**Object oriented programming using JAVA - LAB**

**Assignment 1**

**Topic:** Simple Java Programs.

|  |  |
| --- | --- |
| **Sl. No.** | **Question** |
| 1. | Write a java program that will display “This is my first Java Program”. |
| 2. | Write a java program that will print your name 10 times. (Hints: Use for Loop) |
| 3. | Write a Java program to test the number is prime or composite. |
| 4. | Write a java program that will take input as two +ve integers M and N where M<N.  a). The program will display the numbers between M and N, which are divisible by 3 and 5.  b). Count the numbers which are divisible by 3 and 5(between M and N), then add those numbers and display. |
| 5. | Write a java program that will count the number of primes between 37 and 129. |
| 6. | Write a java program that will compute GCD of two numbers. |
| 7. | Write a Java program that will evaluate the following exponential series. |
| 8. | Write a method that takes a number between 0 and 9 as an argument and prints on the screen “Number green bottles standing on the wall” (where Number is replaced by the word version of the number). If the numeric parameter is 1 then your method should print bottle rather than bottles. |
| 9. | If one looks at the square numbers and the differences between each adjacent pair of square numbers a pattern emerges.  squares: 0 1 4 9 16 25 36 49… differences: 1 3 5 7 9 11 13… Write a program to print out the square numbers, ten in a line, *without* using multiplication operator. |
| 10\*. | The body mass index (BMI) of a person is defined as the ratio of body mass in kilograms to the square of body height in meter. For example, a person of 1.8m tall with weight 55kg has BMI 55/(1.8)2 = 16.975308641975307. A person with BMI between 20 and 25 is considered to have a healthy amount of body fat. A person with BMI of less than 20 is regarded as underweight, and one with a BMI of more than 25 is regarded as overweight. Write a program to calculate the body mass index (BMI) of the user. Ask the user for their height and their mass. Allow the user to enter both in inches and meters, and both in pounds and kilograms. Also output the meaning of the BMI. Note that 1kg = 2.2046lb, and 1inch = 0.0254 meter.  **Sample input/output:**  What is your weight? **55**  In lb or kg? (1=lb, 2=kg) **2**  What is your height? **1.8**  In in or m? (1=in, 2=m) **2**  Your BMI is 16.975308641975307  You are underweight. |
| 11\*. | Write a method called ***absDiff***, which takes two integers as parameters and returns the absolute value of the difference between the two integers. Using the method ***absDiff*** write a method called ***intRoot***, which takes an integer as a parameter and returns the integer value closest to the square root of the parameter. Given a sequence where the ith term is the closest integer to the square root of i, write a method called ***printSequence***, which takes an integer n as a parameter and prints on the screen the first n integers in the sequence, ten in a line. You may use previously written and inbuilt methods. So ***printSequence***(10) would print on the screen:  1 1 2 2 2 2 3 3 3 3 |
| 12\*. | Write a method ***smallestFactor*** that accept an integer argument, and return the smallest factor larger than 1 of this number. It should return the argument itself if it is a prime or it is 1, and return 0 if it is not positive. By calling the method ***smallestFactor***, write the main method of a program that read a number from the user and completely factorize it. |

