

Lab1:

Program1:

```
import java.util.Scanner;

class MatrixMul {
    public static void main(String args[]) {
        int r1 = Integer.parseInt(args[0]);
        int c1 = Integer.parseInt(args[1]);
        int r2 = Integer.parseInt(args[2]);
        int c2 = Integer.parseInt(args[3]);
        Scanner sc = new Scanner(System.in);
        int[][] a = new int[r1][c1];
        int[][] b = new int[r2][c2];
        int[][] c = new int[r1][c2];
        if ((r1 <= 0) || (r2 <= 0) || (c1 <= 0) || (c2 <= 0)) {
            System.out.println("Invalid dimensions\n");
        }
        else if (c1 != r2) {
            System.out.println("Matrix multiplication not possible\n");
        }
        else {
            System.out.println("Enter elements of first array:");
            for (int i = 0; i < r1; i++) {
                for (int j = 0; j < c1; j++) {
                    a[i][j] = sc.nextInt();
                }
            }
            System.out.println("Enter elements of second array:");
            for (int i = 0; i < r2; i++) {
                for (int j = 0; j < c2; j++) {
                    b[i][j] = sc.nextInt();
                }
            }
            for (int i = 0; i < r1; i++) {
```

```

        for (int j = 0; j < c2; j ++ ) {
            c[i][j] = 0;
            for (int k = 0; k < r2; k ++ ) {
                c[i][j] += a[i][k] * b[k][j];
            }
        }

    }

    System.out.println("The resultant matrix is:");
    for ( int i = 0; i < r1; i ++ ) {
        for (int j = 0; j < c2; j ++ ) {
            System.out.print((c[i][j] + " "));
        }
        System.out.println();
    }
}
}
}

```

```

Portkey@taruni:~/Desktop/javaprogs/lab1$ javac MatrixMul.java
Portkey@taruni:~/Desktop/javaprogs/lab1$ java MatrixMul 0 2 2 1
Invalid dimensions

Portkey@taruni:~/Desktop/javaprogs/lab1$ java MatrixMul 1 2 3 4
Matrix multiplication not possible

Portkey@taruni:~/Desktop/javaprogs/lab1$ java MatrixMul 2 2 2 2
Enter elements of first array:
1
2
3
4
Enter elements of second array:
1
1
1
1
The resultant matrix is:
3 3
7 7
Portkey@taruni:~/Desktop/javaprogs/lab1$

```

Pre lab questions:

1. JVM is an interpreter which is installed in each client machine that is updated with latest security updates by the internet .
2. Java is said to be Robust because of its auto-garbage collection feature. The JVM takes care of security and the lack of pointers in java account for its security.
3. When we compile a java file, the compiler converts the source code into a byte code which is stored in binary form in a .class file. This is called as bytecode.
4. Compiler converts source code into bytecode(.class file) and interpreter(JVM) converts it into machine code at runtime to give the output.
5. Lang package is the default package in java
6. Object
7. No it is not illegal to declare in this way, but memory is not allocating here. After we declare in this we, we have to allocate memory by, `arr = new int[3][3];`
8. `type[] arrayName;` // to declare array
(or) `type arrayName[size];`
They can be initialised by , `arrayName = new type[size]` // to allocate memory to array
9. `parseInt()` method is used to get the primitive data type of a string in Java.
10.

```
public class Debug { //first brace missing
    public static void main(String args[]) { //parenthesis missing
        System.out.println("Hello world"); //quotes missing
        System.out.println("this is my first program"); //s is small, question mark is outside
//the quotes and p is capital
        System.out.println(); //semicolon is missing
        System.out.println("debug it"); //closing parenthesis was missing
    }
}
```

Lab2:

Program1:

```
class Person {
    private String firstName;
    private String lastName;
    private String gender;
    public Person() {
        this.firstName = "firstName";
        this.lastName = "lastName";
        this.gender = "gender";
    }
    public Person( String firstName, String lastName, String gender) {
        this.firstName = firstName;
        this.lastName = lastName;
        this.gender = gender;
    }
    public String getfirstName() {
        return this.firstName;
    }
    public String getlastName() {
        return this.lastName;
    }
    public String getgender() {
        return this.gender;
    }
    public void setfirstName(String firstName) {
        this.firstName = firstName;
    }
    public void setlastName(String lastName) {
        this.lastName = lastName;
    }
    public void setgender(String gender) {
        this.gender = gender;
    }
    public String toString(){
        return firstName + lastName + gender;
    }
}
```

```

}
class Instructor extends Person {
    private int id;
    private String highestQualification;
    private String department;
    public Instructor() {
        super();
        this.id = 0;
        this.highestQualification = "highestQualification";
        this.department = "department";
    }
    public Instructor(int id, String department, String highestQualification) {
        super();
        this.id = id;
        this.highestQualification = highestQualification;
        this.department = department;
    }
    public String getdepartment() {
        return this.department;
    }
    public int getid() {
        return this.id;
    }
    public String gethighestQualification() {
        return this.highestQualification;
    }
    public void setdepartment(String department) {
        this.department = department;
    }
    public void setid(int id) {
        this.id = id;
    }
    public void sethighestQualification(String highestQualification) {
        this.highestQualification = highestQualification;
    }
    public String toString(){
        return Integer.toString(id) + " " + highestQualification + " " + department;
    }
}

```

```

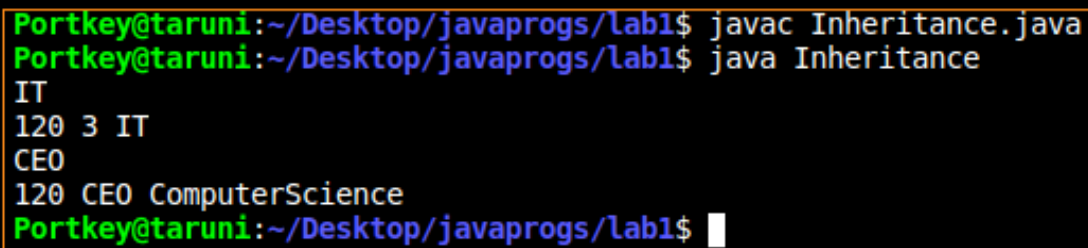
}
class Student extends Person {
    private int rnum;
    private int currSem;
    private String branch;
    public Student() {
        super();
        this.rnum = 0;
        this.currSem = 0;
        this.branch = "branch";
    }
    public Student(int rnum, int currSem, String branch) {
        super();
        this.rnum = rnum;
        this.currSem = currSem;
        this.branch = branch;
    }
    public String getbranch() {
        return this.branch;
    }
    public int getrnum() {
        return this.rnum;
    }
    public int getcurrSem() {
        return this.currSem;
    }
    public void setbranch(String branch) {
        this.branch = branch;
    }
    public void setrnum(int rnum) {
        this.rnum = rnum;
    }
    public void setcurrSem(int currSem) {
        this.currSem = currSem;
    }
    public String toString(){
        return Integer.toString(rnum) + " " + Integer.toString(currSem) + " " + branch;
    }
}

