LAB CYCLE:1

DATE:13/03/23

EXPERIMENT NO:1

WELCOME JAVA

**Aim:** Write a java program to display message “Welcome Java”.

**Algorithm:**

Step 1: Define main class.

Step 2: Write the main method, public static void main(String args[])

Step 3: Inside the main method write the display statement and message using, System.out.println()

Step 4: End of the main class.

**Result:** Program executed successfully and output is verified.

LAB CYCLE:1 DATE:13/03/23

EXPERIMENT NO:2

AREA OF RECTANGLE

**Aim:** Create a class Rectangle with instance variable length and breadth. Define a method setdata for setting values of instance variables and a method getArea to return the area of Rectangle using the class. Find the area of the rectangle using the values length = 12.48 and breadth = 13.

**Algorithm:**

Step 1: Define a class Rectangle with instance variables length and breadth.

Step 2: Define methods setData() and getArea() as member functions of Rectangle

class to set the values of instance variables and to calculate the area

respectively.

Step 3: Create an object to access the members of Rectangle class.

Step 4: Invoke the method setData(12.48,13).

Step 5: Invoke method getArea().

Step 6: Print area of the rectangle.

**Result:** Program executed successfully and output is verified.

LAB CYCLE:1 DATE:13/03/23

EXPERIMENT NO:3

EVEN OR ODD

**Aim:** Write a program to read integer from keyboard and check whether the number is even or odd.

**Algorithm:**

Step 1: Import the Java ‘Scanner’ class.

Step 2: Read an integer (num) from user.

Step 3: If num % 2 == 0 , then print “num is even”.

Step 4: Else, print “num id odd”.

**Result:** Program executed successfully and output is verified.

LAB CYCLE:1 DATE:20/03/23

EXPERIMENT NO:4

CONSTRUCTOR IMPLEMENTATION

**Aim:** Define a class 'product' with data members pcode, pname and price. Create three objects of the class and find the product having the lowest price.

**Algorithm:**

Step 1: Define a class Product with data members pcode,pname,price.

Step 2: Read the details (product code,name,price) of three products from the user.

Step 3: Create three objects for Product class and initialize it with the details read from the user.

Step 4: Compare the prices of three products.

Step 5: Print the product which has the lowest price.

**Result**: Program executed successfully and output is verified.

LAB CYCLE:1 DATE:20/03/23

EXPERIMENT NO:5

MATRIX ADDITION

**Aim:** Write a java program to read two matrices from the console and perform matrix addition.

**Algorithm:**

Step 1: Import the Scanner class.

Step 2: Define the class and read the count of rows and columns (r,c).

Step 3: Declare two matrices and read the values of the matrices,A[r][c],B[]r[c].

Step 4: Declare a new matrix to store the sum of other two matrices,C[r][c].

Step 5: Print matrix C[r][c].

**Result:** Program executed successfully and output is verified.

LAB CYCLE:1 DATE:20/03/23

EXPERIMENT NO:6

COMPLEX NUMBER ADDITION

**Aim:** Write a program to perform complex number addition.

**Algorithm:**

Step 1: Import the Scanner class.

Step 2: Define the class and read the real and imaginary part of two complex numbers.

Step 3: Create two objects to access the complex numbers and temp object to store the sum.

Step 4: Create static method with class type to perform the addition,add().

Step 5: Return temp.

Step 6: Call the class method add().

Step 7: Print the sum.

**Result:** Program executed successfully and output is verified.

LAB CYCLE:1 DATE:20/03/23

EXPERIMENT NO:7

SYMMETRIC MATRIX

**Aim:** Write a java program to read a matrix from the console and check whether it is symmetric or not.

**Algorithm:**

Step 1: Import the Scanner class.

Step 2: Define the class and read the count of rows, rows and columns, cols.

Step 3: Read the values of matrix.

Step 4: Check if number of rows and columns are equal, then read the values of matrix.

Step 5: Check each element with transpose of matrix are equal, then the matrix is symmetric.

Step 6: If not equal then matrix is not symmetric.

**Result:** Program executed successfully and output is verified.

LAB CYCLE:1 DATE:24/03/23

EXPERIMENT NO:8

LEAP YEAR

**Aim:** Write a program to print the leap years within the given range.

**Algorithm:**

Step 1: Import the Scanner class.