**\*: Optional**

**Assignment 2**

**Topic:** Array

|  |  |
| --- | --- |
| **Sl. No.** | **Question** |
| 1. | Write a java program that will search a number from an array of 10 numbers. |
| 2. | Write a java program that will sort an array of N inputted numbers. |
| 3. | Write a java program that will read array of N numbers, and it will remove the duplicate numbers (if any) and display the array. |
| 4. | A secret code encrypts a message by putting it in an array and reading down the columns (blanks are replaced by asterisks and full stops are added to fill up the array). Write a program that encrypts an input string.  LETS\*G  O\*TO\*T  HE\*SAN  DWICH\*  SHOP\*T  ODAY..  **Sample input/output:**  **Input**: “LETS GO TO THE SANDWICH SHOP TODAY”  **Output**: “LOHDSOE\*EWHDTT\*IOASOSCPY\*\*AH\*.GTN\*T.” |
| 5\*. | Sudoku, originally called Number Place, is a logic-based, combinatorial number placement puzzle. The objective is to fill a 9×9 grid with digits so that each column, each row, and each of the nine 3×3 sub-grids that compose the grid (also called "boxes", "blocks", "regions", or "sub-squares") contains all of the digits from 1 to 9. The puzzle setter provides a partially completed grid, which typically has a unique solution. Completed puzzles are always a type of Latin square with an additional constraint on the contents of individual regions. For example, the same single integer may not appear twice in the same 9×9 playing board row or column or in any of the nine 3×3 sub-regions of the 9×9 playing board. The puzzle was popularized in 1986 by the Japanese puzzle company Nikoli, under the name Sudoku, meaning single number. It became an international hit in 2005.  Write a java program to model the Sodoku game. Create a 9 × 9 matrix inside main. Write a function (or functions) to check whether the solution is as per the rule. |

**\*: Optional**

**Assignment 3**

**Topic:** Class & Objects

|  |  |
| --- | --- |
| **Sl. No.** | **Question** |
| 1. | Define a class *Stack*, which perform the basic operation of stack. Define another driver class to demonstrate the basic operations. |
| 2. | Develop a java program that will deal with employee information of an organization.  Define a class *Employee*. Minimum number of data member and member function are as follows:  **Data members:***empName*, *empNo*, *basicSal*, *da*, *hra*, *grossSal*  **Methods:***calGrossSal()*, *showEmpDetails()*  You are free to add more number of relevant data member and member function.Define parameterized constructor to intilaize*empName*, *empNo* and *basicSal*. Create anytwo objects of Employee class and initialize their data members while object creation. Use the method *calGrossSal()*to calculate the gross salary and method *showEmpDetails()* to display the detail informationof the employees in a tabular manner. Note that *da* is 20% of *basicSal* and *hra* is 10% of *basicSal*. *grossSal* is the sum of *basicSal*, *da* and *hra.* |
| 3. | Create a class named *Item* that holds data about an item in a retail store.   * The class should have the following three fields:   1. *name*: the name field is a String object that holds the name of the item.   2. *price*: the price field is a double variable that holds the item's retail price   3. *quantity*: the quantity field is an int variable that holds the number of units currently in inventory * Write a public constructor method that accepts three arguments, name, price, & quantity and stores the values of the arguments passed into it in the object's instance fields. * Write four public methods to retrieve the values from the three fields and their current inventory value   1. *String getName( )* returns the item name   2. *double getPrice( )* returns the price of the item   3. *intgetQuantity( )* returns the number of quantities   4. *double getValue( )* that returns the current inventory value (quantity \* price) * Write a separate class called *Inventory* with a main method that creates three Item objects and then produces a neatly formatted table of the store's inventory displaying the three items, their current inventory value, and the total inventory value for the store. * Duplicate the format of the output exactly shown below. Test your output with different items in inventory. |
| 4. | Declare a variable *Student* which consists of a student’s*name*, *markfor Programming*, *mark for Logic* and a *grade* for Lab. A mark is a number (between 0 and100) and a grade is a letter (between A and F). Write a predicate (a boolean method) *isStronger*, which takes two students and returns true if and only if the first student has done better than the second in the ordering below.   1. the Programming mark is most important, 2. numerical order of Logic marks is the determining factor when two students have the same Programming mark, 3. alphabetical order of Lab grades is the determining factor when two students have the same Programming and Maths marks. |
| 5. | Suppose we want to represent a planet in the solar system as an object of class *Planet*. In the object, it contains two fields: one called *name* is a String containing its name, and one called *satellites* is an array of String containing the names of all its satellites. Define the class, with a constructor taking a String and an array of String as argument for initializing the object.  Write a method print() in the class to print out the name of the planet and all the satellites. For example, if we have  String[] earth\_sat = { "Moon" };  Planet earth = new Planet("Earth", earth\_sat);  Then we want earth.print(); to print out the following:  Earth has 1 satellite(s):  Moon. |