```

```

}
public class Inheritance {
    public static void main(String args[]) {
        Student s1 = new Student();
        Student s2 = new Student(120, 3, "IT");
        s1.setfirstName("abc");
        System.out.println(s2.getbranch());
        System.out.println(s2);
        Instructor i1 = new Instructor();
        Instructor i2 = new Instructor(120, "ComputerScience", "CEO");
        i1.setdepartment("CSE");
        System.out.println(i2.gethighestQualification());
        System.out.println(i2);
    }
}

```



```

Portkey@taruni:~/Desktop/javaprogs/lab1$ javac Inheritance.java
Portkey@taruni:~/Desktop/javaprogs/lab1$ java Inheritance
IT
120 3 IT
CEO
120 CEO ComputerScience
Portkey@taruni:~/Desktop/javaprogs/lab1$ 

```

Prelab Questions:

1. Data hiding can be achieved using encapsulation in Java. This means we can only show the required/necessary attributes and methods to the user and hide the implementation from the user. This also increases the reusability of code.
2. When the static modifier is not used for the main method, the JVM is unable to call the main method as there is no object of the class, and thus raises an error.
3. Direct multiple inheritance is not supported in Java because it raises ambiguity in choosing which method to invoke if the classes have the same method names.
4. Inheritance derives one class from another whereas composition defines a class as a sum of its parts. Inheritance is a is-a relationship whereas composition is a has -a relationship.
5. When superclass and subclass contain the same static methods having the same signatures, the method in the super class will be hidden by the one that is in the sub class. This is called method hiding.
6. `super.i = 10;`
7. Yes, we can because the subclass inherits all the attributes and methods of the superclass and subclass "is -a" superclass.
8. No constructors are not inherited by subclasses in Java. The constructors of base classes are invoked before the subclasses.
9. Yes, static methods can be inherited in Java, but they can not be overridden but the static methods of the base class are hidden.

Lab3:

Program1:

```
import java.util.Scanner;
```

```
class Employee {  
    public String name;  
    public int id;  
    public int wagePerDay;  
    public int noOfDaysWorked;  
    public int hra;  
    public Employee(String name, int id, int wagePerDay, int noOfDaysWorked, int hra) {  
        this.name = name;  
        this.id = id;  
        this.wagePerDay = wagePerDay;  
        this.noOfDaysWorked = noOfDaysWorked;  
        this.hra = hra;  
    }  
    public String getname() {  
        return this.name;  
    }  
    public int getid() {  
        return this.id;  
    }  
    public int getwagePerDay() {
```

```
        return this.wagePerDay;
    }

    public int getnoOfDaysWorked() {
        return this.noOfDaysWorked;
    }

    public int gethra() {
        return this.hra;
    }

    public void setname(String name) {
        this.name = name;
    }

    public void setid(int id) {
        this.id = id;
    }

    public void setwagePerDay(int wagePerDay) {
        this.wagePerDay = wagePerDay;
    }

    public void setnoOfDaysWorked(int noOfDaysWorked) {
        this.noOfDaysWorked = noOfDaysWorked;
    }

    public void sethra(int hra) {
        this.hra = hra;
    }
}
```

```

    public double calculateSalary() {
        return this.wagePerDay * this.noOfDaysWorked;
    }
}

class GeneralManager extends Employee{

    public GeneralManager(String name, int id, int wagePerDay, int noOfDaysWorked, int hra) {
        super(name, id, wagePerDay, noOfDaysWorked, hra);
    }

    public double calculateSalary() {
        return super.calculateSalary() + ( 0.2 * super.hra);
    }
}

public class Sal {

    public static void main(String args[]) {

        Scanner sc = new Scanner(System.in);

        System.out.println("Enter employee name:");

        String name = sc.next();

        System.out.println("Enter employee id:");

        int id = sc.nextInt();

        System.out.println("Enter employee wage per day:");

        int wagePerDay = sc.nextInt();

        System.out.println("Enter number of days worked:");

        int noOfDaysWorked = sc.nextInt();

        System.out.println("Enter employee hra:");
    }
}

```

```

        int hra = sc.nextInt();

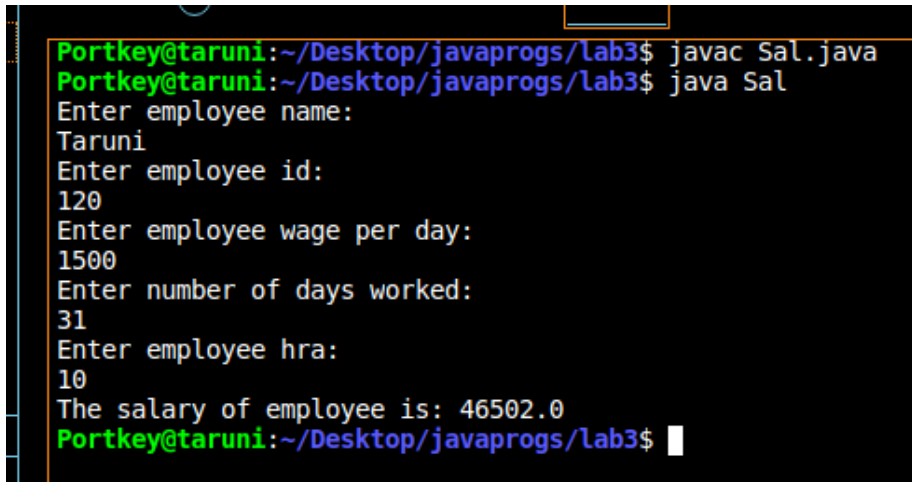
        Employee e = new GeneralManager(name, id, wagePerDay, noOfDaysWorked, hra);

        System.out.println("The salary of employee is: " + e.calculateSalary());

    }

}

```



```

Portkey@taruni:~/Desktop/javaprogs/lab3$ javac Sal.java
Portkey@taruni:~/Desktop/javaprogs/lab3$ java Sal
Enter employee name:
Taruni
Enter employee id:
120
Enter employee wage per day:
1500
Enter number of days worked:
31
Enter employee hra:
10
The salary of employee is: 46502.0
Portkey@taruni:~/Desktop/javaprogs/lab3$

