Assignment No.:1 Topic: User defined class and objects

Objective:

To get familiar with fundamental OOP concept of Encapsulation that binds together the data and functions that manipulate the data, and that keeps both safe from outside interference and misuse. Data encapsulation led to the important OOP concept of data hiding.

To understand the concept of encapsulation, overloading constructors, implicit and explicit parameters and apply it to define classes and create objects.

Problem Statement:

1. Write a Java Program to print Hello World.

2. Create a class Employee having emp\_id,name,salary,designation.

Use constructor overloading for designing 3 types of employees--

i) Freshers (name should be user given and other parameters should be fixed)

ii)Executive (Name , salary and designation should be user given)

iii)Temporary member(All the parameter having fixed values)

Create an employee array and create and store 3 Employee objects and print the detail.

**Solution:**

import java.io.\*;

public class Employee

{

final String name; //To store Employee Name

String designation; //To store Employee designation

int emp\_id; //To store Employee ID

static int id=1; //To store Employee count

double salary; //To store Employee salary

//Constructor Overloading

Employee() //Initialising Temporary Employee object

{

name="Temporary";

designation="Temporary";

emp\_id=id++;

salary=10000;

}

Employee(String n) //Initialising Freshers Employee object

{

name=n;

salary=25000;

designation="Freshers";

emp\_id=id++;

}

Employee(String n, double sal) //Initialising Executive Employee object

{

name=n;

designation="Executive";

emp\_id=id++;

salary=sal;

}

void display() //To display the Employee details

{

System.out.println("\nName : "+this.name);

System.out.println("Designation : "+this.designation);

System.out.println("Id. No : "+this.emp\_id);

System.out.println("Salary : Rs. "+this.salary);

}

public static void main(String args[])throws IOException

{

String n;

double p,s;

BufferedReader in = new BufferedReader(new InputStreamReader(System.in));

Employee e[] = new Employee[3];

e[0] = new Employee();

System.out.println("Enter the name of the Freshers:");

n=in.readLine();

e[1] = new Employee(n);

System.out.println("Enter the name of the Executive and initial salary:");

n=in.readLine();

s=Double.parseDouble(in.readLine());

e[2] = new Employee(n,s);

System.out.println("\n \*\*\*\*\*\*\*Employee Details\*\*\*\*\*\*\* \n ");

e[0].display();

e[1].display();

e[2].display();

}

}

Assignment No.:2 Topic: I/O operations using API classes

Assignment No.:2 Topic: I/O operations using API classes

Not completed: Assignment No.:2 Topic: I/O operations using API classes. Select to mark as complete.

Objective:

To remember and apply the functionality provided in the API classes System, Scanner and BufferedReader; and via the command line arguments so as to apply and create relevant operational constructs towards getting input and output from keyboard.

Problem Statement:

1. Write a Java Program to illustrate Scanner, BufferReader and System class for input and output from terminal.

2. Write a Java Program to implement simple calculator using command line arguments.

Solution:

//To design a calculator using command line arguments.

public class Calculator

{

//To add 2 numbers.

public static void add(int x, int y)

{

int result =x+y;

System.out.println(x+" + "+y+" = "+result);

}

//To subtract one number from the other.

public static void subtract(int x, int y)

{

int result =x-y;

System.out.println(x+" - "+y+" = "+result);

}

//To multiply 2 numbers

public static void multiply(int x, int y)

{

int result =x\*y;

System.out.println(x+" \* "+y+" = "+result);

}

//To divide one number by the other.

public static void divide(int x, int y)

{

int result =x/y;

System.out.println(x+" / "+y+" = "+result);

}

public static void main(String args[])

{

int x = Integer.parseInt(args[0]);

int y = Integer.parseInt(args[1]);

//Calling all the static functions to calculate the results.

add(x,y);

subtract(x,y);

multiply(x,y);

divide(x,y);

}

}

Assignment No.:3 Topic: Parameter passing, static fields and methods

Assignment No.:3 Topic: Parameter passing, static fields and methods

Not completed: Assignment No.:3 Topic: Parameter passing, static fields and methods. Select to mark as complete.

Objective:

To understand the concept of passing primitive and object parameters as arguments to a function and return them from a function; furthering to apply, analyze and create programming constructs to justify why java uses pass by value only. In this assignment we also engage to understand the concept of static fields and methods and make appropriate use so as to construct, differentiate and actualize.

Problem Statement:

1. (a) In class Employee created in Assignment 1 add static instance field emp\_count.

