BCSCCS407R02 / BITCIT407R02 / BICCIC408

JAVA PROGRAMMING LABORATORY

School of Computing
Shanmugha Arts, Science, Technology and Research Academy [SASTRA]
Thanjavur – 613 401

List of Experiments

1. Program to demonstrate Polymorphism.
2. Program to demonstrate Inheritance.
3. Program to demonstrate Interfaces and Packages.
4. Program to demonstrate Exception handling.
5. Program to demonstrate Multithreading.
6. Program using ArrayList
7. Program using Set, Comparable interface and Comparator class
8. Program to create applet with AWT controls.
9. Program to create Frame with AWT controls.
10. Program using Layout Managers.
11. Program to demonstrate file I/O.
12. Program using JDBC.

Additional Exercises

1. Program demonstrating nested classes
2. Understand and handle mouse events and keyboard events
3. Program to demonstrate String and StringBuffer classes
4. Program using Adapter Classes.
5. Communicate between systems using TCP/UDP

1. Polymorphism in Java

Objective: To demonstrate polymorphism in Java.

Pre-Lab Exercise: Demonstrating classes and objects

Create a class called Book that contains instance variables like BKName, BKId and BKAuthor, a parameterized constructor to initialize its instance variables, a method BKUpdateDetails(String name, int id, String author), that accepts new values for name, Id and author as parameters and updates the corresponding instance variable values of that object and another method BKDisplay() to display the book details. Create a class BookDemo and provide main method for instantiate a Book object, display the original book details, update its details with new values, and display the updated book details.

Lab Exercise: To demonstrate compile time polymorphism in Java.

Create a Point class that has two data members x and y of double type. Write a default constructor which initializes its data members to zero. Write a parameterized constructor which takes two parameters of double type and assigns them to its data members. Write another parameterized constructor that takes one Point object as parameter and copies the values of the passed object's data member to the calling object's data members. Write a find_distance method that takes two double parameters representing the x, y values of a point and finds the distance between the calling object and the passed parameters and returns the distance as double value. Overload the find_distance method, that takes single Point object parameter and computes the distance between both the points and returns the distance as a double value. Write a display method to print the point in the format of "(x, y)". In main method create three point objects p1, p2 and p3. Initialize p1 with (3.25, 7.89), p2 with (5.37, 18.12) and p3 with p2. Find distance between p1 and (7.9, 16.25) using first find_distance method and between p1 and p3 using second find_distance method.

2. Inheritance in Java

Objective: To demonstrate the concept of inheritance in Java

Pre-Lab Exercise:

Create an abstract class called Shape2d that has two data members of type double and two abstract methods area() and display(). Create two derived classes Rectangle and Triangle. In both classes define the method area() to compute the area of that shape and return it as a double value and define the method display() to display the value of the data members with appropriate caption and the calculated area. Create a class called Shape2dDemo and provide main method to instantiate the objects of Rectangle and Triangle for demonstration of the above classes.

Lab Exercise:

Create a class called Account that has the protected data members accnumber of int type, balance of double, and constructors for initialization.

Derive a class called SBAccount and define methods deposit(double), withdraw(double), calc_interest(). Provide a parameterized constructor with two parameters account number, and init_balance and assign these values to the appropriate data members of the class using super class constructor. The deposit method should take one double type argument amount and adds the amount to the balance if the amount is positive. The withdraw method, should take one double type argument amount and checks if balance - amount is greater than Rs.1000/-. If so, it should subtract the amount from balance. Otherwise it should display error message. The calc_interest() method should compute interest for the balance amount available @ 4% for one year and the interest amount should be credited to the balance.

Derive another class called FDAccount that has a data member period of int type. Provide parameterized constructor with parameters for accno, period and deposit amount for initializing them. Provide a method called calc_interest() for calculating interest for the deposit amount for the given period @ 8.25% p.a. and returns the calculated interest, and another method called close() which calls calc_interest() add it to the balance.

