OBJECT ORIENTED PROGRAMMING USING JAVA LAB

Lab 1: String Operations

Aim:
To write a program to perform various String Operations
Description:
String: A string is the sequence of characters enclosed within double quotes. Strings are very useful because most of the data on internet will be in the form of strings only.
For example: name, vehicle number, address, credit card number etc. will come under strings.
The following methods are uses to solve various string method classes and various string buffer classes.

- **charAt(index)**
  Give the charcater at the specified index.
- **compareTo(str2)**
  Compare two strings and returns integer.(case sensitive)
- **compareToIgnoreCase(str2)**
  Compare two strings and returns integer.(not case sensitive)
- **equals(str2)**
  returns true if two strings are equal(case sensitive)
- **equalsIgnoreCase(str2)**
  returns true if two strings are equal(not case sensitive)
- **concat(str3)**
  combines string1 and string2
- **toLowerCase()**
  Convert the given string to lower case.
- **toUpperCase()**
  Convert the given string to lower case.
- **length()**
  returns actual number of characters in the string buffer.
- **trim()**
  remove the space from right and left side of a string.
- **substring(start inex, end index)**
  Returns a substring from the starting index till the cahracter before the end index.
- **replace(character1,character2)**
  replace the character1 by character 2.

1.Program for various string operations

```
import java.util.*;
public class AllStringFuctionExample
	{
		public static void main(String[] args)
	{
		String str = "GM INFORMATICS ";
		String str1= "LATEST JOB NEWS";
		String str2= "latest job news";
		String str3= "UG and PG Previous papers";
		String tempstr = "    String trimming example    ";
System.out.println("Character at the index 6 is   :" + str.charAt(6));
System.out.println("Compare b/w two strings  :" + str1.compareTo(str2));
System.out.println("Compare b/w two strings:" +
str1.compareToIgnoreCase(str2));
System.out.println("Difference b/w two strings   :" + str1.equals(str2));
System.out.println("Difference b/w two strings :" +
str1.equalsIgnoreCase(str2));
System.out.println("Concatenation of  two strings :" + str.concat(str3));
String Lowercase = str.toLowerCase();
System.out.println("Lower case String              :" + Lowercase);
String Uppercase = str.toUpperCase();
System.out.println("Upper case String              : " + Uppercase);
System.out.println("Length of the given string      :" + str.length());
System.out.println("String before trimming          : " + tempstr);
System.out.println("String after trimming           : " + tempstr.trim());
System.out.println("String between index 3 to 9 is :"+ str.substring(3, 9));
System.out.println("Difference between two strings:" +
str1.replace('J','M'));
System.out.println("String after replacement :"+ str.replace("GM", "DM"));
}
}
```

Lab 2. Class and Object

Aim:
Write a program on class and object in java.
Description:
To write a java program to display total marks of 5 students using student class. Use the following attributes HTNO, Name , Marks in 3 subjects Maths, Statistics/Physics, Computers and Total.

2. Program for student Marks list

```
import java.util.*;
class Student
{
	int HTNO,TOTAL;
	String NAME;
	int marks[]=new int[3];
	void getDetails()
	{
		Scanner sc =new Scanner(System.in);
		System.out.println("Enter the Hall ticket Number");
		HTNO=sc.nextInt();
		System.out.println("Enter the name");
		NAME=sc.next();
		System.out.println("Enter Marks in Maths");
		marks[0]=sc.nextInt();
		System.out.println("Enter marks Stat/Physics: ");
		marks[1]=sc.nextInt();
		System.out.println("Enter Marks in Computer science");
		marks[2]=sc.nextInt();
		TOTAL=marks[0]+marks[1]+marks[2];
	}
	void display()
	{
		System.out.println(HTNO +"\t"+NAME + "\t" + marks[0] +"\t"
+marks[1]+ "\t"+marks[2]+"\t"+TOTAL);
	}
}
class Marks
{
```

```
        public static void main(String[] args)
        {
                Student s[]=new Student[5];
                for(int i=0;i<5;i++)
                {
                        s[i]=new Student();
                        s[i].getDetails();
                }
                System.out.println("\t\t\t Marks List");
                System.out.println("****************************************");

        System.out.println("HTNO\tNAME\tMaths\tStat\tComputers\tTotal");
                for(int i=0;i<5;i++)
                        s[i].display();
        }
}
```

3. Method Overloading and Method Overriding

Aim:

Write a program to illustrate Function Overloading & Function Overriding methods in Java

Description:

Method: A function written in a class is called a method. In java we have only methods.