Use emp\_count for getting emp\_id in proper sequence. Also include appropiate

accessor method for the instance field emp\_count

(b) Write a static method in Employee class.

(c) Write methods to compare two Employees based upon their salary and return object having higher salary.

(d) Write two overloading methods in your Employee class.

2. Write a java program to illustrate - “Java uses pass by value”

Solution:

import java.io.\*;

public class Employee

{

final String name; //To store Employee name

String designation; //To store Employee designation

int emp\_id; //To store Employee ID

static int emp\_count=1; //To store Employee Count

double salary; //To store Employee salary

Employee() //To store the details of Temporary Employee

{

name="Temporary";

designation="Temporary";

emp\_id=emp\_count++;

salary=10000;

}

Employee(String n) //To store the details of Freshers Employee

{

name=n;

salary=25000;

designation="Freshers";

emp\_id=emp\_count++;

}

Employee(String n, double sal) //To store the details of Executive Employee

{

name=n;

designation="Executive";

emp\_id=emp\_count++;

salary=sal;

}

void inc() //To increment the salary of Temporary Employee

{

salary+=2000;

}

void inc(double p,double b) //To increment the salary of Executive Employee

{

salary+=(salary\*p/100.0)+b;

}

void inc(double b) //To increment the salary of Freshers Employee

{

salary+=b;

}

void display() //To display the details of an Employee

{

System.out.println("\nName : "+this.name);

System.out.println("Designation : "+this.designation);

System.out.println("Id. No : "+this.emp\_id);

System.out.println("Salary : Rs. "+this.salary);

}

Employee compare(Employee ob) //To compare the salries of the Employee

{

if(this.salary>ob.salary)

{

System.out.println(" Higher salary is recieved by "+this.name+".");

return this;

}

else if( ob.salary>this.salary)

{

System.out.println(" Higher salary is recieved by "+ob.name+".");

return ob;

}

else

{

System.out.println(" Both have the same salary!!!");

return null;

}

}

public static void main(String args[])throws IOException

{

String n;

double p,s;

BufferedReader in = new BufferedReader(new InputStreamReader(System.in));

Employee e[] = new Employee[3];

e[0] = new Employee();

System.out.println("Enter the name of the Freshers:");

n=in.readLine();

e[1] = new Employee(n);

System.out.println("Enter the name of the Executive and initial salary:");

n=in.readLine();

s=Double.parseDouble(in.readLine());

e[2] = new Employee(n,s);

System.out.println("\n \*\*\*\*\*\*\*Initial Values\*\*\*\*\*\*\* \n \n");

e[0].display();

e[1].display();

e[2].display();

e[0].inc();

System.out.println("Enter the BONUS of the Freshers:");

s=Double.parseDouble(in.readLine());

e[1].inc(s);

System.out.println("Enter the PERCENTAGE & BONUS of the Freshers:");

p=Double.parseDouble(in.readLine());

s=Double.parseDouble(in.readLine());

e[2].inc(p,s);

System.out.println("\n \*\*\*\*\*\*\*After Increment\*\*\*\*\*\*\* \n \n");

e[0].display();

e[1].display();

e[2].display();

Employee c;

System.out.println("\n Comparison between "+e[0].name+" and "+e[1].name+" :");

c=e[0].compare(e[1]);

System.out.println("\n Comparison between "+e[1].name+" and "+e[2].name+" :");

c=e[1].compare(e[2]);

System.out.println("\n Comparison between "+e[2].name+" and "+e[0].name+" :");

c=e[2].compare(e[0]);

}

}

public class Swap\_No

{

//In the swap() method, formal parameters x & y receive their values by pass by values.

public static void swap(int x, int y)

{

x = x+y;

y = x-y;

x = x-y;

System.out.println("\nInside Swap Method:\nx=" +x+"\ny="+y);

}

public static void main(String[] args)

{

int x = 10;

int y = 20;

System.out.println("\nInitial values:\nx=" +x+"\ny="+y);

swap(x, y);

System.out.println("\nAfter returning to the main() method:\nx=" +x+"\ny="+y);

}

}

Assignment No.:4 Topic: String class, StringBuffer class

Assignment No.:4 Topic: String class, StringBuffer class

Not completed: Assignment No.:4 Topic: String class, StringBuffer class. Select to mark as complete.

Objective:

To analyze and understand the appropriate use of the methods provided in the String and StringBuffer class and apply those programming constructs to create objects of those classes.

