective**:
  To learn how to create simple containers: `JFrame` and `JPanel`, and create `JComponents` and put them in the `Container`
- **Suggestion**:
  👉 To explain your TA, student SHOULD define **a variable name** as the following format:
  `<JComponents in lowercase>_<function/role>`
  For example,
  `JButton button_Stop = new JButton("Stop");`
- **Instruction:**
  Download Microwave.java from Google Classroom and complete a class `Microwave` to create a user interface for a Microwave oven. Write a class `MicrowaveComponents` to show Microwave GUI





👉 *Students MUST consider details of the GUI design, and arrange a sequence before adding components*

This JFrame of "Microwave" contains the first panel "`panal01_MicrowaveLayout`" that has its layout on two sides: CENTER and EAST.

- The CENTER part contains the button "`button_Foods`" for placing foods, and defines "`Food to be placed here`"
- This ControlPanel comprises of 12 equal size buttons that are represented by four rows and three
- The EAST part contains the second panel "`panal02_NumberLayout`" that composes of 12 equal size buttons represented by four rows and three as below:
  1. The NORTH part contains a textfield "`textfield_TimeDisplay`" to display microwave's time, and defines "`Microwave Time Display`"
  2. The CENTER part contains the matrix layout (3x3) to display buttons by 1, 2, 3, 4, 5, 6, 7, 8, 9, respectively.

  *Students MUST use a FOR loop to add those buttons

  3. The CENTER part also contains three remaining buttons to display "Stop", "0", and "Start", respectively.

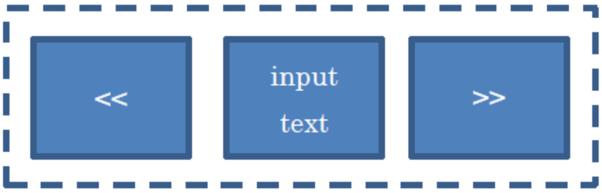
Exercise 2: (2 points)

- **Java Project**: <Student_ID>_LAB06_Television
- **Objective**:
  To learn how to create simple containers: `JFrame` and `JPanel`, and create `JComponents` and put them in the `Container`
- **Suggestion**:
  👉 To explain your TA, student SHOULD define **a variable name** as the following format:

  ```
  <JComponents in lowercase>_<function/role>
  ```

  For example,

  ```
  JButton button_Stop = new JButton("Stop");
  ```
- **Instruction:**
- Download AdjustPanel.java from Google Classroom and complete a class `AdjustPanel` which is a subclass of Panel. The panel comprises of three buttons: decrease (<<) on the left, increase (>>) on the right. The display text on the middle is obtained from the input argument of the constructor. Use an appropriate LayoutManager to arrange these two components according to the plan below.



Exercise 3: (2 points)

- **Java Project**: <Student_ID>_LAB06_ Television
- **Objective**: To learn how to create simple containers: `JFrame` and `JPanel`, and create `JComponents` and put them in the `Container`
- **Suggestion**:

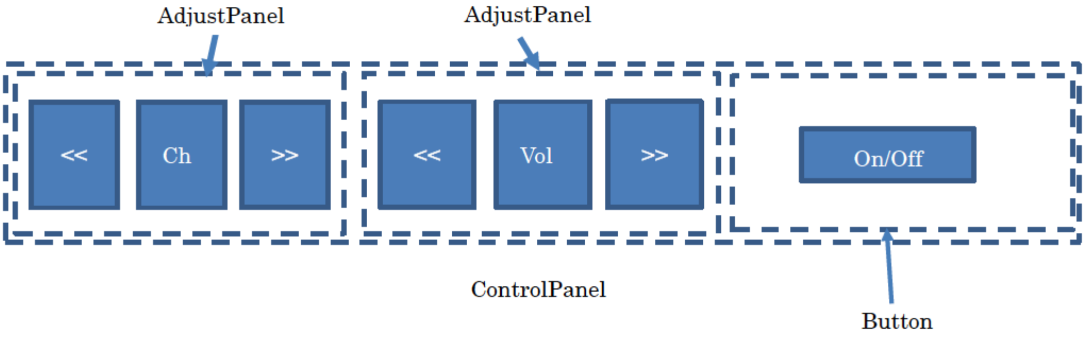
  👉 To explain your TA, student SHOULD define **a variable name** as the following format:

  ```
  <JComponents in lowercase>_<function/role>
  ```

  For example,

  ```
  JButton button_Stop = new JButton("Stop");
  ```
- **Instruction:**

  Download ControlPanel.java from Google Classroom and complete a class `ControlPanel` which is a subclass of `Panel`. The `ControlPanel` comprises of 3 equal size subpanels. Use an appropriate `LayoutManager` to arrange these two components according to the plan below.