Method Overloading: Having the same name of method with different parameters is method overloading. If we define multiple methods with the same name with different signatures in a class, such concept is known as method overloading. The difference may be number of parameters order of parameters and type of parameters. We implement polymorphism in java using this concept.

Method Overriding: Having same type of methods same signature in the parent and child is nothing but overriding.

Changing the definition of the class method in the sub class is known as method overriding. Method overriding happens when classes are inheritance procedure.Super class method and sub class method signatures should be the same.Their return type should be the same.

Program for Method overloading

```
class Areas
{
	int l,b;
	void setSides(int x)
	{
		l=b=x;
	}
	void setSides(int x,int y)
	{
		l=x;
		b=y;
	}
	int getArea()
	{
		return l*b;
	}
}
class ExMethodOverLoading
{
	public static void main(String args[])
{
	Areas sc1=new Areas();
	sc1.setSides(3);
	Areas sc2=new Areas();
	sc2.setSides(4,5);
	System.out.println("Areas of the square"+sc1.getArea());
	System.out.println("Areas of the rectangle"+sc2.getArea());
}
}
```

Program for Method overriding

```
class RBI
{
	public double getROI(){
		return 10.5;
	}
}
class SBI extends RBI
{
	public double getROI()
		{
			return super.getROI()-0.5;
```

```
            }
        public double calculateInterest(double p,double t)
        {
            return(p*t*getROI())/100;
        }
        public static void main(String args[])
        {
          SBI sbi=new SBI();
           System.out.println("SBI: "+sbi.getROI());
        }
}
```

4. Abstract classes

Aim:

Write a program to illustrate the implementation of abstract class

Description:

Abstract classes:

- These classes are used to provide abstraction in java.
- An abstract class should start with a keyword “abstract”.
- The abstract classes can contain both abstract methods and concrete methods(implemented / non- implemented method)

Program for abstract class

```
abstract class Animal
{
        public abstract String sound();
        public abstract String nature();
        public String walksWith()
        {
            return "By legs";
        }
}
        class Lion extends Animal
        {
            public String sound()
            {
                return "Roar Roar";
            }
            public String nature()
```

```
		{
			return "Wild Animal";
		}
	}
	class  Dog extends Animal
	{
		public String sound()
		{
			return "Bow Bow";
		}
		public String nature()
		{
			return "Domestic Animal";
		}
	}
public class  AnimalTest
	{
	public static void main(String[] args)
	{
		//Animal a=new Animal();
		//walksWith();
		Dog d=new Dog();
		Lion l=new Lion();
		System.out.println("=======Dog behaviour======");
		System.out.println("sounds like "+d.sound());
		System.out.println("Nature "+d.nature());
		System.out.println("Walks with "+d.walksWith());
		System.out.println("=========Lion behaviour=====");
		System.out.println("sounds like "+ l.sound());
		System.out.println("Nature "+l.nature());
		System.out.println("Walks with "+l.walksWith());
	}
}
```

5.Exception Hndling

Aim:

Write a program to implement Exception handling

Description:

Program for exception handling

```
import java.util.*;
class ExException
  {
   int i;
   String fun()throws Exception
     {
     try
       {
       Scanner s1=new Scanner(System.in);
       System.out.println("Enter Amount:");
       i=s1.nextInt();
            if(i<1000)
       throw new Exception();
       }
catch(Exception ee)
           {
             System.out.println("catch block");
             return "insufficient balance";
           }
                    finally
           {
           System.out.println("State Bank Of India");
           }
                    return "sufficient balance";
     }
     public static void main(String args[])
       {
        try
          {
```

```
        ExException obj=new ExException();
        System.out.println(obj.fun());
        }
         catch(Exception ee)
          {
             System.out.println("in main catch");
          }
      }
 }
```