Create a class called Customer. The data members of the class are cust_id of int type, name, and address of string type, and objects of SBAccount, and FDAccount. Provide parameterized constructor with cust_id, name and address as parameters for initializing. Provide a method called createAccount(int type). Based on the value of type (SB, or FD) create a new account of given type. Provide a method transaction(int type) {type may be withdraw, deposit or calc_interest on SBAccount, or closing of FDAccount} to perform the requested transaction on the requested Account object.

To demonstrate all the functionalities of above classes, create a class called BankDemo and declare a main method to create an array of 5 elements of Customer objects, and perform manipulations on the objects by creating SB and FD accounts, depositing and withdrawing amount from SB account, and closing FD account.

3. Interfaces and Packages in Java

Objective: To implement the concept of interfaces and packages in Java.

Pre-Lab Exercise:

Create a package called shape2d. In that package declare an interface called twod and three classes Square, Rectangle and Circle. In all three classes implement the twod interface. In interface twod declare two methods area() and perimeter(). Both returns double value. Square class contains one instance variable called side of double type, a parameterized constructor for initializing that member, area() and perimeter() methods for computing and returning the area and perimeter of the square using appropriate formulae. Similarly Rectangle class contains two instance variables called length and breadth of double type, a parameterized constructor with two parameters for initializing those members, area() and perimeter() methods for computing and returning the area and perimeter of the rectangle using appropriate formulae. Circle class contains one instance variable called radius of double type, a parameterized constructor for initializing that member, area() and perimeter() methods for computing and returning the area and perimeter of the circle using appropriate formulae.

Create another package called shape3d. In that package declare an interface called threed and three classes Cube, Cuboid and Sphere. In all three classes implement the threed interface. In interface threed declare two methods volume() and surfaceArea(). Both returns double value. Cube class extends the Square class. It contains a parameterized constructor for setting the value for the member side of its super class through the super class constructor, volume() and surfaceArea() methods for computing and returning the volume and surface area of the cube using appropriate formulae. Similarly Cuboid class extends the Rectangle class, one instance variable called height and a

parameterized constructor with 3 parameters len, bdth, and hgt for setting the values for its member height and its super class members length and breadth through the super class constructor, volume() and surfaceArea() methods for computing and returning the volume and surface area of the cuboid using appropriate formulae. Sphere class extends Circle class contains a parameterized constructor for initializing its super class member radius through super class constructor, volume() and surfaceArea() methods for computing and returning the volume and surface area of the sphere using appropriate formulae. Create a class ShapeDemo outside the packages and declare a main method for creating objects of all 2D and 3D shapes, calculate area, perimeter, volume and surface area accordingly using the appropriate interface references and display.

Lab Exercise:

Create a package named pkbanking. In that package define a sub package named pkinterface. The package pkinterface contains an interface named Transaction with a data member min_balance initialized to 500, two methods namely void withdraw(double) and void deposit(double) and another interface named InterestRate with two data members sbrate with 4%, fdrate with 8.25%.

Create another package named pkaccount. In that package create an abstract class called Account that has the protected data members accnumber of int type, balance of double, constructors for initialization. Define two subpackages sb and fd in pkaccount. In sb package derive a class called SBAccount from Account class that implements both Transaction and InterestRate interfaces. Also it has constructors for initialization using super class constructors, deposit(double), withdraw(double), and calc_interest() methods. Provide a parameterized constructor with two parameters account number, and init_balance and assign these values to the appropriate

data members of the class using super class constructor. The deposit method should take one double type argument amount and adds the amount to the balance if the amount is positive. The withdraw method, should take one double type argument amount and checks if balance - amount is greater than min_balance. If so, it should subtract the amount from balance. Otherwise it should display error message. The calc_interest() method should compute interest for the balance amount available @ sbrate p.a. and the interest amount should be credited to the balance.

In fd package derive a class called FDAccount from Account class that implements InterestRate interface. It has one data member period of int type. Provide parameterized constructor with parameters for accno, period and deposit amount for initializing them using super class constructor. Provide a method called calc_interest() for calculating interest for the deposit amount for the given period @ fdrate p.a. and returns the calculated interest, and another method called close() which calls calc_interest() add it to the balance.