Step 2: Define the class and read the start and end range.

Step 3: Check for the leap year condition also for century year.

if(i%4==0||(i%100!=0)&&(i%400==0)) then

print “Leap year”.

Step 4: Print the leap year within the range.

**Result:** Program executed successfully and output is verified.

LAB CYCLE:1 DATE:27/03/23

EXPERIMENT NO:9

NESTED CLASS

**Aim:** Create CPU with attribute price. Create an inner class processor (no of cores, manufacturer) and static nested class RAM (memory, manufacturer). Create an object of CPU and print information of processor and RAM.

**Algorithm:**

Step 1: Import the Scanner class.

Step 2: Define the class CPU with data member price, define an inner class Processor with attributes ncores and manft and define a static class RAM with attributes manf and mm.

Step 3: Create the object for CPU and call the Processor and its methods.

Step 4: Create an object for static class RAM and call its methods .

Step 5: Print the details of Processor and RAM.

**Result:** Program executed successfully and output is verified.

LAB CYCLE:2 DATE:03/04/23

EXPERIMENT NO:10

MARKS AND PERCENTAGE OF A STUDENT

**Aim:** Write a program which accepts the marks of a student into a 1D array from the keyboard. Calculate and display total marks & percentage obtained by the student.

**Algorithm:**

Step 1: Import Scanner class.

Step 2: Read the number of subjects, n.

Step 3: Read the total marks of each subject, max.

Step 4: Read the marks of each subject, mar[i].

Step 5: Check if the mark entered is greater than maximum mark,then

print “Enter the mark less than or equal to max”

Step 6: Cumulate the marks of each subjects, total=total+marks[i].

Step 7: Calculate the grand total of all subject, t=n\*max.

Step 8: Calculate the percentage, per=(total/t)\*100.

Step 9: Print percentage and total.

**Result:** Program executed successfully and output is verified.

LAB CYCLE:2 DATE:03/04/23

EXPERIMENT NO:11

SORT STRINGS

**Aim:** Program to sort strings.

**Algorithm:**

Step 1: Define class declare the string variable to store the temporary values.

Step 2: Read the number of strings to be sorted.

Step 3: Read the strings.

Step 4: Using compareTo() compare the string with next string. If the next string is lower then the previous swap the strings till range.

Step 5: Print sorted strings.

**Result:** Program executed successfully and output is verified.

LAB CYCLE:2 DATE:03/04/23

EXPERIMENT NO:12

SORT CHARACTER OF A STRING

**Aim:** Program to sort character from a string.

**Algorithm:**

Step 1: Define a class.

Step 2: Declare a string variable, temp.

Step 3: Read the string.

Step 4: Initialize a character array strarr[].

Then convert temp to character using toCharArray() method.

Step 5: Sort array using Arrays.sort()

Step 6: Print the new string.

**Result:** Program executed successfully and output is verified.

LAB CYCLE:2 DATE:03/04/23

EXPERIMENT NO:13

SEARCH ELEMENT IN AN ARRAY

**Aim:** Program to search an element in an array.

**Algorithm:**

Step 1: Import the scanner class.

Step 2: Read the limit of the array.

Step 3: Declare an array.

Step 4: Read the array items.

Step 5: Read the item to be searched.

Step 6: Visit each element in array if it matches the item, print “Item is found”.

If not matched print “Item not found”.

Step 7: Print item with position.

**Result:** Program executed successfully and output is verified.

LAB CYCLE:2 DATE:10/04/23

EXPERIMENT NO:14

STRING MANIPULATION

**Aim:** Perform string manipulation (using Built-in methods of String Class and StringBuffer Class).

**Algorithm:**

Step 1: Read two strings str1 and str2.

Step 2: a) Concatenate 2 strings using concat().

b) Find the length of string using length().

c) Convert the string to uppercase using toUpperCase().

d) Replace a character by other using replace().

e) Convert a string to character array using toCharArray().

f) Find the index position of a substring using indexOf().

Step 3: String manipulation using StringBuffer class.

Step 4: a) Concatenate strings using append().