**Assignment 4**

**Topic:** Array of objects, static, nested class, command line argument

|  |  |
| --- | --- |
| **Sl. No.** | **Question** |
| 1. | Define a class Employee with the following members:  **Data members:** private String *empName*  private String *empNo*  private int*dependentCnt*  **Methods:***Employee(String name, String eno, intdepcnt*): constructor  void *showEmpDetails()*:displays*empNo*and*empName*  int*depCount()*: returns *dependentCnt*  Write a separate class called *EmpTest* with a main method thatdefine an array of *n* employees where the value of *n* will be inputted from the user. Read and store the information of all *n* employees. Display the details of the employees with more than two dependents. |
| 2. | Create a class *Account* having data members *accNo*, *balance*, *timePeriod* and *intInYears*(as static and initialize with **7.5%**). The class should also contain the following methods:   * float*calculateInterst*() which calculates and returns the interest amount. * void*showAccDetails*() which displays account number, balance and calculated interest amount. * staticvoid*changeIntRate*(float *newRate*) which changes the interest rate to *newRate*.   Create an array of object of the class *Account*. Store the details of each object through the parameterized constructor. Display all the account details by calling the method *showAccDetails*().Change the interest rate to a new one by calling the method *changeIntRate*(). Finally display the account details after the change in interest rate. |
| 3. | Create a class *Student* having data members *name*, *roll* and *address*. Note that *address* is an object of inner class *Address* having data members city and pin. Create some student class objects. Read, store (using constructor) and display their information. While creating the objects your program should display a message “Creating student number n” from the constructor. You can get the value of *n* by using a static member of Student class which is initialized to 0. |
| 4. | Write a program that will take two integer numbers from the command prompt and find their GCD and LCM. If the user does not provide exactly two numbers of arguments then the program should display error message. |

**Assignment 5**

**Topic:** Strings

|  |  |
| --- | --- |
| **Sl. No.** | **Question** |
| 1. | Write a JAVA program to reverse the following string containing days of a week.  Suppose, String days = "Monday Tuesday Wednesday Thursday Friday Saturday Sunday";  Using the methods of String class manipulate the string *days* in such a way that when we print the string *days* it should display Sunday Saturday Friday Thursday Wednesday Tuesday Monday |
| 2. | Write a JAVA program to remove multiple spaces in a string.  **Sample Output:** If the given string is String With Multiple Spaces  The output should be String With Multiple Spaces |
| 3. | Write a JAVA program to sort an array of names (strings) as per alphabetical order.  **Sample Output:** If the array of names contains Ram John Harish Anand Ajay  After sorting the output should be Ajay Anand John Harish Ram |
| 4. | Write a JAVA program to find all substrings of a given string.  **Sample Output:**  Enter a string to find its sub-strings: shore  Sub-strings of the string "shore" are  s  sh  sho  shor  shore  h  ho  hor  hore  o  or  ore  r  re  e |
| 5. | Write a JAVA program to read a string from the key board and print its reverse.  **Sample Output:**  Enter a string: shore  Reverse of shore is erohs |
| 6. | Write a JAVA program to read a string and print the count of occurrence of each character of the string.  **Sample Output:**  Enter the String: impossible  The Character i has occurred 2 times  The Character m has occurred 1 times  The Character p has occurred 1 times  The Character o has occurred 1 times  The Character s has occurred 2 times  The Character b has occurred 1 times  The Character l has occurred 1 times  The Character e has occurred 1 times |
| 7. | Write a JAVA program to capitalize first letter of each word in an inputted sentence from keyboard.  **Sample Output:**  Enter a sentence: india is great  India Is Great |
| 8. | Write a JAVA program to check two strings are Anagram or not.  **Sample Output:**  Enter the 2 strings to check Anagram :  Debit card  Bad credit  The input strings are Anagram |

