**Computer Science Department**

**Laboratory Manual**

**CS220: Object Oriented Programming 2**

**2nd Semester 2017-2018**

**Dr. Imad Eldin Ahmed**

**Dr Hafiz Farooq Ahmad**

**Dr Nauman Qureshi**

**Dr.Noor Hafsa**

**Ms Muneera Alhajri**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| List of Experiments  [Lab 1: OOP1 Review – Structured Programming 3](#_Toc419319000)  [Lab 2: OOP1 Review - Arrays 7](#_Toc419319004)  [Lab 3: JOptionPane and JLabel 10](#_Toc419319008)  [Lab 4: JOptionPane and JLabel 13](#_Toc419319012)  [Lab 5: JTextField, JPasswordField, and ActionListner 21](#_Toc419319015)  [Lab 6: JTextArea, JScrollPane, and JButton 28](#_Toc419319018)  [Lab 7: JCheckBox, ItemListener 37](#_Toc419319021)  [Lab 8: JRadioButton, ItemListener, JPanel and BoxLayout 44](#_Toc419319024)  [Lab 9: JComboBox, JList, ItemListener, ListSelectionListener 52](#_Toc419319027)  [Lab 10: JComboBox, JList, ItemListener, ListSelectionListener 59](#_Toc419319030)  [Lab 11: JApplets 67](#_Toc419319033)  [Lab 12: Multi-Frame Application, Color and Font 74](#_Toc419319036)  Lab 13 : Database Access Using JDBC 81 Computer Science Department  **CS220: OOP2** Lab 1: OOP1 Review – Structured Programming **Second Semester: (2017/2018)**  **Lab 1: OOP1 Review – Structured Programming**  **Introduction**  In this lab you are going review some OOP1 structured programming concepts through solving several exercises.  **Objectives**   1. Practicing declaring variables 2. Practicing control statements 3. Practicing repetition structures 4. Practicing arrays 5. Practicing user defined functions   **Tools/Software Requirement**   1. Netbeans  **Exercise Description:** Q1) Write a program that ask the user to insert a width and length of a rectangle as integer values. Then the program will compute and print the area of the corresponding rectangle:  Sample run:  Insert width:  3  Insert length:  5  The rectangle area= 15  Q2) Write a program prompts the user to insert an integer. If the inserted number was odd the program prints “it is odd.”, else it prints “it is even.”  Sample run:  Insert integer: 4  It is Even  Q3) Write a program prompts the user to insert 5 integers. If the inserted number was odd the program prints “it is odd.”, else it prints “it is even.”. *Note: you have to use for loop.*  Sample run:  Insert integer: 4  It is even  Insert integer: 5  It is odd  Insert integer: 0  It is even  Insert integer: -1  It is odd  Insert integer: 13  It is odd  Q4) Write a program prompts the user to insert integers. While the inserted number doesn’t equal zero, the program will print “it is odd.” If the inserted integer is odd, else it prints “it is even.”. *Note: you have to use while loop.*  Sample run:  Insert integer: 4  It is even  Insert integer: 5  It is odd  Insert integer: 0  It is even  Q5) Write a program prompts the user to insert 10 integers. After the user finishes inserting the 10 integers, the program will prints the 10 inserted numbers and their summation.  Sample run:  Insert integer: 1  Insert integer: 2  Insert integer: 3  Insert integer: 4  Insert integer: 5  Insert integer: 6  Insert integer: 7  Insert integer: 8  Insert integer: 9  Insert integer: 10  The inserted numbers:  1, 2, 3, 4, 5, 6, 7, 8, 9, 10,  The summation= 55  Q6) write two user defined functions called area which computes and prints square and rectangle areas. *Note: you have to overload the function area.*  Then write a program that calls them.  Sample run:  Calling Area for a rectangle:  Insert width: 3  Insert length: 5  The rectangle area= 5  Calling Area for a square:  Insert side: 3  The square area= 9 Computer Science Department  **CS220: OOP2** Lab 2: OOP1 Review - Arrays **Second Semester: (2017/2018)**  **Lab 2: OOP1 Review - Arrays**  **Introduction**  In this lab you are going to review how to work with arrays.  **Objectives**   1. Practicing declaring arrays 2. Practicing initializing arrays 3. Practicing searching in arrays 4. Practicing 2D arrays 5. Practicing Recursion   **Tools/Software Requirement**   1. Netbeans  **Exercise Description:** Q1) Write a program in which you:  1- Declare an array of letters of size 5.  2- Initialize the array elements to space, using for loop.  3- Then Insert the letters {a,b,c,d,e}, in order, into the array, using a *for* loop.  4- Output the array content, using a *for* loop.  5- Search for letter ‘c’, when c is found your program should print ‘The littler ‘c’ has  been found at index: 2’  Sample run:  Insert a letter: a  Insert a letter: b  Insert a letter: c  Insert a letter: d  Insert a letter: e  You have inserted:  a, b, c, d, e,  The littler ‘c’ has been found at index: 2  Q2) Write a program that reads the first name, last name, and ID of 3 students, and then prints them back. *Hint: you have to use multidimensional array of type string or use OOP with Array list and method toString().*  Sample run:  Student 1:  Insert first name: Ahmed  Insert last name: Otaibi  Insert ID: 231098  Student 2:  Insert first name: Ali  Insert last name: Hasan  Insert ID: 654789  Student 3:  Insert first name: Khalid  Insert last name: Rasasi  Insert ID: 201534  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  You have inserted:  231098, Ahmed Otaibi  654789, Ali Hasan  201534, Khalid Rasasi    Q3) Write a program to calculate the factorials using: 1) for loop. 2) Recursion.    For example, 5! is clearly equal to 5 · 4!, as shown by the following equations:  5! = 5 · 4 · 3 · 2 · 1  *Hint: a recursive declaration of the factorial method is arrived at by observing the following*  *relationship:*  n! = n · (n – 1)! So, 5! = 5\*4!  Homework: (0.5 point)  Q4) Write a program to calculate the summation of a number recursively (use an array)  *Hint: the summation of 10:*  *Sum (10 )= 10 + Sum(9)*  *Sum (10 )= 10 + 9+ Sum(8)*  *…*  *Sum (10 )= 10 + 9+ 8+…+1* Computer Science Department  **CS220: OOP2** Lab 3: Object Oriented Programming **Second Semester: (2017/2018)**  **Lab 3: OOP1 Review – Object Oriented Programming**  **Introduction**  In this lab you are going to review OOP concepts through solving exercises.  **Objectives**   1. Declaring classes. 2. Practice inheritance. 3. Understand constructor/destructor invoking order. 4. Practice overriding/redefining base class members.   **Tools/Software Requirement**   1. Netbeans  Exercise: **Q1) Write a program that implements the following class diagram.**   |  | | --- | | ***employeeType*** | | -**employeeId:**  int  -**employeeNm:**  String | | +**setId**(int):void  +**setNm**(String):void  +**getId()**: int  +**getNm()**: String  *+****print()****: void*  +**employeeType()**  +**employeeType(**int, String**)** |  |  | | --- | | **fullTimeEmployee** | | -**empSalary** : int | | + **setSalary**( int): void  + **getSalary():** int  + **print():** void  + **fullTimeEmployee ()**  + **fullTimeEmployee (** int **,** String**,** int**)** |   **1- Split you solution into three files:**   * Lab3\_Ex.java * EmployeeType * FulTimeEmployee   - ***employeeType member functions’ descriptions* :**  - **setId**: // Function to set the employeeId.  - **setNm**: // Function to set the employeeNm.  - **getId:** //Function to retrieve the employeeId.  - **getNm:** //Function to retrieve the employeeNM.  *-* ***print:*** // output employee's data.  - **employeeType():**//default constructor initialize employeeId to zero and employeeNm to the word “Empty”  - **employeeType(int, String):** //overloaded constructor with parameters Sets the emploeeID and employeeNm  **- *fullTimeEmployee member functions’ descriptions* :**  **- setSalary:** //Function to set the salary.  **- getSalary:** //Function to retrieve the salary.  **- print:** //Function to output the employeeId,employeeNm, and salary  **- fullTimeEmployee():** //default constructor. Sets the employeeId to zero, employeeNm to “Empty”, and salary to zero.  **- fullTimeEmployee(int, String, int):** // overloaded constructor with parameters Sets the emploeeID, employeeNm, and salary  In the main function:   1. Declare an object of type *employeeType* using the default constructor 2. Declare an object of type *employeeType* using the overloaded constructor and pass the needed values. 3. Declare an object of type *FullTimeEmployee* using the default constructor 4. Declare an object of type *FullTimeEmployee* using the overloaded constructor and pass the needed values. 5. Prints all object data.*Hint: by calling the prints functions*   Sample RUN:  Employee 1 Information:  Employee ID: 0  Employee Name: Empty  Employee 2 Information:  Employee ID: 1  Employee Name: Dr. Imad Eldin Ahmed  Employee 3 Information:  Employee ID: 0  Employee Name: Empty  Employee Salary: 0  Employee 4 Information:  Employee ID: 2  Employee Name: Ali Ali  Employee Salary: 2500 |

# Computer Science Department

**CS220: OOP2**

# Lab 4: JOptionPane and JLabel

**Second Semester: (2017/2018)**

**Lab 4: JOptionPane and JLabel**

**Introduction**

This lab presents developing simple GUI-based input/output applications using JOptionPane. In addition to knowing how to create a JFrame. Furthermore, how to add JLabel into a JFrame.