Problem Statement:

1. Write a java program to illustrate following String API methods.

charAt() , compareTo(), equals(), equalsIgnoreCase(), indexOf(), length() , substring(), toCharArray() , toLowerCase(), toString(), toUpperCase() , trim() , valueOf()

2. Write a java program to illustrate following StringBuffer API methods.

append(), capacity(), charAt(), delete(), deleteCharAt(), ensureCapacity(), getChars(),

indexOf(), insert(), length(), setCharAt(), setLength(), substring(), toString() methods),

3. Write a java program for explaining the concept of mutable and immutable string.

Solution:

//To explain the concepts of mutable and immutable Strings.

public class Mut\_Immut

{

//This function tries to add a string "Added" to the original String object.

static void change(String s)

{

s = s + " Added";

}

//This function appends a string "Added" to the original StringBuffer object.

static void change(StringBuffer s)

{

s.append(" Added");

}

public static void main(String[] args)

{

StringBuffer sb = new StringBuffer("Value");

String str = "Value";

System.out.println("Original StringBuffer: "+sb);

System.out.println("Original String: "+str+"\n\nAfter Executing change function:--");

change(sb);

change(str);

System.out.println("StringBuffer: "+sb); //Displays that StringBuffer object is mutable.

System.out.println("String: "+str); //Displays that String object is immutable.

}

}

//To illustrate the use of various String API methods.

import java.io.\*;

import java.util.Date;

class StringApi

{

public static void main(String args[])throws IOException

{

BufferedReader in = new BufferedReader(new InputStreamReader(System.in));

System.out.println("Enter the text:");

String s=in.readLine();//To create a new String object.

System.out.println("Operations on String:");

System.out.println(s+" charAt 3 is: "+s.charAt(3));

System.out.println("String "+s+" compareTo \"computer\" is: "+s.compareTo("computer"));

System.out.println("String "+s+" equals \"computer\" is: "+s.equals("computer"));

System.out.println("String "+s+" equalsIgnoreCase \"computer\" is: "+s.equalsIgnoreCase("computer"));

System.out.println("String "+s+" indexof first t is: "+s.indexOf('t'));

System.out.println("String "+s+" length is: "+s.length());

System.out.println("String "+s+" substring(0,3) is: "+s.substring(0,3));

System.out.println("String "+s+" toUpperCase is: "+s.toUpperCase());

System.out.println("String "+s+" toLowerCase is: "+s.toLowerCase());

System.out.println("String "+s+" trim remove spaces from ends: "+s.trim());

Date date=new Date();

System.out.println("Current Date toString(): "+date.toString());

Double days=Double.valueOf("7");

System.out.println("Double days = Double.valueOf(\"7\") \nDisplaying days="+days);

}

}

Assignment No.:5 Topic: Packages, Access Specifiers

Assignment No.:5 Topic: Packages, Access Specifiers

Not completed: Assignment No.:5 Topic: Packages, Access Specifiers. Select to mark as complete.

Objective:

To understand and analyze the concept of packages and create packages. In this assignment we also focus on understanding the concept of access specifiers and analyze and actualize them in further details in view of the concept of packages.

Problem Statement:

1. Java program to implement the concept of importing classes from user defined package and creating packages.

2. Write a java program to explain the use of access specifiers - Public,Protected, Default, Private

package Academics;

import College.Students; //Importing the Students class present in the package College.

public class Bppimt //Bppimt class is defined inside the Academics package.

{

public static void main(String args[])

{

double a=10.0,b=20.0,c=30.0;

System.out.println("Subjects:\nSubject 1 : "+a+"\nSubject 2 : "+b+"\nSubject 3 : "+c);

Students s = new Students();

s.average(a,b,c); //Calling the average() method present in the imported package.

s.maxMarks(a,b,c);//Calling the maxMarks() method present in the imported package.

}

}

package College;

public class Students //Students class is defined inside the College package.

{

//To calculate the average marks of the students.

public void average(double s1,double s2,double s3)

{

System.out.println("\nAverage : "+((s1+s2+s3)/3));

return;

}

//To determine the Maximum marks obtained.

public void maxMarks(double s1,double s2,double s3)

{

double max=(s1>s2)?((s1>s3)?s1:s3):((s2>s3)?s2:s3);

System.out.println("\nMaximum marks : "+max);

}

Assignment No.:6 Topic: Inheritance, Polymorphism

Assignment No.:6 Topic: Inheritance, Polymorphism

Not completed: Assignment No.:6 Topic: Inheritance, Polymorphism. Select to mark as complete.