Program For Packages

```
/*Book detail class in Package Book*/
// javac -d . BookDetails.java
package book;
public class BookDetails
{
        String name,author;
        float price;
        int year;
        public BookDetails(String n,String a,float p, int y)
        {
                name=n;
                author=a;
                price=p;
                year=y;
        }
public void display( )
        {
                System.out.println("Book Name   :  "+name);
                System.out.println("Book Author  :  " +author);
                System.out.println("Book Price  :  " +price);
                System.out.println("Year of publishing :  " +year);
        }
}

/* Class that imports book package*/
import book.BookDetails;
class  BookDemo
```

```
{
        public static void main(String[] args)
        {
           BookDetails b=new BookDetails("java programming",
"Bala",190.00f,2007);
                b.display();
        }
}
```

Program for Interface

```
interface AquaticAnimal
        {
                //String LivesIn="water";
                String oceanName();
                String eats();
                boolean isLays();
        }
        class BlueWhale implements AquaticAnimal
        {
                public String oceanName()
                 {
                        return "Pacific";
                  }
                public String eats()
                {
                        return "Fishes and small creatures";
                }
                public boolean isLays()
                {
                        return false;
                }
        }
        class StarTortoise implements AquaticAnimal
                {
                public String oceanName()
                        {
                                return "Bay of bengal";
                        }
```

```
                public String eats()
                        {
                        return "small creatures and under water plants";
                        }
                public boolean isLays()
                        {
                                return true;
                        }
                }
public class  AquaticTest
        {
        public static void main(String[] args)
        {
                BlueWhale whale =new BlueWhale();
                StarTortoise tot=new StarTortoise();
                System.out.println("Bluewhale behaviour");
                System.out.println("eats "+whale.eats());
                System.out.println("Lays eggs "+whale.isLays());
                System.out.println("available in"+whale.oceanName());
                System.out.println("=====================");

                System.out.println("Tortoise behaviour");
                System.out.println("eats "+tot.eats());
                System.out.println("Lays eggs "+tot.isLays());
                System.out.println("available in"+tot.oceanName());
                System.out.println("=====================");
        }
}
```

Program for multiple Threads

```
class MyThreadOne extends Thread
{
        public void run( )
        {
           for(int i=1;i<=20;i++)
           System.out.println("Hello");
        }
 }
```

```
class MyThreadTwo extends Thread
{
        public void run( )
        {
            for(int i=1;i<=20;i++)
                System.out.println("Hai");
        }
}
class MultiThreadingExample
{
        public static void main(String[] args)
        {
                MyThreadOne t1=new MyThreadOne( );
                MyThreadTwo t2= new MyThreadTwo( );
                t1.start();
                t2.start();
                System.out.println("This is Mahesh");
        }
}
```

Program for Multiple Inheritance

```
abstract class Animal
        {
        public abstract String sound();
        public abstract String nature();
        public void walksWith()
                {
                        System.out.println("By legs");
                }
        }
interface AquaticAnimal
                {
                        String LivesIn="water";
                        String oceanName();
                        String eats();
                        boolean isLays();
                }

class BlueWhale extends Animal implements AquaticAnimal
{
```

```
        public String oceanName(){
                return "Pacific";
        }
public String eats(){
        return "Fishes and small creatures";
}
public boolean isLays(){
        return false;
}
        public String sound()
        {
                return "Queek Queek";
        }
        public String nature()
        {
                return "Wild nature";
        }
}
class ExMultipleInheritance
        {
        public static void main(String[] args)
        {
                BlueWhale whale =new BlueWhale();
                StarTortoise tot=new StarTortoise();
                System.out.println("Bluewhale behaviour");
                System.out.println("eats "+whale.eats());
                System.out.println("Lays eggs "+whale.isLays());
                System.out.println("available in"+whale.sound());
                System.out.println("Sounds like"+whale.oceanName());
                System.out.println("Nature"+whale.nature());
                System.out.println("Lives in   "+whale.LivesIn);
                System.out.println("====================");

        }
}
```

Program for Priorities to thread

```
class TestMultiPriority extends Thread{
 public void run(){
   System.out.println("running thread name
is:"+Thread.currentThread().getName());
   System.out.println("running thread priority
is:"+Thread.currentThread().getPriority());

  }
 public static void main(String args[]){
  TestMultiPriority m1=new TestMultiPriority();
  TestMultiPriority m2=new TestMultiPriority();
  m1.setPriority(Thread.MIN_PRIORITY);
  m2.setPriority(Thread.MAX_PRIORITY);
  m1.start();
  m2.start();

 }
}
```