KEMENTERIAN PENGAJIAN TINGGI
JABATAN PENDIDIKAN POLITEKNIK DAN KOLEJ KOMUNITI

POLITEKNIK MALAYSIA
KUALA TERENGGANU

JAVA GUI BUNDLE

A COMPLETE GUIDES FOR PROGRAMMING PRACTICES

SUHAILA BINTI MOHD ZAIN
Politeknik Kuala Terengganu

JAVA GUI BUNDLE

A Complete Guides for Programming Practices

Author:

Suhaila Binti Mohd Zain

Published by:

Politeknik Kuala Terengganu
20200 Jalan Sultan Ismail
Kuala Terengganu, Terengganu

Perpustakaan Negara Malaysia Cataloguing-in-Publication Data

Suhaila Mohd. Zain, 1979-
JAVA GUI BUNDLE : A COMPLETE GUIDES FOR PROGRAMMING PRACTICES / SUHAILA BINTI MOHD ZAIN.
Mode of access: Internet
eISBN 978-967-2240-36-5
1. Java (Computer program language).
2. Object-oriented programming languages.
3. Programming languages (Electronic computers).
4. Government publications--Malaysia.
5. Electronic books.
I. Title.

005.133

Table of Content

Introduction iii

Section 1 Revise of Object Oriented Programming

❑ CLASSES AND OBJECT 2

❑ INHERITANCE 4

❑ POLYMORPHISM 6

❑ INTERFACE 7

❑ MULTITHREADING CONCEPTS 8

❑ REMINDER OF DATA TYPES 9

Section 2 Short Notes to Java GUI

❑ BASIC GUI IN JAVA PROGRAMMING 12

❑ EVENT HANDLING CONCEPTS 31

❑ DATABASE AND GUI 37

Section 3 Simple Questions

❑ EXERCISE ON BASIC JAVA PROGRAMMING 44

❑ EXERCISE INPUT OUTPUT STREAM PROGRAMMING 47

❑ EXERCISE INTRODUCTION TO OOP 50

Section 4 Activities

❑ INTRODUCTION TO GUI 58

❑ MORE ON GUI 66

Introduction

The aims of this book is for readers to apply object oriented approach and get programming skills in Java application.This book start with the concepts of object oriented programming. The concept is then applied to create GUI in Java programming. It then addresses on the creation of GUIs through standalone front-end applications. This book has four main section, which focus on the AWT and Swing library for GUI, then continued with event handling and finally with database knowledge in the development of an application. Upon completion the contents of this book, readers are able to design, code, test, and debug at intermediate level.

The content flow is arranged from basic to more advanced skill. However it can be skipped according to readers level and skills. The IDE used in executing codes in this book is NetBeans IDE 8.0.2 with jdk1.8.0_121. Hopefully this book will assit the readers in the process of learning and gaining skills for Java programming and project development.

Convention used:

The following are specific notes for iconic symbols used in this book.

Icon	Description
💡	This icon is used to highlight and give ideas of related terms and concept discussed
Discussion	This icon is to express that the exercise will have short discussion.
JAVA source code	This icon is to indicate that the text is a java codes.
🔔	This icon is to indicate a reminder for alert the reader
✓ Check your answer	This icon is to indicate a sample answer for a given problem.
📋	This icon is to indicate a check list or steps or guides for given problem.

Notices:

Images used is free source modified from the Internet and Freepik. The originals are solely held by their owner.

Introduction

Acknowledgement:

The task for completing this book has get attention from the Politeknik Kuala Terengganu management. Their support, guide and advice for finishing "JAVA GUI BUNDLE, A COMPLETE GUIDES FOR PROGRAMMING PRACTICES" really has made the task smooth and easy. The opportunity to write this book not only for achieving the organization mission, but also to increase the writers' knowledge.

Thanks again to all the members involved, without them, this book would not have been successfully completed.

Suhaila Binti Mohd Zain

Section 1

Revise of Object Oriented Programming

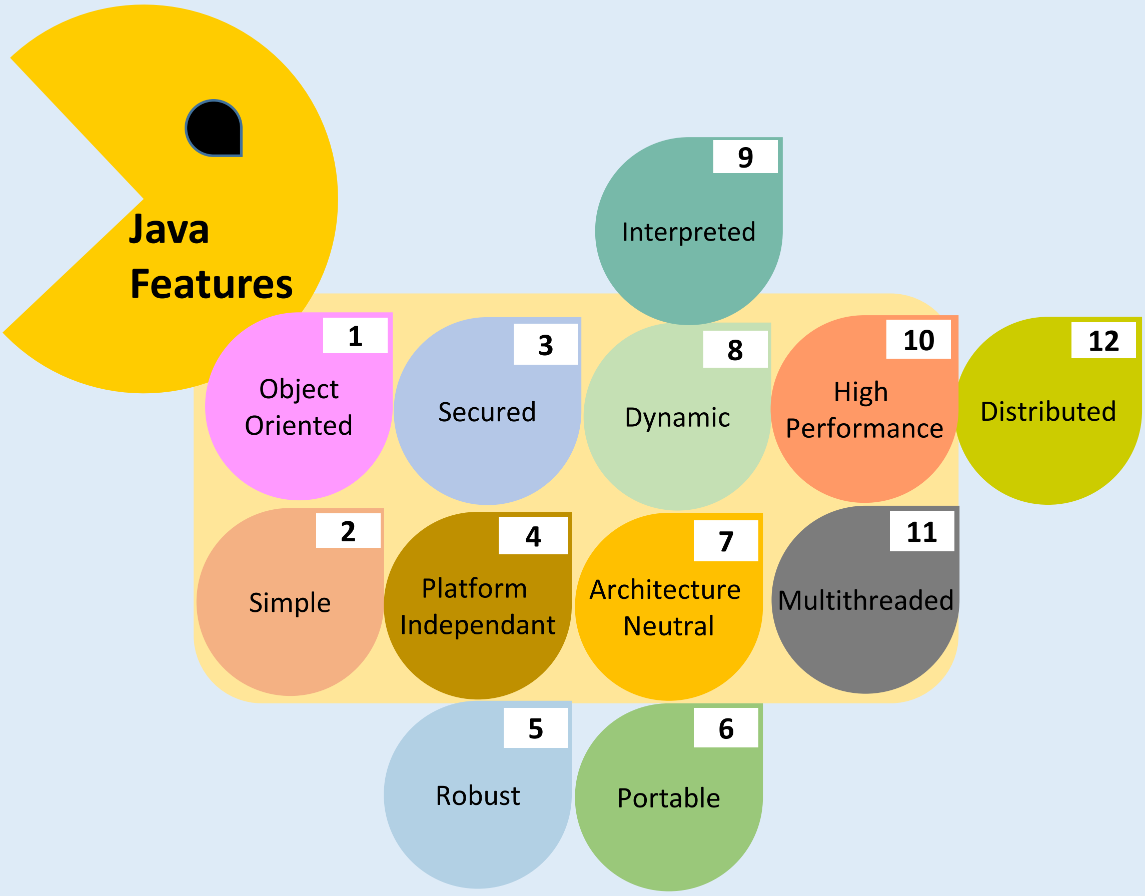
- ❑ CLASSES AND OBJECT
- ❑ INHERITANCE
- ❑ POLYMORPHISM
- ❑ INTERFACE
- ❑ MULTITHREADING CONCEPTS
- ❑ REMINDER OF DATA TYPES

Classes and Object

Java is a powerful general purpose programming language that has been around for over 23 years now since its inception in 1996. The Java Programming environment consists of :

1.Java Language — used by programmers to write the application
2.The Java Virtual Machine (JVM) — used to execute the application
3.The Java Ecosystem — provides additional value to the developers using the programming language

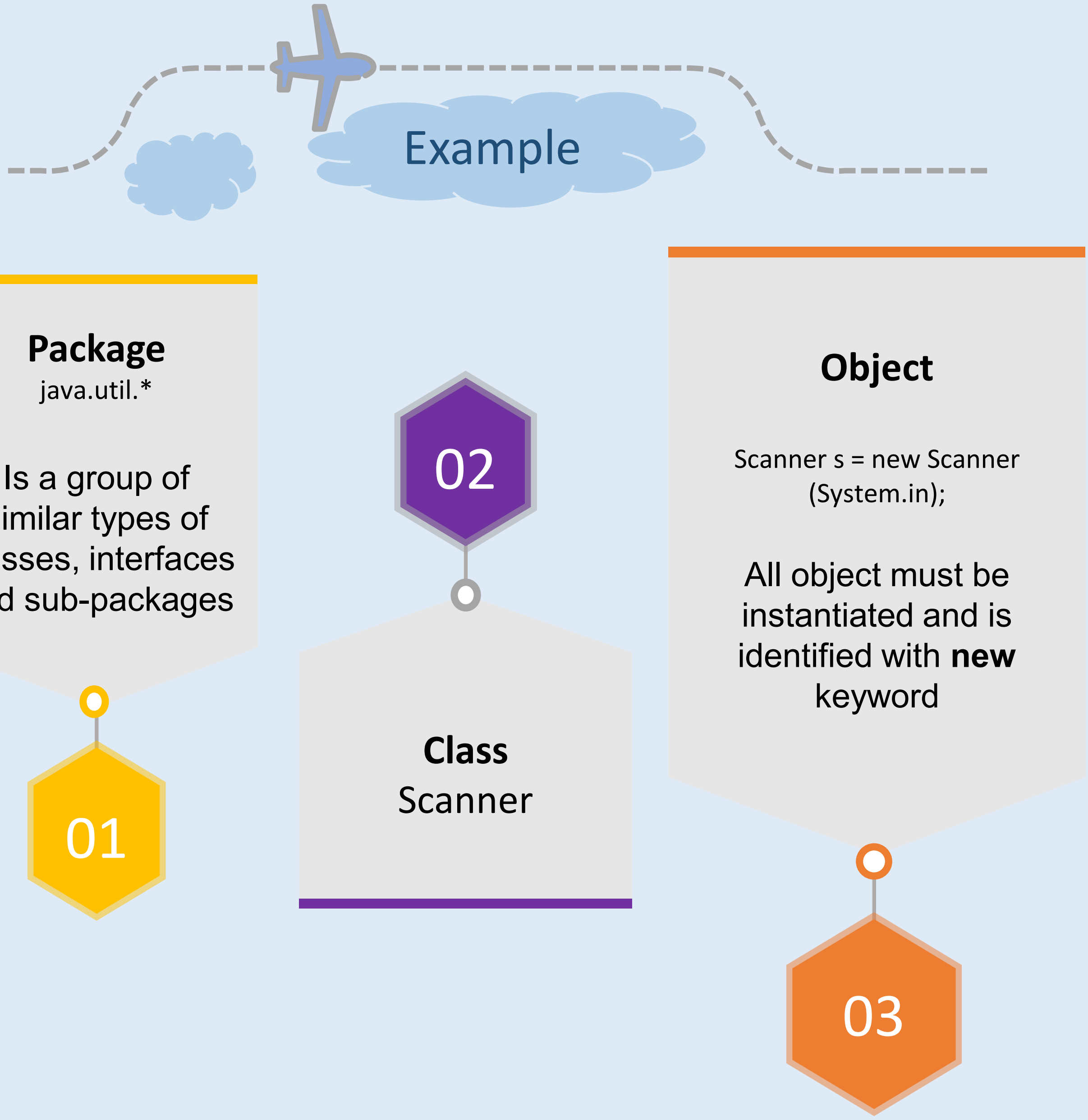
The features of Java is summarize in the following diagram.



It is *class-based* and *object-oriented* in nature.

Classes and Object

Fundamental of Java language is it consist of classes stored in a package.



Example

Package
java.util.*

Is a group of similar types of classes, interfaces and sub-packages

01

02

Class
Scanner

Object

Scanner s = new Scanner (System.in);

All object must be instantiated and is identified with **new** keyword

03

Discussion

Where is the best place to define an object? Consider you have nested class and methods.

Inheritance

Inheritance is denoted by keyword **extends**. There are various types of inheritance as demonstrated below:

Single Inheritance

```
public class A{ .....
}
public class B extends A{ .....
}
```

Multi Level Inheritance

```
public class A{ ..... }
public class B extends A{ ..... }
public class C extends B{ ..... }
```

Hierarchical Inheritance

```
public class A{ ..... }
public class B extends A{ ..... }
public class C extends A{ ..... }
```

Java does not support multiple inheritance.

Inheritance

Usage of **super** keyword in inheritance context

super can be used to refer immediate parent class instance variable

Super can be used to invoke immediate parent class method

super() can be used to invoke immediate **parent class constructor**

Usage of **this** keyword

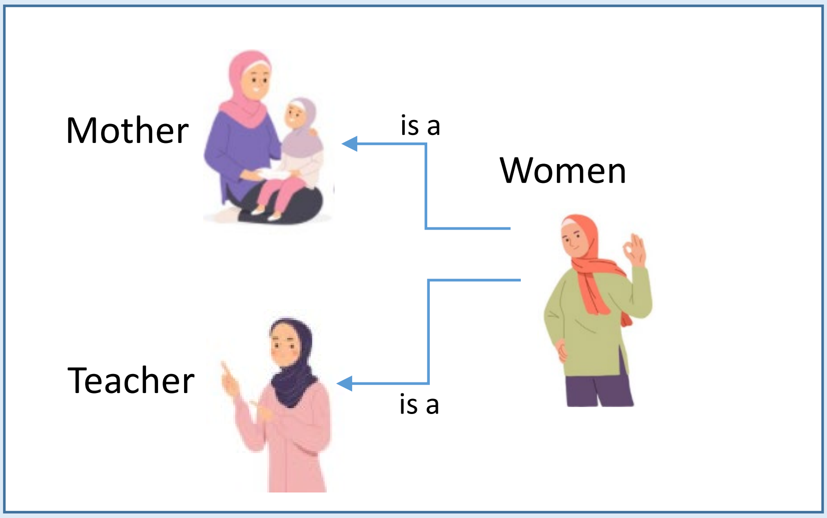
Refer to current class variable

Refer to current class method

This is an argument in the method call or in the constructor call

Polymorphism

Polymorphism in Java is **the ability of an object to take many forms**. To put it simply, polymorphism in Java allows us to perform the same action in many different ways. Polymorphism is denoted by keyword **implements**



Explain method overloading and method overriding?