Objective:

To understand the concept of inheritance and construct programming constructs to analyze inheritance and understand and analyze how polymorphism can be appropriately used and to be actualized both at compile time and run time.

Problem Statement:

1. Create a class Shape having atmost two dimensions. Define two subclasses circle and rectangle of Shape.

i) Override a method area.

ii)Show compile time and run time polymorphism(Dynamic method dispatch).

iii)Use a final method for display.

iv) use super keyword.

Solution:

//To implement Polymorphism and Inheritance

class Shape //Parent class inherited by Circle and Rectangle

{

protected int x;

protected int y;

protected double area;

//Constructor Overloading

Shape()

{}

Shape(int x)

{

this.x=x;

}

Shape(int x,int y)

{

this.x=x;

this.y=y;

}

void area()

{

System.out.println("We are inside the Shape class");

}

final void display() //This method cannot be overridden due to the final keyword.

{

System.out.println("Displaying final method : "+area);

}

}

class Circle extends Shape //Circle is the child class of the Shape

{

Circle(int a)

{

super(a); //Calls the constructor of the parent class

}

void area() //Method Overriding

{

System.out.println("Area of circle having radius "+x+" = "+(3.14\*x\*x));

area=3.14\*x\*x;

}

}

class Rectangle extends Shape //Rectangle is the child class of the Shape

{

Rectangle(int a,int b)

{

super(a,b); //Calls the constructor of the parent class

}

void area() //Method Overriding

{

System.out.println("Area of Rectange having length "+x+" ,breadth "+y+" = "+(x\*y));

area=x\*y;

}

}

//To test the program

class Test

{

public static void main(String args[])

{

Shape p=new Shape();

Shape circle=new Circle(3);

Shape rect=new Rectangle(4,5);

p.area();

p.display();

circle.area();

circle.display();

rect.area();

rect.display();

}

}

Assignment No.:7 Topic: Interface, Abstract class

Assignment No.:7 Topic: Interface, Abstract class

Not completed: Assignment No.:7 Topic: Interface, Abstract class. Select to mark as complete.

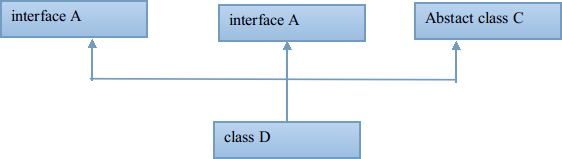
Objective:

To understand and analyze the concept of interfaces and abstract classes and actualize them

Problem Statement:

1. Define a abstract class figure. Define the area and volume method in the child classes. Use dynamic method dispatch.

Implenent the following design with suitable example classes.

**Solution:**

**public interface Area**

**{**

**public void area();**

**}**

**abstract class Figure**

**{**

**public double ar,vol;**

**public abstract void area(); //Abstract method Area**

**public abstract void volume(); //Abstract method Volume**

**public Figure()**

**{**

**ar=0.0;**

**vol=0.0;**

**}**

**}**

**public interface Volume**

**{**

**public void volume();**

**}**

**public class Sphere extends Figure**

**{**

**private double radius;**

**public Sphere(double r)**

**{**

**super();**

**radius=r;**

**}**

**public void area() //Defining the abstract method area()**

**{**

**ar=4\*3.14\*this.radius\*this.radius;**

**System.out.println("Area:"+ar);**

**}**

**public void volume() //Defining the abstract method volume()**

**{**

**vol=4\*3.14\*this.radius\*this.radius\*this.radius/3.0;**

**System.out.println("Volume:"+vol);**

**}**

**public static void main(String args[])**

**{**

**Figure s = new Sphere(10.0);**

**System.out.println("For radius = 10.0");**

**s.area();**

**s.volume();**

**}**

**}**

**Assignment No.:8 Topic: Array of objects, Wrapper class**

**Assignment No.:8 Topic: Array of objects, Wrapper class**

**Not completed: Assignment No.:8 Topic: Array of objects, Wrapper class. Select to mark as complete.**

**Objective:**

**To understand the concept of constructing array of objects and actualize. In this assignment we also analyze the appropriate use of wrapper class and actualize.**

**Problem Statement:**

**1. Create a Student class having roll no. , name, dept., marks. Use array of objects to store details of 5 students. List the name of the student a) having highest marks b) lowest marks c) Marks more than avarage.**