Create third package named pkcustomer which contains a class Customer. The data members of the class are cust_id of int type, name, and address of string type, and Array of objects of SBAccount and FDAccount. Provide parameterized constructor with cust_id, name and address as parameters for initializing. Provide a method called createAccount(int type). Based on the value of type (SB, or FD) create a new account of given type and store it in the corresponding array. Provide a method transaction(int type) {type may be withdraw, deposit or calc_interest on SBAccount, closing of FDAccount} to perform the requested transaction on the requested account object.

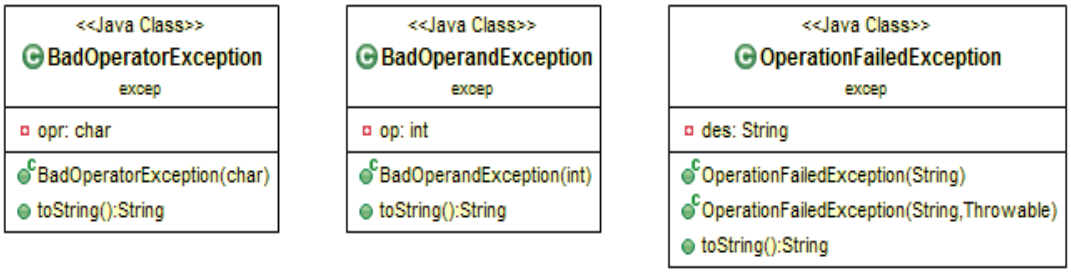
To demonstrate all the functionalities of above classes, create a class called BankDemo and declare a main method to create an array of 5 elements of Customer objects, and perform manipulations on the objects by creating SB and FD accounts, depositing and withdrawing amount from SB accounts, and closing FD accounts.

4. Exception Handling in Java

Objective: To implement the concept of exception handling in Java.

Pre-Lab Exercise:

Create three exception classes as detailed in the following class diagram:



In the main() method get input for two operands and one operator from the user and check if the first operand is within the range 10,000 to 50,000 and second operand is within the range 500 to 5000 and the operator is one of the characters '+', '-', '*' or '/'. If the operand is not within the range then create and throw an OperationFailedException with BadOperandException as its root cause. If the operator is not one of the listed characters create and throw an OperationFailedException with BadOperatorException as its root cause. If exception is thrown, print the exception and its root cause; otherwise print the result of the evaluated expression.

Lab Exercise:

Create a package named pkbanking. In that package define two sub packages named pkinterface and pkexception. The package pkinterface contains an interface named Transaction with a data member min_balance initialized to 500, two methods namely void withdraw(double) throws InsufficientFundsException and void deposit(double) and another interface named InterestRate with three data members sbrate with 4%, rdrate with 6%, fdrate with 8.25%. The package pkexception contains a user defined exception class named InsufficientFundsException which has a String type member named

description. Provide both default constructor that assigns “Insufficient Funds” to description and a parameterized constructor that takes one string argument and stores it in its member description. Override the toString() method to return description.

Create another package named pkaccount. In that package create a class called Account that has the protected data members accnumber of int type, balance of double, constructors for initialization. Define two subpackages sb and fd in pkaccount. In sb package derive a class called SBAccount from Account class that implements both Transaction and InterestRate interfaces. Also it has constructors for initialization using super class constructors, deposit(double), withdraw(double), and calc_interest() methods. Provide a parameterized constructor with two parameters account number, and init_balance and assign these values to the appropriate data members of the class using super class constructor. The deposit method should take one double type argument amount and adds the amount to the balance if the amount is positive. If the amount is negative, it should throw an IllegalArgumentException. The withdraw method, should take one double type argument amount and checks if balance - amount is greater than min_balance. If so, it should subtract the amount from balance. Otherwise it should throw InsufficientFundsException. The calc_interest() method should compute interest for the balance amount available @ sbrate for one year and the interest amount should be credited to the balance.