**Assignment 6**

**Topic:** Inheritance& Polymorphism

|  |  |
| --- | --- |
| **Sl. No.** | **Question** |
| 1. | |  |  | | --- | --- | |  | Create a class **Point2D** with the data member and methods shown in the class diagram. Note that the items with a minus sign (-) indicate private members and items with a plus sign (+) indicate public members. Create a subclass called **Point3D** which is derived from the superclass **Point2D**.Test the methods of both the classes by creating objects in the main method of another class. | |
| 2. | Create a derived class **Circle** inherited from the class **Point2D** (created in the previous question) with the data member and methods shown in the following class diagram. Create a subclass called **Cylinder** which is derived from the superclass Circle. Test the methods of Circle and Cylinder classes by creating objects in the main method of another class. |
| 3. | |  |  | | --- | --- | |  | We are required to model students and teachers in an application. We can define a superclass called **Person** to store common properties such as name and address, and subclasses **Student** and **Teacher** for their specific properties. For students, we need to maintain the courses taken and their respective grades; *add a course with grade*, *print all courses taken* and *the average grade*. Assume that a student takes no more than **6** courses for the entire program. For teachers, we need to maintain the courses taught currently, and able to add or remove a course taught. Assume that a teacher teaches not more than **5** courses concurrently.  Test the methods of both the derived classes by creating objects of the derived classes in the main method of another class. | |
| 4. | |  |  | | --- | --- | |  | Write a superclass called **Shape** (as shown in the class diagram), which contains:   * Two instance variables **color** (String) and **filled** (boolean). * Two constructors: a no-argument constructor that initializes the color to "green" and filled to true, and a constructor that initializes the color and filled to the given values. * Getter and setter for all the instance variables. By convention, the getter for a boolean variable xxx is called isXXX() (instead of getXxx() for all the other types). * A toString() method that returns "A Shape with color of xxx and filled/Not filled".   Write a test program to test all the methods defined in Shape.  Write two subclasses of Shape called **Circle** and **Rectangle**, as shown in the class diagram.  The **Circle** class contains:   * An instance variable **radius** (double). * Three constructors as shown. The no-arg constructor initializes the radius to 1.0. * Getter and setter for the instance variable radius. * Methods **getArea**() and **getPerimeter**(). * Override the **toString**() method inherited, to return "A Circle with radius=xxx, which is a subclass of yyy", where yyy is the output of the toString() method from the superclass. | | The **Rectangle** class contains:   * Two instance variables **width** (double) and **length** (double). * Three constructors as shown. The no-arg constructor initializes the width and length to 1.0. * Getter and setter for all the instance variables. * Methods **getArea**() and **getPerimeter**(). * Override the toString() method inherited, to return "A Rectangle with width=xxx and length=zzz, which is a subclass of yyy", where yyy is the output of the toString() method from the superclass.   Write a class called **Square**, as a subclass of Rectangle. Square has no instance variable, but inherits the instance variables width and length from its superclass Rectangle.   * Provide the appropriate constructors (as shown in the class diagram).   Hint: public Square(double side) {  super(side, side); // Call superclass Rectangle(double, double)  }   * Override the **toString**() method to return "A Square with side=xxx, which is a subclass of yyy", where yyy is the output of the toString() method from the superclass. * Override the **setLength**() and **setWidth**() to change both the width and length, so as to maintain the square geometry. | | |
| 5. | In this exercise, Shape shall be defined as an abstract class, which contains:   * Two protected instance variables color(String) and filled(boolean). * The protected variables can be accessed by its subclasses and classes in the same package. They are denoted with a '#' sign in the class diagram. * Getter and setter for all the instance variables, and toString(). * Two abstract methods getArea() and getPerimeter() (shown in italics in the class diagram). * The Subclasses Circle and Rectangle shall *override* the abstract methods getArea() and getPerimeter() and provide the proper implementation. They also *override* the toString().   Write a test class to test these statements involving polymorphism and explain the outputs. Some statements may trigger compilation errors. Explain the errors, if any.  Shape s1 = new Circle(5.5, "RED", false); // Upcast Circle to Shape  System.out.println(s1); // which version?  System.out.println(s1.getArea()); // which version?  System.out.println(s1.getPerimeter()); // which version?  System.out.println(s1.getColor());  System.out.println(s1.isFilled());  System.out.println(s1.getRadius());  Circle c1 = (Circle)s1; // Downcast back to Circle  System.out.println(c1);  System.out.println(c1.getArea());  System.out.println(c1.getPerimeter());  System.out.println(c1.getColor());  System.out.println(c1.isFilled());  System.out.println(c1.getRadius());  Shape s2 = new Shape();  Shape s3 = new Rectangle(1.0, 2.0, "RED", false); // Upcast  System.out.println(s3);  System.out.println(s3.getArea());  System.out.println(s3.getPerimeter());  System.out.println(s3.getColor());  System.out.println(s3.getLength());  Rectangle r1 = (Rectangle)s3; // downcast  System.out.println(r1);  System.out.println(r1.getArea());  System.out.println(r1.getColor());  System.out.println(r1.getLength());  Shape s4 = new Square(6.6); // Upcast  System.out.println(s4);  System.out.println(s4.getArea());  System.out.println(s4.getColor());  System.out.println(s4.getSide());  // Take note that we downcast Shape s4 to Rectangle,  // which is a superclass of Square, instead of Square  Rectangle r2 = (Rectangle)s4;  System.out.println(r2);  System.out.println(r2.getArea());  System.out.println(r2.getColor());  System.out.println(r2.getSide());  System.out.println(r2.getLength());  // Downcast Rectangle r2 to Square  Square sq1 = (Square)r2;  System.out.println(sq1);  System.out.println(sq1.getArea());  System.out.println(sq1.getColor());  System.out.println(sq1.getSide());  System.out.println(sq1.getLength()); |