**2. Use wrapper class for explaining autoboxing and unboxing.**

**Solution:**

**// To create a Student class having student details using array of objects**

**import java.util.\*;**

**public class Student**

**{**

**private int marks,roll;**

**private String name,dept;**

**Student(int r,int m,String n,String d) //Parameterized constructor to initialize the Student type objects.**

**{**

**roll=r;**

**marks=m;**

**name=n;**

**dept=d;**

**}**

**static void max(Student[] s) //To find the student who obtained Maximum marks.**

**{**

**int i;**

**int max=0;**

**int a=-1;**

**for(i=0;i<5;i++)**

**{**

**if(max<s[i].marks)**

**{**

**max=s[i].marks;**

**a=i;**

**}**

**}**

**System.out.println("Maximum marks: "+s[a].name);**

**}**

**static void min(Student[] s) //To find the student who obtained Minimum marks.**

**{**

**int i;**

**int min=99;**

**int a=-1;**

**for(i=0;i<5;i++)**

**{**

**if(min>s[i].marks)**

**{**

**min=s[i].marks;**

**a=i;**

**}**

**}**

**System.out.println("Minimum marks: "+s[a].name);**

**}**

**static void avg(Student[] s) //To determine the average marks & to find the students who obtained Above Average marks.**

**{**

**int i;**

**int avg;**

**int total=0;**

**for(i=0;i<5;i++)**

**{**

**total=total+s[i].marks;**

**}**

**avg=total/5;**

**System.out.println("Average marks:"+avg);**

**System.out.println("Marks greater than average:");**

**for(i=0;i<5;i++)**

**{**

**if(avg<s[i].marks)**

**System.out.println(s[i].name);**

**}**

**}**

**public static void main(String[] args)**

**{**

**int i;**

**Student[] s= new Student[5]; //Creating an array of 5 Student type objects.**

**Scanner in= new Scanner(System.in);**

**int m,r;**

**String n,d;**

**for(i=0;i<5;i++) //Accepting the details of the 5 students and storing them in the array of objects.**

**{**

**System.out.println("Enter details of student: "+(i+1));**

**System.out.print("Enter roll: ");**

**r=Integer.parseInt(in.nextLine());**

**System.out.print("Enter marks: ");**

**m=Integer.parseInt(in.nextLine());**

**System.out.print("Enter name: ");**

**n=in.nextLine();**

**System.out.print("Enter department: ");**

**d=in.nextLine();**

**s[i]=new Student(r,m,n,d);**

**}**

**max(s); //To find the student who obtained Maximum marks.**

**min(s); //To find the student who obtained Minimum marks.**

**avg(s); //To find the students who obtained Above Average marks.**

**}**

**}**

**//To use wrapper class for explaining autoboxing and unboxing.**

**public class Wrapper**

**{**

**//In this program we are using the Integer Wrapper Class and int primitive data type.**

**public static void main(String args[])**

**{**

**//Converting int into Integer**

**int a=10;**

**Integer i=Integer.valueOf(a);//converting int into Integer explicitly.**

**Integer j=a;//autoboxing, since the compiler will write Integer.valueOf(a) internally.**

**System.out.println("a="+a+" i="+i+" j="+j);**

**//Converting Integer to int**

**Integer b=new Integer(20);**

**int x=b.intValue();//converting Integer to int explicitly.**

**int y=b;//unboxing, since the compiler will write a.intValue() internally.**

**System.out.println("b="+b+" x="+x+" y="+y);**

**}**

**}**

**Assignment No.:9 Topic: Exception Handling**

**Assignment No.:9 Topic: Exception handling**

**Not completed: Assignment No.:9 Topic: Exception handling . Select to mark as complete.**

**Objective:**

**To understand the concept of exceptions and make appropriate uses of the system defined exceptions and/or create user defined exceptions and actualize them.**