In fd package derive a class called FDAccount from Account class that implements InterestRate interface. It has one data member period of int type. Provide parameterized constructor with parameters for accno, period and deposit amount for initializing them using super class constructor. Provide a method called calc_interest() for calculating interest for the deposit amount for the given period @ fdrate p.a. and returns the calculated interest, and another method called close() which calls calc_interest() add it to the balance.

Create third package named pkcustomer which contains a class Customer. The data members of the class are cust_id of int type, name, and address of string type, and Array of objects of Account. Provide parameterized constructor with cust_id, name and address as parameters for initializing. Provide a method called createAccount(int type). Based on the value of type (SB, or FD) create a new account of given type and store it in the corresponding array. Provide a method transaction(int type) {type may be withdraw, deposit or calc_interest on SBAccount, closing of FDAccount} to perform the requested transaction on the requested account object.

To demonstrate all the functionalities of above classes, create a class called BankDemo and declare a main method to create an array of 5 elements of Customer objects, and perform manipulations on the objects by creating SB and FD accounts, depositing and withdrawing amount from SB accounts, and closing FD accounts. Provide exception handling mechanisms to handle the exceptions.

5. Program to create multiple threads in JAVA.

Objective: To implement the concept of threads in JAVA.

Pre-Lab Exercise:

Create three threads namely Factorial, SumOfSeries and MultiplicationTable. Fact thread should produce factorial value. Sum thread should produce sum of natural numbers. MultiplicationTable thread class should produce multiplication table. From the main class, create the thread objects. Demonstrate the use of the methods like setPriority(), getPriority(), sleep(), suspend(), resume(), isAlive() and join() methods.

Lab Exercise:

Create a class called queue which has an array of integers, and front and rear position values to represent the front and rear of the queue, and two Boolean flags full and empty. Implement two synchronized methods get and put to retrieve next item to consume from the queue and to store the newly produced item into the queue. Set the flags appropriately according to the state of the queue. Create one object of the queue and two threads namely producer thread and consumer thread and share the queue for both the threads. Producer thread should produce items and place it in the queue and the consumer will consume the items one by one from the queue. If the queue is full the producer has to wait until one of the items is consumed by the consumer. Similarly if the queue is empty the consumer has to wait until the producer places at least one item into the queue.

6. ArrayList

Pre-Lab Exercise:

Declare three methods: reverse(ArrayList <String> words) that reverses the order of the elements in words, capitalizePlurals(ArrayList <String> words) that replaces every word ending with an "s" with its uppercased version, removePlurals(ArrayList <String> words) that removes every word in the list ending with an "s", case-insensitively. In main method create an ArrayList <String> and add words to it. Display all the words, call the above three methods and display the contents of the ArrayList after each operation.

Lab Exercise:

Create a Point class that has two data members x and y of double type. Write a default constructor which initializes its data members to zero. Write a parameterized constructor which takes two parameters of double type and assigns them to its data members. Write another parameterized constructor that takes one Point object as parameter and copies the values of the passed object's data member to the calling object's data members. Write a find_distance method that takes two double parameters representing the x, y values of a point and finds the distance between the calling object and the passed parameters and returns the distance as double value. Overload the find_distance method, that takes single Point object parameter and computes the distance between both the points and returns the distance as a double value. Write a display method to print the point in the format of "(x, y)". In main method create an ArrayList <Point> objects. Find distance between every pair of points and display it.

7. Set, Comparable interface and Comparator class

Pre-Lab Exercise:

Declare three methods: reverse(TreeSet <String> words) that reverses the order of the elements in words, capitalizePlurals(TreeSet <String> words) that replaces every word ending with an "s" with its uppercased version, removePlurals(TreeSet <String> words) that removes every word in the list ending with an "s", case-insensitively. In main method create a TreeSet <String> and add words to it. Display all the words, call the above three methods and display the contents of the TreeSet after each operation.