Method overloading	Method overriding
Having multiple methods with same name but with different signature (number, type and order of parameters)	When a subclass contains a method with the same name and signature as in the super class

Method overriding is applied to use many interface

Interface

An **interface** is a blueprint or template of a class. It is much similar to the Java class but the only difference is that it has **abstract methods** (there is no method body inside these abstract methods) and **static constants**.

The class that implements the interface should be abstract, otherwise, we need to define all the methods of the interface in the class.

Difference between Class and Interface in Java

Class	Interface
Class can instantiate variable and create object	Interface can not instantiate variable and create object
Class can contain concrete methods	The interface can not contain concrete methods
The access specifier used with classes are private, protected and public	In interface only one public specifier is used

Example ActionListener interface

JAVA source code

```
public interface ActionListener extends
EventListener
{

public abstract void actionPerformed(A
ctionEvent e);
}
```

JAVA source code

```
class Task implements Runnable {
@Override
  public void run() {
    for (int i = 0; i < 10; i++) {
         System.out.println("Running Task");
         // Other statements task go here
       }
  }
}
```

Multithreading Concepts

Any application can have multiple processes (instances). Each of this process can be assigned either as a single thread or multiple threads.

Multithreading in Java is a process of executing two or more threads simultaneously to maximum utilization of CPU. Multithreaded applications execute two or more threads run concurrently. Hence, it is also known as Concurrency in Java.

A single-thread program has a single entry point (the main() method) and a single exit point. A multi-thread program has an initial entry point (the main() method), followed by many entry and exit points, which are run concurrently with the main(). The term "*concurrency*" refers to doing multiple tasks at the same time.

In GUI applications, multithreading is essential in providing a *responsive* user interface.

The SwingUtilities.invokeLater() method is an extremely important method for writing a Java application that uses multithreading and if the interface uses Swing.

Example SwingUtilities abstract class

JAVA source code

```
private static void createGUI() {
    JFrame.setDefaultLookAndFeelDecorated(true);
    JFrame frame = new JFrame("JMenu Demo");
   ....
}
public static void main(String[] args) {
    javax.swing.SwingUtilities.invokeLater(new Runnable() {
        public void run() {
            createGUI();
        }
    });
}
```

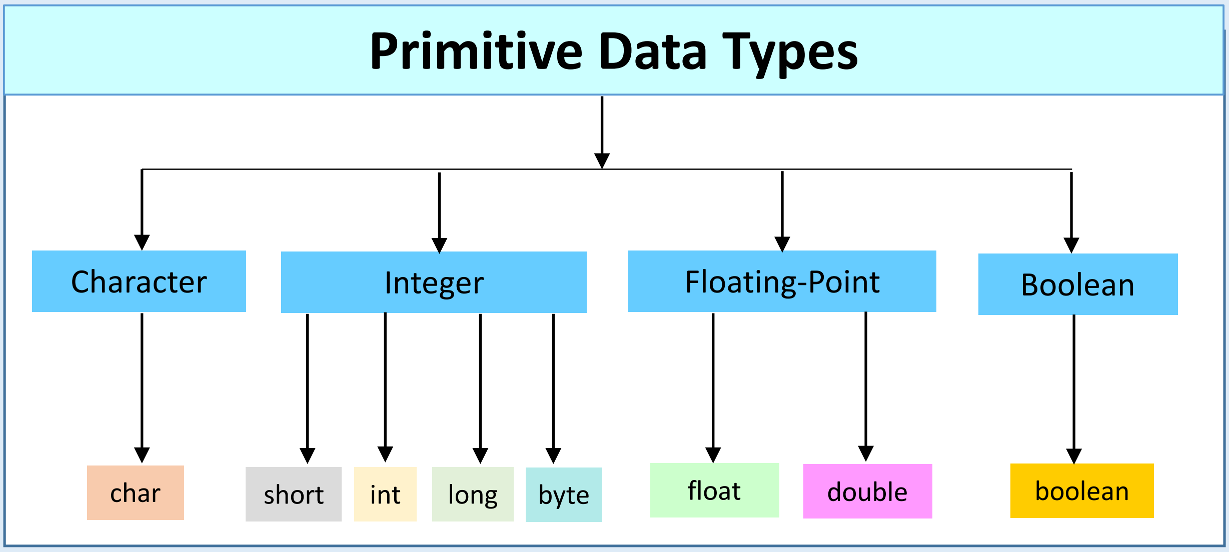
Reminder Of Data Types

Data types incorporate storage categories like integers, floating-point values, strings, characters, etc. Before moving towards the Java Data types, you must know the types of languages.

There are majorly two types of languages:

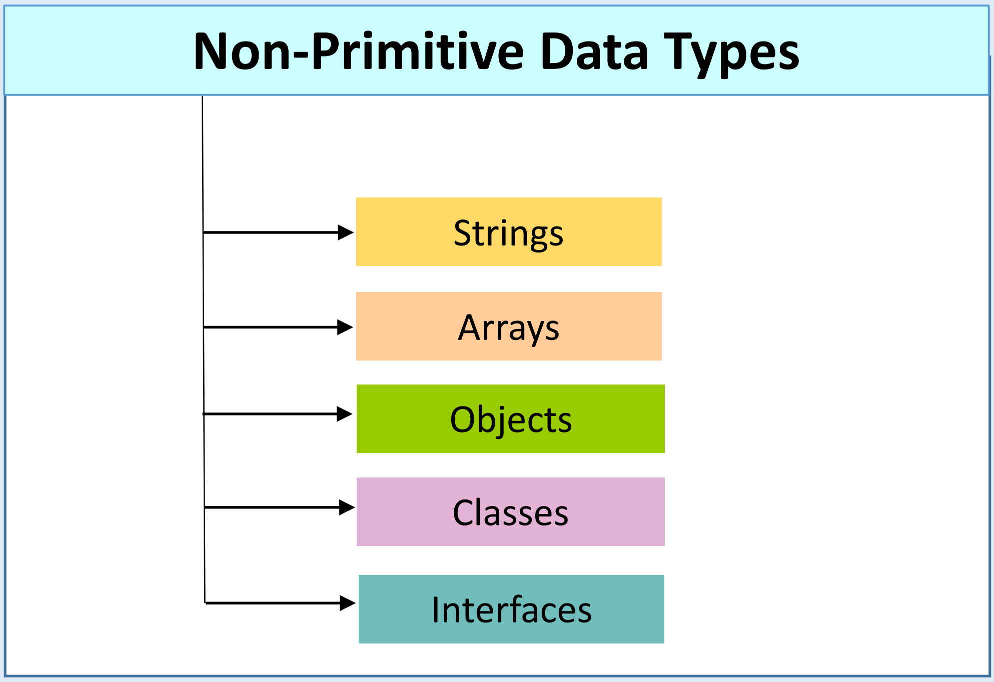
The first is statically typed language in which the data type of each variable has to be defined during the compile time. That is, we have to declare the type of the variable before we can use it.
Once we declare a variable of a specific data type, then we cannot change its data type again. However, they can be converted to other types by using explicit type casting in Java, except boolean. Some statically typed languages are C, C++, C#, Java, and Scala.

The other is dynamically typed language. In this type of language, the data types can change with respect to time and the variables are checked during run-time.
Some dynamically typed languages are Ruby, Python, Erlang, Perl, VB, and PHP.



https://techvidvan.com/tutorials/data-types-in-java/

Reminder Of Data Types



Non-Primitive Data Types

Array
Arrays are fixed length homogeneous data types. Arrays store fixed length different values of same type. Arrays can be of 1-dimensional, 2-dimensional or multi-dimensional.

Example Declaration of Arrays:

JAVA
source code

```
byte[] buffer;
int[][] matrix; //2-Dimensional
int matrix[][][]; // 3-Dimensional
double[] units[][]; //3-Dimensional equivalent to
double[][][] units;
JLabel labels[] = new JLabel[10]; //array of objects
```

Section 2

Short Notes to Java GUI

- ❑ BASIC GUI IN JAVA PROGRAMMING
- ❑ EVENT HANDLING CONCEPTS
- ❑ DATABASE AND GUI

Basic GUI In Java Programming

Understand the concepts of:

Container

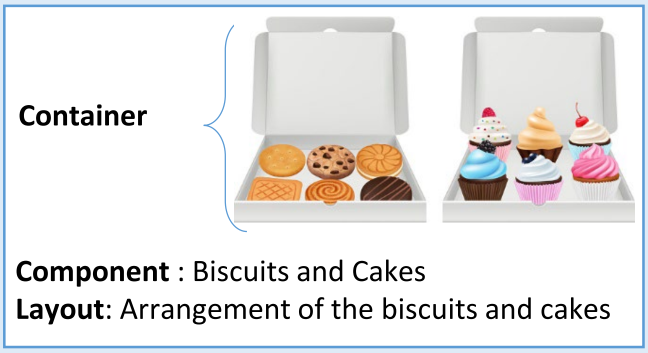
01

Component

02

Layout

03



Component : Biscuits and Cakes

Layout: Arrangement of the biscuits and cakes

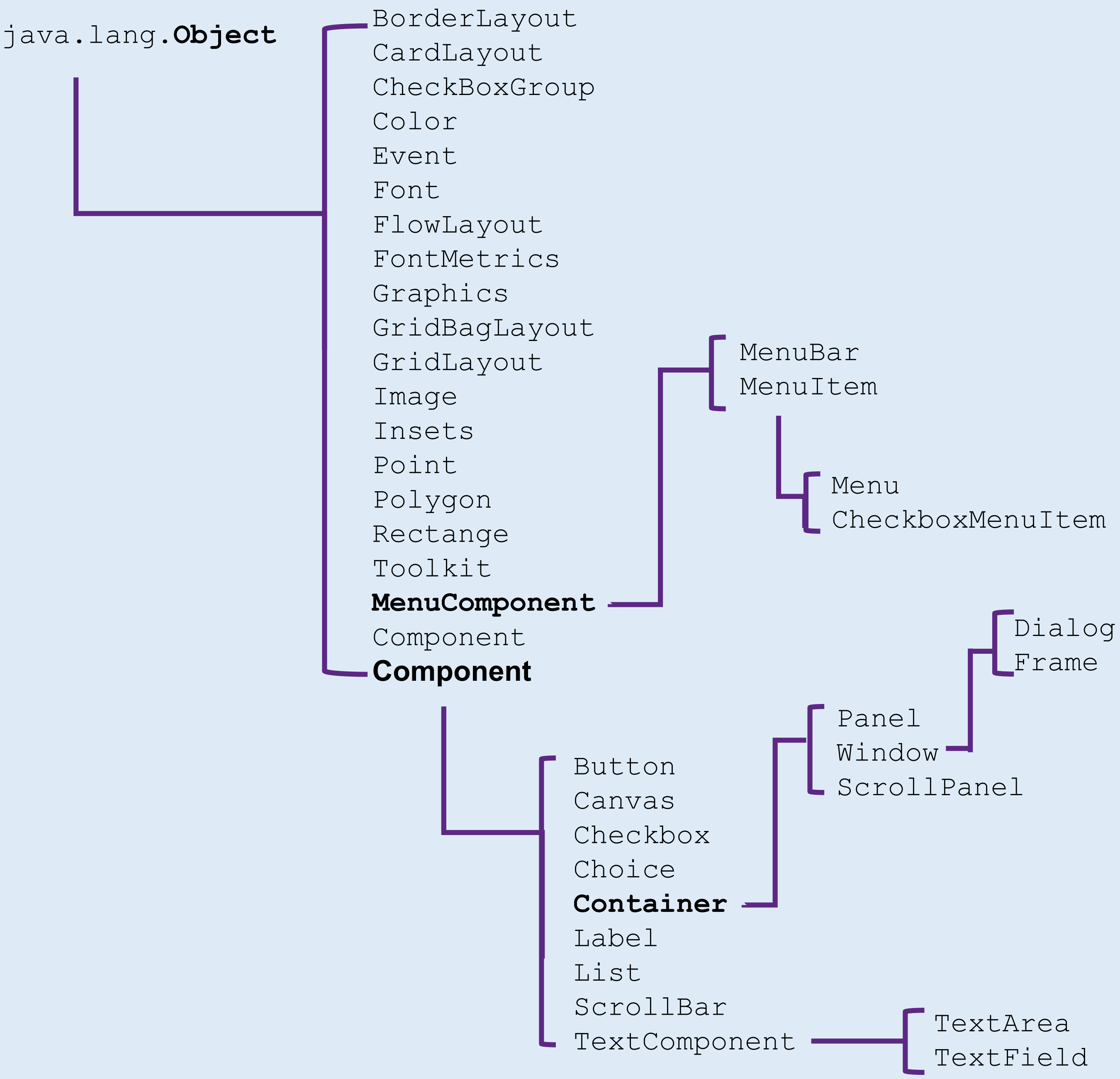
Java GUI component consist of frame, buttons, text fields, combo boxes and etc.