**Objectives**

1. Develop a simple GUI application using JOptionPane
2. Creating JFrame
3. Practice using JLabel

**Tools/Software Requirement**

1. Netbeans

**Description**

1. JOptionPane

*JOptionPane* class is from java *swing* library. It provides prepackaged dialog boxes for both input and output. These dialogs are displayed by invoking static JOptionPane methods. To develop a simple application just follow the below steps:

1. Import Javax.Swing.JOptionPane to use the JOptionPane class

import javax.swing.JOptionPane;

1. To display text use *showMessageDialog()* method

JOptionPane.showMessageDialog(component parentComponent", Object message, string title, int messageType, Icon Icon);

Example:

JOptionPane.showMessageDialog(null, "Welcome to KFU", "Greeting Message", JOptionPane. INFORMATION\_MESSAGE);

1. To prompt the user to insert a value use *showInputDialog()*method

JOptionPane.JOptionPane.showInputDialog (component parentComponent", Object message, string title, int messageType, Icon Icon);

Example:

String name=JOptionPane.showInputDialog(null, "Insert your name:", "Information", JOptionPane.QUESTION\_MESSAGE);

1. JFram

A frame is a window in which different components/control are added and arranged on it. To create a *JFrame* you have to do the following:

1. Import the following libraries

import javax.swing.JFrame; //provide the basic window/frame features

import java.awt.FlowLayout; //Specifies how components are arranged

1. Extend the existing class by JFram class

public class JFrameSamp1 extends JFrame

1. Declare an object of the existing class

JFrameSamp1 frm=new JFrameSamp1(); /\* This object will be used to set the features of the frame and adding the control \*/

1. Set the size of the frame

frm.setSize(int width, int height);

1. Specifies the default close operation for the window/frame

frm.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);//Terminates the window\frame

1. Specify the layout of the frame One such layout manager is *FlowLayout*, in which GUI components are placed on a container from left to right in the order in which the program attaches them to the container. When there is no more room to fit components left to right, components continue to display left to right on the next line. If the container is resized, a FlowLayout reflows (i.e., rearranges) the components to accommodate the new width of the container

frm.setLayout(new FlowLayout());//

1. Add controls to the frame one-by-one

frm.add(Component comp); //The component can be Jlabel,TextField,..etc object

1. Set the frame to be visible

frm.setVisible(true);

|  |  |
| --- | --- |
| Example: | Output |
| package tstFrm;  import java.awt.FlowLayout;  import javax.swing.JFrame;  public class TstFrm extends JFrame{  public static void main(String[] args) {  TstFrm frm=new TstFrm();  frm.setTitle("Frame Title");  frm.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  frm.setSize(300, 400);  frm.setLayout(new FlowLayout());  /\*Add controls here\*/  frm.setVisible(true);  }    } |  |

1. JLable

Alabel is used to identify the purpose of every component unless the GUI designer provides text instructions or information stating the purpose of each component. To create and display a JLabel on a frame do the following:

1. Import the following libraries:

import javax.swing.JLabel; // displays text and images

import javax.swing.SwingConstants; // common constants used with Swing

import javax.swing.Icon; // interface used to manipulate images

import javax.swing.ImageIcon; // loads images

1. Declare an object of type JLabel as a member of the existing class

JLabel lbl=new JLabel(String text, Icon icon, int HorizontalAlignment);

1. Add the Jlabel object to the already declared JFrame object

frm.add(frm.lbl);

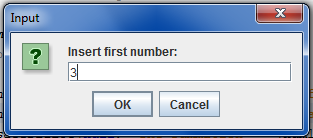
|  |  |
| --- | --- |
| Example1: | Output |
| package tstfrm;  import java.awt.FlowLayout;  import javax.swing.JFrame;  import javax.swing.JLabel;  public class TstFrm extends JFrame{  private JLabel nmaeLbl=new JLabel("Name: Ahmed Al Shemari");  public static void main(String[] args) {  TstFrm frm=new TstFrm();  frm.setSize(300, 400);  frm.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  frm.setLayout(new FlowLayout());  frm.add(frm.nmaeLbl);  frm.setVisible(true);  }    } |  |

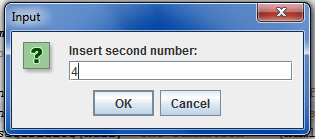
|  |  |
| --- | --- |
| Example2: | Output |
| package tstfrm;  import java.awt.FlowLayout;  import javax.swing.JFrame;  import javax.swing.JLabel;  import javax.swing.SwingConstants;  import javax.swing.Icon;  import javax.swing.ImageIcon;  public class TstFrm extends JFrame{  private JLabel nameLbl=new JLabel("Name: Ahmed Al Shemari");  Icon pic=new ImageIcon(getClass().getResource("pic.jpg"));  private JLabel pictureLbl=new JLabel(pic,SwingConstants.LEFT);  public static void main(String[] args) {  TstFrm frm=new TstFrm();  frm.setSize(300, 200);  frm.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  frm.setLayout(new FlowLayout());  frm.add(frm.pictureLbl);  frm.add(frm.nameLbl);  frm.setVisible(true);  }    } |  |

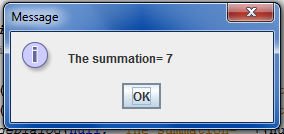
**Lab Tasks**

1. Write a program that prompts the user to insert two numbers. Then it prints their summation. You have to use JOptionPane dialogs for input and output.

Sample RUN:







1. Create a JFrame of size 500X200 and titled as “Welcome Window” that contains two JLabel objects. The first JLabel display KFU logo picture, while the other one display the text “Welcome to KFU”.

Sample Run:



**Deliverables**

When you are finished save zip the tasks projects into one file and \*\*??!

# Computer Science Department

**CS220: OOP2**

# Lab 5: JTextField, JPasswordField, and ActionListner

**Second Semester: (2017/2018)**

**Lab 5: JTextField, JPasswordField, and ActionListner**

**Introduction**

This lab presents developing simple GUI-based input/output applications using JtextFields, and JPasswordField. In addition to knowing how to implement ActionListeners.

**Objectives**

1. Develop a simple GUI application using JTextField
2. Practice using JPasswordPane
3. Practice using ActionListener

**Tools/Software Requirement**

1. Netbeans

**Description**

1. JTextField

*JTextField* class is from java *swing* library. It provides an area with predefined length to enable the user to insert text. This component is a single-line area in which the user can enter text via the keyboard. Applications can also display text in a JTextField. To implement a JTextField follow the below steps:

1. Import Javax.Swing.JTextField

import javax.swing.JTextField;

1. Declare an object of type JTextField. Note bellow that the JTextField ***txt*** is declared as static member of the class, thus, it will be accessed directly without the need for an object of type JFrame

|  |
| --- |
| static JTextField txt= new JTextField(); |

1. Specify the JTextField length in temp of number of character it can show at once

|  |
| --- |
| txt.setColumns(10); |

1. By default the JTextField is editable, but if you want to make it uneditable set the auditability into false

|  |
| --- |
| txt.setEditable(false); |

1. Add the JTextField control to the JFram object

|  |
| --- |
| frm.add(Txt); |

|  |  |
| --- | --- |
| Example: | Output |
| package jtxt;  import java.awt.FlowLayout;  import javax.swing.JFrame;  import javax.swing.JLabel;  import javax.swing.JTextField;  public class JTxt extends JFrame{  static JLabel edtLbl= new JLabel("Editable JTextField");  static JTextField edtTxt= new JTextField();  static JTextField nEdtTxt= new JTextField("You are NOT allowed to write here",20);  public static void main(String[] args) {  JTxt frm=new JTxt();  frm.setTitle("Frame Tittle");  frm.setLayout(new FlowLayout());  frm.setSize(250, 150);  frm.setDefaultCloseOperation(frm.EXIT\_ON\_CLOSE);    frm.add(edtLbl); // adding the nameLbl to the frame frm    /\*Set edtTxt proporties/features\*/  edtTxt.setColumns(10); // specify the width of the JTextField  frm.add(edtTxt); // addding the JTextField nameTxt to the frame frm    /\*Set nEdtTxt proporties/features\*/  nEdtTxt.setEditable(false);  frm.add(nEdtTxt); // addding the JTextField nEdtTxt to the frame frm    frm.setVisible(true);  }  } |  |

1. JPasswordField

Class JPasswordField extends JTextField and adds several methods that are specific to processing passwords. A JPasswordField shows that characters are being typed as the user enters them, but hides the actual characters with an **echo character**, assuming that they represent a password that should remain known only to the user. This component is a single-line area in which the user can enter text via the keyboard To create a JPasswordField you have to do the following:

1. Import Javax.Swing.JPasswordField

|  |
| --- |
| import javax.swing.JPasswordField; |

1. Declare an object of type JPasswordField. Note bellow that the JPasswordField ***passtxt*** is declared as static member of the class, thus, it will be accessed directly without the need for an object of type JFrame

|  |
| --- |
| static JPasswordField passTxt=new JPasswordField(); |

1. Specify the JPasswordField length in temp of number of character it can show at once

|  |
| --- |
| passTxt.setColumns(10); |

1. Add the JPasswordField control to the JFram object

|  |
| --- |
| frm.add(passTxt); |

|  |  |
| --- | --- |
| Example: | Output |
| package jtxt;  import java.awt.FlowLayout;  import javax.swing.JFrame;  import javax.swing.JLabel;  import javax.swing.JPasswordField;  public class JTxt extends JFrame{  static JLabel passLbl= new JLabel("JPasswordField");  static JPasswordField passTxt=new JPasswordField(10);// the width of the  //JPasswordField is specified in the constructor  public static void main(String[] args) {  JTxt frm=new JTxt();  frm.setTitle("Frame Tittle");  frm.setLayout(new FlowLayout());  frm.setSize(250, 150);  frm.setDefaultCloseOperation(frm.EXIT\_ON\_CLOSE);    frm.add(passLbl); // addding the JLabel passLbl to the frame frm  frm.add(passTxt); // addding the JPasswordField passTxt to the frame frm    frm.setVisible(true);  }  } |  |

1. ActionListener

Action listeners are probably the easiest — and most common — event handlers to implement. You implement an action listener to define what should be done when a user performs certain operation. An action event occurs, whenever an action is performed by the user. Examples: When the user clicks a [button](http://docs.oracle.com/javase/tutorial/uiswing/components/button.html), chooses a [menu item](http://docs.oracle.com/javase/tutorial/uiswing/components/menu.html), presses Enter in a [text field](http://docs.oracle.com/javase/tutorial/uiswing/components/textfield.html). The result is that an ActionPerformed message is sent to all action listeners that are registered on the relevant component. To write an Action Listener, follow the steps given below:

1. Import the following libraries

|  |
| --- |
| import java.awt.event.ActionListener;  import java.awt.event.ActionEvent; |

1. Implements an ActionListener interface:

|  |
| --- |
| public class MyClass implements ActionListener {…} |

1. Register an instance of the event handler class as a listener on one or more components. Below an ActionListener is added for a JTextField

|  |
| --- |
| txt.addActionListener(instanceOfMyClass); |

1. Include code that implements the methods in listener interface

|  |
| --- |
| public void actionPerformed(ActionEvent e) {  //code that reacts to the action  } |

|  |  |
| --- | --- |
| Example1: | Output |
| package actionlistenerapp;  import java.awt.event.ActionListener;  import java.awt.event.ActionEvent;  import java.awt.FlowLayout;  import javax.swing.JFrame;  import javax.swing.JLabel;  import javax.swing.JTextField;  public class ActionListenerApp extends JFrame implements ActionListener{  static JLabel fstNumLbl= new JLabel("Insert Number");  static JLabel sndNumLbl= new JLabel("Inser Number");  static JLabel resultLbl= new JLabel("Result: ");  static JTextField fstNumTxt= new JTextField(10);  static JTextField sndNumTxt= new JTextField(10);  static JTextField resultTxt= new JTextField(10);  public static void main(String[] args) {  ActionListenerApp frm=new ActionListenerApp();  frm.setTitle("Add Two Numbers");  frm.setLayout(new FlowLayout());  frm.setSize(250, 150);  frm.setDefaultCloseOperation(frm.EXIT\_ON\_CLOSE);    frm.add(fstNumLbl);  frm.add(fstNumTxt);  frm.add(sndNumLbl);  frm.add(sndNumTxt);  frm.add(resultLbl);  resultTxt.setEditable(false);  frm.add(resultTxt); // addding the JTextField nEdtTxt to the frame frm    frm.setVisible(true);  sndNumTxt.addActionListener(frm);  }  public void actionPerformed(ActionEvent e) {  int num1=Integer.parseInt(fstNumTxt.getText());  int num2=Integer.parseInt(fstNumTxt.getText());  int result=num1+num2;  resultTxt.setText(num1+"+"+num2+"= "+result+"\n");  }  } |  |

**Lab Tasks**

1. Create a LogIn JFrame that enables a student to insert his username and password. Once the, user presses the enter key in the JPasswordField, compare the inserted username and password with already stored username “aalhowaide” and password “P@ss0rd”. If they are matched show a JOptionMassageDialog saying “Welcome aalhowaide”, else it prints in an none-editable JTextField “Either Username or Password incorrect!”. The frame contains two JLabels, two JTextField, and JPasswordField objects, and set the frame size to 250x200. Set the non-editable JTextField size to 20, while the other JTextField and JPasswordField size to 10.