Step 5: b) Find the length of string using length().

c) Read the position to insert a substring using insert().

Step 6: Print the resultant string.

**Result:** Program executed successfully and output is verified.

LAB CYCLE:2 DATE:10/04/23

EXPERIMENT NO:15

ARRAY OF OBJECTS

**Aim:** Program to create a class for Employee having attributes eno, ename,esal. Read n employ information and search for an employee given eno using the concept of array of objects.

**Algorithm:**

Step 1: Define a class Employee with attributes eno, ename, esal .

Step 2: Initialize the attributes by constructors.

Step 3: Define main class.

Step 4: Define number of employees, n ,then create array of objects for class

Employee[n].

Employee[] obj = new Employee[n];

Step 5: Read the values of employee.

Step 6: Display employee details.

Step 7: Read the eno to be searched ,item.

Step 8: Check if the obj[i].eno == item, then

print “employee not found.

else

“Not found”.

**Result:** Program executed successfully and output is verified.

LAB CYCLE:3 DATE:17/04/23

EXPERIMENT NO:16

METHOD OVERLOADING

**Aim:** Using the concept of method overloading find the area of different shapes

rectangle, circle and square.

**Algorithm:**

Step 1: Define a class with three methods having same name with different arguments.

Step 2: Define each method with required data members and store the calculated result for area of rectangle, circle and square.

Step 3: Read the measurements for the shapes inside the main function.

Step 4: Create the object for class and invoke the methods to display the areas of shapes.

**Result:** Program executed successfully and output is verified.

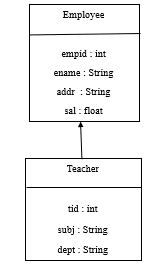
LAB CYCLE:3 DATE:02/06/23

EXPERIMENT NO:17

SIMPLE INHERITANCE

**Aim:** Create a class ‘Employee’ with data members Empid, Name, Salary, Address and constructors to initialize the data members. Create another class ‘Teacher’ that inherit the properties of class employee and contain its own data members department, Subjects taught and constructors to initialize these data members and also include display function to display all the data members. Use array of objects to display details of N teachers.

**Class Diagram:**

****

**Algorithm:**

Step 1: Create a class Employee with data members empid, employee name, salary, address with a constructor.

Step 2: Create a class Teacher inherited from Employee, also access the data members in Employee.

Step 3: Include methods to display the details in Teacher.

Step 4: Create the array of objects and read the details of employee and teacher.

Step 5: Call the method by object and display details.

**Result:** Program executed successfully and output is verified.

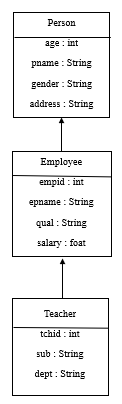
LAB CYCLE:3 DATE:05/06/23

EXPERIMENT NO:18

MULTILEVEL INHERITANCE

**Aim:** Create a class ‘Person’ with data members Name, Gender, Address, Age and a constructor to initialize the data members and another class ‘Employee’ that inherits the properties of class Person and also contains its own data members like Empid, Company name, Qualification, Salary and its own constructor. Create another class ‘Teacher’ that inherits the properties of class Employee and contains its own data members like Subject, Department,Teacher id and also contain constructors and methods to display the data members. Use array of objects to display details of N teachers.

**Class Diagram:**



**Algorithm:**

Step 1: Create a class Person with attributes name, age, gender, address with constructor.

Step 2: Create class Employee inherited from Person with attributes initialize values with constructors.

Step 3: Create class Teacher inherited from Employee with attributes initialize values with constructors.

Step 4: Create a method to display details.

Step 5: Create array of objects and read values.

Step 6: Call the method and display details.

**Result:** Program executed successfully and output is verified.

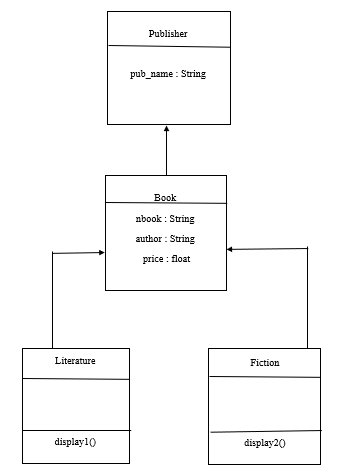
LAB CYCLE:3 DATE:12/06/23

EXPERIMENT NO:19

HIERARCHIAL INHERITANCE

**Aim**: Write a program has class Publisher, Book, Literature and Fiction. Read the information and print the details of books from either the category, using inheritance.