Each type of GUI defined in a class such as Frame , Button , TextField , ComboBox etc

Basic package involves are:

- java.awt.*;
- javax.swing.*

GUI Hierarchy – **java.awt Package**:



java.lang.**Object**

BorderLayout
CardLayout
CheckBoxGroup
Color
Event
Font
FlowLayout
FontMetrics
Graphics
GridBagLayout
GridLayout
Image
Insets
Point
Polygon
Rectangle
Toolkit
MenuComponent
Component
Component

MenuBar
MenuItem

Menu
CheckboxMenuItem

Button
Canvas
Checkbox
Choice
Container
Label
List
ScrollBar
TextComponent

Panel
Window
ScrollPanel

Dialog
Frame

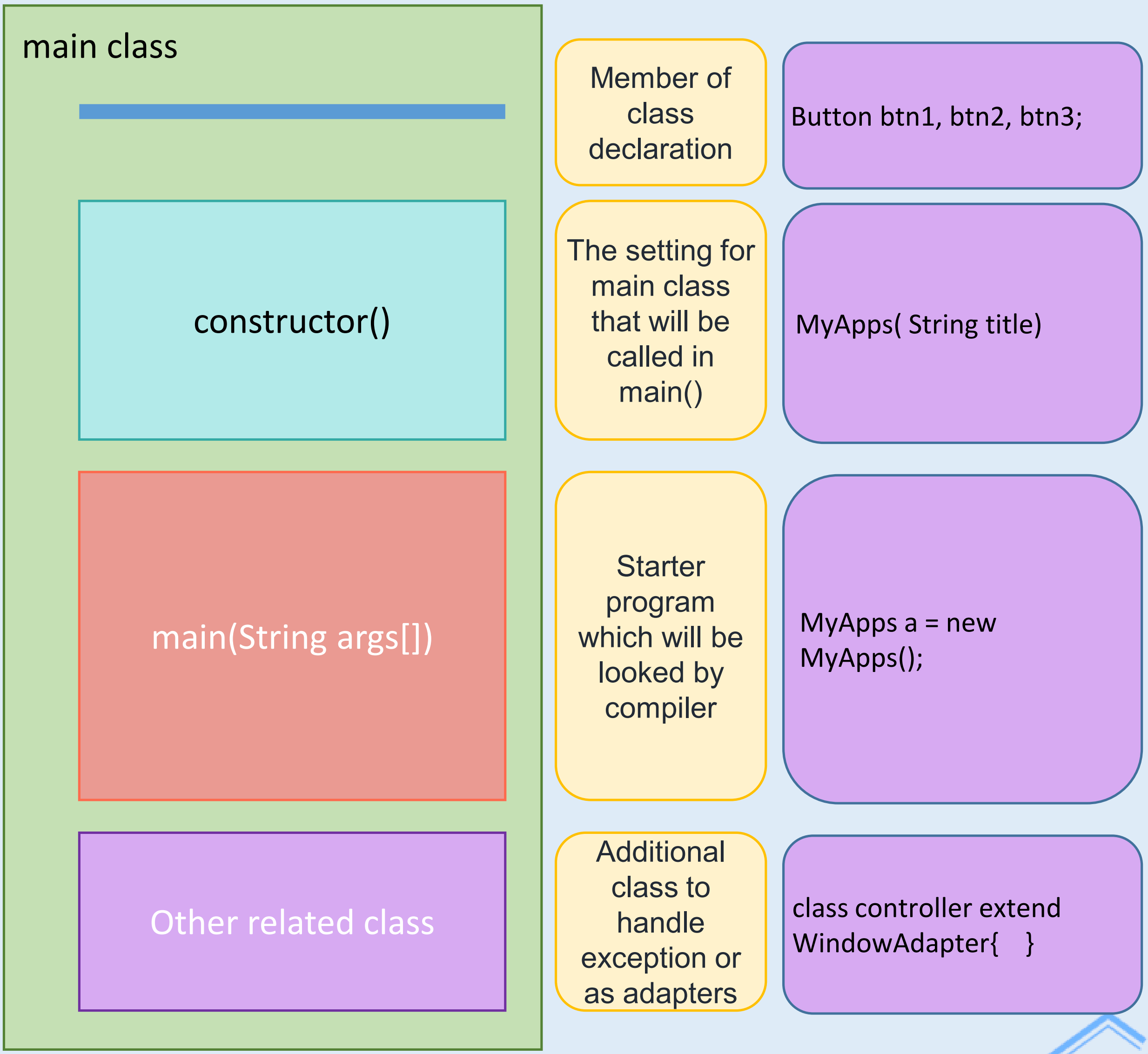
TextArea
TextField

Creating Java program

a) Setup a project, and the structure of files, consider good file name and easy access

b) Then identify the main class. Should it extends from any class?

c) Content of main() method, as wil be an entry point to start the execution of a program, consider the access and relation to main class, any object for execution



main class

	Member of class declaration	Button btn1, btn2, btn3;
constructor()	The setting for main class that will be called in main()	MyApps(String title)
main(String args[])	Starter program which will be looked by compiler	MyApps a = new MyApps();
Other related class	Additional class to handle exception or as adapters	class controller extend WindowAdapter{ }

Create FRAME

The class **Frame** is a top level window with border and title. It uses BorderLayout as default layout manager. The class is from **java.awt.Frame**

	Constructor & Description
1	**Frame()** Constructs a new instance of Frame that is initially invisible.
2	**Frame(GraphicsConfiguration gc)** Constructs a new, initially invisible Frame with the specified GraphicsConfiguration.
3	**Frame(String title)** Constructs a new, initially invisible Frame object with the specified title.
4	**Frame(String title, GraphicsConfiguration gc)** Constructs a new, initially invisible Frame object with the specified title and a GraphicsConfiguration.

Steps on creating basic container using Frame

1. Create blank java file
2. Write import statement to use package
3. Write a complete structure file consist of main class, main method and constructor of main class
4. Set container (Frame) in constructor
5. Create object of main class

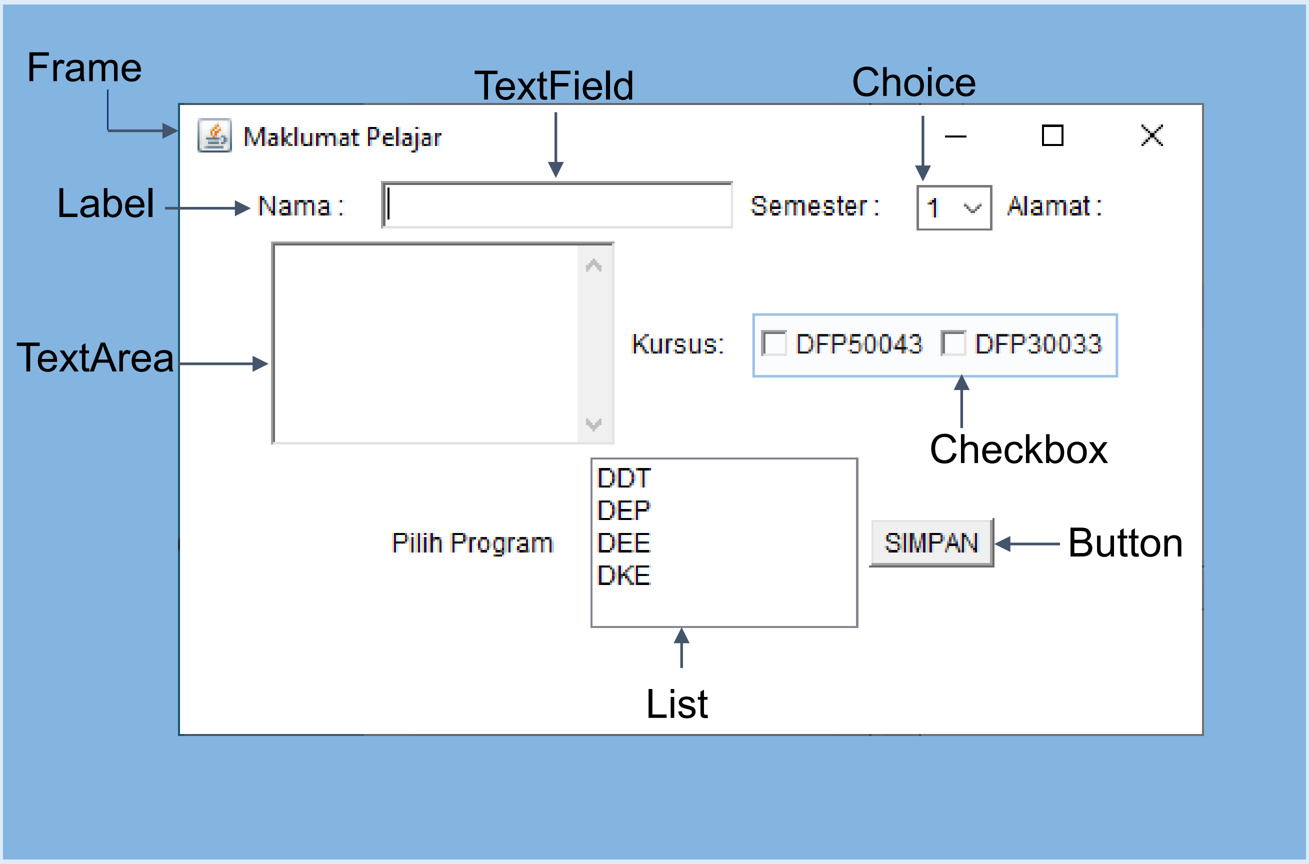
Create MENU

The Menu class represents pull-down menu component which is deployed from a menu bar from **java.awt.Menu** class

	Constructor & Description
1	**Menu()** Constructs a new menu with an empty label.
2	**Menu(String label)** Constructs a new menu with the specified label.
3	**Menu(String label, boolean tearOff)** Constructs a new menu with the specified label, indicating whether the menu can be torn off.

Identify awt Components

a) Frame
b) Label
c) Button
d) CheckBox.
e) Choice
f) List
g) ScrollPane
h) TextField
i) TextArea

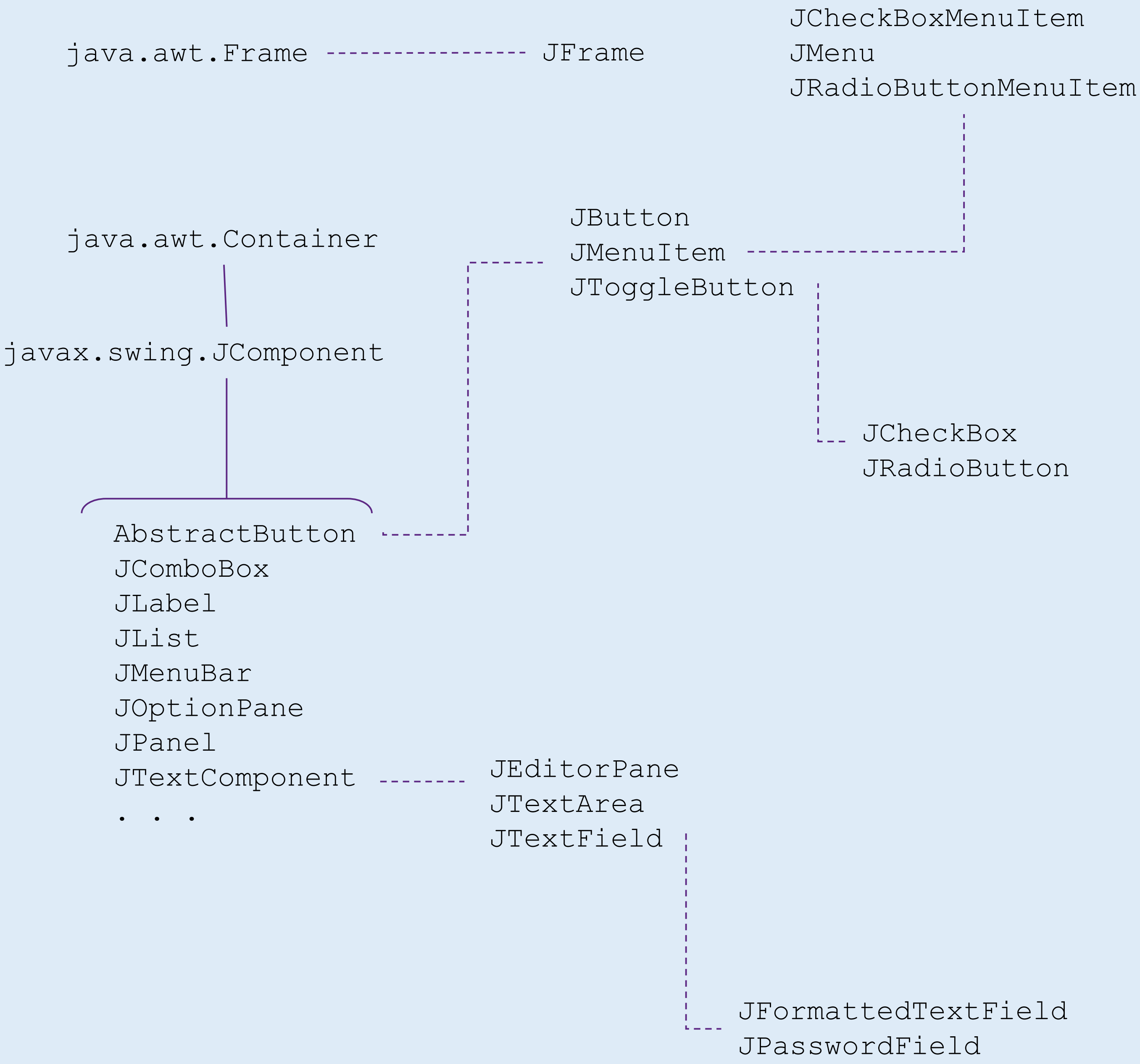


Example Components Application

JAVA source code

```
public AllAWTcomponents(String str){
    super(str);
    setLayout(new FlowLayout());
    Label lName=new Label("Nama :");
    TextField tname=new TextField(20);
    Label lSem=new Label("Semester :");
    Choice pilihSem = new Choice();
    pilihSem.add("1");
    pilihSem.add("2");
    pilihSem.add("3");
    pilihSem.select("1");
    Label lAlamat=new Label("Alamat :");
    TextArea talamat=new TextArea(5,20);
    Label lCourse=new Label("Kursus:");
    Checkbox cIPT=new Checkbox("DFP50043");
    Checkbox cHCI=new Checkbox("DFP30033");
    Label lProgram=new Label("Pilih Program");
    List lstProgram=new List(5,false);
    String p[ ]={"DDT","DEP","DEE","DKE"};

    Button bSimpan=new Button("SIMPAN");

    add(lName);
    add(tname);
    add(lSem);
    add(pilihSem);
    add(lAlamat);
    add(talamat);
    add(lCourse);
    add(cIPT);
    add(cHCI);
    add(lProgram);

    for(int i=0; i<p.length; i++) lstProgram.add(p[i]);

    add(lstProgram);
    add(bSimpan);
  }
```