Sample RUN:

|  |  |
| --- | --- |
| **Incorrect username and password** | **Correct username and password** |
|  |  |

# Computer Science Department

**CS220: OOP2**

# Lab 6: JTextArea, JScrollPane, and JButton

**Second Semester: (2017/2018)**

**Lab 6: JTextArea, JScrollPane, and JButton**

**Introduction**

This lab presents developing simple GUI-based input/output applications using JTextArea with JScrollPane and JButton. In addition to how to add several ActionListeners and perform different events for each control.

**Objectives**

1. Develop a simple GUI application using JTextArea with JScrollPane
2. Practice using JButton
3. Practice using multi-ActionListeners

**Tools/Software Requirement**

1. Netbeans

**Description**

1. JTextArea

*JTextArea* class is from java *swing* library. It provides a multiline area with predefined length and width to enable the user to insert text. Applications can also display text in a JTextArea. To implement a JTextArea follow the below steps:

1. Import Javax.Swing.JTextArea

import javax.swing.JTextArea;

1. Declare an object of type JTextArea. Note bellow that the JTextArea ***txtArea*** is declared as static member of the class, thus, it will be accessed directly without the need for an object of type JFrame. Below the number of rows and columns is specified and passed to the JTextArea constructor

|  |
| --- |
| static JTextArea txtArea=new JTextArea(10,20); |

1. Specify the text wrapping style

|  |
| --- |
| txtArea.setLineWrap(true);  txtArea.setWrapStyleWord(true); |

1. Add the JTextField control to the JFram object

|  |
| --- |
| frm.add(txtArea); |

* Try the below example with specifying the wrapping style and without. Did you notice the difference?

|  |  |
| --- | --- |
| Example: | Output |
| package fstjtextarea;  import java.awt.FlowLayout;  import javax.swing.JFrame;  import javax.swing.JLabel;  import javax.swing.JTextArea;  public class FstJTextArea extends JFrame {  static JLabel TxtArLbl=new JLabel("Descripe yourself:");  static JTextArea txtArea=new JTextArea(10,20);//set the row and columns  public static void main(String[] args) {  FstJTextArea frm=new FstJTextArea();  frm.setTitle("JTextArea");  frm.setLayout(new FlowLayout());  frm.setSize(250, 300);  frm.setDefaultCloseOperation(EXIT\_ON\_CLOSE);  frm.setVisible(true);    frm.add(TxtArLbl);  /\*Set the propoerties of txtArea control\*/  txtArea.setLineWrap(true);  txtArea.setWrapStyleWord(true);  frm.add(txtArea);  }  } |  |

1. JScrollPane

Class JScrollPane is used to add scroll bars to an JTextArea control. To create a JScrollPane you have to do the following:

1. Import Javax.Swing.JScrollPane

|  |
| --- |
| import javax.swing.JScrollPane; |

1. Declare an object of type JScrollPane. And   in your JTextArea to your JScrollPane constructor. Also you can specify what scrolls you need and how they will be provided

|  |
| --- |
| static JScrollPane scroll=new JScrollPane(txtArea,  JScrollPane.VERTICAL\_SCROLLBAR\_AS\_NEEDED,  JScrollPane.HORIZONTAL\_SCROLLBAR\_AS\_NEEDED); |

1. Add the JScrollPane control to the JFram object. NOTE: while you already add the JTextArea to the JScrollPane there is no need to add the JTextArea one more time, if you did, the JScrollPane will be not displayed

|  |
| --- |
| frm.add(scroll); |

|  |  |
| --- | --- |
| Example: | Output |
| package fstjtextarea;  import java.awt.FlowLayout;  import javax.swing.JFrame;  import javax.swing.JLabel;  import javax.swing.JTextArea;  import javax.swing.JScrollPane;  public class FstJTextArea extends JFrame implements ActionListener{  static JLabel TxtArLbl=new JLabel("Descripe yourself:");  static JTextArea txtArea=new JTextArea(10,20);//set the row and columns  static JScrollPane scroll=new JScrollPane(txtArea,JScrollPane.VERTICAL\_SCROLLBAR\_AS\_NEEDED,JScrollPane.HORIZONTAL\_SCROLLBAR\_AS\_NEEDED);  public static void main(String[] args) {  FstJTextArea frm=new FstJTextArea();  frm.setTitle("JTextArea");  frm.setLayout(new FlowLayout());  frm.setSize(250, 300);  frm.setDefaultCloseOperation(EXIT\_ON\_CLOSE);  frm.setVisible(true);    frm.add(TxtArLbl);  /\*Set the propoerties of txtArea control\*/  txtArea.setLineWrap(true);  txtArea.setWrapStyleWord(true);    frm.add(scroll);  }  } |  |

1. JButton

A button is a component the user clicks to trigger a specific action. A Java application can use several types of buttons, including command buttons, checkboxes, toggle buttons and radio buttons. A command button generates an ActionEvent when the user clicks the button. Command buttons are created with class JButton. The text on the face of a JButton is called a button label. A GUI can have many JButtons, but each button label typically should be unique in the portion of the GUI that is currently displayed. Below are the steps how to create a JButton with action listener:

1. Import the following libraries

|  |
| --- |
| import javax.swing.JButton; |

1. Declare an object of type JButton:

|  |
| --- |
| static JButton resetBtn=new JButton(); |

1. Specify the JButton text:

resetBtn.setText("Reset");

1. Add the JButton control to the JFram object

|  |
| --- |
| frm.add(resetBtn); |

* In the example below an event handler is added to two controls. Note how it is distinguished between them.

|  |  |
| --- | --- |
| Example1: | Output |
| package fstjtextarea;  import java.awt.FlowLayout;  import java.awt.event.ActionEvent;  import java.awt.event.ActionListener;  import javax.swing.JFrame;  import javax.swing.JLabel;  import javax.swing.JTextField;  import javax.swing.JTextArea;  import javax.swing.JScrollPane;  import javax.swing.JButton;  public class FstJTextArea extends JFrame implements ActionListener{  static JLabel TxtArLbl=new JLabel("Descripe yourself:");  static JTextField validationTxt=new JTextField(20);  static JTextArea txtArea=new JTextArea(10,20);//set the row and columns  static JScrollPane scroll=new JScrollPane(txtArea,  JScrollPane.VERTICAL\_SCROLLBAR\_AS\_NEEDED,  JScrollPane.HORIZONTAL\_SCROLLBAR\_AS\_NEEDED);  static JButton resetBtn=new JButton();  static JButton dnBtn=new JButton();// will validate your description  public static void main(String[] args) {  /\*Setting up the frame properties\*/  FstJTextArea frm=new FstJTextArea();  frm.setTitle("JTextArea");  frm.setLayout(new FlowLayout());  frm.setSize(250, 300);  frm.setDefaultCloseOperation(EXIT\_ON\_CLOSE);  frm.setVisible(true);    frm.add(TxtArLbl);  /\*Set the propoerties of txtArea control\*/  txtArea.setLineWrap(true);  txtArea.setWrapStyleWord(true);  frm.add(scroll);    /\*Setting the JButtons text\*/  resetBtn.setText("Reset");  frm.add(resetBtn);  dnBtn.setText("Done");  frm.add(dnBtn);  /\*Setting the validationTxt properties\*/  validationTxt.setEditable(false);  frm.add(validationTxt);    resetBtn.addActionListener(frm);  dnBtn.addActionListener(frm);  }  public void actionPerformed(ActionEvent e) {  if(e.getSource()==dnBtn)  {  validationTxt.setText("");  if(txtArea.getText().isEmpty())  validationTxt.setText("\* You did Not Discripe yourself!");  }  else if(e.getSource()==resetBtn)  {  txtArea.setText("");  }  }  } | * At startup:      * After pressing Done button without writing anything:      * After writing and pressing on Done buttoin:      * After pressing the reset button: |

**Lab Tasks**

1. Create a simple create account JFrame that enables a student to insert his first name and last name. Once the, user click on Validate button, it will check what ever either the first name of the last name text field are empty or not. In case one of them is empty or both the button with add a text to the validate label satisfying the empty text field, else it prints “welldone” in addition to the student name. While the reset button will clear all the text in all the text fields. The frame of size 250X200 contains two JLabels, two JTextFields of size 10, one non-editable JTextArea of size 2X10, and two JButtons with ActionListeners.

Sample RUN:

|  |  |
| --- | --- |
| **At Start** | **Validate when Both text fields are empty** |
|  |  |
| **Validate when first name text field is empty** | **Validate when last name text field is empty** |
|  |  |
| **Pressing the reset button** |  |
|  |  |

# Computer Science Department

**CS220: OOP2**

# Lab 7: JCheckBox, ItemListener

**Second Semester: (2017/2018)**

**Lab 7: JCheckBox, ItemListener**

**Introduction**

This lab presents developing simple GUI-based application using JCheckBox with ItemListener. Furthermore, ActionListener will be used with ItemListener in more complex examples.

**Objectives**

1. Develop a simple GUI application
2. Practice using JCheckBox
3. Practice using ItemListener

**Tools/Software Requirement**

1. Netbeans

**Description**

1. JCheckBox

The Swing GUI components contain three types of state buttons—JToggleButton, JCheckBox and JRadioButton—that have on/off or true/false values. Classes JCheckBox and JRadioButton are subclasses of JToggleButton. We first discuss class JCheckBox.