**Problem Statement:**

**1. Write a program to implement the concept of Exception Handling using predefined exception.**

**2. Write a program to implement the concept of Exception Handling by creating user defined exceptions.**

**NOTE: Use throws, throw, try, catch and finally keywords in your program.**

**Solution:**

**//To implement the concept of Exception Handling using predefined exception.**

**public class PredefinedExceptionHandler**

**{**

**public static void printSubString(String s, int a, int b) throws NullPointerException**

**{**

**String str = s.substring(a, b); //Throws NullPointerException in this line**

**System.out.println(str); //The control doesn't reach this line.**

**}**

**public static void main(String args[])**

**{**

**try**

**{**

**printSubString(null, 1, 5);**

**}**

**catch (NullPointerException npe)**

**{**

**System.out.println("Exception caught!");**

**}**

**finally**

**{**

**System.out.println("Final Block!"); //Always executed, even if the exception isn't caught**

**}**

**}**

**}**

**//To implement the concept of Exception Handling by creating user defined exceptions.**

**class UserDefinedException extends Exception //UserDefinedException class inherits class Exception**

**{**

**UserDefinedException(String s)**

**{**

**super(s); //Calls constructor of parent class Exception**

**}**

**}**

**public class UserDefinedExceptionHandler**

**{**

**public static int divide(int a, int b) throws UserDefinedException**

**{**

**if(b == 0)//Divided by zero exception occurs**

**{**

**throw new UserDefinedException("Can't divide by 0!");**

**}**

**else**

**{**

**return a / b;**

**}**

**}**

**public static void main(String args[])**

**{**

**try**

**{**

**int x = divide(4, 0);**

**System.out.println(x); //The control doesn't reach this line.**

**}**

**catch (UserDefinedException ude)//Catches the UserDefinedException type object thrown explicitly.**

**{**

**System.out.println("Exception caught!");**

**}**

**finally**

**{**

**System.out.println("Final Block!"); //Always executed, even if the exception isn't caught.**

**}**

**}**

**}**

**Assignment No.:10 Topic: Applet using awt and swing**

**Assignment No.:10 Topic: Applet using awt and swing**

**Not completed: Assignment No.:10 Topic: Applet using awt and swing. Select to mark as complete.**

**Objective:**

**To understand the concepts of graphics programming using awt and swing and appropriately use them to achieve the actualization of awt and swing applets.**

**Problem Statement:**

**1. Java program for printing your name ,roll no,year of admission, Dept. and section by using Applet. Configure the Applets by passing parameters.**

**2. Use getDocumentBase() and getCodeBase() methods in the Applet.**

**3. Write a program for key event handling.**

**Solution:**

**/\***

**\* To change this license header, choose License Headers in Project Properties.**

**\* To change this template file, choose Tools | Templates**

**\* and open the template in the editor.**

**\*/**

**package keyEventPackage;**

**import java.applet.Applet;**

**import java.awt.Graphics;**

**import java.awt.event.KeyListener;**

**import java.awt.event.KeyEvent;**

**/\*\***

**\***

**\* @author User**

**\*/**

**public class AppletKeyEvent extends Applet implements KeyListener**

**{**

**/\*\***

**\* Initialization method that will be called after the applet is loaded into**

**\* the browser.**

**\*/**

**char ch;**

**String str;**

**public void init() // link the KeyListener with Applet**

**{**

**addKeyListener(this);**

**}**

**// override all the 3 abstract methods of KeyListener interface**

**public void keyPressed(KeyEvent e) { }**

**public void keyReleased(KeyEvent e) { }**

**public void keyTyped(KeyEvent e)**

**{**

**ch = e.getKeyChar();**

**if(ch == 'a')**

**str = "a for apple";**

**else if(ch == 'e')**

**str = "e for ear";**

**else if(ch == 'i')**

**str = "i for ice";**

**else if(ch == 'o')**

**str = "o for ox";**

**else if(ch == 'u')**

**str = "u for umbrella";**

**else**

**str = "Type only vowels a, e, i, o, u only";**

**repaint();**

**}**

**public void paint(Graphics g)**

**{**

**g.drawString(str, 100, 150);**

**showStatus("You typed " + ch + " character");**

**}**

**// TODO overwrite start(), stop() and destroy() methods**

**}**

**import java.applet.Applet;**

**import java.awt.\*;**

**public class MyApplet extends Applet {**

**/\*\***

**\* Initialization method that will be called after the applet is loaded into**

**\* the browser.**

**\*/**

**private String defaultMessage = "Hello!";**

**public void paint(Graphics g) {**

**String s1 =this.getParameter("name");**

**String s2 =this.getParameter("roll");**

**String s3 =this.getParameter("year");**

**String s4 =this.getParameter("dept");**

**String s5 ="Section - B";**

**String codebase="Codebase: " + getCodeBase().toString();**

**String documentbase="Documentbase: " + getDocumentBase().toString();**

**if(s1 == null)**

**{**

**s1=defaultMessage;g.drawString(s1, 50, 25);**

**}**

**else**

**{**

**g.drawString(s1, 50, 25);**

**g.drawString(s2, 50, 45);**

**g.drawString(s3, 50, 65);**

**g.drawString(s4, 50, 85);**

**g.drawString(s5, 50, 105);**

**g.drawString(codebase, 50, 125);**

**g.drawString(documentbase, 50, 145);**

**}**

**}**

**// TODO overwrite start(), stop() and destroy() methods**

**}**

**Assignment No.:11 Topic: Multithreading extending Thread class**

**Assignment No.:11 Topic: Multithreading extending Thread class**

**Not completed: Assignment No.:11 Topic: Multithreading extending Thread class. Select to mark as complete.**

**Objective:**

**To understand the concept of multithreading and actualize concurrency by appropriately using concepts of inheritance and methods overriding with regards to the Thread class.**