GUI Hierarchy – **javax.swing Package**:



java.awt.Frame

JFrame

JCheckBoxMenuItem
JMenu
JRadioButtonMenuItem

java.awt.Container

javax.swing.JComponent

JButton
JMenuItem
JToggleButton

JCheckBox
JRadioButton

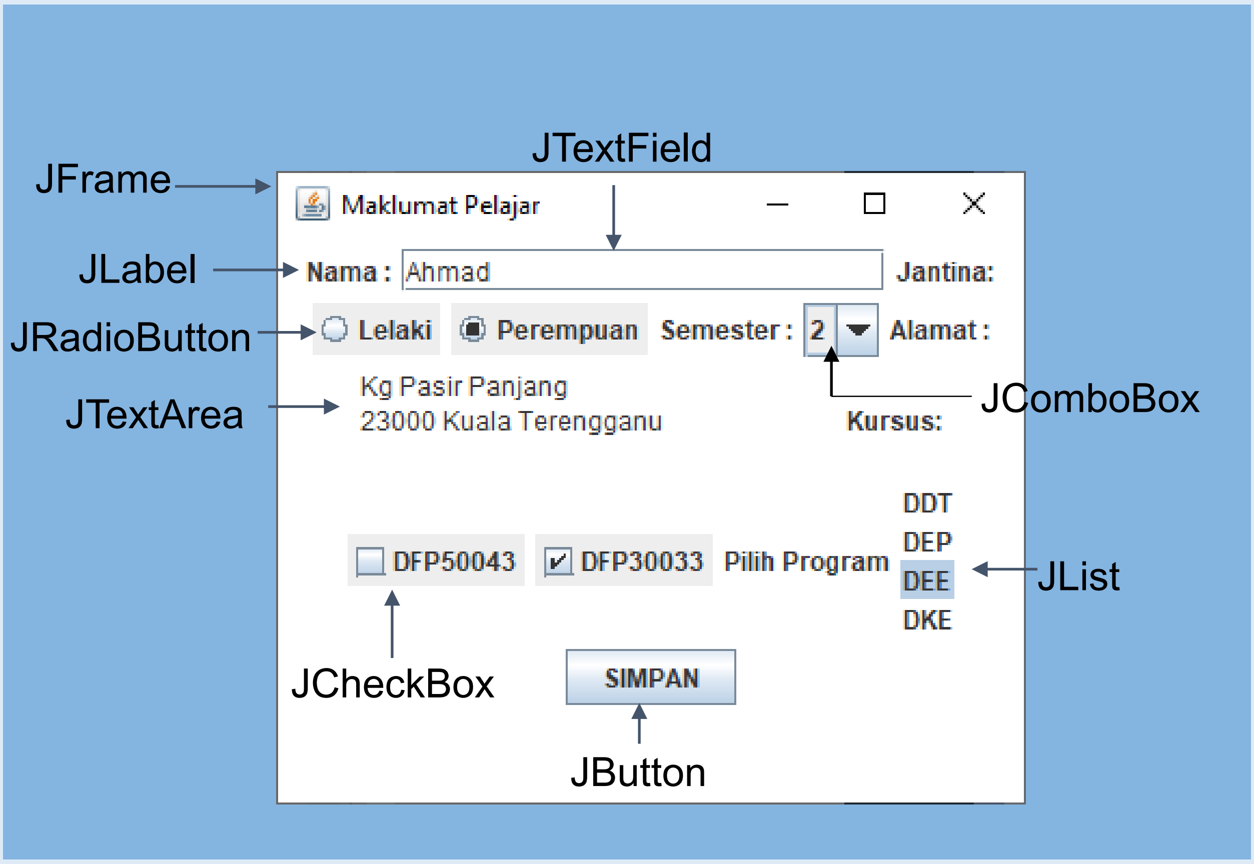
AbstractButton
JComboBox
JLabel
JList
JMenuBar
JOptionPane
JPanel
JTextComponent
. . .

JEditorPane
JTextArea
JTextField

JFormattedTextField
JPasswordField

Identify swing Components

a) JFrame
b) JLabel
c) JButton
d) JCheckBox.
e) JRadioButton
f) JList
g) JScrollPane
h) JTextField
i) JTextArea
j) JOptionPane



Maklumat Pelajar

Nama : Ahmad

Jantina: ○ Lelaki ● Perempuan

Semester : 2

Alamat : Kg Pasir Panjang
23000 Kuala Terengganu

Kursus: □ DFP50043 ☑ DFP30033

Pilih Program: DDT, DEP, DEE, DKE

SIMPAN

Example Components Application

JAVA source code

```
public AllSwingComponent(String str){
    super(str);
    setLayout(new FlowLayout());

    JLabel lName=new JLabel("Nama :");
    JTextField tname=new JTextField(20);
    JLabel lSem=new JLabel("Semester :");
    JComboBox pilihSem = new JComboBox();
    pilihSem.addItem("1");
    pilihSem.addItem("2");
    pilihSem.addItem("3");
    pilihSem.setSelectedIndex(2);

    JLabel lAlamat=new JLabel("Alamat :");
    JTextArea talamat=new JTextArea(3,20);

    JLabel lCourse=new JLabel("Kursus:");
    JCheckBox cIPT=new JCheckBox("DFP50043");
    JCheckBox cHCI=new JCheckBox("DFP30033");

    JLabel lgender=new JLabel("Jantina:");
    JRadioButton rbmale = new JRadioButton("Lelaki");
    JRadioButton rbfemale = new JRadioButton("Perempuan",true);
    ButtonGroup group = new ButtonGroup();
    group.add(rbmale); group.add(rbfemale);
    JLabel lProgram=new JLabel("Pilih Program");
    String p[ ]={"DDT","DEP","DEE","DKE"};
    JList lstProgram=new JList(p);
    JButton bSimpan=new JButton("SIMPAN");
    add(lName);
    add(tname);
    add(lgender);
    add(rbmale);add(rbfemale);
    add(lSem);
    add(pilihSem);
    add(lAlamat);
    add(talamat);
    add(lCourse);
    add(cIPT);
    add(cHCI);
    add(lProgram);
    add(lstProgram);
    add(bSimpan);
}
```

JOptionPane

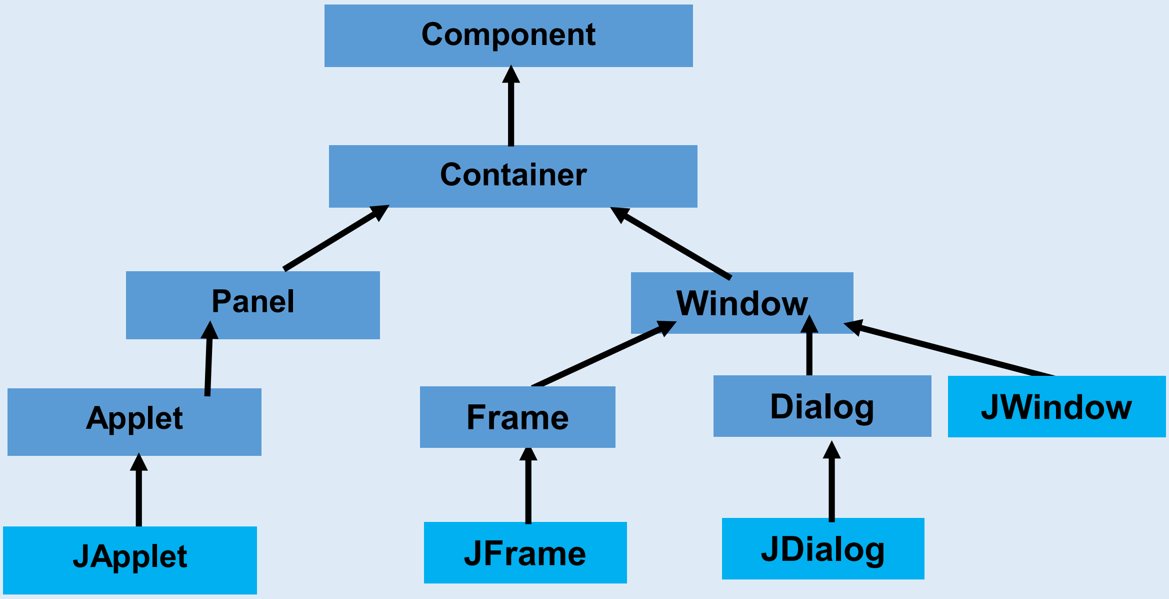
The JOptionPane is a subclass of JComponent class which includes static methods for creating and customizing modal dialog boxes using a simple code. The JOptionPane is used instead of JDialog to minimize the complexity of the code. The JOptionPane displays the dialog boxes with one of the four standard icons (question, information, warning, and error) or the custom icons specified by the user.

JOptionPane class is used to display **four** types of dialog boxes.

JAVA source code

```
  JFrame frame = new JFrame("JOptionPane Test");
     frame.setSize(400, 400);
     frame.setVisible(true);
     JOptionPane.showMessageDialog(frame, "Wake up!");
     JOptionPane.showMessageDialog(frame, "You are late, please hurry","Urgent message",
JOptionPane.WARNING_MESSAGE);
     int result = JOptionPane.showConfirmDialog(null, "Do you want to enter school?");
     switch(result) {
       case JOptionPane.YES_OPTION:
            System.out.println("Yes");
       break;
       case JOptionPane.NO_OPTION:
            System.out.println("No");
       break;
       case JOptionPane.CANCEL_OPTION:
            System.out.println("Cancel");
       break;
       case JOptionPane.CLOSED_OPTION:
            System.out.println("Closed");
       break;
     }
     String name = JOptionPane.showInputDialog(null, "Please enter your name.");
     System.out.println(name);
     JTextField userField = new JTextField();
     JPasswordField passField = new JPasswordField();
     String message = "Please enter your user name and password.";
     result = JOptionPane.showOptionDialog(frame, new Object[]  {message, userField, passField},
                    "Login", JOptionPane.OK_CANCEL_OPTION, JOptionPane.QUESTION_MESSAGE, null,
null, null);
     if (result == JOptionPane.OK_OPTION)
             System.out.println(userField.getText() + " " + new String(passField.getPassword()));
```

Relationship of awt and swing package



	AWT	SWING
1.	It is an API used to develop window-based applications in Java.	Swing is a graphical user interface (GUI) and a part of Oracle's Java Foundation Classes that are used to design different applications.
2.	Its components are heavy weighted*.	Its components are light weighted*.
3.	In Java AWT, the components are platform dependent.	In Java swing, the components are independent.
4.	The functionality of JAVA AWT is less as compared to the Java swing.	The functionality of the JAVA swing is higher than AWT.
5.	It requires more time for execution.	It requires less time for execution.
6.	It has less powerful components compared to the Java swing.	It has more powerful components than Java AWT.

Organizes Layout with LayoutManager

Layout manager determine the size and position of components within a container.
The layout manager is responsible for deciding the layout policy and size of each of its container.