The user is able to select more than one JCeckBox, when a JCheckBox is selected its value is true, else it is false. To implement a JCheckBox follow the below steps:

1. Import Javax.Swing. JCheckBox

import javax.swing.JCheckBox;

1. Declare an object of type JCheckBox. Note bellow that the JCheckBox ***csCbx*** is declared as static member of the class, thus, it will be accessed directly without the need for an object of type JFrame. In addition, the text of the JCeckBox is also set to “CS220” using the constructor.

|  |
| --- |
| static JCheckBox cs220Cbx=new JCheckBox("CS220"); |

1. Add the JCheckBox control to the JFram object

|  |
| --- |
| frm.add(cs220Cbx); |

|  |  |
| --- | --- |
| Example: | Output |
| package jcheckbox\_eg;  import java.awt.FlowLayout;  import java.awt.event.ActionEvent;  import java.awt.event.ActionListener;  import javax.swing.JOptionPane;  import javax.swing.JFrame;  import javax.swing.JLabel;  import javax.swing.JButton;  import javax.swing.JCheckBox;  public class JCheckBox\_Eg extends JFrame implements ActionListener{  static JLabel crslbl=new JLabel("Select your courses for this semester:");  static JButton dnBtn=new JButton("Done");  static JCheckBox cs220Cbx=new JCheckBox("CS220");  static JCheckBox IS222Cbx=new JCheckBox("IS222");  static JCheckBox IS313Cbx=new JCheckBox("IS313");    public static void main(String[] args) {  JCheckBox\_Eg frm=new JCheckBox\_Eg();  frm.setTitle("CheckBox Example");  frm.setLayout(new FlowLayout());  frm.setDefaultCloseOperation(EXIT\_ON\_CLOSE);  frm.setSize(250, 250);  frm.setVisible(true);  frm.add(crslbl);  frm.add(cs220Cbx);  frm.add(IS222Cbx);  frm.add(IS313Cbx);  frm.add(dnBtn);  dnBtn.addActionListener(frm);  }  @Override  public void actionPerformed(ActionEvent e) {  if(cs220Cbx.isSelected()||IS222Cbx.isSelected()||IS313Cbx.isSelected())  {  JOptionPane.showMessageDialog(rootPane, "You have selected: "+  (cs220Cbx.isSelected()?cs220Cbx.getText()+", ":" ")+  (IS222Cbx.isSelected()?IS222Cbx.getText()+", ":" ")+  (IS313Cbx.isSelected()?IS313Cbx.getText():""));  }  else  {JOptionPane.showMessageDialog(rootPane, "You did NOT select a course, \n Please, Select a course!","Error!",JOptionPane.ERROR\_MESSAGE);  }  } |  |

1. ItemListener

When the user clicks a JCheckBox, an ItemEvent occurs. This event can be handled by an ItemListener object, which must implement method itemStateChanged. To write an Action Listener, follow the steps given below:

1. Import the following libraries

|  |
| --- |
| import java.awt.event.ItemEvent;  import java.awt.event.ItemListener; |

1. Implements an ActionListener interface:

|  |
| --- |
| public class JCheckBox\_Eg extends JFrame implements ItemListener {…} |

1. Register an instance of the event handler class as a listener on one or more components. Below an ActionListener is added for a JCheckBox

|  |
| --- |
| cs220Cbx.addItemListener(frm); |

1. Include code that implements the methods in listener interface

|  |
| --- |
| public void itemStateChanged(ItemEvent e){  //code that reacts to the action  } |

* Note in the example below when the user uncheck a JCheckBox, its corresponding text is deleted from the JTextArea.

|  |  |
| --- | --- |
| Example: | Output |
| package jcheckbox\_eg;  import java.awt.FlowLayout;  import java.awt.event.ItemEvent;  import java.awt.event.ItemListener;  import javax.swing.JFrame;  import javax.swing.JLabel;  import javax.swing.JTextArea;  import javax.swing.JCheckBox;  public class JCheckBox\_Eg extends JFrame implements ItemListener{  static JLabel crslbl=new JLabel("Select your courses for this semester:");  static JTextArea crsTA=new JTextArea(5,20);  static JCheckBox cs220Cbx=new JCheckBox("CS220");  static JCheckBox IS222Cbx=new JCheckBox("IS222");  static JCheckBox IS313Cbx=new JCheckBox("IS313");    public static void main(String[] args) {  JCheckBox\_Eg frm=new JCheckBox\_Eg();  frm.setTitle("CheckBox Example");  frm.setLayout(new FlowLayout());  frm.setDefaultCloseOperation(EXIT\_ON\_CLOSE);  frm.setSize(250, 200);  frm.setVisible(true);  crsTA.setEditable(false);  frm.add(crslbl);  frm.add(cs220Cbx);  frm.add(IS222Cbx);  frm.add(IS313Cbx);  frm.add(crsTA);  cs220Cbx.addItemListener(frm);  IS222Cbx.addItemListener(frm);  IS313Cbx.addItemListener(frm);  }  @Override  public void itemStateChanged(ItemEvent e) {  if(e.getSource()==cs220Cbx)  {  if(cs220Cbx.isSelected())  crsTA.setText(crsTA.getText()+cs220Cbx.getText()+"\n");  if(!(cs220Cbx.isSelected()))  crsTA.setText(crsTA.getText().replaceAll(cs220Cbx.getText()+"\n", ""));  }  if(e.getSource()==IS222Cbx)  {  if(IS222Cbx.isSelected())  crsTA.setText(crsTA.getText()+IS222Cbx.getText()+"\n");  if(!(IS222Cbx.isSelected()))  crsTA.setText(crsTA.getText().replaceAll(IS222Cbx.getText()+"\n", ""));  }  if(e.getSource()==IS313Cbx)  {  if(IS313Cbx.isSelected())  crsTA.setText(crsTA.getText()+IS313Cbx.getText()+"\n");  if(!(IS313Cbx.isSelected()))  crsTA.setText(crsTA.getText().replaceAll(IS313Cbx.getText()+"\n", ""));  }  }  } |  |

**Lab Tasks**

1. Create a simple course registration application that enables a student to select courses for this semester. Once the, user checks a JCheckBox, its text will be added to the JTextArea. When the user unchecks a jCheckBox, its corresponding text will be deleted from the JTextArea. The JButton will check whatever the students selected some courses. In case the students didn’t select any course and pressed on the *Done* button, it will show a massage asking the user to select some courses, else it will show a massage saying “The Registration Was Successfully DONE”. The frame of size 250X200 contains one JLabels, three JCheckBoxes, one non-editable JTextArea of size 5X20, and a JButton with ActionListeners.

Sample RUN:

|  |  |
| --- | --- |
| **At start** | **Some courses are checked/selected** |
|  |  |
| **Pressing *Done* button after selecting some courses** | **Press *Done* button without selecting any course** |
|  |  |

# Computer Science Department

**CS220: OOP2**

# Lab 8: JRadioButton, ItemListener, JPanel and BoxLayout

**Second Semester: (2017/2018)**

**Lab 8: JRadioButton, ItemListener, JPanel and BoxLayout**

**Introduction**

This lab presents developing GUI application using JRadioButton with ItemListener. Furthermore, JPanel with BoxLayout will be introduced.

**Objectives**

1. Develop a simple GUI application
2. Practice using JRadioButton
3. Practice using ItemListener
4. Practice using JPanel and BoxLayout

**Tools/Software Requirement**

1. Netbeans

**Description**

1. JRadioButton

Radio buttons (declared with class JRadioButton) are similar to checkboxes in that they have two states—selected and not selected (also called deselected). However, radio buttons normally appear as a group in which only one button can be selected at a time. Selecting a different radio button forces all others to be deselected. Radio buttons are used to represent mutually exclusive options (i.e., multiple options in the group cannot be selected at the same time). The logical relationship between radio buttons is maintained by a ButtonGroup object (package javax.swing), which itself is not a GUI component. A ButtonGroup object organizes a group of buttons and is not itself displayed in a user interface. Rather, the individual JRadioButton objects from the group are displayed in the GUI. To implement a JCheckBox follow the below steps:

1. Import the following libraries

import javax.swing.JRadioButton;

import javax.swing.ButtonGroup;

1. Declare an object of type JRadioButton. Note bellow that the JRadioButton ***mRbtn*** is declared as static member of the class, thus, it will be accessed directly without the need for an object of type JFrame. In addition, the text of the JRadioButton is also set to “Male” using the constructor.

|  |
| --- |
| static JRadioButton mRbtn= new JRadioButton( "Male"); |

1. Declare an object of type ButtonGroup to add the declared JRadioButtons to it, thus only one JRadioButton will be selected

|  |
| --- |
| static ButtonGroup gndGroup= new ButtonGroup(); // buttongroup to hold radio buttons |

1. Add the JRadioButton to the ButtonGroup

|  |
| --- |
| gndGroup.add(mRbtn); |

1. Add the JRadioButton to the JFrame

|  |
| --- |
| frm.add(mRbtn); |

|  |  |
| --- | --- |
| Example: | Output |
| package jradiobutton\_eg;  import java.awt.FlowLayout;  import javax.swing.JFrame;  import javax.swing.JLabel;  import javax.swing.JRadioButton;  import javax.swing.ButtonGroup;  public class JRadioButton\_Eg extends JFrame{  static JLabel gndLbl=new JLabel("Select your gender:");  static ButtonGroup gndGroup= new ButtonGroup();  // buttongroup to hold radio buttons  static JRadioButton mRbtn= new JRadioButton( "Male");  static JRadioButton fRbtn= new JRadioButton( "Female");  public static void main(String[] args) {  JRadioButton\_Eg frm=new JRadioButton\_Eg();  frm.setLayout(new FlowLayout());  frm.setTitle("JRadioButton Example");  frm.setSize(300, 100);  frm.setDefaultCloseOperation(EXIT\_ON\_CLOSE);  frm.setVisible(true);  frm.add(gndLbl);  gndGroup.add(mRbtn);  gndGroup.add(fRbtn);  frm.add(mRbtn);  frm.add(fRbtn);  }  } |  |

1. ItemListener

When the user clicks/selects a JRadioButton, an ItemEvent occurs. This event can be handled by an ItemListener object, which must implement method itemStateChanged. To write an Action Listener, follow the steps given below:

1. Import the following libraries

|  |
| --- |
| import java.awt.event.ItemEvent;  import java.awt.event.ItemListener; |

1. Implements an ActionListener interface:

|  |
| --- |
| public class JRadioButton\_Eg extends JFrame implements ItemListener{…} |

1. Register an instance of the event handler class as a listener on one or more components. Below an ActionListener is added for a JRadioButton

|  |
| --- |
| cs220Cbx.addItemListener(frm); |

1. Include code that implements the methods in listener interface

|  |
| --- |
| public void itemStateChanged(ItemEvent e){  //code that reacts to the action  } |

* **Note:** in the example below JPanel was used, in addition to BoxLayout. JPanel is treated just like a JFrame. Add the components into the JPanel instead of adding them to the JFrame. Then add the JPanel into the JFrame. The JPanel is used here to enable us to add control in vertical layout. FlowLayout doesn’t enable us to add controls/components in vertical layout, thus another layout used called BoxLayout. For more information about BoxLayout and other layouts visit:
* https://docs.oracle.com/javase/tutorial/uiswing/layout/visual.html
* Below when the user selects a JRadioButton from the ***eduGroup*** an event will be performed, which is displaying the user selected information in a JOptionPane dialog box.