**Problem Statement:**

**Write a program to create multiple threads by using thread class.**

**Solution:**

**//To create multiple threads by using Thread class.**

**class First\_Thread extends Thread //The first thread 'first' object is created of this type in main() method.**

**{**

**public void run()**

**{**

**for (int i = 0; i < 10; i++)**

**{**

**System.out.println(i);**

**try**

**{**

**Thread.sleep(200);**

**}**

**catch(InterruptedException ie)**

**{**

**System.out.println("Thread has been interrupted!");**

**}**

**}**

**}**

**}**

**class Second\_Thread extends Thread //The second thread 'second' object is created of this type in main() method.**

**{**

**public void run()**

**{**

**for (int i = 0; i < 20; i = i + 2)**

**{**

**System.out.println(i);**

**try**

**{**

**Thread.sleep(100);**

**}**

**catch(InterruptedException ie)**

**{**

**System.out.println("Thread has been interrupted!");**

**}**

**}**

**}**

**}**

**class MultiThread**

**{**

**public static void main(String args[])**

**{**

**First\_Thread first = new First\_Thread(); // First\_Thread type object is created.**

**Second\_Thread second = new Second\_Thread(); // Second\_Thread type object is created.**

**first.start(); //first thread starts its execution at this point.**

**second.start(); //second thread starts its execution at this point.**

**}**

**}**

**Assignment No.:12 Topic: Multithreading implementing Runnable Interface**

**Assignment No.:12 Topic: Multithreading implementing Runnable Interface**

**Not completed: Assignment No.:12 Topic: Multithreading implementing Runnable Interface. Select to mark as complete.**

**Objective:**

**To understand the concept of multithreading and actualize concurrency by appropriately using concepts of interface and implementing them with regards to the Runnable Interface.**