Java Layout Managers are:

- FlowLayout
- BorderLayout
- GridLayout
- BoxLayout

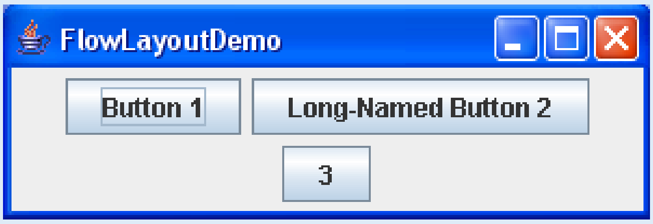
FlowLayout

Features of the layout are:

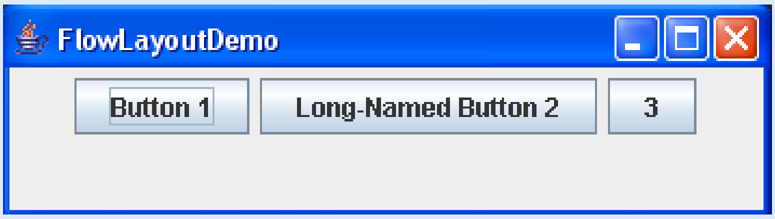
- Default layout for Panel class
- Components are added from left to right
- Default alignment is centered
- Uses component preferred size

Creating and Setting up FlowLayout

Method / Constructor	**Purpose**
FlowLayout()	To construct new flow layout for the frame



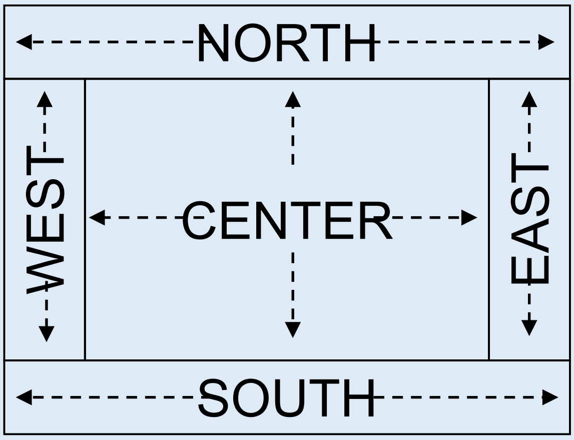
After program or user resizes



BorderLayout

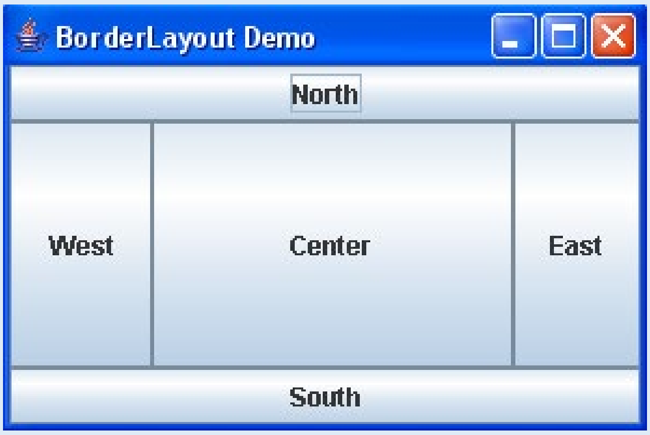
Features of the layout are:

- Default layout for Frame class
- Components added to **specific region**:



Creating and Setting up BorderLayout

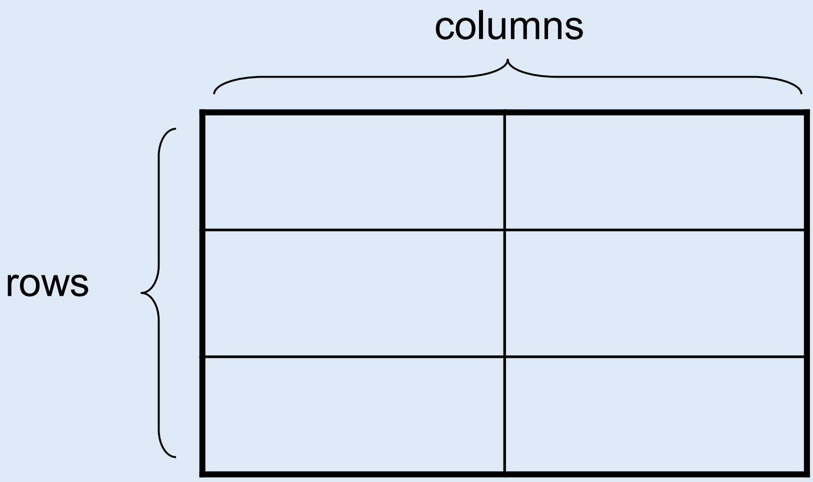
Method / Constructor	**Purpose**
BorderLayout()	To construct new border layout for the frame



GridLayout

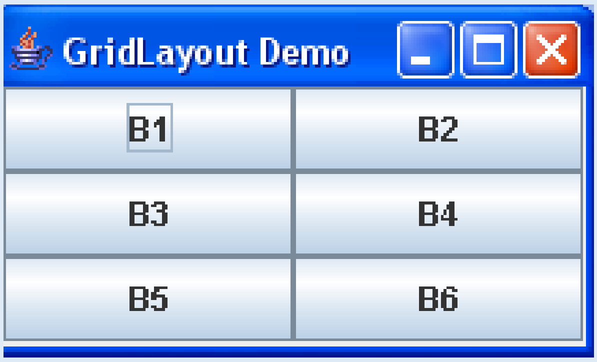
Features of the layout are:

- Components are added from left to right and top to bottom
- All regions have equal size
- The constructor specifies the rows and columns as the following:



Creating and Setting up GridLayout

Method / Constructor	**Purpose**
GridLayout(int, int)	To construct new grid layout for the frame



JOptionPane

The **JOptionPane** is a subclass of **JComponent** class which includes static methods for creating and customizing **modal dialog boxes** using a simple code. The **JOptionPane** is used instead of **JDialog** to minimize the complexity of the code. The **JOptionPane** displays the dialog boxes with one of the four standard icons (**question, information, warning, and error**) or the custom icons specified by the user.

Example creating custom content for dialog

JAVA source code

```
public static void main(String[] args{
   // create a simple jpanel
   JPanel panel = new JPanel();
   panel.setBackground(Color.BLUE);

   JButton btn = new JButton("TEsting");
   JButton btn1 = new JButton("TEsting");
   JButton btn2 = new JButton("TEsting");
   panel.add(btn);panel.add(btn1);panel.add(btn2);

   JOptionPane.showMessageDialog(null, panel);

   System.exit(0);
}
```

JOptionPane

Another example creating custom content for dialog

JAVA source code

```
public static void main(String[] args)
 {
   final String s = "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean eu nulla urna. Donec sit amet risus nisl, a porta enim. Quisque luctus, ligula eu scelerisque gravida, tellus quam vestibulum urna, ut aliquet sapien purus sed erat. Pellentesque consequat vehicula magna, eu aliquam magna interdum porttitor. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Sed sollicitudin sapien non leo tempus lobortis. Morbi semper auctor ipsum, a semper quam elementum a. Aliquam eget sem metus.";
   final String html1 = "<html><body style='width: ";
   final String html2 = "px'>";

   Runnable r = new Runnable() {

     @Override
     public void run() {
       JOptionPane.showMessageDialog(
           null, new JLabel(html1 + "200" + html2 + s));
       JOptionPane.showMessageDialog(
           null, new JLabel(html1 + "300" + html2 + s));
     }
   };
   SwingUtilities.invokeLater(r);
 }
```

JOptionPane

Another example creating custom content for dialog

JAVA source code

```
public class TestDialog
{
  // *** your image path will be different *****
  private static final String IMG_PATH = "src/images/index.jpg";

  public static void main(String[] args) {
       try {
             BufferedImage img = ImageIO.read(new File(IMG_PATH));
             ImageIcon icon = new ImageIcon(img);
             JLabel label = new JLabel(icon);
             JOptionPane.showMessageDialog(null, label);
        }catch (IOException e) {
            e.printStackTrace();
        }
   }
}
```

Event Handling Concepts

Event as a signal to the program that something has happened.

In the event model, there are **three** participants:

- The event source, the object whose state changes
- The event object, encapsulates the state changes in the event source
- The event listener, the object that wants to be notified

The event source is triggered by external user actions, for example:

- Example actions: mouse movement, button clicks, keystrokes, internal program activities (timer)
- Example application: java.util.EventObject, event class is EventObject which creates events instance

Event object deals with special types of event, example classes are:

- ActionEvent
- MouseEvent
- KeyEvent
- ItemEvent
- WindowEvent

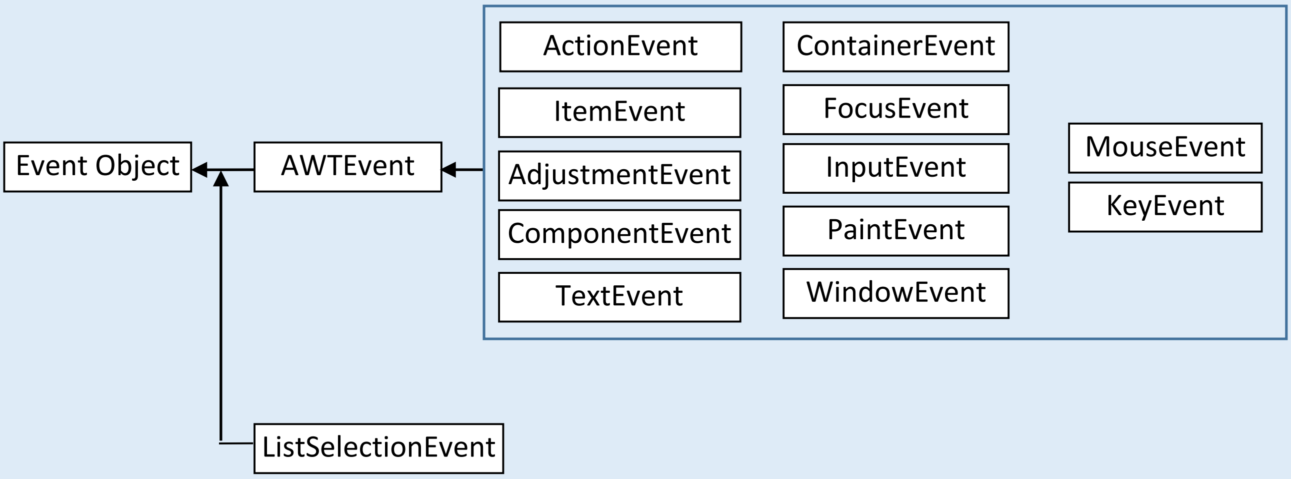
For example a WindowEvent is generated by an instance of the Window class or its subclass. JButton is not a subclass of Window, therefore, it cannot generate the WindowEvent. JButton can generate MouseEvent and ActionEvent.

Event listener are classes that will catch the event object, event listener are:

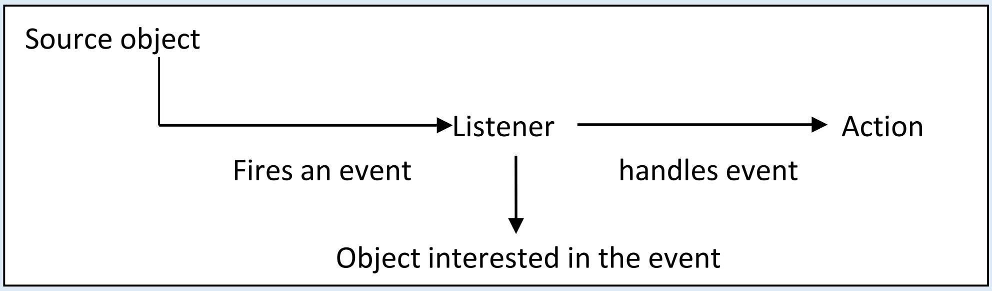
- ActionListener
- ItemListener
- MouseListener
- KeyListener

Event Handling Concepts

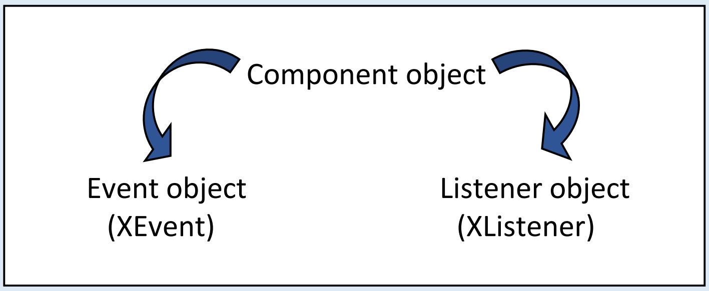
The relationship and classes involved are as the following:



Component which an event is fired or generated are executed as following:



The diagram below is simplified of Event processing:



Event Handling Concepts

Summary Of Listener:

Implements listener in main class

JAVA source code

```
public class Listener1 extends JFrame implements ActionListener{}
```

Writing a listener as a nested class

JAVA source code

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

class ButtonListener implements ActionListener{

  public void actionPerformed(ActionEvent ae){
    Toolkit.getDefaultToolkit().beep();
  }
}

public class UseActionListener {

  public static void main(String[] a) {

       JFrame frame = new JFrame("Popup JComboBox");
       frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

       JButton source = new JButton("Ring the bell!");
       source.addActionListener(new ButtonListener());
       frame.add(source, BorderLayout.SOUTH);
       source.addActionListener(new ButtonListener());
       frame.setSize(300, 200);
       frame.setVisible(true);
    }
}
```

Event Handling Concepts

Writing listener as anonymous class

JAVA source code

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

public class Listener3 {

  public static void main(String[] args) {
    JFrame.setDefaultLookAndFeelDecorated(true);
    JFrame frame = new JFrame();
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    JButton button = new JButton("Select File");

    button.addActionListener(new ActionListener() {

      public void actionPerformed(ActionEvent e) {
        JFileChooser fileChooser = new JFileChooser();
        int returnVal = fileChooser.showOpenDialog(null);

        if (returnVal == JFileChooser.APPROVE_OPTION) {
          System.out.println(fileChooser.getSelectedFile().getName());
        }
      }
    });

    frame.add(button);
    frame.pack();
    frame.setVisible(true);
  }
}.
```

The three approaches are written depends to the flow of a program

Only CLASSES which IMPLEMENTS ActionListener will have ActionPerformed() method

Program Discussion:

JAVA source code

```
import javax.swing.event.*;
import java.awt.*;
import javax.swing.*;
public class solve extends JFrame implements ListSelectionListener{
   JList b,b1,b2;
   JLabel l1;

   public solve(){
      JPanel p =new JPanel();
      JLabel l= new JLabel("select your birthday");
      l1= new JLabel();

      //String array to store weekdays
      String month[]= { "January", "February", "March","April", "May", "June", "July", "August",
      "September", "October", "November", "December"};

      //create a array for months and year
      String date[]=new String[31],year[]=new String[31];

      //add month number and year to list
      for(int i=0;i<31;i++){
            date[i]=""+(int)(i+1);
            year[i]=""+(int)(2018-i);
      }

      //create lists
      b= new JList(date);
      b1= new JList(month);
      b2= new JList(year);

         //set a selected index
         b.setSelectedIndex(2);
         b1.setSelectedIndex(1);
         b2.setSelectedIndex(2);
         l1.setText(b.getSelectedValue()+" "+b1.getSelectedValue()
                                          +" "+b2.getSelectedValue());
```