**Assignment 7**

**Topic:** Packages

|  |  |
| --- | --- |
| **Sl. No.** | **Question** |
| 1. | Create a Package ***btech*** which has one class ***Student***. Accept student detail through parameterized constructor of ***Student*** class. Write a method ***display*** ()to display the student details. Create another class ***Test*** containing the main method which will use the package ***btech*** and calculate total marks and percentage of marks. One sample output is shown below. |
| 2. | [Create a sub-package called ***arithmetic***under the package ***btech***. The ***arithmetic*** package should contain a class ***MyMath*** having methods to deal with different arithmetic operations (addition, subtraction, multiplication, division and mod). Create a class ***Test*** containing the main method which will use the methods of sub-package ***arithmetic***.](http://www.smartclass.co/2011/09/create-package-called-arithmetic-that.html) |
| 3. | Create a sub-package named ***shapes***under a package ***org***. Create some classes in the package representing some common geometric shapes like ***Square***, ***Triangle***, ***Circle*** and so on. The classes should contain the ***area*( )** and ***perimeter*( )** methods in them. Compile the package. Use this package to find area and perimeter of different shapes as chosen by the user. |
| 4. | Run the programs under section 7.5 (Access Protection) present in page 82 of class note to test the visibility of class members in subclasses in the same package, non-subclasses in the same package, subclasses in different packages, classes that are neither in the same package nor subclasses. Uncomment the commented lines, test and analyse the output. |