```

Program2:

```

import java.util.Scanner;

abstract class Shape {
    public double area;
    public double perimeter;
    abstract void calcArea();
    abstract void calcPerimeter();
    public void setarea(double area) {
        this.area = area;
    }
    public void setperimeter(double perimeter) {
        this.perimeter = perimeter;
    }
    public double getarea() {
        return this.area;
    }
    public double getperimeter() {
        return this.perimeter;
    }
}

```

```

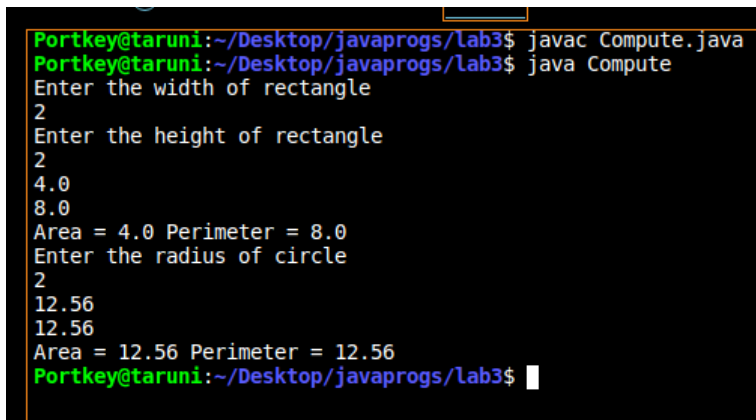
}
class Rectangle extends Shape {
    public double width = 0;
    public double height = 0;
    public Rectangle(double width, double height) {
        this.width = width;
        this.height = height;
    }
    public void setwidth(double width) {
        this.width = width;
    }
    public void setheight(double height) {
        this.height = height;
    }
    public void calcArea() {
        this.area = this.width * this.height;
    }
    public void calcPerimeter() {
        this.perimeter = 2 * (this.width + this.height);
    }
    public String toString() {
        return "Area = " + this.area + " Perimeter = " + this.perimeter;
    }
}
class Circle extends Shape {
    public double radius;
    public double pi = 3.14;
    public Circle(double radius) {
        this.radius = radius;
    }
    public void setradius(double radius) {
        this.radius = radius;
    }
    public void calcArea() {
        this.area = this.pi * this.radius * this.radius;
    }
    public void calcPerimeter() {
        this.perimeter = 2 * this.radius * this.pi;
    }
    public String toString() {
        return "Area = " + this.area + " Perimeter = " + this.perimeter;
    }
}
public class Compute {

```

```

public static void main(String args[]) {
    Scanner sc = new Scanner(System.in);
    System.out.println("Enter the width of rectangle");
    double width = sc.nextDouble();
    System.out.println("Enter the height of rectangle");
    double height = sc.nextDouble();
    Shape r = new Rectangle(width, height);
    r.calcArea();
    r.calcPerimeter();
    System.out.println(r.getarea());
    System.out.println(r.getperimeter());
    System.out.println(r);
    System.out.println("Enter the radius of circle");
    double radius = sc.nextDouble();
    Shape c = new Circle(radius);
    c.calcArea();
    c.calcPerimeter();
    System.out.println(c.getarea());
    System.out.println(c.getperimeter());
    System.out.println(c);
}
}

```



```

Portkey@taruni:~/Desktop/javaprogs/lab3$ javac Compute.java
Portkey@taruni:~/Desktop/javaprogs/lab3$ java Compute
Enter the width of rectangle
2
Enter the height of rectangle
2
4.0
8.0
Area = 4.0 Perimeter = 8.0
Enter the radius of circle
2
12.56
12.56
Area = 12.56 Perimeter = 12.56
Portkey@taruni:~/Desktop/javaprogs/lab3$

```

Pre Lab Questions:

1. No, an abstract class can not be final in java.
2. Yes, it is mandatory for an abstract class to have at least one abstract method.
3. Yes, an abstract class can contain the main method because abstract classes also contain non-abstract methods.
4. Garbage collection in Java is when we remove(free) the memory occupied by an object. It is used when there is no longer the need of memory used. The finalize() method is

invoked when we want to remove/ free the memory (i.e right before the object is destroyed).

5. When the final keyword is used with a method, it cannot be overridden. Final Variables can not change its value. Final Classes can not be extended or inherited.
6. A static method cannot be overridden. The static methods of the sub class only hide those in the base class (with same name and signatures). There is no runtime polymorphism in this, hence it cannot be called as overriding.
7. b
c 1
a 2
c 1
b
c 1
b 2
c 2
c
c 1
c 2
b
d 1
b 2
c 2
b 2
c 2

Lab4:

Program1:

```
package prog1;
interface Arithmetic {
    public double add(double num1, double num2);//abstract methods
    public double subtract(double num1, double num2);//abstract methods
    public double multiply(double num1, double num2);//abstract methods
    public double divide(double num1, double num2);//abstract methods
}
interface Trigonometric {
    public double pi = 3.14;
    public double sin(double x, int n);//abstract methods
    public double cos(double x, int n);//abstract methods
    public double tan(double x, int n);//abstract methods
}
public class Calculator implements Arithmetic, Trigonometric { //implementation class

    public double add(double num1, double num2) { //defining abstract method add
        return num1 + num2 ;
    }
    public double subtract(double num1, double num2) { //defining abstract method subtract
        return num1 - num2;
    }
    public double multiply(double num1, double num2) { //defining abstract method multiply
        return num1 * num2;
    }
    public double divide(double num1, double num2) { //defining abstract method divide
        return num1 / num2;
    }
    public int factorial(int n ) {
        int f = 1;
        if ( n == 0) {
            return 1;
        }
        while ( n > 0) {
            f *= n;
            n -= 1;
        }
        return f;
    }
    public double sin(double x, int n) { //defining abstract method sin
        x = x * pi / 180;
```



```

        double sum = 0;
        int j = 1;
        for (int i = 0; i <= n ; i++) {
            sum += (Math.pow(x, 2 * i + 1) / this.factorial( 2 * i + 1)) * j;
            j = j * (-1);
        }
        return sum;
    }

    public double cos(double x, int n) { //defining abstract method cos
        double sum = 0;
        x = x * pi / 180;
        int j = 1;
        for (int i = 0; i <= n ; i++) {
            sum += (Math.pow(x, 2 * i) / this.factorial( 2 * i)) * j;
            j = j * (-1);
        }
        return sum;
    }

    public double tan(double x, int n) { //defining abstract method tan
        return sin ( x, n) / cos(x, n);
    }
}

```

```

import java.util.Scanner;
import prog1.Calculator;