|  |  |
| --- | --- |
| Example: | Output |
| package jradiobutton\_eg;  import javax.swing.BoxLayout;  import java.awt.event.ItemEvent;  import java.awt.event.ItemListener;  import javax.swing.JFrame;  import javax.swing.JLabel;  import javax.swing.JRadioButton;  import javax.swing.ButtonGroup;  import javax.swing.JPanel;  import javax.swing.JOptionPane;  public class JRadioButton\_Eg extends JFrame implements ItemListener{  static JPanel p1=new JPanel();  static JLabel gndLbl=new JLabel("Select your gender:");  static JLabel eduLbl=new JLabel("Select your Education Level:");  static ButtonGroup gndGroup= new ButtonGroup(); // buttongroup to hold radio buttons  static ButtonGroup eduGroup= new ButtonGroup(); // buttongroup to hold radio buttons  static JRadioButton mRbtn= new JRadioButton( "Male"); // selects plain text  static JRadioButton fRbtn= new JRadioButton( "Female"); // selects bold text  static JRadioButton schlRbtn= new JRadioButton( "High Schoole"); // selects plain text  static JRadioButton underRbtn= new JRadioButton( "Under-Graduate"); // selects bold text  static JRadioButton gradRbtn= new JRadioButton( "Graduate"); // selects plain text    public static void main(String[] args) {  JRadioButton\_Eg frm=new JRadioButton\_Eg();  frm.setTitle("JRadioButton Example");  frm.setSize(300, 200);  frm.setDefaultCloseOperation(EXIT\_ON\_CLOSE);  frm.setVisible(true);  p1.setLayout(new BoxLayout(p1, BoxLayout.Y\_AXIS));  p1.add(gndLbl);  gndGroup.add(mRbtn);  gndGroup.add(fRbtn);  p1.add(mRbtn);  p1.add(fRbtn);  p1.add(eduLbl);  eduGroup.add(schlRbtn);  eduGroup.add(underRbtn);  eduGroup.add(gradRbtn);  p1.add(schlRbtn);  p1.add(underRbtn);  p1.add(gradRbtn);  frm.add(p1);    schlRbtn.addItemListener(frm);  underRbtn.addItemListener(frm);  gradRbtn.addItemListener(frm);  }  @Override  public void itemStateChanged(ItemEvent e) {  String gender=new String();  if(mRbtn.isSelected())  gender="Male";  if(fRbtn.isSelected())  gender="Female";  if(schlRbtn.isSelected())  {  JOptionPane.showMessageDialog(rootPane,"You are a "+gender+"\nYou are a " +schlRbtn.getText()+"Student","Students Information",JOptionPane.INFORMATION\_MESSAGE);  }  if(underRbtn.isSelected())  {  JOptionPane.showMessageDialog(rootPane,"You are a "+gender+"\nYou are a " +underRbtn.getText()+"Student","Students Information",JOptionPane.INFORMATION\_MESSAGE);  }  if(gradRbtn.isSelected())  {  JOptionPane.showMessageDialog(rootPane,"You are a "+gender+"\nYou are a " +gradRbtn.getText()+"Student","Students Information",JOptionPane.INFORMATION\_MESSAGE);  }  }  } |  |

**Lab Tasks**

1. Create a student registration form that enables a student to specify his/her gender, department and status. Once the, user select a JRadioButton, its text will be added to the JTextArea. When the user selects another JRadioButton, its corresponding text will be replaced with the new select JRadioButton text in the JTextArea. The *Done* JButton will check whatever the students specified all the required information. In case the student didn’t specify some of the required information and pressed on the *Done* button, it will show a massage asking the user to specify the remaining information as shown in the sample run, else it will show a massage saying “Registration Succeed”. The frame of size 300X400. The JTextArea of size 4X20

*Hint: You have to use JPanel and the BoxLayot. You may need 3 Panels, one for JradioButtons, one for JTextArea and the last one for JButtons.*

Sample RUN:

|  |  |
| --- | --- |
| **At start** | **Pressing Done and nothing is selected** |
|  |  |
| **Pressing *Done* button after selecting gender** | **Press *Done* button without selecting some radio buttons** |
|  |  |
| **Pressing *Done* button after selecting all the radio buttons** | |
|  | |

# Computer Science Department

**CS220: OOP2**

# Lab 9: JComboBox, JList, ItemListener, ListSelectionListener

**Second Semester: (2017/2018)**

**Time: 13:00 / 14:15**

**Lab 9: JComboBox, JList, ItemListener, ListSelectionListener**

**Introduction**

This lab presents developing GUI application using JComboBox with ItemListener. In addition to JList with ListSelectionListener.

**Objectives**

1. Develop a simple GUI application
2. Practice using JComboBox with ItemListener
3. Practice using JList with ListSelectionListener

**Tools/Software Requirement**

1. Netbeans

**Description**

1. JComboBox with ItemListener

A combo box (sometimes called a drop-down list) provides a list of items from which the user can make a single selection. Comboboxes are implemented with class JComboBox, which extends class JComponent. JComboBoxes generate ItemEvents like JCheckBoxes and JRadioButtons. To implement a JComboBox follow the below steps:

1. Import the following libraries

import javax.swing.JComboBox;

1. Declar an array of string to store the JComboBox items text

static String[] departments={"","Computer Science","Information System","Computer Networks"};

1. Declare an object of type JComboBox. Note bellow that the JComboBox ***depCB*** is declared as static member of the class, thus, it will be accessed directly without the need for an object of type JFrame. In addition, items of the JComboBox are assigned using the constructor.

|  |
| --- |
| static JComboBox depCb=new JComboBox(departments); |

1. Add the JComboBoc to the JFrame

|  |
| --- |
| frm.add(depCB); |

* ToDo: in the example below change the number of displayed items in the JComboBox by using the following code and note the difference.

depCb.setMaximumRowCount(2);

|  |  |
| --- | --- |
| Example: | Output |
| package jcombobox\_eg;  import java.awt.FlowLayout;  import java.awt.event.ItemEvent;  import java.awt.event.ItemListener;  import javax.swing.JFrame;  import javax.swing.JLabel;  import javax.swing.JComboBox;  import javax.swing.JOptionPane;  public class JComboBox\_Eg extends JFrame implements ItemListener{  static JLabel depLbl=new JLabel("Select your department:");  static String[] departments={"","Computer Science","Information System","Computer Networks"};  static JComboBox depCb=new JComboBox(departments);  public static void main(String[] args) {  JComboBox\_Eg frm=new JComboBox\_Eg();  frm.setLayout(new FlowLayout());  frm.setTitle("JComboBox Example");  frm.setSize(300, 200);  frm.setDefaultCloseOperation(EXIT\_ON\_CLOSE);  frm.setVisible(true);    frm.add(depLbl);  frm.add(depCb);    depCb.addItemListener(frm);  }  @Override  public void itemStateChanged(ItemEvent e) {  if(e.getStateChange()==1)//execute when it is selection action  if(depCb.getSelectedIndex()==0)  JOptionPane.showMessageDialog(rootPane, "You have to select a department", "Department", JOptionPane.ERROR\_MESSAGE);  else  for(int i=1;i<4;i++)  if(depCb.getSelectedItem()==departments[i])  {  JOptionPane.showMessageDialog(rootPane, "You have selected: "+depCb.getSelectedItem(), "Department", JOptionPane.INFORMATION\_MESSAGE);  } }    } |  |

1. JList with ListSelectionListener

A list displays a series of items from which the user may select one or more items. Lists are created with class JList, which directly extends class JComponent. Class JList supports single-selection lists (which allow only one item to be selected at a time) and multiple-selection lists (which allow any number of items to be selected). To write an Action Listener, follow the steps given below:

1. Import the following libraries

|  |
| --- |
| import javax.swing.JList; |

1. Declar an array of string to store the JComboBox items text

|  |
| --- |
| static String[] hobbies={"Swimming","Reading","Writing", "Other"}; |

1. Declare an object of type JList. Note bellow that the JList ***hbLst*** is declared as static member of the class, thus, it will be accessed directly without the need for an object of type JFrame. In addition, items of the JList are assigned using the constructor.

|  |
| --- |
| static JList hbLst=new JList(hobbies); |

1. Add the JList to the JFrame

|  |
| --- |
| frm.add(hbLst); |

* **Note:** Multi-selection can be enabled by the following steps:

1. Import:

import javax.swing.ListSelectionModel;

1. Set the selection model to multiple

hbLst.setSelectionMode(ListSelectionModel.MULTIPLE\_INTERVAL\_SELECTION);

|  |  |
| --- | --- |
| Example: | Output |
| package jlist\_eg;  import java.awt.FlowLayout;  import javax.swing.event.ListSelectionEvent;  import javax.swing.event.ListSelectionListener;  import javax.swing.JFrame;  import javax.swing.JLabel;  import javax.swing.JList;  import javax.swing.JOptionPane;  import javax.swing.ListSelectionModel;  public class JList\_Eg extends JFrame implements ListSelectionListener{  static JLabel hbLbl=new JLabel("Select your department:");  static String[] hobbies={"Swimming","Reading","Writing", "Other"};  static JList hbLst=new JList(hobbies);  public static void main(String[] args) {  JList\_Eg frm=new JList\_Eg();  frm.setLayout(new FlowLayout());  frm.setTitle("Jlist Example");  frm.setSize(300, 200);  frm.setDefaultCloseOperation(EXIT\_ON\_CLOSE);  frm.setVisible(true);  hbLst.setSelectionMode(ListSelectionModel.SINGLE\_SELECTION);  frm.add(hbLbl);  frm.add(hbLst);    hbLst.addListSelectionListener(frm);  }  @Override  public void valueChanged(ListSelectionEvent e) {  for(int i=0;i<4;i++)  if(hbLst.getSelectedValue()==hobbies[i])  {  JOptionPane.showMessageDialog(rootPane, "You have selected: "+hbLst.getSelectedValue(), "Department", JOptionPane.INFORMATION\_MESSAGE);  }  }    } |  |

**Lab Tasks**

1. Create a student registration form that enables a student to specify his/her department and his/her hobbies. When the user selects hobbies their text will be add to the text area in addition to the select department. The frame of size 255X300. The JTextArea of size 4X20

Sample RUN:

|  |  |
| --- | --- |
| **At start** | **When the use select empty for the department** |
|  |  |
| **The user selected one hobby** | **The user selected multiple hobbies** |
|  |  |

# Computer Science Department

**CS220: OOP2**

# Lab 10: JComboBox, JList, ItemListener, ListSelectionListener

**Second Semester: (2017/2018)**

**Lab 10: MouseListener, MouseMotionListener**

**Introduction**

This lab presents developing GUI application uses MouseListener and MouseMotionListener event-listener interfaces for handling mouse events. Mouse events can be trapped for any GUI component that derives from java.awt.Component. Package javax.swing.event contains interface MouseInputListener, which extends interfaces MouseListener and MouseMotionListener to create a single interface containing all the MouseListener and MouseMotionListener methods. The MouseListener and MouseMotionListener methods are called when the mouse interacts with a Component if appropriate event-listener objects are registered for that Component.