**Assignment 8**

**Topic:** Interfaces

|  |  |
| --- | --- |
| **Sl. No.** | **Question** |
| 1. | Suppose that we have a set of objects with some common behaviours: they could move up, down, left or right. The exact behaviours (such as how to move and how far to move) depend on the objects themselves. One common way to model these common behaviors is to define an *interface* called **Movable**, with abstract methods **moveUp()**, **moveDown()**, **moveLeft()**and **moveRight()**. The classes that implement the Movable interface will provide actual implementation to these abstract methods.  Write two concrete classes - **MovablePoint** and **MovableCircle** - that implement the **Movable** interface.    The code for the interface Movable is as follows:  public interface **Movable** { // saved as "Movable.java"  public void moveUp();  ......  }  For the **MovablePoint** class, declare the instance variable **x**, **y**, **xSpeed** and **ySpeed** with package access as shown with '~' in the class diagram (i.e., classes in the same package can access these variables directly). For the **MovableCircle** class, use a **MovablePoint** to represent its center (which contains four variable x, y, xSpeed and ySpeed). In other words, the **MovableCircle** composes a **MovablePoint**, and its radius.  public class **MovablePoint implements Movable** { // saved as "MovablePoint.java"  // instance variables  int x, y, xSpeed, ySpeed; // package access  // Constructor  public MovablePoint(int x, int y, intxSpeed, intySpeed) {  this.x = x;  ......  }  ......  // Implement abstract methods declared in the interface Movable  @Override  public void moveUp() {  y -= ySpeed; // y-axis pointing down for 2D graphics  }  ......  }  public class **MovableCircle implements Movable** { // saved as "MovableCircle.java"  // instance variables  private MovablePointcenter; // can use center.x, center.y directly  // because they are package accessible  private int radius;  // Constructor  public MovableCircle(int x, int y, intxSpeed, intySpeed, int radius) {  // Call the MovablePoint's constructor to allocate the center instance.  center = new MovablePoint(x, y, xSpeed, ySpeed);  ......  }  ......  // Implement abstract methods declared in the interface Movable  @Override  public void moveUp() {  center.y -= center.ySpeed;  }  ......  }  Write a test program and try out these statements:  Movable m1 = new MovablePoint(5, 6, 10, 15); // upcast  System.out.println(m1);  m1.moveLeft();  System.out.println(m1);  Movable m2 = new MovableCircle(1, 2, 3, 4, 20); // upcast  System.out.println(m2);  m2.moveRight();  System.out.println(m2); |
| 2. | Write a new class called **MovableRectangle**, which composes two MovablePoints (representing the top-left and bottom-right corners) and implementing the **Movable** Interface. Make sure that the two points has the same speed. |

**Assignment 9**

**Topic:** Exception Handling

|  |  |
| --- | --- |
| **Sl. No.** | **Question** |
| 1. | Write a java program to read two numbers ***a*** and ***b*** and calculate ***a/(a−b)***. The program should check the value of ***a−b***before dividing with a, it should throw an exception if ***a−b*** is zero. In the exception handler the program should display appropriate message to the user. |
| 2. | Write a class called **Account** with the following properties and methods:  Properties: String **name**, int**acc\_no**,double **balance**  Methods: void **deposit**(intamt)  void **withdraw**(intamt),  void **transfer** (Account acc1, Account acc2, intamt)  Assume that an account needs to have a minimum balance of 500. If an attempt is made to withdraw or transfer, which results in balance going below 500, throw auser defined exception called **MinimumBalanceException**. Use throw and throws wherever necessary. |
| 3. | Write a java program to compare names of two persons (first name and last name) and return result as  a. Fully matched if both first and last names are same  b. Same first names if only first name matches  c. Same last names if only last name matches  d. No match otherwise.  e. **NameFormatException** if any of the Person name has any illegal entry i.e.  numberswithin name or name consisting of either one part or three parts. |