```

```

public class Mycalculator {
    public static void main(String[] args) {
        Calculator testing = new Calculator();//making an object
        Scanner sc = new Scanner( System.in);
        System.out.println("Testing the calculator implementation class!");
        System.out.println("Enter two numbers: ");
        double num1 = sc.nextDouble();
        double num2 = sc.nextDouble();
        System.out.println("Adding : " + testing.add(num1, num2)); //calling add function
        System.out.println("subtracting : " + testing.subtract(num1, num2)); //calling subtract
function
        System.out.println("Multiplying : " + testing.multiply(num1, num2)); //calling multiply function
        System.out.println("Dividing : " + testing.divide(num1, num2)); //calling divide function
        System.out.println("Enter x value(in degrees): ");
        double x = sc.nextDouble();
        System.out.println("Sin(" + x + ") : " + testing.sin(x, 10)); //calling sin function
        System.out.println("Cos(" + x + ") : " + testing.cos(x, 10)); //calling cos function
        System.out.println("Tan(" + x + ") : " + testing.tan(x, 10)); //calling tan function
    }
}

```

```
}
}
```

```
Portkey@taruni:~/Desktop/javaprogs/Lab4$ javac Mycalculator.java
Portkey@taruni:~/Desktop/javaprogs/Lab4$ java Mycalculator
Testing the calculator implementation class!
Enter two numbers:
3.5 0.5
Adding : 4.0
subtracting : 3.0
Multiplying : 1.75
Dividing : 7.0
Enter x value(in degrees):
30
Sin(30.0) : 0.4997701026430513
Cos(30.0) : 0.8661580944053981
Tan(30.0) : 0.5769964003928572
Portkey@taruni:~/Desktop/javaprogs/Lab4$ java Mycalculator
Testing the calculator implementation class!
Enter two numbers:
-2 3
Adding : 1.0
subtracting : -5.0
Multiplying : -6.0
Dividing : -0.6666666666666666
Enter x value(in degrees):
60
Sin(60.0) : 0.8657598077481052
Cos(60.0) : 0.5004596899295196
Tan(60.0) : 1.7299291534749408
```

```
Portkey@taruni:~/Desktop/javaprogs/Lab4$ java Mycalculator
Testing the calculator implementation class!
Enter two numbers:
0 0
Adding : 0.0
subtracting : 0.0
Multiplying : 0.0
Dividing : NaN
Enter x value(in degrees):
15
Sin(15.0) : 0.258690844053802
Cos(15.0) : 0.9659601685383986
Tan(15.0) : 0.2678069474078098
Portkey@taruni:~/Desktop/javaprogs/Lab4$ java Mycalculator
Testing the calculator implementation class!
Enter two numbers:
12 0.1
Adding : 12.1
subtracting : 11.9
Multiplying : 1.2000000000000002
Dividing : 120.0
Enter x value(in degrees):
45
Sin(45.0) : 0.7068251809540194
Cos(45.0) : 0.7073882691620831
Tan(45.0) : 0.9992039898983189
Portkey@taruni:~/Desktop/javaprogs/Lab4$
```

Program2:

package prog2;

```
public class Person {
    public String firstName, lastName, gender;
    public Person() {
        this.firstName = "firstName";
        this.lastName = "lastName";
        this.gender = "gender";
    }
    public Person(String firstName, String lastName, String gender) {
        this.firstName = firstName;
        this.lastName = lastName;
        this.gender = gender;
    }
    public String toString() {
        return firstName + " " + lastName + " " + gender;
    }
    public void setfirstName(String firstName) { //mutator method
        this.firstName = firstName;
    }
    public void setlastName(String lastName) { //mutator method
        this.lastName = lastName;
    }
    public void setgender(String gender) { //mutator method
        this.gender = gender;
    }
}
```

```

    }
    public String getfirstName() { //accessor method
        return this.firstName;
    }
    public String getlastName() { //accessor method
        return this.lastName;
    }
    public String getgender() { //accessor method
        return this.gender;
    }
}

```

```

package prog2;

```

```

public interface Athlete {
    public String getSport();
    public boolean isAmateur();
}

```

```

import prog2.Person;

```

```

import java.util.Scanner;

```

```

import prog2.Athlete;

```

```

class Student extends Person implements Athlete{
    public int rnum, currSem;
    String branch, sport;
    public Student() {
        this.rnum = 0;
        this.currSem = 0;
        this.branch = "branch";
    }
    public Student(int rnum, int currSem, String sport, String branch) {
        this.rnum = rnum;
        this.currSem = currSem;
        this.branch = branch;
        this.sport = sport;
    }
    public int getrnum() {
        return this.rnum;
    }
    public int getcurrSem() {
        return this.currSem;
    }
}

```

```

    }
    public String getbranch() {
        return this.branch;
    }
    public void setrnum(int rnum) {
        this.rnum = rnum;
    }
    public void setcurrSme(int currSem) {
        this.currSem = currSem;
    }
    public void setbranch(String branch) {
        this.branch = branch;
    }
    public String toString() {
        return firstName + " " + lastName + " " + branch;
    }
    public String getSport() {
        return this.sport;
    };
    public void setSport(String sport) {
        this.sport = sport;
    }
    public boolean isAmateur() {
        return true;
    }
}

public class TestStudent {
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        Student obj = new Student();
        System.out.println("Enter first name");
        String firstName = sc.next();
        obj.setfirstName(firstName);
        System.out.println("Enter last name");
        String lastName = sc.next();
        obj.setlastName(lastName);
        System.out.println("Enter roll number ");
        int rnum = sc.nextInt();
        obj.setrnum(rnum);
        System.out.println("Enter current sem number ");
        int currSem = sc.nextInt();
        obj.setcurrSme(currSem);
        System.out.println("Enter branch number ");
        String branch = sc.next();
    }
}

```

```
    obj.setbranch(branch);  
    System.out.println(obj);  
}  
}
```

```
Portkey@taruni:~/Desktop/javaprogs/lab4$ javac TestStudent.java  
Portkey@taruni:~/Desktop/javaprogs/lab4$ java TestStudent  
Enter first name  
N  
Enter last name  
Taruni  
Portkey@taruni:~/Desktop/javaprogs/lab4$ javac TestStudent.java  
Portkey@taruni:~/Desktop/javaprogs/lab4$ java TestStudent  
Enter first name  
Hermione  
Enter last name  
Granger  
Enter roll number  
1  
Enter current sem number  
7  
Enter branch number  
CSE  
Hermione Granger CSE  
Portkey@taruni:~/Desktop/javaprogs/lab4$ java TestStudent  
Enter first name  
Ronald  
Enter last name  
Weasely  
Enter roll number  
100  
Enter current sem number  
5  
Enter branch number  
IT  
Ronald Weasely IT  
Portkey@taruni:~/Desktop/javaprogs/lab4$ java TestStudent  
Enter first name  
Harry  
Enter last name  
Potter
```