Lab Exercise:

Create a Point class that implements Comparable Interface and has two data members x and y of double type. Write a default constructor which initializes its data members to zero. Write a parameterized constructor which takes two parameters of double type and assigns them to its data members. Write another parameterized constructor that takes one Point object as parameter and copies the values of the passed object's data member to the calling object's data members. Write a find_distance method that takes two double parameters representing the x, y values of a point and finds the distance between the calling object and the passed parameters and returns the distance as double value. Overload the find_distance method, that takes single Point object parameter and computes the distance between both the points and returns the distance as a double value. Define the compareTo method that returns difference between the distances of the two points from origin. Write a display method to print the point in the format of "(x, y)". In main method create a HashSet <Point> objects. Create and add 10 Point objects in the HashSet. Display the points in the ascending order of the points based on the distance from the origin to that point. Also find distance between every pair of points and display it.

8. Simple Applet using AWT controls

Objective: To create an Applet in JAVA, that shows the manipulation of Labels, Text Field and Buttons (Calculator application)

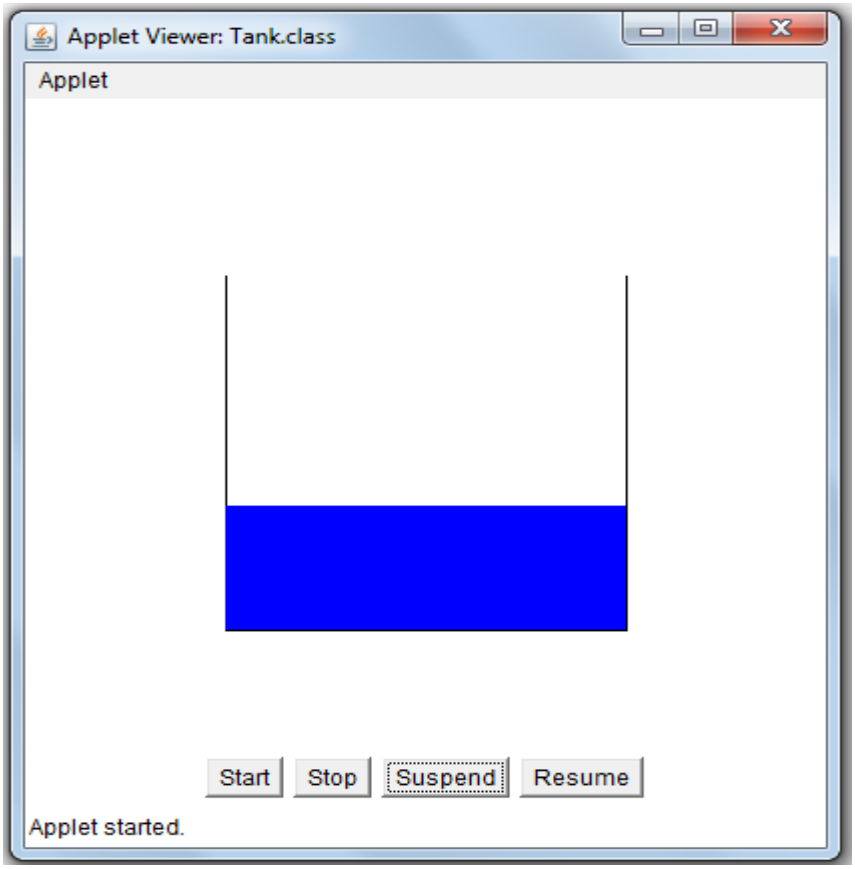
Pre-Lab Exercise:

Write an Applet program to draw various graphical shapes like circle, oval, square, rectangle, polygon with and without filled area using the methods available in Graphics class

Lab Exercise:

Write a JAVA Applet program that displays the current water tank level and buttons like start, stop, suspend and resume. When clicking on the start button the water level in the tank should be increased by some fixed quantity for every second. When suspend button is pressed the filling process should be suspended temporarily and once resume button is pressed the filling of water should be resume from the point left. When stop button is pressed the filling process should be terminated. Also it should be able to restart again the filling process from beginning by pressing the start button again. The applet should check for all possible error conditions and should display the error message in the status bar. For example if stop button is pressed without starting the filling process, if the suspend button is pressed without starting the process, if resume is pressed without suspend, if water overflows.