**Assignment 10**

**Topic:** Multi-Threading

|  |  |
| --- | --- |
| **Sl. No.** | **Question** |
| 1. | Write a java program that will create one child thread. The child thread has to display all odd numbers between m and n, and the main thread will display all the even numbers between m and n. The sample output is shown below: |
| 2. | Write a java program to create two threads. First thread should find the square of the number, second thread should find the sum of the digits of the squared number. |
| 3. | Write a java program that will compute product of two vector (1D array) using multithreading. The program should read two vectors (of same size) from the user. First thread should multiply the corresponding elements present in the odd index position and second thread should multiply the corresponding elements present in the even index position. Main thread should display the result. |
| 4. | Write a simple Java thread program to compute the sum of *n*natural numbers. The program should read the number of threads m and value of n from the user. Each of the threads should add its share of assigned number to a global variable. When all the threads are done, the global variable should contain the result. The programshould use a Synchronized block to make sure that only one thread is updating the global variable at a given time. |
| 5. | Write a Java thread program to search the minimum number in a given array. The program should read the number of elements in the array, number of threads to be created and the array elements from the user. Each thread should find minimum element in an assigned block of elements and compare to global minimum element. When all the threads are done, the global variable should contain the minimum element. It should use a Synchronized block to make sure that only one thread is updating the global minimum variable at any given time |
| 6. | Write a java program in which main thread should create two child threads (Producer and Consumer). First child thread (Producer) should produce ten random integers between 1 to 100 and the second child thread (Consumer) should check whether the generated number is even or odd. At the end the second child thread (Consumer) should print total number of even numbers received. Both the threads should wait and notify each other wherever necessary. The sample output is shown below: |

**Assignment 11**

**Topic:** Collection Framework

|  |  |
| --- | --- |
| **Sl. No.** | **Question** |
| 1. | Write a Java program to create an ArrayList. The program should able to replace a specified element with a new value. |
| 2. | Write a Java program to create two linked lists containing different colors (as strings) and compare the two linked lists to check whether both containing the same elements or not. |
| 3. | Write a Java program to compare two HashSets and using the common elements create a TreeSet. |
| 4. | Write a Java program to retrieve and remove the element of a tree set which are less than a specified value. |
| 5. | Using Stack write java program to reverse the words of a sentence. |
| 6. | Write a Java program to get the portion of a map whose keys range from a given key to another key. |

**Assignment 12**

**Topic:**AWT, Applet, Event Handling

|  |  |
| --- | --- |
| **Sl. No.** | **Question** |
| 1. | Write a java program to print your name in an applet. In the same applet print “My first applet program” in the status. |
| 2. | Write a java program to print smiling face in an applet. |
| 3. | Write java program to show landing of an aeroplane in an applet. |
| 4. | Write a java program to create three color buttons (with caption “Red”, “Blue” and “Green”) and a text in a frame. Initially the text should be displayed in black color. When the user clicks on a particular color button the text should be changed to that particular color. |
| 5. | Write java program to design a calculator using Frame. |

1. import java.applet.Applet;
2. import java.awt.\*;
4. publicclassSmileyExcextendsApplet{
6. publicvoid paint(Graphics g){
7. g.setColor(Color.yellow);
8. g.fillOval(20,20,150,150);// For face
9. g.setColor(Color.black);
10. g.fillOval(50,60,15,25);// Left Eye
11. g.fillOval(120,60,15,25);// Right Eye
12. int x[]={95,85,106,95};
13. int y[]={85,104,104,85};
14. g.drawPolygon(x, y,4);// Nose
15. g.drawArc(55,95,78,50,0,-180);// Smile
16. g.drawLine(50,126,60,116);// Smile arc1
17. g.drawLine(128,115,139,126);// Smile arc2
18. }
19. }
21. /\* <applet code="SmileyExc.class" width="200" height="200">
22. </applet>
23. \*/

**Assignment 13**

**Topic:** Swings

|  |  |
| --- | --- |
| **Sl. No.** | **Question** |
| 1. | Design an application containing a button named Show. You should add an image of “Silicon” in the button. When we click on the button it should display a message “Welcome to Silicon” as text in a label. |
| 2. | Design an application containing a Textfield and a button. When you type some text in the text field and hit Enter, it should show the text entered in the text field as the message in a dialog. If the field is empty, the OK button should be disabled. |
| 3. | Design an application containing four buttons, a text area and a label. When we click the submit button, the application should count the number of lines entered in the text area. By clicking on the button plain, bold or italic it should apply the design on the text present in the text area. |

https://www.w3resource.com/java-exercises/collection/index.php#arraylist