Pre Lab questions:

1. Abstract classes can contain non-abstract methods but interfaces contain only abstract methods.
2. Public class is visible in other packages, field is visible everywhere (class must be public too)
private : Private variables or methods may be used only by an instance of the same class that declares the variable or method, A private feature may only be accessed by the class that owns the feature.
protected : Is available to all classes in the same package and also available to all subclasses of the class that owns the protected feature. This access is provided even to subclasses that reside in a different package from the class that owns the protected feature.
default : What you get by default ie, without any access modifier (ie, public private or protected). It means that it is visible to all within a particular package.
3. Yes, importing a package also imports its sub-packages.
4. Multiple inheritance can be achieved either by a class implementing an interface or by another interface extending an interface.
5. This is the same as having only one method, and these two methods would be undistinguishable.
6. Yes, an abstract class can implement interfaces in Java. No, it does not require you to implement all methods.
7. When we have some non abstract methods along with abstract methods, we prefer abstract classes over interfaces.
8. The previous versions of Java only had abstract methods in interfaces, but Java 8 allows interfaces to have default and static methods also.

Lab5:

Program1:

```
import java.util.Scanner;

class NegativeNumberException extends Exception{
    public int num;
    public NegativeNumberException(int num) {
        this.num = num;
    }
    public String toString() {
        return "NegativeNumberException[" + num + "]\n";
    }
}

class NotAnIntegerException extends Exception{
    public String num;
    public NotAnIntegerException(String num) {
        this.num = num;
    }
    public String toString() {
        return "NotAnIntegerException[" + num + "]\n";
    }
}

class OutOfMemoryException extends Exception{
    public int num;
    public OutOfMemoryException(int num) {
        this.num = num;
    }
    public String toString() {
        return "OutOfMemoryException[" + num + "]\n";
    }
}

public class Factorials {
    static void checkException(String n) throws OutOfMemoryException,
    NegativeNumberException, NotAnIntegerException {
        try {
            int num = Integer.parseInt(n);
            if ( num > 16) {
                throw new OutOfMemoryException(num);
            }
            else if ( num < 0) {
                throw new NegativeNumberException(num);
            }
        }
    }
}
```

```

    }
    catch (NumberFormatException e) {
        throw new NotAnIntegerException(n);
    }
}
static int fact(int n) {
    int f = 1;
    for ( int i = 2; i <= n; i ++ ) {
        f *= i;
    }
    return f;
}
public static void main (String args[]) {
    Scanner sc = new Scanner(System.in);
    int count = 0;
    String n;
    int num = Integer.parseInt(args[0]);
    while ( count < num ) {
        try {
            System.out.print("Enter an integer: ");
            n = sc.next();
            checkException(n);
            System.out.println("Factorial of " + n + " is: " + fact(Integer.parseInt(n)) + "\n");
            count += 1;
        }
        catch (NegativeNumberException err) {
            System.out.println(err);
        }
        catch ( NotAnIntegerException e ) {
            System.out.println(e);
        }
        catch ( OutOfMemoryException e ) {
            System.out.println(e);
        }
    }
}
}

```



```

Portkey@taruni:~/Desktop/javaprogs/lab5$ javac Factorials.java
Portkey@taruni:~/Desktop/javaprogs/lab5$ java Factorials 3
Enter an integer: aaab
NotAnIntegerException[aaab]

Enter an integer: -7
NegativeNumberException[-7]

Enter an integer: 24
OutOfMemoryException[24]

Enter an integer: 0
Factorial of 0 is: 1

Enter an integer: 10
Factorial of 10 is: 3628800

Enter an integer: 12
Factorial of 12 is: 479001600
Portkey@taruni:~/Desktop/javaprogs/lab5$

```

```

Portkey@taruni:~/Desktop/javaprogs/lab5$ javac Factorials.java
Portkey@taruni:~/Desktop/javaprogs/lab5$ java Factorials 4
Enter an integer: -1
NegativeNumberException[-1]

Enter an integer: 4
Factorial of 4 is: 24

Enter an integer: 20
OutOfMemoryException[20]

Enter an integer: name
NotAnIntegerException[name]

Enter an integer: 3
Factorial of 3 is: 6

Enter an integer: 2
Factorial of 2 is: 2

Enter an integer: 5
Factorial of 5 is: 120
Portkey@taruni:~/Desktop/javaprogs/lab5$

```

Pre lab questions:

1. Throw is a keyword which is used to throw an exception explicitly in the program inside a function or inside a block of code. Throws is a keyword used in the method signature used to declare an exception which might get thrown by the function while executing the code.
2. Final - keyword is used for variables (have constant value), classes (these cannot be inherited) and methods (these cannot be overridden).
Finally - The finally block is used after a try and catch statements, to implement the things even when there are some exceptions that have been caught.
Finalize () - This object is invoked in java for garbage collection when the memory/objects are no longer in use.
3. Exceptions and errors both are subclasses of Throwable class. The error indicates a problem that mainly occurs due to the lack of system resources and our application should not catch these types of problems. Some of the examples of errors are system crash error and out of memory error. Errors mostly occur at runtime that's they belong to an unchecked type.
Exceptions are the problems which can occur at runtime and compile time. It mainly occurs in the code written by the developers. Exceptions are divided into two categories such as checked exceptions and unchecked exceptions
4. In the given scenario, when there is an exception caught in the statement2, the control goes over to the catch block, so the statements following statement2, i.e statement3, does not get executed.
5. When we are keeping multiple catch blocks, the order of catch blocks must be from most specific to most general ones. i.e subclasses of Exception must come first and superclasses later. If we keep superclasses first and subclasses later, the compiler will throw an unreachable catch block error.
6. Checked exceptions are checked at compile-time while unchecked exceptions are checked at runtime.
7. Yes, the finally block gets executed no matter what.