Program to create a list and add itemListener to it

Event Handling Concepts

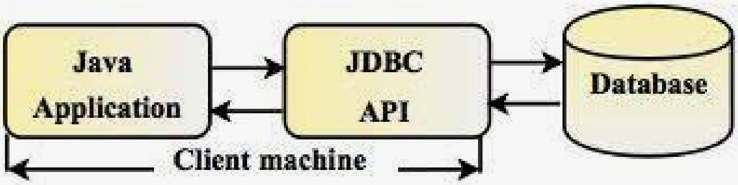
JAVA source code

```
//add item listener
        b.addListSelectionListener(this);
        b1.addListSelectionListener(this);
        b2.addListSelectionListener(this);

        //add list to panel
        p.add(l);
        p.add(b);
        p.add(b1);
        p.add(b2);
        p.add(l1);

        add(p);
        setSize(500,600);
        show();
  }
  public static void main(String[] args){
    solve s=new solve();
  }
  public void valueChanged(ListSelectionEvent e){
        //set the text of the label to the selected value of lists
        l1.setText(b.getSelectedValue()+" "+b1.getSelectedValue()
                                  +" "+b2.getSelectedValue());
  }
}
```

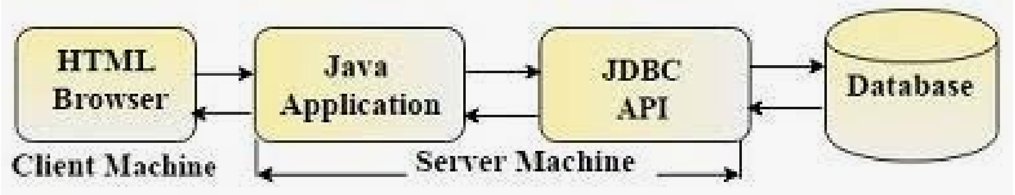
Database and GUI

Introduction to database connection in Java:

Java Database Connectivity(JDBC) is the Java Application Programming Interface (API) that manages connecting to a database, issuing queries and commands, and handling result sets obtained from the database. These includes all tabular data stored in relational databases such as Oracle, MySQL, MS Access and many others.
The JDBC architecture consists of two-tier and three-tier processing models to access a database. They are as described below:



Two-tier processing models



Three-tier processing models

JDBC drivers are client-side adapters (installed on the client machine, not on the server) that convert requests from Java programs to a protocol that the DBMS can understand. There are 4 types of JDBC drivers:

- Type-1 driver or JDBC-ODBC bridge driver
- Type-2 driver or Native-API driver
- Type-3 driver or Network Protocol driver
- Type-4 driver or Thin driver

JDBC Net pure Java driver(Type 4 driver) is the fastest driver for localhost and remote connections because it directly interacts with the database by converting the JDBC calls into vendor-specific protocol calls

Introduction to database connection in Java:

Interfaces of JDBC API

A list of popular interfaces of JDBC API is given below:

- Driver interface
- Connection interface
- Statement interface
- PreparedStatement interface
- CallableStatement interface
- ResultSet interface
- ResultSetMetaData interface
- DatabaseMetaData interface
- RowSet interface

Database and GUI

Introduction to database connection in Java:

Example JDBC application

JAVA source code

```
import java.sql.*;
public class JDBCDemo {
   public static void main(String args[])       throws SQLException, ClassNotFoundException
   {
      String driverClassName  = "sun.jdbc.odbc.JdbcOdbcDriver";
      String url = "jdbc:odbc:XE";
      String username = "scott";
      String password = "tiger";
      String query  = "insert into students values(109, 'bhatt')";

      Class.forName(driverClassName);                  // Load driver class

      // Obtain a connection
      Connection con = DriverManager.getConnection(url, username, password);

      Statement st = con.createStatement();         // Obtain a statement

      // Execute the query
      int count = st.executeUpdate(query);
      System.out.println("number of rows affected by this query= " + count);

      con.close();                                              // Closing the connection
   }
} // class
```

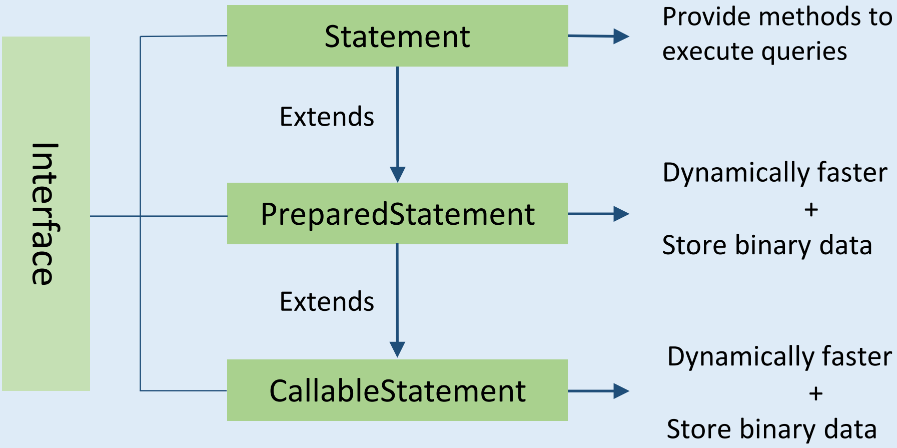
Database is not part of Java environment. It is external entity to Java.

All Java codes related to database (connection, operation, processing) need to be handled, therefore they must be throws or catched else there will be error.

Introduction to database connection in Java:

Understanding types of statement interface

- Statement interface
- PreparedStatement interface
- CallableStatement interface


Interface
Statement
Extends
PreparedStatement
Extends
CallableStatement
Provide methods to execute queries
Dynamically faster + Store binary data
Dynamically faster + Store binary data

Statement: It is used for general-purpose access to the database by executing the static SQL query at runtime. Example:

```
Statement st = conn.createStatement( );
ResultSet rs = st.executeQuery();
```

PreparedStatement: It is used when we need to give input data to the query at runtime and also if we want to execute SQL statements repeatedly. It is more efficient than a statement because it involves the pre-compilation of SQL. Example:

```
String SQL = "Update item SET limit = ? WHERE itemType = ?";
PreparedStatement ps = conn.prepareStatement(SQL);
ResultSet rs = ps.executeQuery();
```

CallableStatement: It is used to call stored procedures on the database. It is capable of accepting runtime parameters. Example:

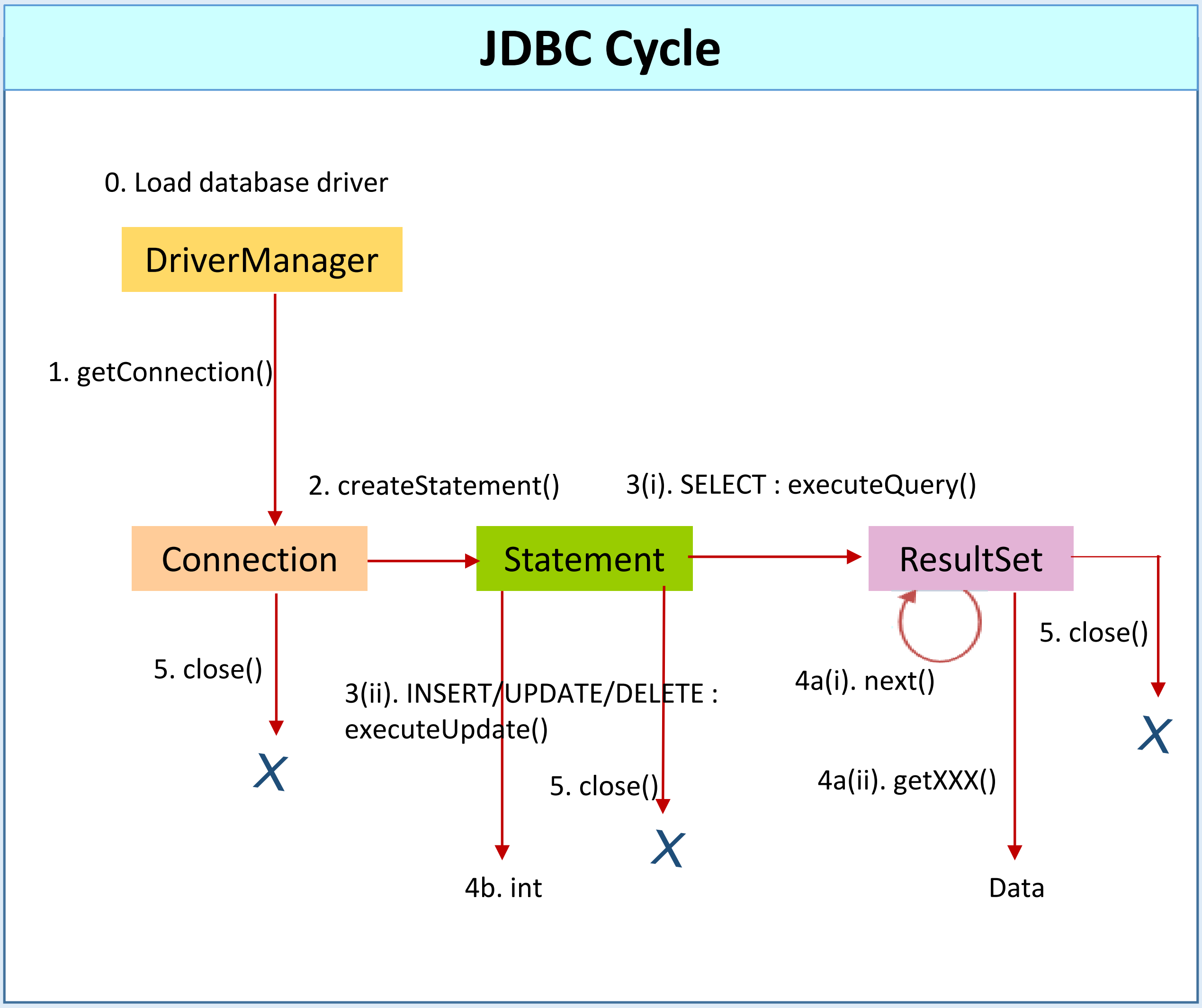
```
CallableStatement cs = con.prepareCall("{call SHOW_CUSTOMERS}");
ResultSet rs = cs.executeQuery();
```

Introduction to database connection in Java:

Comparison of Statement Types

Statement	PreparedStatement	CallableStatement
It is used to execute normal SQL queries.	It is used to execute parameterized or dynamic SQL queries.	It is used to call the stored procedures.
It is preferred when a particular SQL query is to be executed only once.	It is preferred when a particular query is to be executed multiple times.	It is preferred when the stored procedures are to be executed.
You cannot pass the parameters to SQL query using this interface.	You can pass the parameters to SQL query at run time using this interface.	You can pass 3 types of parameters using this interface. They are – IN, OUT and IN OUT.
This interface is mainly used for DDL statements like CREATE, ALTER, DROP etc.	It is used for any kind of SQL queries which are to be executed multiple times.	It is used to execute stored procedures and functions.
The performance of this interface is very low.	The performance of this interface is better than the Statement interface (when used for multiple execution of same query).	The performance of this interface is high.

Summary of database integration in Java program:



JDBC Cycle

0. Load database driver

DriverManager

1. getConnection()

2. createStatement()

3(i). SELECT : executeQuery()

Connection

Statement

ResultSet

5. close()

3(ii). INSERT/UPDATE/DELETE : executeUpdate()

4a(i). next()

5. close()

X

5. close()

4a(ii). getXXX()

X

X

4b. int

Data

Your project must first have a database driver in library, then to make database connection the driver is declared to open the connection. Query to database require object Statement using either executeQuery() or executeUpdate(), its depends whether to modify or not the database content. The result returned from database is then processed typically using looping to fetch records row by row.

Section 3

Simple Questions

- ❑ EXERCISE ON BASIC JAVA PROGRAMMING
- ❑ EXERCISE INPUT OUTPUT STREAM PROGRAMMING
- ❑ EXERCISE INTRODUCTION TO OOP

Exercise On Basic Java Programming

Question 1
Indicate True or False for the following statements

a.	Every element in an array has the same type	True/False
b.	The array size is fixed after it is declared	True/False
c.	The array size is fixed after it is created	True/False
d.	The element in the array must be of primitive data type	True/False

Question 2
Which of the following statements is valid array declarations?

a.	int i = new int (30);	
b.	double d[ ] = new double [30];	
c.	char [ ] r = new char(1 . . 3)	
d.	int i [ ] = (3 , 4, 3, 2);	
e.	float f [ ] = {2.3, 4.5, 5.6 }	
f.	char [ ] c = new char();	

Exercise On Basic Java Programming

Question 3.

Choose True or False for the following statements

1	Throwable is a base class for Exception and Error class	True/False

2	The block statement below are valid:
	try{ . . . }
	catch(Exception e){ . . .}
		True/False

3	The block statement below are valid:
	try{ . . . }
	try{ . . . }
	finally{ . . .}
		True/False

4	The block statement below are valid:
	try{ . . . }
	catch(IOException i){ . . .}
	catch(Exception e){ . . .}
	finally{ . . .}
		True/False

Question 4.