**Class Diagram:**

****

**Algorithm:**

Step 1: Create class Publisher with the attributes.

Step 2: Create class Book inherited from Publisher with its attribute.

Step 3: Create class Literature Fiction inherited from book with its attributes. Also include method to display details of literature and fiction.

Step 4: Create objects for subclasses and display details.

**Result:** Program executed successfully and output is verified.

LAB CYCLE:3 DATE:12/06/23

EXPERIMENT NO:20

MULTIPLE INHERITANCE

**Aim:** Create classes Student and Sports. Create another class Result inherited from Student and Sports. Display the academic and sports score of a student.

**Algorithm:**

Step 1: Create two interface Sports and Student with method.

Step 2: Create class Result implemented from interfaces. Include data members and methods to display details.

Step 3: Read the values for result.

Step 4: Create object display the details.

**Result**: Program executed successfully and output is verified.

LAB CYCLE:3 DATE:19/06/23

EXPERIMENT NO:21

INTERFACE HAVING PROTOTYPE

**Aim**: Create an interface having prototypes of functions area() and perimeter(). Create two classes Circle and Rectangle which implements the above interface. Create a menu driven program to find area and perimeter of objects.

**Algorithm**:

Step 1: Create an interface with a prototype method.

Step 2: Create class Circle, Rectangle implemented from interface.

Step 3: Give the definition of methods in subclasses.

Step 4: Read the values and create the objects.

Step 5: Use a selection case to display details.

**Result**: Program executed successfully and output is verified.

LAB CYCLE:3 DATE:03/07/23

EXPERIMENT NO:22

CREATE A BILL

**Aim :** Prepare bill with the given format using calculate method from interface.

Order No.

Date :

Product Id Name Quantity unit price Total

* - - - -
* - - - -

Net. Amount

**Algorithm:**

Step 1: Import Simple Date Format.

Step 2: Create an inference with a prototype method.

Step 3: Implement a class from interface with data members id , quatity , order.no , price , total price with constructors.

Step 4: Define the function method , create an object for the Data class. Also create object for Simple Date Format.

Step 5: Initialize the random no. for Random class with upper bound.

Step 6: Set the display format.

Step 7: Read the values of product.

Step 8: Calculate the total and net price.

Step 9: Create the object and call the methods to display details.

**Result**: Program executed successfully and output is verified.

LAB CYCLE:3 DATE:26/06/23

EXPERIMENT NO:23

METHOD OVERRIDING

**Aim**: Using the concept of method overriding, find the area of shapes Rectangle, Circle and Square.

**Algorithm**:

Step 1: Create class Dimen to hold two values with its constructor.

Step 2: Inherit shapes like square, rectangles and circle.And define the function method with same name override in all shapes.

Step 3: Create object for different shape .Read the values.

Step 4: Display details of methods.

**Result**: Program executed successfully and output is verified.

LAB CYCLE:3 DATE:26/06/23

EXPERIMENT NO:24

ABSTRACT CLASS

**Aim**: Create an Abstract Class ‘Shape’ with an abstract method find Area to find the area of different shapes. Create subclasses Rectangle, Circle and Square from Shape. Calculate and display area of Rectangle, Circle and Square.

**Algorithm**:

Step 1: Define an abstract class Shape with abstract method area(). [Use “abstract” keyword].

Step 2: Define three sub classes Rectangle, Circle and Square of class Shape.

Step 3: Each class should define the abstract method area() in such a way to calculate the area of different shapes.

a) area() of class Rectangle finds the area of rectangle.[area=length\*breadth]

b) area() of class Circle finds the area of circle.[area=3.14\*rad\*rad]

c) area() of class Square finds the area of square.[area=a\*a]

Step 4: Read the parameters of the different shapes and invoke the method area() of corresponding classes.

Step 5: Display the results.

**Result**: Program executed successfully and output is verified.