8. No, it gives unreachable code error. Because, control is returning from the finally block itself. Compiler will not see the statements after it.
9. 1. `addSuppressed(Throwable exception)`: This method appends the specified exception to the exceptions that were suppressed in order to deliver this exception.
Syntax:
`public final void addSuppressed(Throwable exception)`
Returns: This method does not return anything.
2. `fillInStackTrace()`: Fills in the execution stack trace. This method records information about the current state of the stack frames for the current thread within the current Throwable object.
Syntax:
`public Throwable fillInStackTrace ()`
Returns: a reference to the current Throwable instance.
3. `getCause()`: It returns the cause that was supplied via one of the constructors requiring a Throwable , or that was set after creation with the `initCause()` method.
Syntax:
`public Throwable getCause ()`
Returns: the cause of current Throwable. If the cause is nonexistent or unknown, it returns null.
4. `getLocalizedMessage()`: This method creates a localized description of current Throwable.
Syntax:
`public String getLocalizedMessage ()`
Returns: The localized description of current Throwable
5. `getMessage()`: Returns the detailed message string of current throwable.
Syntax:
`public String getMessage ()`
Returns: the detailed message string of current Throwable instance(may also return null)
10. Chained Exception helps to identify a situation in which one exception causes another Exception in an application.
11. 1. try block, catch block, finally block
2. try block, catch block
3. try block, multiple catch blocks, finally block
4. try block, multiple catch blocks
12. `printStackTrace()` method prints this throwable and its backtrace to the standard error stream. It prints a stack trace for this Throwable object on the error output stream that is the value of the field `System`.

Lab6:

Program1:

```
import java.util.*;
class Factorial {
    public int num, fact;
    public Factorial() {
        this.num = 1;
        this.fact = 1;
    }
    public Factorial(int num, int fact) {
        this.num = num;
        this.fact = fact;
    }
    public int getfact() {
        return this.fact;
    }
    public void setfact(int fact) {
        this.fact= fact;
    }
    public int getnum() {
        return this.num;
    }
    public void setnum(int num) {
        this.num = num;
    }
    synchronized public void compute() {
        this.fact = 1;
        for ( int i = 1; i <= num; i ++ ) {
            this.fact *= i;
        }
    }
    public String toString() {
        return "Number = " + num + " Factorial = " + fact;
    }
}
class ObjectCreator extends Thread {
    public int size;
    public Factorial objArr[];
    public ObjectCreator(Factorial objArr[], int size) {
        this.size = size;
        this.objArr = objArr;
    }
}
```

```

    public void run() {
        for( int i = 0; i < size; i ++ ) {
            this.objArr[i] = new Factorial();
            System.out.println("Object" + ( i + 1 ) + " created");
        }
    }
}

class MethodTester extends Thread {
    public int size;
    public Factorial objArr[];
    public MethodTester(Factorial objArr[], int size) {
        this.size = size;
        this.objArr = objArr;
    }
    Scanner sc = new Scanner(System.in);
    public void run() {
        for (int i = 0; i < size; i ++ ) {
            System.out.println("Enter n vale:");
            int n = sc.nextInt();
            objArr[i].setnum(n);
            objArr[i].compute();
            System.out.println(objArr[i]);
        }
    }
}

public class TestFactorial {
    public static void main(String args[]) throws InterruptedException {
        int n = Integer.parseInt(args[0]);
        Factorial objArr[] = new Factorial[n];
        ObjectCreator t1 = new ObjectCreator(objArr, n);
        MethodTester t2 = new MethodTester(objArr, n);
        t1.start();
        t1.join();
        t2.start();
        t2.join();
    }
}

```

```

Portkey@taruni:~/Desktop/javaprogs/lab6$ javac TestFactorial.java
Portkey@taruni:~/Desktop/javaprogs/lab6$ java TestFactorial 3
Object1 created
Object2 created
Object3 created
Enter n vale:
5
Number = 5 Factorial = 120
Enter n vale:
2
Number = 2 Factorial = 2
Enter n vale:
4
Number = 4 Factorial = 24
Portkey@taruni:~/Desktop/javaprogs/lab6$ █

```

Prelab Questions:

1. A program in execution is often referred to as a process. A thread is a subset(part) of the process. A process consists of multiple threads. A thread is a smallest part of the process that can execute concurrently with other parts(threads) of the process.
2.
 1. extend the Thread Class (lang package)
 2. implement Runnable interface (lang package)

When we want to inherit/extend other classes we can implement the runnable interface, otherwise we can extend the thread class.
3. Thread behaviour is unpredictable because execution of Threads depends on Thread scheduler, thread scheduler may have different implementation on different platforms like windows, unix etc. There is no surety that which threads will complete first.
4. We can use the Thread.join() method to make sure all the threads created by the program are dead before finishing the main function. In this way we can ensure that the main thread is the last to be executed.
5. When a program calls the start() method, a new thread is created and then the run() method is executed. But if we directly call the run() method then no new thread will be created and run() method will be executed as a normal method call on the current thread itself and no multithreading will take place.
 In multithreading, we can't call the start() method twice otherwise it will throw an IllegalStateException whereas run() method can be called multiple times as it is just a normal method calling.
 The start() method is defined in the java.lang Thread class whereas the run() method is defined in the java.lang Runnable interface.
6. No, we cannot start() the thread twice even if the run() method is completed.
7. When a method is synchronized, no other method can use this method in the whole object while this method is being run by one method. It means this method can only be run by the object once at a time and cannot be run multiple times simultaneously.
 Similarly synchronised block means the statements in a given block (between two braces {}) cannot be run simultaneously by many objects and can be run once at a time.
8. Race condition occurs in java when multiple threads operation the same object and their processes rely upon each other. This can be overcome by using synchronization.

9. Threads communicate with each other about their status by the use of methods like wait(), notify() and notifyAll().

Example:

```
class Table{
    synchronized void printTable(int n){//synchronized method
        for(int i=1;i<=5;i++){
            System.out.println(n*i);
            try{
                Thread.sleep(400);
            }catch(Exception e){System.out.println(e);}
        }
    }
}
```

```

}
}

class MyThread1 extends Thread{
    Table t;
    MyThread1(Table t){
        this.t=t;
    }
    public void run(){
        t.printTable(5);
    }
}
```

```

}
class MyThread2 extends Thread{
    Table t;
    MyThread2(Table t){
        this.t=t;
    }
    public void run(){
        t.printTable(100);
    }
}
```

```

public class TestSynchronization2{
    public static void main(String args[]){
        Table obj = new Table();//only one object
        MyThread1 t1=new MyThread1(obj);
        MyThread2 t2=new MyThread2(obj);
        t1.start();
        t2.start();
    }
}
```

10. These methods belong to the Object class. If wait() and notify() were on the Thread instead then each thread would have to know the status of every other thread and there is no way to know thread1 that thread2 was waiting for any resource to access. Hence, notify, wait, notifyAll methods are defined in object class in Java.