What is the output for the following program if user enters “GOOD DAY”?

```
inputStr = Scan.next()
try{
        X= Integer.parseInt(inputStr);
        if(X>100){
                 throw new Exception(“Out of Bound”);
                 }
}catch(Exception e){
                 System.out.println(“Cannot convert to int”);
}finally{
                  System.out.println(“Done”);
}
```

Exercise On Basic Java Programming

Check your answer

Question 1

a.	True
b.	False
c.	True
d.	False

Question 2

a.	Invalid
b.	Valid
c.	Invalid
d.	Invalid
e.	Valid
f.	Invalid

Question 3

1	True
2	True
3	False
4	True

Question 4
The output should be:

```
Cannot convert to int
Done
```

Exercise Input Output Stream Programming

Question 1.

Write the following codes and analyze the result, what is the solution if something wrong or not right fro the output?

```
import java.io.*;
class IntegerInput
{
 public static void main (String[] args) throws IOException
 {
    InputStreamReader inStream = new InputStreamReader(System.in);
    BufferedReader stdin = new BufferedReader(inStream);
    int n;
    System.out.println("Enter 1 or 2 : ");
    n=stdin.read();
        switch(n){
                   case 1:
                     System.out.println("one");
                   case 2:
                     System.out.println("two");
                   default:
                     System.out.println("not one and two");
         }
     }
}
```

Exercise Input Output Stream Programming

Question 2

Complete the blank line in program codes based on comment given.
Then write in suitable IDE to see the result.

```
____________________ // add classes related to program
class IntegerInputSample
{
  public static void main (String[] args) throws IOException
  {
    InputStreamReader inStream = new InputStreamReader(System.in);
    BufferedReader stdin = new BufferedReader( __________ ); // InputStreamReader object
    String str;
    _________________  // declare an int variable
     System.out.println("Enter an integer:");
    str = stdin.readLine();
    __________________ // convert str to integer data type
    System.out.println("Integer Value: "+num);
  }
}
```

Exercise Input Output Stream Programming

Check your answer

Question 1

The following is sample output that will be displayed if user enters 1.

```
Enter 1 or 2 :
1
not one and two
```

The program has flaws at switch block. The program will continue running although it does not agree with the case. To correct the code, one must add break for each case.

Question 2

Integer variable defined in the program must be used to assigned value converted from String type.

```
import java.io.*;
class IntegerInput
{
  public static void main (String[] args) throws IOException
  {
    InputStreamReader inStream = new InputStreamReader(System.in);
    BufferedReader stdin = new BufferedReader(inStream);
    String str;
    int num;
    System.out.println("Enter an integer:");
    str = stdin.readLine();
    num = Integer.parseInt(str);
    System.out.println("Integer Value: "+num);
  }
}
```

Exercise Introduction To OOP

QUESTION 1

1. If class A inherits from Class B, which is a superclass? Which is a subclass?

2. Draw a diagram that shows Class A is inheriting from Class B

3. What are the other name for superclass and subclass?

4. If we have Animal, Insect and Mammal classes, which one will be a superclass?

5. Model different types of vehicles, using inheritance. Include Vehicle, Automobile, Motorcycle, Sports Car, Sedan, Bicycle.

QUESTION 2

1. Graphically represent a Vehicle class and three Vehicle objects named car1, car2, car3.

2. Suppose the Vehicle class is used in a program that keeps track of vehicle registration for the Department of Motor Vehicles. What kinds of instance variables would you define for such Vehicle class?

3. Suppose the following formulas are used to compute the annual vehicle registration fee
   - For cars, the annual fee is 2 percent of its value
   - For trucks, the annual fee is 5 percent of its value.

   - Define 2 new classes for Car and Truck as subclasses of Vehicle. Hint: Associate class and instance variables to both Car and Truck to Vehicle.

QUESTION 3

Write the following code and understand the code:

```
class Book
{
 Book(String bname, String aname, int nopages)
 {
    book_name = bname;
    author_name = aname;
    no_of_pages = nopages;
 }

 public static void main(String args[])
  {
    Book b1 = new Book ("Java 2","Herbert Schildt",100);
  }
}
```

Exercise Introduction To OOP

QUESTION 4

Write the following code and understand the code:

```
public class person {
    String name;

    int age;
    float salary;

    public void getData(){
        name="suhaila";
        age=23;
        salary=150;
    }
    public void displayData(){
        System.out.println("Name is "+name+", with age "+age+", "+salary+ " is the salary");

    }
    public static void main(String[] args) {
        person me = new person();
        me.getData();
        me.displayData();
    }
}
```

Discussion What is the meaning of defining person as public class?

Exercise Introduction To OOP

QUESTION 5

Write the following code and understand the code:

```
public class bank_account {
    String name;
    float actnum;
    String acttype;
    float actbal;

    public void getData(){
        name="suhaila";
        actnum=233758578;
        acttype="saving";
        actbal=30000;
    }
    public void displayData(){

        System.out.println("Name: "+name);
        System.out.println("Act balance: "+actnum);
        System.out.println("Act type: "+acttype);
        System.out.println("Act bal: "+ actbal);
    }
    public static void main(String[] args) {
        bank_account me = new bank_account();
        me.getData();
        me.displayData();
    }
}
```

Discussion

What is accessor and mutator in java? Can you apply it for the program in Question 5?

QUESTION 6

Understand the following code :

```
interface calculator{

  public void add(int x, int y);

}
```

Then create calculatorScientific that will implements the interface.

Discussion What is the characteristic of an interface?

QUESTION 7

Write the following code and understand the code:

```
public abstract class auto {
    private String made;
    private double price;

    public String getMade() {        return made;
    }

    public double getPrice() {        return price;
    }

    public void setMade(String made) {        this.made = made;
    }

    public abstract double setPrice(double price);
}

public class chevy extends auto {
   public double setPrice(double price){        return price;
    }
}

public class ford extends auto{
    public double setPrice(double price){        return price;
    }
}

public class exec {
    public static void main(String[] args) {
      chevy c = new chevy();
      ford f = new ford();

      c.setMade("Chevy");
      c.setPrice(420000);
      f.setMade("Ford");
      f.setPrice(230000);
    }
}
```

Section 4

Activities

- ❑ **INTRODUCTION TO GUI**
- ❑ **MORE ON GUI**

This part consists of laboratory activities for simple problem solving using graphical user interfaces then followed by more advanced GUI manipulation. These activities requires only 20 to 35 minutes to finished. These exercise will evaluate student understanding and skills towards the practice set.

ACTIVITY 1

1. Create an empty frame by creating object of JFrame class. Set minimum method so that it can be displayed.

2. Then using another java file, create a frame by inheriting the JFrame class

3. Create multiple objects of JFrame dynamically using array.

Discussion

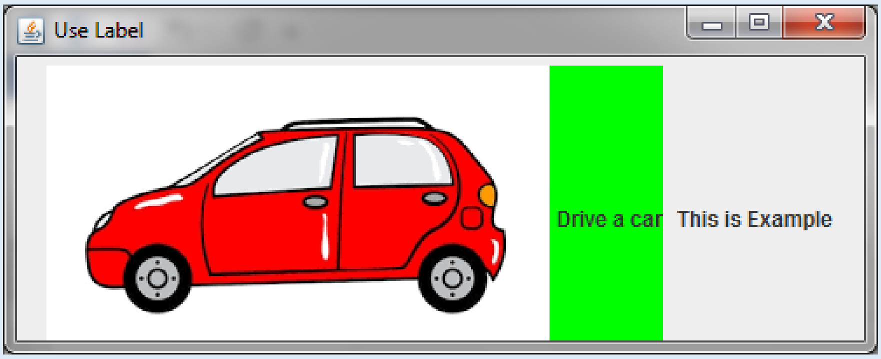
Both will produce the same output. What is the difference of both way of writing frame object?

ACTIVITY 2

Write the following codes and run it

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

public class ManipulateLabel extends JFrame{
    public ManipulateLabel(){
        ImageIcon myIcon = new ImageIcon("car.png");
        JLabel lbl1 = new JLabel(" This is Example");
        JLabel lbl = new JLabel("Drive a car", myIcon, JLabel.CENTER);
        lbl.setBackground(Color.green);
        lbl.setOpaque(true);
        setVisible(true);
        setLayout(new FlowLayout());
        setSize(400,200);
        setTitle("Use Label");
        add(lbl);
        add(lbl1);
    }
    public static void main(String[] args) {
        ManipulateLabel m = new ManipulateLabel();
        }
}
```

Sample Output:



ACTIVITY 3

Write the following codes and run it

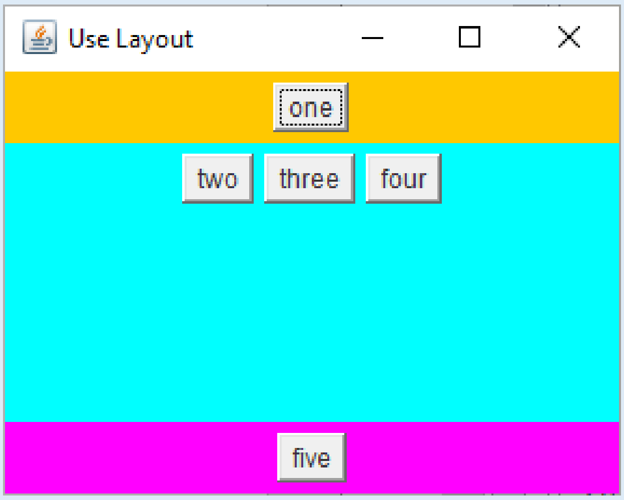
```
import java.awt.*;
import javax.swing.*;

public class obj extends JFrame {
   JButton btns[] = new JButton[5];
   String teks[] = {"Enter","Exit","Save","Edit","Add"};

   public obj(){
      setSize(300,400);
      setLayout(new FlowLayout());
      setTitle("Dynamic components");
      setVisible(true);

      for(int i=0;i<btns.length;i++){
         btns[i] = new JButton();
         btns[i].setText(teks[i]);
         add(btns[i]);
      }