- The left subpanel composes of three buttons:
  decrease (<<) on the left, increase (>>) on the right, and a `textField` which display text(`Ch`) in the middle.
- The middle subpanel comprises of three buttons:
  decrease (<<) on the left, increase (>>) on the right, and a `textField` which display text(`Vol`) in the middle.
- The right subpanel composed of one button (`On/Off`)

Exercise 4: (2 points)

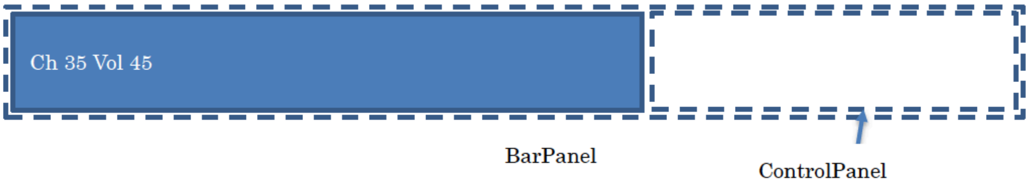
- **Java Project**: <Student_ID>_LAB06_Television
- **Objective**:
  To learn how to create simple containers: `JFrame` and `JPanel`, and create `JComponents` and put them in the `Container`
- **Suggestion**:

  👉 To explain your TA, student SHOULD define **a variable name** as the following format:

  ```
      <JComponents in lowercase>_<function/role>
  ```

  For example,

  ```
      JButton button_Stop = new JButton("Stop");
  ```
- **Instruction:**
- Download BarPanel.java from Google Classroom and complete a `BarPanel` class which is a subclass of Panel. The panel comprises a TextField that displays text Ch 35 Vol 45 and a ControlPanel that you defined in Exercise 3. Use an appropriate LayoutManager to arrange these two components according to the plan below.



Exercise 5: (2 points)

- **Java Project**: <Student_ID>_LAB06_Television
- **Objective**: To learn how to create simple containers: JFrame and JPanel, and create JComponents and put them in the Container
- **Suggestion**:

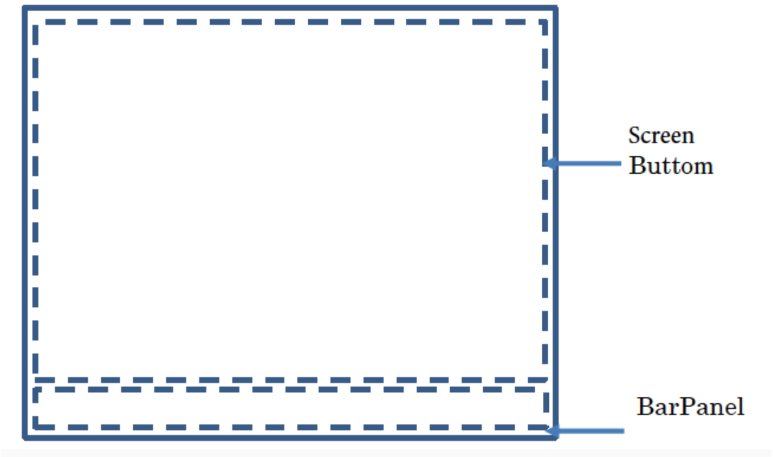
  👉 To explain your TA, student SHOULD define **a variable name** as the following format:

  ```
          <JComponents in lowercase>_<function/role>
  ```

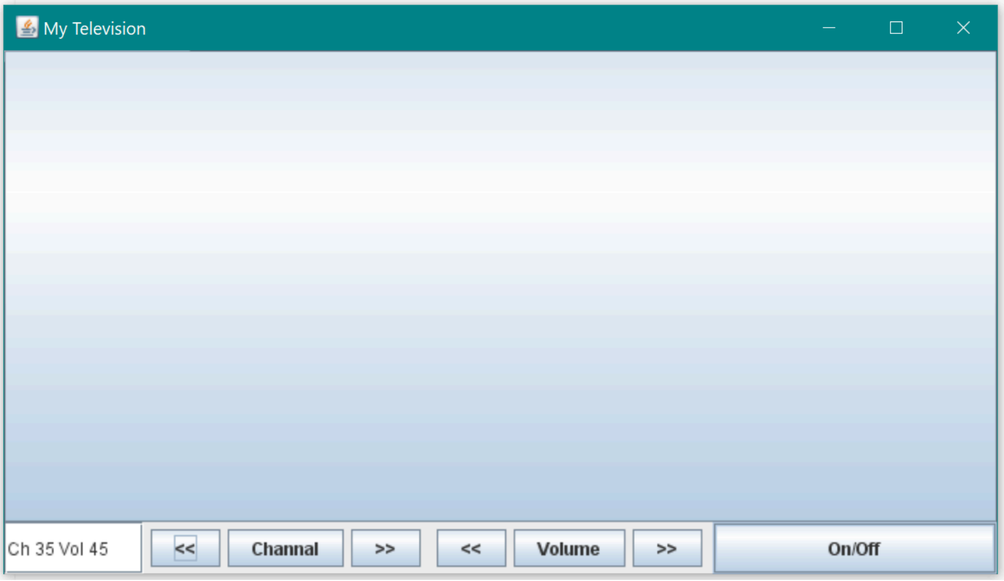
  For example,

  ```
          JButton button_Stop = new JButton("Stop");
  ```
- **Instruction:**

  Download Television.java from Google Classroom and complete a Television class which is a subclass of JFrame. The GUI of Television consists of a screen button above and a bar panel below that you created from Exercise 4.



Write a class TelevisionTesting, and run the final output of television GUI should look like


My Television
Ch 35 Vol 45
<<
Channal
>>
<<
Volume
>>
On/Off