**Objectives**

1. Develop GUI application
2. Practice using MouseListener
3. Practice using MouseMotionListener

**Tools/Software Requirement**

1. Netbeans

**Description**

1. MouseListener

There are five methods of the interface MouseListener are summarized below.

|  |  |
| --- | --- |
| **Method** | **Description** |
| public void mousePressed( MouseEvent event ) | Called when a mouse button is pressed while the mouse cursor is on a component. |
| public void mouseClicked( MouseEvent event ) | Called when a mouse button is pressed and released while the mouse cursor remains stationary on a component.This event is always preceded by a call to mousePressed. |
| public void mouseReleased( MouseEvent event ) | Called when a mouse button is released after being pressed.This event is always preceded by a call to mousePressed and one or more calls to mouseDragged. |
| public void mouseEntered( MouseEvent event ) | Called when the mouse cursor enters the bounds of a component. |
| public void mouseExited( MouseEvent event ) | Called when the mouse cursor leaves the bounds of a component. |

The programmer have to declare all the five methods from MouseListener interface. To implement a MouseListener follow the below steps:

1. Import the following libraries

import java.awt.event.MouseEvent;

import java.awt.event.MouseListener;

1. Add MouseListener to any component/control you

|  |
| --- |
| frm.addMouseListener(frm); // add a MouseListener to JFrame  btn.addMouseListener(frm); // add a MouseListener to JButton |

|  |  |
| --- | --- |
| Example: | Output |
| package mouselistener\_eg;  import java.awt.FlowLayout;  import java.awt.event.MouseEvent;  import java.awt.event.MouseListener;  import javax.swing.JButton;  import javax.swing.JFrame;  import javax.swing.JLabel;  public class MouseListener\_Eg extends JFrame implements MouseListener{  static JLabel lbl=new JLabel("Mouse Event: ");  static JButton btn=new JButton("Click Me");  public static void main(String[] args) {  MouseListener\_Eg frm=new MouseListener\_Eg();  frm.setLayout(new FlowLayout());  frm.setTitle("MouseListener");  frm.setSize(500, 200);  frm.setDefaultCloseOperation(EXIT\_ON\_CLOSE);  frm.add(lbl);  frm.add(btn);  frm.addMouseListener(frm);  btn.addMouseListener(frm);  frm.setVisible(true);  }  @Override  public void mouseClicked(MouseEvent e) {  if(e.getSource()==btn)  lbl.setText("Mouse Event: Mouse Clicked on the Button at: "+e.getX()+","+e.getY());  if(e.getSource()!=btn)  lbl.setText("Mouse Event: Mouse Clicked on the frame at: "+e.getX()+","+e.getY());  }  @Override  public void mousePressed(MouseEvent e) {  if(e.getSource()==btn)  lbl.setText("Mouse Event: Mouse Pressed on the Button at: "+e.getX()+","+e.getY());  if(e.getSource()!=btn)  lbl.setText("Mouse Event: Mouse Pressed on the frame at: "+e.getX()+","+e.getY());  }  @Override  public void mouseReleased(MouseEvent e) {  if(e.getSource()==btn)  lbl.setText("Mouse Event: Mouse Releasedd on the Button at: "+e.getX()+","+e.getY());  if(e.getSource()!=btn)  lbl.setText("Mouse Event: Mouse Released on the frame at: "+e.getX()+","+e.getY());  }  @Override  public void mouseEntered(MouseEvent e) {  if(e.getSource()==btn)  lbl.setText("Mouse Event: Mouse Enteredd on the Button at: "+e.getX()+","+e.getY());  if(e.getSource()!=btn)  lbl.setText("Mouse Event: Mouse Entered on the frame at: "+e.getX()+","+e.getY());  }  @Override  public void mouseExited(MouseEvent e) {  if(e.getSource()==btn)  lbl.setText("Mouse Event: Mouse Exited on the Button at: "+e.getX()+","+e.getY());  if(e.getSource()!=btn)  lbl.setText("Mouse Event: Mouse Exited on the frame at: "+e.getX()+","+e.getY());  }  } |  |

1. MouseMotionListener

There are two methods of the interface MouseMotionListener are summarized below.

|  |  |
| --- | --- |
| **Method** | **Description** |
| public void mouseDragged( MouseEvent event ) | Called when the mouse button is pressed while the mouse cursor is on a component and the mouse is moved while the mouse button remains pressed. This event is always preceded by a call to mousePressed. All drag events are sent to the component on which the user began to drag the mouse. |
| public void mouseMoved( MouseEvent event ) | Called when the mouse is moved when the mouse cursor is on a component. All move events are sent to the component over which the mouse is currently positioned. |

The programmer have to declare both methods from MouseMotionListener interface. To implement a MouseMotionListener follow the below steps:

1. Import the following libraries

import java.awt.event.MouseEvent;

import java.awt.event.MouseMotionListener;

1. Add MouseMotionListener to any component/control you

|  |
| --- |
| frm.addMouseMotionListener(frm); // add a MouseMotionListener to JFrame  btn. addMouseMotionListener(frm); // add a MouseMotionListener to JButton |

|  |  |
| --- | --- |
| Example: | Output |
| package mouselistener\_eg;  import java.awt.FlowLayout;  import java.awt.event.MouseEvent;  import java.awt.event.MouseMotionListener;  import javax.swing.JButton;  import javax.swing.JFrame;  import javax.swing.JLabel;  public class MouseListener\_Eg extends JFrame implements MouseMotionListener{  static JLabel lbl=new JLabel("Mouse Event: ");  static JButton btn=new JButton("Click Me");  public static void main(String[] args) {  MouseListener\_Eg frm=new MouseListener\_Eg();  frm.setLayout(new FlowLayout());  frm.setTitle("MouseMotionListener");  frm.setSize(500, 200);  frm.setDefaultCloseOperation(EXIT\_ON\_CLOSE);  frm.add(lbl);  frm.add(btn);  frm.addMouseMotionListener(frm);  btn.addMouseMotionListener(frm);  frm.setVisible(true);  }  @Override  public void mouseDragged(MouseEvent e) {  if(e.getSource()==btn)  lbl.setText("Mouse Event: Mouse Dragged on the Button at: "+e.getX()+","+e.getY());  if(e.getSource()!=btn)  lbl.setText("Mouse Event: Mouse Dragged on the frame at: "+e.getX()+","+e.getY());  }  @Override  public void mouseMoved(MouseEvent e) {  if(e.getSource()==btn)  lbl.setText("Mouse Event: Mouse Moved on the Button at: "+e.getX()+","+e.getY());  if(e.getSource()!=btn)  lbl.setText("Mouse Event: Mouse Moved on the frame at: "+e.getX()+","+e.getY());  }  } | Compare the output with MouseListener Example |

**Lab Tasks**

1. Create a student registration form that enables a student to insert his/her name, age, and department. When the user clicks, presses, enters, drags the mouse, or moves the mouse over the text fields, a hint will be printed in the hint label. The hints will be "*Insert your full name*" for name, "*Insert your age as integer value*" for age, "*Insert your department full title*" for department. When the user clicks, presses, enters, drags the mouse, or moves the mouse over the ***Done*** label the message "*Thanks, your registration done successfully*" will be printed in the hint label if all the text fields are filled-in, else it will print "*Please, insert all the required fields*". When the user clicks, presses, enters, drags the mouse, or moves the mouse over the form the message "*Please, finish the form*" will be printed in the hint label. The frame of size 350X200. The JTextFields of size 10

Sample RUN:

|  |  |
| --- | --- |
| **At start** | **When the user clicks, presses, enters, drags the mouse, or moves the mouse over the name text fields** |
|  |  |
| **When the user clicks, presses, enters, drags the mouse, or moves the mouse over the form** | **When the user clicks, presses, enters, drags the mouse, or moves the mouse over the *Done* label and all text fields are filled-in** |
|  |  |
| **When the user clicks, presses, enters, drags the mouse, or moves the mouse over the *Done* label and some text fields are NOT filled-in** | |
|  | |

# Computer Science Department

**CS220: OOP2**

# Lab 11: JApplets

**Second Semester: (2017/2018)**

**Lab 11: JApplets**

**Introduction**

An applet is an object that is used by another program, typically a Web browser. The Web browser is the application program and (at least conceptually) holds the main() method. The applet part of a Web page provides services (methods) to the browser when the browser asks for them. An applet object has many instance variables and methods. Most of these are defined in the JApplet class. To access these definitions, your program should import javax.swing.JApplet.

**Objectives**

1. Develop GUI application using JApplet
2. Practice using GUI Controls

**Tools/Software Requirement**

1. Netbeans

**Description**

1. JApplet
2. Import the following libraries

import javax.applet.JApplet

1. Extends the class to JApplet

public class Lab12\_Eg extends JApplet{…}

1. Implement  init() method. This method is called just after the applet object has been created and before it appears on the screen. Its purpose is to give the applet a chance to do any necessary initialization. Again, this method is called by the system, not by your program. Your job as a programmer is just to provide a definition of the init() method. The definition of the method must have the form:

|  |
| --- |
| public void init() {  // do initialization  } |

|  |  |
| --- | --- |
| Example: | Output |
| package lab11\_eg;  import javax.swing.BoxLayout;  import java.awt.event.ItemEvent;  import java.awt.event.ItemListener;  import javax.swing.JLabel;  import javax.swing.JRadioButton;  import javax.swing.ButtonGroup;  import javax.swing.JPanel;  import javax.swing.JOptionPane;  import javax.swing.JApplet;    public class Lab11\_Eg extends JApplet implements ItemListener{  static JPanel p1=new JPanel();  static JLabel gndLbl=new JLabel("Select your gender:");  static JLabel eduLbl=new JLabel("Select your Education Level:");  static ButtonGroup gndGroup= new ButtonGroup(); // buttongroup to hold radio buttons  static ButtonGroup eduGroup= new ButtonGroup(); // buttongroup to hold radio buttons  static JRadioButton mRbtn= new JRadioButton( "Male"); // selects plain text  static JRadioButton fRbtn= new JRadioButton( "Female"); // selects bold text  static JRadioButton schlRbtn= new JRadioButton( "High Schoole"); // selects plain text  static JRadioButton underRbtn= new JRadioButton( "Under-Graduate"); // selects bold text  static JRadioButton gradRbtn= new JRadioButton( "Graduate"); // selects plain text  public void init(){  //this.setsetTitle("JRadioButton Example"); /\*TODo: How can you set the title of JApplet?\*/  this.setSize(300, 200);  this.setVisible(true);  p1.setLayout(new BoxLayout(p1, BoxLayout.Y\_AXIS));  p1.add(gndLbl);  gndGroup.add(mRbtn);  gndGroup.add(fRbtn);  p1.add(mRbtn);  p1.add(fRbtn);  p1.add(eduLbl);  eduGroup.add(schlRbtn);  eduGroup.add(underRbtn);  eduGroup.add(gradRbtn);  p1.add(schlRbtn);  p1.add(underRbtn);  p1.add(gradRbtn);  this.add(p1);  schlRbtn.addItemListener(this);  underRbtn.addItemListener(this);  gradRbtn.addItemListener(this);  }  @Override  public void itemStateChanged(ItemEvent e) {  String gender=new String();  if(mRbtn.isSelected())  gender="Male";  if(fRbtn.isSelected())  gender="Female";  if(schlRbtn.isSelected())  {  JOptionPane.showMessageDialog(rootPane,"You are a "+gender+"\nYou are a " +schlRbtn.getText()+"Student","Students Information",JOptionPane.INFORMATION\_MESSAGE);  }  if(underRbtn.isSelected())  {  JOptionPane.showMessageDialog(rootPane,"You are a "+gender+"\nYou are a " +underRbtn.getText()+"Student","Students Information",JOptionPane.INFORMATION\_MESSAGE);  }  if(gradRbtn.isSelected())  {  JOptionPane.showMessageDialog(rootPane,"You are a "+gender+"\nYou are a " +gradRbtn.getText()+"Student","Students Information",JOptionPane.INFORMATION\_MESSAGE);  }  }  } |  |

**Lab Tasks**

1. Implement the below JApplet as shown in the sample runs, exactly. When the application starts it displays your name and your ID in a JOptionPane message. The application enables the user to compute a circumference value of a square, rectangle and a circle based on the user choice. The user selects a shape by selecting its corresponding radio button. The application keeps ask the user to insert the required dimensions until the user inserts positive value/s. The JTextArea display all the user requests. The Jframe size is 250X300 and the JTextArea size is 10X20.

Equation:

Square circumference=4\* side length

Rectangle circumference=2\*length+2\*width

Circle circumference=2\*3.14\*Radios

**Sample Runs:**

|  |  |
| --- | --- |
| **At start** | **After the user press on Ok button** |
|  |  |
| **When a user select any of the radio buttons, the application asks for the required dimensions to compute the circumference** | **The application keeps asking for inserting the values until the user inserts positive values** |
|  |  |
| **After the user inserts a positive value, the application will prints the result in the JTextArea** | **While the user continue selecting other shapes and inserting the required dimensions, the application keep adding the result into the JTextArea** |
|  |  |

# Computer Science Department

**CS220: OOP2**

# Lab 12: Multi-Frame Application, Color and Font

**Second Semester: (2017/2018)**

**Lab 12: Multi-Frame Application, Color and Font**

**Introduction**

In this lab you are going to learn how to create a multi frame application. In addition to how to practice using Color and Font classes.

**Objectives**

1. Develop Multi-frame GUI application using JApplet
2. Practice using Color and Font classes

**Tools/Software Requirement**

1. Netbeans

**Description**

1. Multi-Frame Application

The idea is about creating another object of type JFrame and make it visible on a specific control action such as a JButton click. You need to pay attention to close the child/new opened frame. To close the child frame call the method dispose(), which is going to destroy the childe JFrame object from the memory. Note the simple multi-frame application bellow:

|  |  |
| --- | --- |
| Example: | Output |
| package lab12\_eg;  import java.awt.event.ActionEvent;  import java.awt.event.ActionListener;  import java.awt.FlowLayout;  import java.awt.event.ActionEvent;  import javax.swing.JFrame;  import javax.swing.JButton;  import javax.swing.JLabel;  import javax.swing.JTextField;  public class Lab13\_Eg extends JFrame implements ActionListener{  static JButton nmBtn=new JButton("Insert");  static JTextField nmTxt=new JTextField(10);  static JLabel nmLbl=new JLabel("Click The Button To Insert Your Name");  static Lab12\_Eg childFrm=new Lab12\_Eg();  public static void main(String[] args) {  Lab12\_Eg mainFrm=new Lab12\_Eg();  mainFrm.setLayout(new FlowLayout());  mainFrm.setTitle("Main Frame");  mainFrm.setSize(300, 200);  mainFrm.setDefaultCloseOperation(EXIT\_ON\_CLOSE);  mainFrm.setVisible(true);  mainFrm.add(nmLbl);  mainFrm.add(nmBtn);  nmBtn.addActionListener(mainFrm);  }  @Override  public void actionPerformed(ActionEvent e) {  if(e.getSource()==nmBtn)  {  JButton dnBtn=new JButton("Done");  childFrm.setLayout(new FlowLayout());  childFrm.setSize(200, 200);  childFrm.add(nmTxt);    childFrm.add(dnBtn);  dnBtn.addActionListener(this);  childFrm.setVisible(true);  }  else  {  nmLbl.setText(nmTxt.getText());//to change nmTxt text value  childFrm.dispose();//to close the childFrm  }  }  } |  |

1. Color

Class Color declares methods and constants for manipulating colors in a Java program. The pre-declared color constants are summarized in the textbook at page 613, and several color methods and constructors are summarized in the textbook at page 614. Note that two of the methods in the textbook at page 614 are Graphics methods that are specific to colors, which mean that you have to import Graphics class. Bellow a simple application that changes JTextField text color:

|  |  |
| --- | --- |
| Example: | Output |
| package lab12\_eg;  import java.awt.Color;  import java.awt.FlowLayout;  import javax.swing.JFrame;  import javax.swing.JTextField;  public class Lab12\_Eg extends JFrame{  static JTextField txt=new JTextField(10);  public static void main(String[] args) {  Lab12\_Eg mainFrm=new Lab12\_Eg();  mainFrm.setLayout(new FlowLayout());  mainFrm.setTitle("Main Frame");  mainFrm.setSize(200, 200);  mainFrm.setDefaultCloseOperation(EXIT\_ON\_CLOSE);  mainFrm.setVisible(true);  txt.setForeground(Color.red);  mainFrm.add(txt);  }  } |  |

* To Open color chooser dialog, which enable the user to select a specific color out of a wide range of colors:

1. Import java.swing.JColorChooser
2. Declare an object of type color
3. Store the select value from the JColorChooser into the color object:

clr=JColorChooser.showDialog(this, "Choose Text Color", clr);

Note: clr is an object of type Color.

1. Font

Class Font’s constructor takes three arguments—the font name, font style and font size. The font name is any font currently supported by the system on which the program is running, such as standard Java fonts Monospaced, SansSerif and Serif. The font style is Font.PLAIN, Font.ITALIC or Font.BOLD (each is a static field of class Font). Font styles can be used in combination (e.g., Font.ITALIC + Font.BOLD). The font size is measured in points. A point is 1/72 of an inch. Graphics method setFont sets the current drawing font—the font in which text will be displayed—to its Font argument. Note the simple application bellow, which sets the font of JTextField:

|  |  |
| --- | --- |
| Example: | Output |
| package lab13\_eg;  import java.awt.Font;  import java.awt.FlowLayout;  import javax.swing.JFrame;  import javax.swing.JLabel;  import javax.swing.JTextField;  public class Lab12\_Eg extends JFrame{  static JTextField normalTxt=new JTextField(10);  static JTextField srfBldTxt=new JTextField(10);  static JLabel lbl1=new JLabel("Normal Text:");  static JLabel lbl2=new JLabel("Serif Bols Text of size 12:");  public static void main(String[] args) {  Lab12\_Eg mainFrm=new Lab12\_Eg();  mainFrm.setLayout(new FlowLayout());  mainFrm.setTitle("Main Frame");  mainFrm.setSize(200, 200);  mainFrm.setDefaultCloseOperation(EXIT\_ON\_CLOSE);  mainFrm.setVisible(true);  srfBldTxt.setFont(new Font("Serif",Font.BOLD ,12 ));  mainFrm.add(lbl1);  mainFrm.add(normalTxt);  mainFrm.add(lbl2);  mainFrm.add(srfBldTxt);  }  } |  |

**Lab Tasks**

1. Implement the below JFrame as shown in the sample runs. This application enable the user to change the text font type, size, and color based on his/her selections from th JComboBoxes. The Jframe size is 450X250 and the JTextArea size is 10X30.

**Sample Runs:**

|  |
| --- |
| **At start** |
|  |
| **When the user click on select color button, the JColorChooser is displayed** |
|  |
| **When the user change the size to 15** |
|  |
| **When the user change the font type to Serif** |
|  |

# Computer Science Department

**CS220: OOP2**

# Lab 13: Database Access Using JDBC

**Second Semester: (2017/2018)**

**Lab 13: Database Access Using JDBC**

**Introduction**

The purpose of this lab is to develop an application which retrieve, update and persist data in database.

**Objectives**

1. To implement Student Information System (SIS) scenario [Reference-Assignment-1].

*“We model Students and Teachers and implement in our school main Student Information System. As a person, both Students and Teachers have common attributes such name:String, address:String. As a Students, they have additional information such as courseNames:String[], numOfCoursesTaking:int and grades:int[]. The Teachers have courseNames:String[] for teaching and numOfCoursesTeaching:int.”*

1. Design database schema for SIS and creating tables in DBMS.
2. Demonstrate establishing connection to SIS database.
3. Retrieving data from SIS database.
4. Persist data into SIS database.

**Tools/Software Requirement**

1. NetBeans
2. JDBC

**Database Management System (DBMS)**

1. Microsoft Access 2010/2013

**Activities Detail**

**Activity 1: Designing and creating database schema**

* Based on requirements of SIS, *SEVEN* tables are required to be created in database.
  1. PERSON
  2. STUDENT
  3. TEACHER
  4. COURSE
  5. TEACHER\_COURSES
  6. STUDENT\_COURSES
  7. TEACHER\_STUDENTS
* **Creating Tables in DBMS**
  1. Creating PERSON
     + Identifying candidate columns and corresponding datatypes:

[pid: AutoNumber, pName: Short Text, pAddress: Long Text]

* + - Designate primary key column (if required).

pid: AutoNumber.

* + - Using Microsoft Access to create *Table* PERSON [Refer to Figure 1]

STEP1: Open MS Access and create Blank Database and named as dbSIS.

STEP2: Go to Tab: Create -> Table

STEP3: Name table as: PERSON

STEP4: Add column one by one and assigning name in “Field Name” and

corresponding datatype in “Data Type”.

STEP5: Assign primary key by right clicking on the corresponding “Field

Name” [here pid].