   }

   public static void main(String[] args) {
      obj o = new obj();
   }

}
```

Discussion

Instead of using array to create object, what is another way to dynamically create object?

ACTIVITY 4

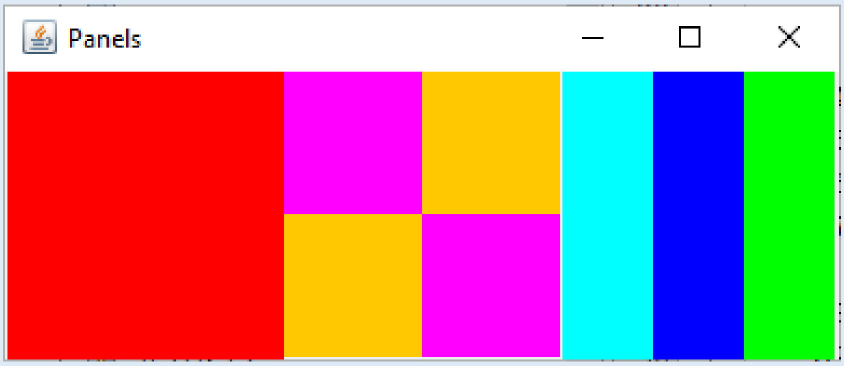
Use suitable layout to display the following GUI



1. Sketch your layout and label the containers for each part.
2. Then define the object name
3. Add the objects to its consecutive containers

ACTIVITY 5

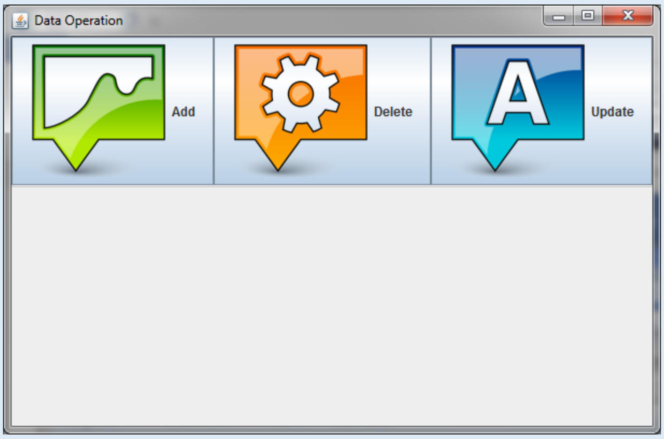
Use suitable layout to display spaces with colors as the following:



1. Sketch your layout and label the containers for each part.
2. Then define the object name
3. Add the object to component

ACTIVITY 6

Write a menu which attached a button on it. Display 3 menus with three different images.



1. Sketch your layout and label the containers for each part.
2. Then define the object name
3. Add the object to component

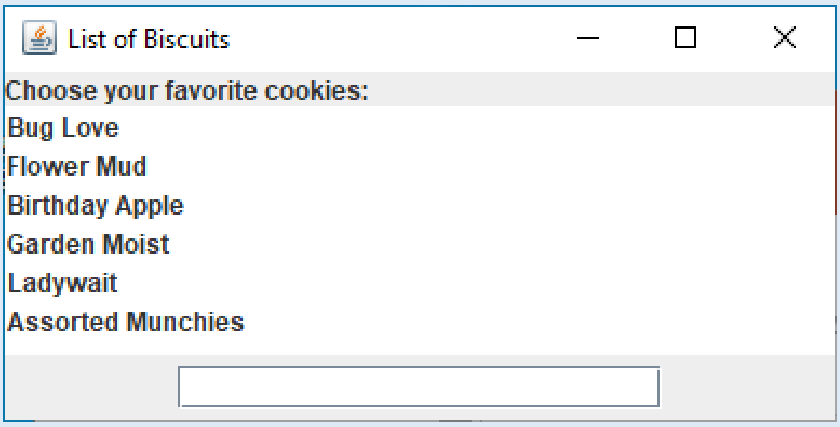
ACTIVITY 7

Creating a JMenu application.

1. Create a blank JFrame
2. Create 3 JMenu object with name Home, Insert and Design
3. Create a JMenuBar object and add to your container
4. Create another 3 JMenuItem (Copy, Pictures, Themes) to attach for each JMenu object.
5. Add listener to all your menu item object
6. Using anonymous listener, display the respond in a JLabel for each menu item selected.

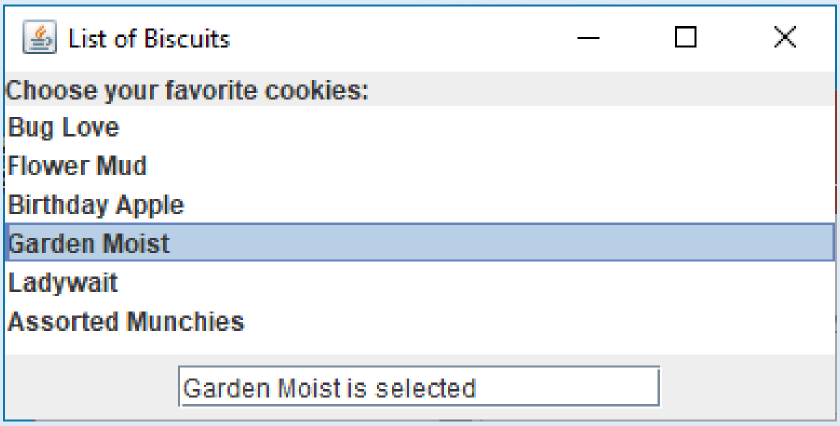
ACTIVITY 8

Write a menu which attached a button on it. Display 3 menus with three different images.



Choose your favorite cookies:
Bug Love
Flower Mud
Birthday Apple
Garden Moist
Ladywait
Assorted Munchies

Before



Choose your favorite cookies:
Bug Love
Flower Mud
Birthday Apple
Garden Moist
Ladywait
Assorted Munchies

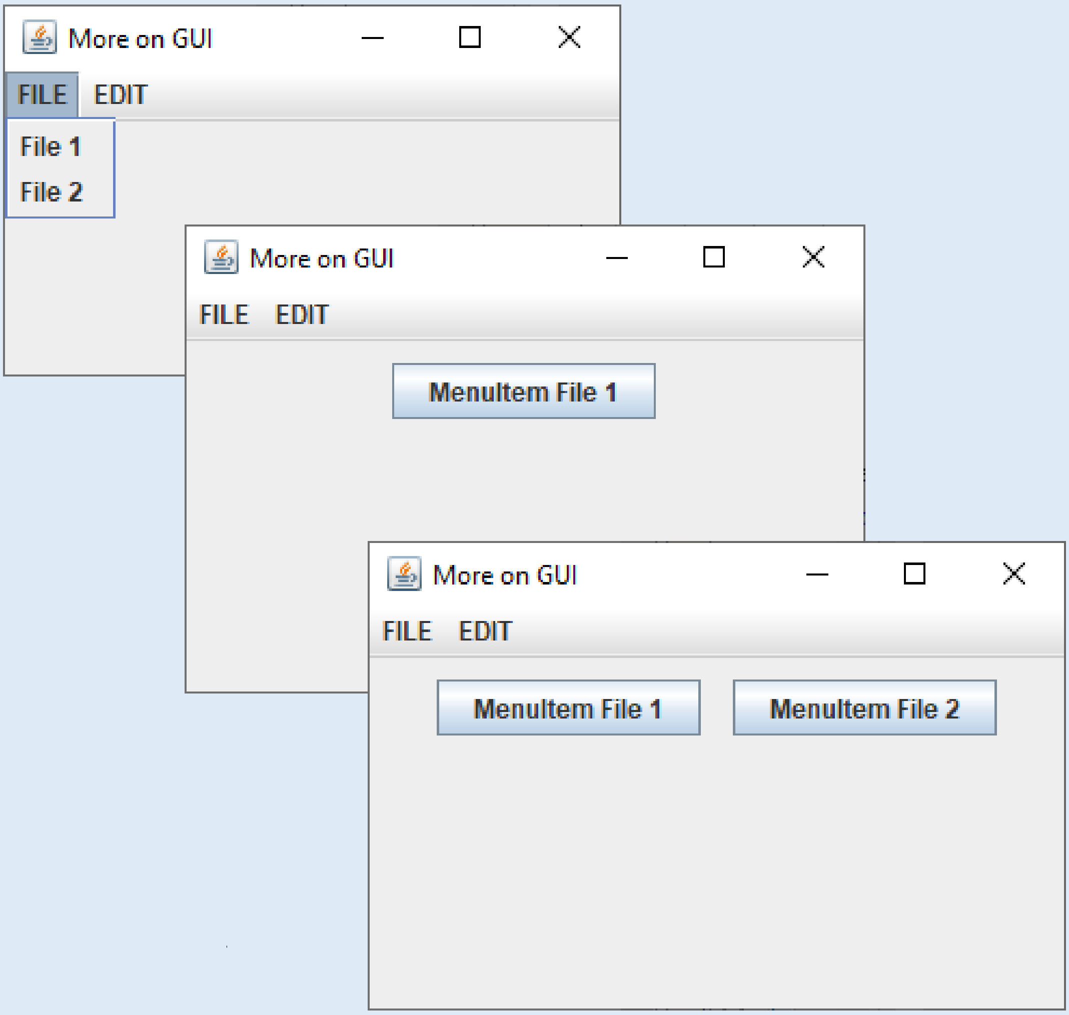
Garden Moist is selected

After

1. Based on Before diagram, create the same GUI.
2. Add event handling for JList to listen to the event
3. Display selected item in the text field as shown in After diagram

ACTIVITY 1

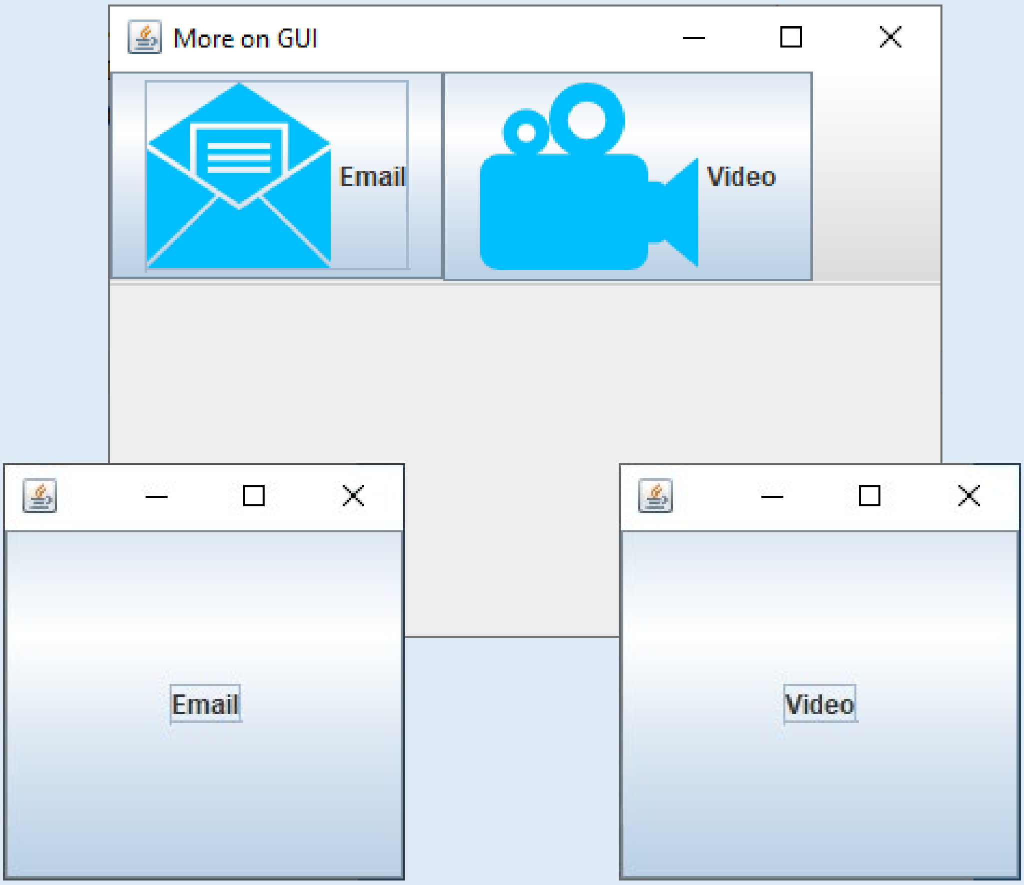
Write a menu (JMenu) based application. Then add menu item (JMenuItem) as the following diagram, the menu item File 1 clicked will add a button labelled MenuItem File 1. Same with menu item File 2, will added another button labelled MenuItem File 2.



1. Create a blank JFrame
2. Create 2 JMenu object for FILE and EDIT
3. Create a JMenuBar object and add to your container
4. Create another 2 JMenuItem for File 1 and File 2 to attach for each JMenu object
5. Add listener to all your menu item object using anonymous listener

ACTIVITY 2

Write a menu based application using button. Display other frame for each button action as in diagram.



1. Create a blank JFrame
2. Create 2 JButton for Email and Video
3. Add image icon for each button.
4. Add listener to all your button object using anonymous listener

ACTIVITY 3

The following codes will check keys entered by user. Analyse the codes and change the result so that it will display in the GUI.

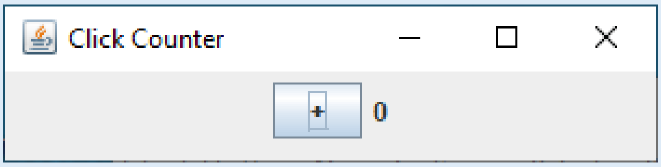
```
import java.awt.event.KeyAdapter;
import java.awt.event.KeyEvent;
import javax.swing.JFrame;
import javax.swing.JTextField;

public class CheckingKeys {
    public static void main(String[] argv) throws
Exception {
    JTextField text = new JTextField();
    text.addKeyListener(new Keychecker());
    JFrame frame = new JFrame();
    frame.add(text);
    frame.setSize(400, 350);
    frame.setVisible(true);
   }
}
class Keychecker extends KeyAdapter {
   @Override
   public void keyPressed(KeyEvent event) {
  char ch = event.getKeyChar();
  System.out.println(event.getKeyChar());
  }
}
```

1. Create another text field to display the result
2. Delete all codes to display the result in Window Output

ACTIVITY 4

The following codes will count the increment of a button click. Complete the codes to execute. Then add more function to decrement the value.

```
public CountButtonClicks(){
    setTitle("Click Counter");
    setSize(new Dimension(250, 80));
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setVisible(true);
    JPanel panel = new JPanel();

    plus = new JButton("+");
    plus.addActionListener(this);
    panel.add(plus);

    label = new JLabel("" + i);
    panel.add(label);
    add(panel, BorderLayout.CENTER);
  }
    @Override
    public void actionPerformed(ActionEvent e) {
       if (e.getSource() == plus) {
        i++;
        label.setText("" + i);
        }
    }
```



1. Add all related class using import statement
2. Define all components in the class
3. Finally create standard main method to execute the class

Bibliography

Anshuman Singh, & Saxena, A. (2022). JDBC Interview Questions. Retrieved May 12, 2022, from https://www.interviewbit.com/jdbc-interview-questions/#fastest-jdbc-driver

Cadenhead, R. (2012). *Sams Teach Yourself Java™ in 24 Hours, 6th ed*. 800 East 96th Street, Indianapolis, Indiana, 46240 USA: Sams Publishing.

Etheridge, D. (2009). *Java: Graphical User Interfaces, An Introduction to Java Programming*: Ventus Publishing ApS.

Goyal, S. (2019). The Java Programming Environment. Retrieved Feb 20, 2022, from https://medium.com/javarevisited/the-java-programming-environment-1bdc7833870e

Horstmann, C. (2007). *Java Concepts, 5th Edition*: John Wiley & Sons, Inc.

Pramodbablad. (2015). Difference Between Statement Vs PreparedStatement Vs CallableStatement In Java. Retrieved 23 June, 2022, from https://javaconceptoftheday.com/statement-vs-preparedstatement-vs-callablestatement-in-java/

Ralph Morelli, & Wade, R. (2021). Java GUIs- From AWT to Swing. Retrieved May 20, 2022, from https://eng.libretexts.org/Bookshelves/Computer_Science/Programming_Languages/Java_Java_Java_-_Object-Oriented_Programming_(Morelli_and_Walde)/13%3A_Graphical_User_Interfaces/13.01%3A_Java_GUIs-_From_AWT_to_Swing#fig-swing2-guis

Rogers Cadenhead, & Lemay, L. (2007). *Sams Teach Yourself Java™ 6 in 21 Days, 5th ed*. 800 East 96th Street, Indianapolis, Indiana 46240, USA: Sams Publishing.

Spell, B. (2015). *Pro Java 8 Programming, 3rd ed*: Apress Berkeley, CA.

TechVidvan. (2022). Data Types in Java Programming with Implementation Examples. Retrieved March 12, 2022, from https://techvidvan.com/tutorials/data-types-in-java/

POLITEKNIK
MALAYSIA
KUALA TERENGGANU

e ISBN 978-967-2240-36-5

9 7 8 9 6 7 2 2 4 0 3 6 5