GUI


Applet Viewer: Tank.class
Applet
Start
Stop
Suspend
Resume
Applet started.

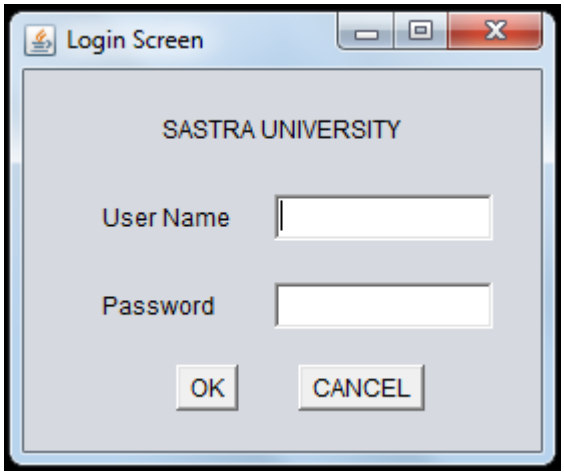
9. Frame using AWT

Objective: To design a Java Application using Frames

Pre-Lab Exercise:

Write a stand-alone application using Frame to create a login screen which has two labels and two text fields to display and get User Name and Password and two buttons namely OK and CANCEL. In the text fields initially "Enter User Name" / "Enter Password" should be displayed with light gray color. When the focus moves to that text field, the message should be cleared. After entering the username and password when the user presses the OK button, it should check for validity and display either "Login Successful" or "Invalid User Name and / or Password" message. Also when the input focus is on OK button if the user presses enter key it should check for validity and display either "Login Successful" or "Invalid User Name and / or Password" message. If the user presses the CANCEL button the current content in the text field should be cleared and "Enter User Name" / "Enter Password" should be displayed with light gray color.

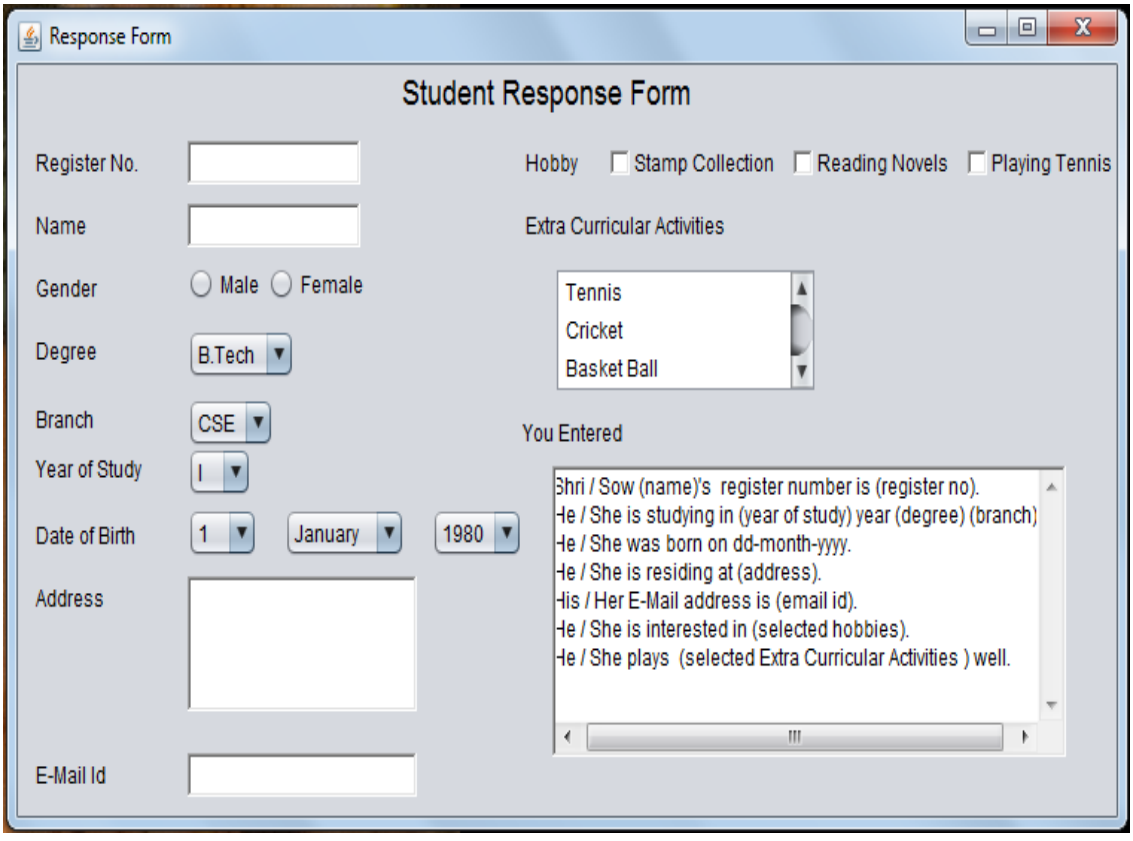
GUI:



Lab Exercise:

Create a Student Response Form, which include details such as Name, register number, sex, degree, branch, year of study date of birth, address (include e-mail id), Hobby and Extra-Curricular activities. For getting these details, use appropriate GUI interface elements such as Label, Textfield, Textarea, List/Choice, radio Button, Checkbox Button etc. Add appropriate event handling mechanism to these controls. The event handler should validate the keyed in input. For example, Name field cannot be blank, register number should be 9 digits, etc. Combine all these collected details in a simple English Sentence form and display it in a Textarea control.

GUI:



10. Layout Managers

Objective: To demonstrate panels and the various layout managers using Applet.

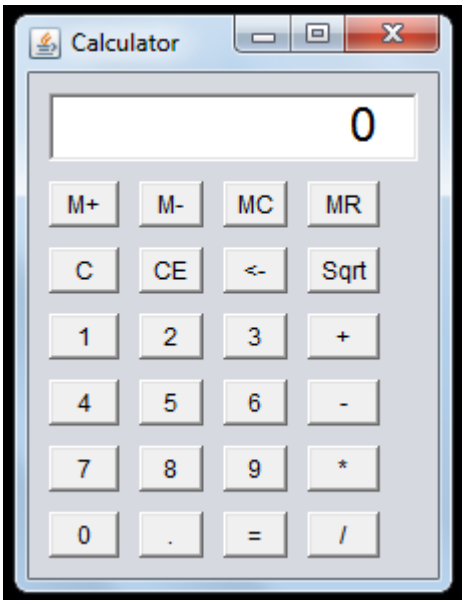
Pre-Lab:

Create an application and create several Panel objects. Use methods that loads and displays images in the panels using flow layout and card layout.

Lab Exercise:

Implement a java application using Frames to design a calculator. Apply Border Layout and Grid Layout. Include the options for Backspace, CE, C, M+, M-, MC, MR, and sqrt. Add buttons numbers 0 to 9 and for basic operations such as +, -, *, / and =. Use Text Field and array of Buttons. Use ActionListener interface to implement the appropriate functionalities when clicking the buttons.

GUI:



11. File I/O

Pre-Lab Exercise:

Write a Java program to copy the contents of one file to another.

Lab Exercise:

Create a Student Response Form, which include details such as Name, register number, sex, degree, branch, year of study date of birth, address (include e-mail id), Hobby and Extra-Curricular activities. For getting these details, use appropriate GUI interface elements such as Label, Textfield, Textarea, List/Choice, radio Button, Checkbox Button etc. Add appropriate event handling mechanism to these controls. The event handler should validate the keyed in input. For example, Name field cannot be blank, register number should be 9 digits, etc. Combine all these collected details in a simple English Sentence form and write it into a file.

12. Student Management using JDBC

Objective: To understand database connectivity, operation and manipulation using JDBC, with the help of student management system.