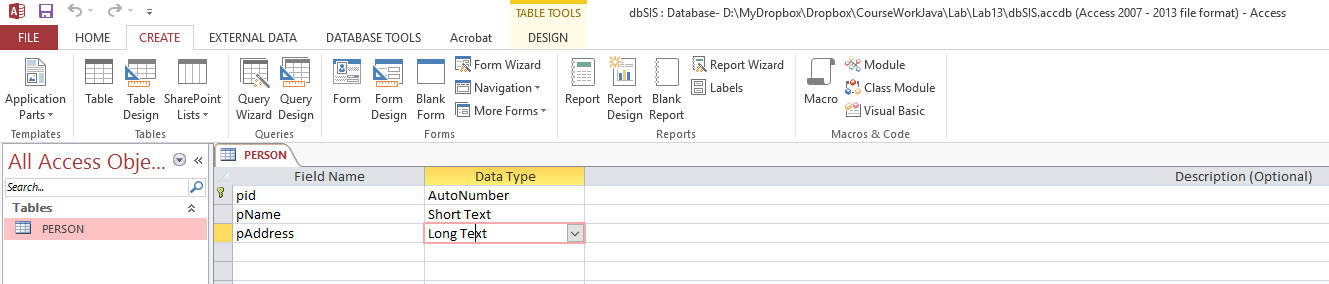


Figure 1: PERSON table creation [design mode]

* 1. Create STUDENT, TEACHER, COURSE, TEACHER\_COURSES, TEACHER\_STUDENTS, and STUDENT\_COURSES
     + Refer to “Student Task-I”.
* **Establishing Foreign Key constraint**
  1. Selecting tables [Refer Figure 2]
     + Select main menu: Design -> Relationships
     + In “Show Table”, select all tables and click “Add” button.

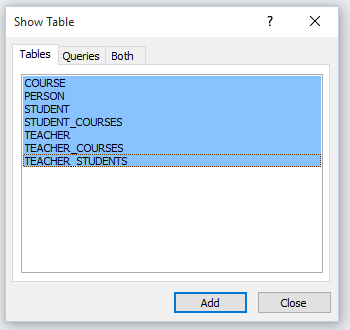


Figure 2: Relationships Diagram

* 1. Creating referential constraints: Foreign Key (FK) [Refer Figure 3]
     + Creating PK-FK between PERSON [pid] and STUDENT [pid]
     + Select pid column in PERSON and keeping press mouse button and drag and draft it on pid in STUDENT table. “Edit Relationships” box will appear. Click button to create the constraint.
     + Creating PK-FK for other tables. Please refer to Student Task-I and create PK-FK for all tables. The final relationship diagram – a schema is shown in Figure 4.

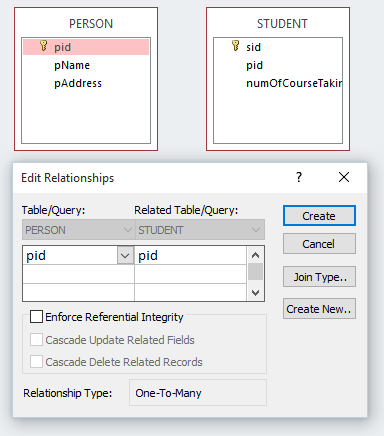
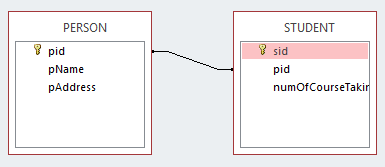
 

Figure 3: FK editing dialog

**Students Task-I**

* 1. Use the procedure mentioned in Activity1 [Step2-Step4] for creation of the following tables.
     + COURSE with columns and datatypes: [cid: AutoNumber, courseName: Long Text] -> *Make cid as Primary Key.*
     + STUDENT with columns and datatypes: [sid: AutoNumber, numOfCourseTaking: Number, pid: Number] -> *Make sid as Primary Key and pid is Foreign Key from PERSON table.*
     + TEACHER with columns and datatypes: [tid: AutoNumber, numOfCoursesTeaching: Number, pid: Number] -> *Make sid as Primary Key and pid is Foreign Key from PERSON table.*
     + TEACHER\_COURSES with columns and datatypes: [tcid: AutoNumber, tid: Number, cid: Number]. *-> Make tcid as Primary Key, tid is Foreign Key from TEACHER table, and cid is Foreign Key from COURSE table.*
     + STUDENT\_COURSES with columns and datatypes: [scid: AutoNumber, sid: Number, tcid: Number, grade: Number]*->Make scid as Primary Key, sid is Foreign Key from STUDENT, and tcid is Foreign Key from TEACHER\_COURSES table.*
     + TEACHER\_STUDENTS with columns and datatypes: [tsid: AutoNumber, tid: Number, sid: Number] *-> Make tsid is Primary Key, tid is Foreign Key from TEACHER, and sid is Foreign Key from STUDENT table.*

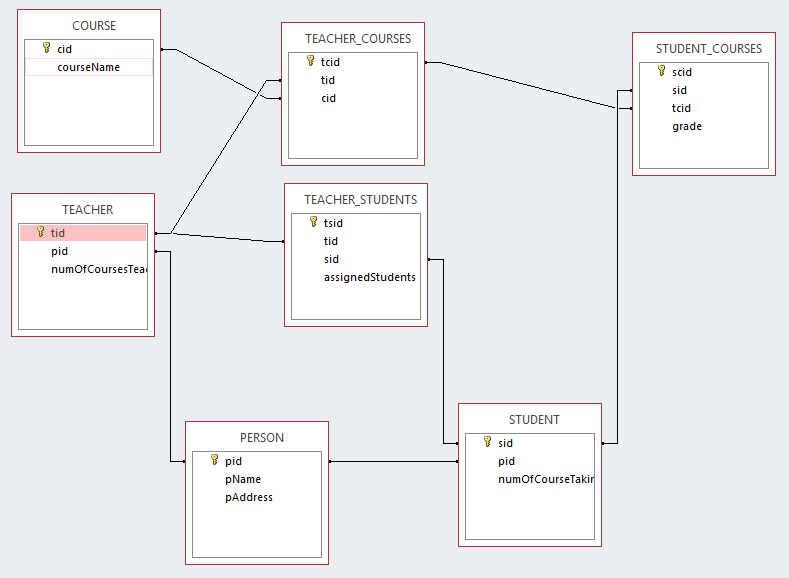


Figure 4: SIS database schema

**Activity 2: Creating ODBC system DNS to Access database**

Step1: Control Panel -> Administrative Tool -> ODBC Data Source -> System DNS -> Add

Step2: Select Microsoft Access Driver(\*.mdb, \*.accdb) -> Finish

Step3: In Data Source Name: DSNLab13Access … and Select the database file of access and OK.

**Activity3: Add Student records in database through Access**

Step1: Double click PERSON table and add student as: pName: Abrahim pAddress: KSA [pid will be automatically generated … for first student it will be 1]

Step2: Double click STUDENT table and add student information: pid: 1 numofCourseTaking: 2

*Repeat step 1-2 for adding Nael of KSA with 1 course*.

**Activity 4: Retrieving Student by id and by name [Query Making]**

*Query1: Retrieving student by id*

SELECT PERSON.pid, PERSON.pName, PERSON.pAddress, STUDENT.sid, STUDENT.numOfCourseTaking FROM PERSON INNER JOIN STUDENT ON PERSON.[pid] = STUDENT.[pid] Where STUDENT.sid = 1

*Query 2: Retrieving student whose name contains provided characters – such as by providing Ab will retrieve* ***Ab****rahim,* ***Ab****rar and R****ab****eel*

SELECT PERSON.pid, PERSON.pName, PERSON.pAddress, STUDENT.sid, STUDENT.numOfCourseTaking FROM PERSON INNER JOIN STUDENT ON PERSON.[pid] = STUDENT.[pid] Where STUDENT. pName like ‘%Ab%’

**Activity 5: Implementation Student Retrieval using JDBC**

**Step 1: Write Person and Student Classes**

public class Person {

protected String name;

protected String address;

public Person() {

}

@Override

public String toString(){

return name + "[" + address + "]";

}

} **----------------------------**

public class Student extends Person{

private int numOfcourses;

public String getName() {

return super.name;

}

public void setName(String name) {

super.name = name;

}

public String getAddress() {

return super.address;

}

public void setAddress(String address) {

super.address = address;

}

public int getNumOfcourses(){

return this.numOfcourses;

}

public void setNumOfcourses(int numOfcourses){

this.numOfcourses = numOfcourses;

}

@Override

public String toString(){

String stdInfo = super.toString() + " Course Details: ";

return stdInfo;

}

}

**Step 2: Write Database access code for retrieving Student.**

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

public class StudentDAOImpl {

static final String DRIVER = "jdbc:odbc:DSNLab13Access";

private static Connection dbCon = null;

public StudentDAOImpl()

{

try{

Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

dbCon = DriverManager.getConnection(DRIVER);

}

catch(SQLException sqlExp)

{

sqlExp.printStackTrace();

}

catch (ClassNotFoundException clex)

{

clex.printStackTrace();

}

}

public Student getStudentByID(int id)

{

Student std = new Student();

ResultSet rs;

try{

Statement stmt = dbCon.createStatement();

rs = stmt.executeQuery("SELECT PERSON.pid, PERSON.pName, PERSON.pAddress, STUDENT.sid, STUDENT.numOfCourseTaking FROM PERSON INNER JOIN STUDENT ON PERSON.[pid] = STUDENT.[pid] Where STUDENT.sid = " + id);

while(rs.next())

{

std.setName( rs.getString("pName"));

std.setAddress(rs.getString("pAddress"));

}

}

catch (SQLException sqlExp)

{

sqlExp.printStackTrace();

}

return std;

}

public Student getStudentByName(String studentName)

{

Student std = new Student();

ResultSet rs;

try{

Statement stmt = dbCon.createStatement();

rs = stmt.executeQuery("SELECT PERSON.pid, PERSON.pName, PERSON.pAddress, STUDENT.sid, STUDENT.numOfCourseTaking FROM PERSON INNER JOIN STUDENT ON PERSON.[pid] = STUDENT.[pid] Where PERSON.pName like \'%" + studentName + "%\'");

while(rs.next())

{

std.setName( rs.getString("pName"));

std.setAddress(rs.getString("pAddress"));

}

}

catch (SQLException sqlExp)

{

sqlExp.printStackTrace();

}

return std;

}

}

**Step 3: Test the database access for student retrieval.**

public class StudentTest {

public static void main(String arg[])

{

StudentDAOImpl studentDataAccess = new StudentDAOImpl();

Student std = studentDataAccess.getStudentByID(2);

System.out.println("Student Name: " + std.getName() + "\t Address: " + std.getAddress());

Student std1 = studentDataAccess.getStudentByName("ab");

System.out.println("Student Name: " + std1.getName() + "\t Address: " + std1.getAddress());

}

}