**Problem Statement:**

**1) Write a program to create multiple threads by implementing Runnable interface.**

**2) Use join(), isAlive(), getPriority(), SetPriority() methods.**

**Solution:**

**//To create multiple threads by implementing Runnable interface.**

**import java.lang.\*;**

**class MyRunnable1 implements Runnable**

**{**

**public void run()**

**{**

**System.out.println("\nPriority of thread1 : " + Thread.currentThread().getPriority());**

**for (int i = 1; i <= 10; i++)**

**{**

**System.out.println(i);**

**try**

**{**

**Thread.sleep(200);**

**}**

**catch(InterruptedException ie)**

**{**

**System.out.println("Thread interrupted!");**

**}**

**}**

**}**

**}**

**class MyRunnable2 implements Runnable**

**{**

**public void run()**

**{**

**System.out.println("\nPriority of thread2 : " + Thread.currentThread().getPriority());**

**for (int i = 1; i <= 19; i = i + 2)**

**{**

**System.out.println(i);**

**try**

**{**

**Thread.sleep(300);**

**}**

**catch(InterruptedException ie)**

**{**

**System.out.println("Thread interrupted!");**

**}**

**}**

**}**

**}**

**class MyRunnable3 implements Runnable**

**{**

**public void run() {**

**System.out.println("\nPriority of thread3 : " + Thread.currentThread().getPriority());**

**for (int i = 1; i <= 28; i = i + 3)**

**{**

**System.out.println(i);**

**try**

**{**

**Thread.sleep(400);**

**}**

**catch(InterruptedException ie)**

**{**

**System.out.println("Thread interrupted!");**

**}**

**}**

**}**

**}**

**public class Runnable\_Interface**

**{**

**public static void main(String[]args)**

**{**

**Thread thread1 = new Thread(new MyRunnable1());**

**Thread thread2 = new Thread(new MyRunnable2());**

**Thread thread3 = new Thread(new MyRunnable3());**

**thread1.setPriority(Thread.MIN\_PRIORITY);**

**thread2.setPriority(Thread.NORM\_PRIORITY);**

**thread3.setPriority(Thread.MAX\_PRIORITY);**

**thread1.start(); //thread1 execution starts at this point.**

**System.out.println("Thread 1 is alive ? " + thread1.isAlive());**

**System.out.println("Thread 2 is alive ? " + thread2.isAlive());**

**System.out.println("Thread 2 is alive ? " + thread3.isAlive());**

**try**

**{**

**thread1.join(); //waiting for thread1 to finish**

**}**

**catch(InterruptedException ie)**

**{**

**System.out.println("Thread interrupted!");**

**}**

**thread2.start(); //thread2 execution starts at this point.**

**System.out.println("Thread 1 is alive ? " + thread1.isAlive());**

**System.out.println("Thread 2 is alive ? " + thread2.isAlive());**

**System.out.println("Thread 2 is alive ? " + thread3.isAlive());**

**try**

**{**

**thread2.join(); //waiting for thread2 to finish**

**}**

**catch(InterruptedException ie)**

**{**

**System.out.println("Thread interrupted!");**

**}**

**thread3.start(); //thread3 execution starts at this point.**

**System.out.println("Thread 1 is alive ? " + thread1.isAlive());**

**System.out.println("Thread 2 is alive ? " + thread2.isAlive());**

**System.out.println("Thread 2 is alive ? " + thread3.isAlive());**

**try**

**{**

**thread3.join(); //waiting for thread3 to finish**

**}**

**catch(InterruptedException ie)**

**{**

**System.out.println("Thread interrupted!");**

**}**

**}**

**}**

**Assignment No.:13 Topic: Beyond Syllabus**

**Assignment No.:13 Topic: Beyond Syllabus**

**Not completed: Assignment No.:13 Topic: Beyond Syllabus. Select to mark as complete.**

**Objective:**

**To interpret the appropriate use of an example API class from order of its fields and methods actions and subsequently construct object(s) and put them to action so as to compose them together towards fulfilling the desired solution to the given problem statement.**

**Problem Statement:**

**Write a program that uses the GregorianCalendar class to display a calendar for the current month, like this:**

**Sun Mon Tue Wed Thu Fri Sat**

**1 2 3 4 5 6**

**7 8 9 10 11 12 13**

**14 15 16 17 18 19 20**

**21 22 23 24 25 26 27**

**28 29\* 30 31**

**The current date should be marked with \***

**Solution:**

**// To write a program that uses the GregorianCalendar class to display a calendar for the current month with a '\*' indicating the current date.**

**import java.util.\*;**

**import java.text.\*;**

**class G\_C**

**{**

**public static void main(String args[])**

**{**

**// To declare an object of inbuilt GregorianCalender class**

**GregorianCalendar d=new GregorianCalendar();**

**//calling the methods of GregorianCalendar class to get system's current date**

**int today = d.get(Calendar.DAY\_OF\_MONTH);**

**int year= d.get(Calendar.YEAR);**

**int month= d.get(Calendar.MONTH);//Where January = 0 - December = 11**

**//Declaring an object if GregorianCalendar class of type Calendar class**

**Calendar gc = new GregorianCalendar(year, month, 1);**

**//method returns a Date object that represents this Calendar's time value**

**Date startDate = new Date(gc.getTime().getTime());**

**int startday=startDate.getDay();**

**System.out.println("Sun\tMon\tTues\tWed\tThurs\tFri\tSat");**

**int c=startday,p=0,t=1;**

**int day[]=new int[7];**

**int dates[][]=new int[5][7];**

**//this array represent total number of days in each month starting from Jan to Dec**

**int getdaycount[]={31,28,31,30,31,30,31,31,30,31,30,31};**

**for(int i=0;i<5;i++)**

**for(int j=0;j<7;j++)**

**dates[i][j]=0;**

**for(int i=0;i<7;i++)**

**day[i]=i;**

**while(t<=getdaycount[month])**

**{**

**if(c<7)**

**{**

**dates[p][c]=t++;**

**c++;**

**}**

**else**

**{**

**c=0;**

**p++;**

**}**

**}**

**for(int i=0;i<5;i++)**

**{**

**for(int j=0;j<7;j++)**

**{**

**if(dates[i][j]!=0)**

**{**

**//date array matches with today date then print "\*" along with the day**

**if(dates[i][j]==today)**

**System.out.print(dates[i][j]+"\*"+"\t");**

**else**

**System.out.print(dates[i][j]+"\t");**

**}**

**else**

**System.out.print(" \t");**

**}**

**System.out.println();**

**}**

**}**

**}**