Lab7:

Program1:

```
import java.io.*;

// Main class Student
public class Student implements Serializable {
    // Encapsulation of Student class
    private int rollNumber;
    private String name;
    private int totalMarks;
    // Unparameterized Constructor for Student class
    public Student() {
        rollNumber = 0;
        name = "N/A";
        totalMarks = 0;
    }
    // Parameterized Constructor for Student class
    public Student(int rollNumber, String name, int totalMarks) {
        this.rollNumber = rollNumber;
        this.name = name;
        this.totalMarks = totalMarks;
    }
    // setter method for name
    public void setName(String name) {
        this.name = name;
    }
    // getter method for name
    public String getName() {
        return name;
    }
    // setter method for totalMarks
    public void setTotalMarks(int totalMarks) {
        this.totalMarks = totalMarks;
    }
    // getter method for totalMarks
    public int getTotalMarks() {
        return totalMarks;
    }
    // setter method for roll number
    public void setRollNumber(int rollNumber) {
        this.rollNumber = rollNumber;
    }
}
```



```

// getter method for roll number
public int getRollNumber() {
    return rollNumber;
}
// overriding toString method
public String toString() {
    return "Name: " + name + "\nRoll Number: " + rollNumber + "\nTotal Marks: " + totalMarks +
"\n";
}
}

```

```
import java.io.*;
```

```

// TestStudent for file-handling operations
public class TestStudent {
    public static void main(String[] args) throws IOException {
        try {
            int n, rNo, totalMarks, i = 0;
            String name;
            // buffered reader for taking input from user without Scanner or CLA
            BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
            System.out.print("Enter The Number Of Students: ");
            n = Integer.parseInt(reader.readLine());
            // Array for storing objects of Student
            Student[] _students = new Student[n];
            // taking user input
            try {
                for (i = 0; i < n; i++) {
                    System.out.print("\nEnter Name: ");
                    name = reader.readLine();
                    System.out.print("Enter Roll Number: ");
                    rNo = Integer.parseInt(reader.readLine());
                    System.out.print("Enter Total Marks: ");
                    totalMarks = Integer.parseInt(reader.readLine());
                    Student obj = new Student(rNo, name, totalMarks);
                    _students[i] = obj;
                }
            } catch (Exception e) {
                System.out.println("Invalid Entry!");
            }
            // Serialization of objects to Student_details.txt
            try {
                ObjectOutputStream objStrm = new ObjectOutputStream(new
FileOutputStream("Student_details.txt"));

```

```

        objStrm.writeObject(_students);
        objStrm.close();
    } catch (Exception e) { // catching Exception during Serialization
        System.out.println("Exception Caught During Serialization To Student_details");
    }
    // Array to hold deserialized Student objects
    Student[] _deserializedStudents = new Student[n];
    // Deserialization of objects from Student_details.txt
    try {
        ObjectInputStream objInpStrm = new ObjectInputStream(new
FileInputStream("Student_details.txt"));
        _deserializedStudents = (Student[]) objInpStrm.readObject();
        objInpStrm.close();
    }
    catch (ClassNotFoundException cnfe) { // catching Exceptions during Deserialization
        System.out.println("Exception Caught During Deserialization From Student_details");
    }
    catch (IOException ioe) {
        System.out.println("Exception Caught During Deserialization From Student_details");
    }
    // Sorting Deserialized Objects based on total marks
    for (i = 0; i < n; i++) {
        for (int j = i+1; j < n; j++) {
            if (_deserializedStudents[i].getTotalMarks() >
_deserializedStudents[j].getTotalMarks()) {
                Student s = _deserializedStudents[i];
                _deserializedStudents[i] = _deserializedStudents[j];
                _deserializedStudents[j] = s;
            }
        }
    }
    // arrays for holding ranks of students
    int[] _ranks = new int[n];
    _ranks[0] = 1;
    // Getting Ranks
    for(i = 1; i < n; i++) {
        if (_deserializedStudents[i-1].getTotalMarks() ==
_deserializedStudents[i].getTotalMarks()) {
            _ranks[i] = _ranks[i-1];
        } else {
            _ranks[i] = (i+1);
        }
    }
    // Writing to Result.txt

```

```

try {
    FileWriter myWriter = new FileWriter("Result.txt");
    BufferedWriter myBuffer = new BufferedWriter(myWriter);
    myBuffer.write("Rank\t| Roll Number\n");
    for (i = 0; i < n; i++) {
        myBuffer.write(" " + _ranks[i] + "\t\t" + _deserializedStudents[i].getRollNumber() +
"\n");
    }
    myBuffer.close();
    myWriter.close();
}
catch (IOException ioe) { // catching Exceptions
    System.out.println("IOException Has Been Caught.");
}
// Reading from Result.txt and displaying on console
System.out.println("");
try {
    BufferedReader bufferReader = new BufferedReader(new FileReader("Result.txt"));
    String current;

    while ((current = bufferReader.readLine()) != null) {
        System.out.println(current);
    }
    bufferReader.close();
}
catch (IOException ioe) { // catching Exceptions
    System.out.println("IOException Has Been Caught.");
}
}
catch (NumberFormatException e) {
    System.out.println("Invalid Entry.");
}
}
}

```

Prelab Questions:

1. They are of public and static type.
2. The byte stream classes provide a convenient means for handling input and output of bytes and character streams provide a convenient means for handling input and output of characters, respectively.
3. INPUT BYTE STREAM CLASSES

BufferedInputStream :contains methods to read bytes from the buffer (memory area)
ByteArrayInputStream :contains methods to read bytes from a byte array
DataInputStream :contains methods to read Java primitive data types
FileInputStream :contains methods to read bytes from a file
FilterInputStream :contains methods to read bytes from other input streams which it uses as its basic source of data
ObjectInputStream :contains methods to read objects
PipedInputStream :contains methods to read from a piped output stream. A piped input stream must be connected to a piped output stream
SequenceInputStream :contains methods to concatenate multiple input streams and then read from the combined stream

OUTPUT BYTE STREAM CLASSES

BufferedOutputStream :Contains methods to write bytes into the buffer
ByteArrayOutputStream :Contains methods to write bytes into a byte array
DataOutputStream :Contains methods to write Java primitive data types
FileOutputStream :Contains methods to write bytes to a file
FilterOutputStream :Contains methods to write to other output streams
ObjectOutputStream :Contains methods to write objects
PipedOutputStream :Contains methods to write to a piped output stream
PrintStream :Contains methods to print Java primitive data types

4. READER CLASS METHODS

BufferedReader :contains methods to read characters from the buffer
CharArrayReader :contains methods to read characters from a character array
FileReader :contains methods to read from a file
FilterReader :contains methods to read from underlying character-input stream
InputStreamReader :contains methods to convert bytes to characters
PipedReader :contains methods to read from the connected piped output stream
StringReader :contains methods to read from a string