Pre-Lab:

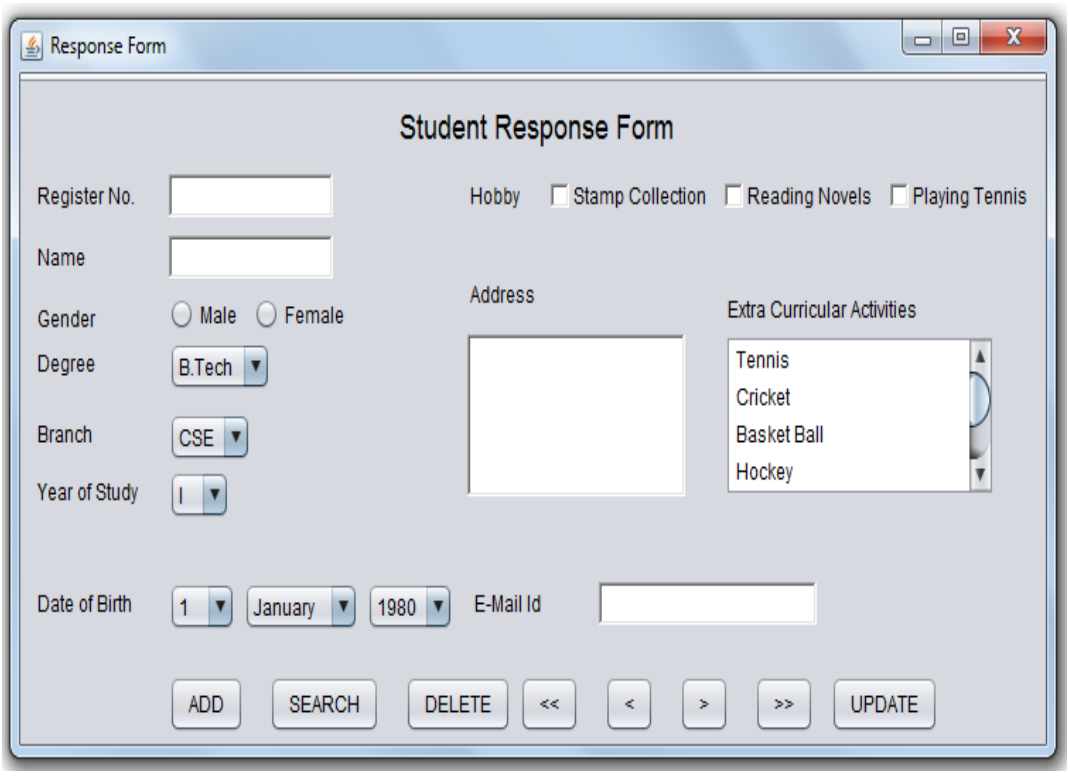
Create a table to store login and password as in Ex-9 (Login Screen). Upon clicking OK the username/password should be validated using the table.

Lab Exercise:

Create a Standalone Java Application using Frames for developing a Student Management System. Design the backend by creating a table Student with the following fields: Register Number, Name, Gender, Degree, Branch, Year of Study, Date of Birth, Address, E-Mail-Id, Hobby, and Extra-Curricular Activities

Using JDBC connect the database and design the following GUI for manipulation of database with buttons for adding, deleting, updating and browsing through students' records. Initially display the first record in the student table. When the user presses Clear button display a blank form. When the user after entering register number presses Search button, display the corresponding student's detail in the form. When the user presses Add button, display a blank form. After entering all the details the user should press the Update button to insert the record into the student table. When the user presses Delete button the current student record that is displayed in the form should be deleted from the student table. Implement the button << for moving to the first record, < for moving to the previous record, > for moving to the next record and >> for moving to the last record. If any student information in the form is updated during search or browse operations, upon clicking Update button the corresponding record should be updated with the current information.

GUI:


Response Form
Student Response Form
Register No.
Name
Gender
Male
Female
Degree
B.Tech
Branch
CSE
Year of Study
I
Hobby
Stamp Collection
Reading Novels
Playing Tennis
Address
Extra Curricular Activities
Tennis
Cricket
Basket Ball
Hockey
Date of Birth
1
January
1980
E-Mail Id
ADD
SEARCH
DELETE
<<
<
>
>>
UPDATE