WRITER CLASS METHODS

BufferedWriter	:Contains methods to write characters to a buffer
FileWriter	:Contains methods to write to a file
FilterWriter	:Contains methods to write characters to underlying output stream
CharArrayWriter	:Contains methods to write characters to a character array
OutputStreamWriter	:Contains methods to convert from bytes to character
PipedWriter	:Contains methods to write to the connected piped input stream
StringWriter	:Contains methods to write to a string

5. FileInputStream is used to read data in the form of a byte stream and FileReader is used for reading data in the form text type.
6. FileNotFoundException
7. RandomAccessFile Creates a random access file stream to read from, and optionally to write to, the file specified by the File argument or file name.
8. No, it will not work because the parent class can only implement serializable interface but the child class cannot implement the serializable interface.
9. Yes, the code works if Test2 doesn't implement Serializable
10. The Externalizable interface provides control over the serialization process. Similar to the Serializable interface, a class that implements the Externalizable interface is marked to be persisted, except those members which are declared as transient or static. Any class that implements the Externalizable interface should override the writeExternal(), readExternal() methods.
11. Closeable: A Closeable is a source or destination of data that can be closed.
DataInput: The DataInput interface provides for reading bytes from a binary stream and reconstructing from them data in any of the Java primitive types.
DataOutput: The DataOutput interface provides for converting data from any of the Java primitive types to a series of bytes and writing these bytes to a binary stream.
Externalizable: Only the identity of the class of an Externalizable instance is written in the serialization stream and it is the responsibility of the class to save and restore the contents of its instances.
FileFilter: A filter for abstract pathnames.
FilenameFilter: Instances of classes that implement this interface are used to filter filenames.
Flushable: A Flushable is a destination of data that can be flushed.
ObjectInput: ObjectInput extends the DataInput interface to include the reading of objects.
Serializable: Serializability of a class is enabled by the class implementing the java.io.Serializable interface.
ObjectStreamConstants: Constants written into the Object Serialization Stream.
ObjectOutput: ObjectOutput extends the DataOutput interface to include writing of objects.
ObjectInputValidation: Callback interface to allow validation of objects within a graph.
12. A. Closeable, Flushable, AutoCloseable
B. DataOutput, Closeable, Flushable, AutoCloseable

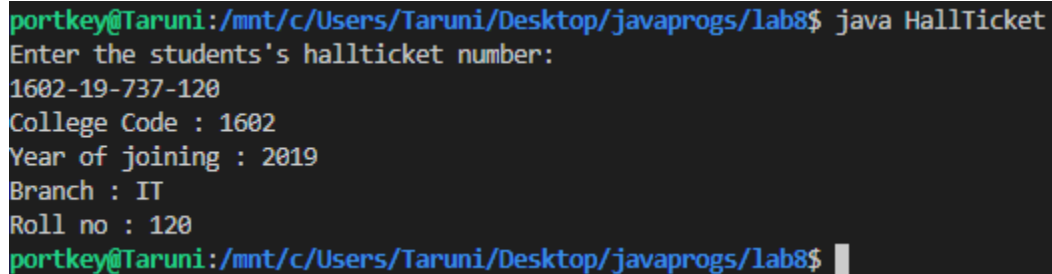
Lab8:

Program1:

```
import java.util.*;

//N Taruni
//IT-B
//1602-19-737-120

public class HallTicket {
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the students's hallticket number: ");
        String str = new String(sc.next()); //reading hallticket from the user
        StringTokenizer tokens = new StringTokenizer(str, "-"); // tokenizing with the delimiter "-"
        System.out.println("College Code : " + tokens.nextToken());
        System.out.println("Year of joining : 20" + tokens.nextToken());
        String branch = new String(tokens.nextToken());
        String res = new String();
        if (branch.equals("732")) res = "Civil";
        else if (branch.equals("733")) res = "CSE";
        else if (branch.equals("734")) res = "EEE";
        else if (branch.equals("735")) res = "ECE";
        else if (branch.equals("736")) res = "Mech";
        else if (branch.equals("737")) res = "IT";
        System.out.println("Branch : " + res);
        System.out.println("Roll no : " + tokens.nextToken());
    }
}
```



```
portkey@Taruni:/mnt/c/Users/Taruni/Desktop/javaprogs/lab8$ java HallTicket
Enter the students's hallticket number:
1602-19-737-120
College Code : 1602
Year of joining : 2019
Branch : IT
Roll no : 120
portkey@Taruni:/mnt/c/Users/Taruni/Desktop/javaprogs/lab8$
```

```
portkey@Taruni:/mnt/c/Users/Taruni/Desktop/javaprogs/lab8$ java HallTicket
Enter the students's hallticket number:
1602-17-733-183
College Code : 1602
Year of joining : 2017
Branch : CSE
Roll no : 183
portkey@Taruni:/mnt/c/Users/Taruni/Desktop/javaprogs/lab8$
```

```
portkey@Taruni:/mnt/c/Users/Taruni/Desktop/javaprogs/lab8$ java HallTicket
Enter the students's hallticket number:
1630-18-732-068
College Code : 1630
Year of joining : 2018
Branch : Civil
Roll no : 068
portkey@Taruni:/mnt/c/Users/Taruni/Desktop/javaprogs/lab8$
```

```
portkey@Taruni:/mnt/c/Users/Taruni/Desktop/javaprogs/lab8$ java HallTicket
Enter the students's hallticket number:
1024-17-736-073
College Code : 1024
Year of joining : 2017
Branch : Mech
Roll no : 073
portkey@Taruni:/mnt/c/Users/Taruni/Desktop/javaprogs/lab8$
```

Program2:

```
import java.util.*;
```

```
//N Taruni
```

```
//IT-B
```

```
//1602-19-737-120
```

```
public class Paragraph {
    static String[] getWords(String para, String wordStart) {
        para = para.replace(".", " ").replace(",", " ").replace("?", " "); // changing all the other
        punctuations to space
        StringTokenizer words = new StringTokenizer(para); // tokenizing with space as delimiter
        //separating into words
        String[] result = new String[words.countTokens()];
        int i = 0;
        while (words.hasMoreTokens()) {
            String s = words.nextToken();
            if (s.toLowerCase().startsWith(wordStart.toLowerCase())) result[i++] =
s.toLowerCase();
        }
    }
}
```

```

        //checking if the word starts with the given string
    }
    return result;
}
public static void main(String args[]) {
    Scanner sc = new Scanner(System.in);
    System.out.println("Enter a para: ");
    String para = sc.nextLine();
    System.out.println("Enter words for start with: ");
    String startsWith = sc.next();
    System.out.println("Words starting with " + startsWith + " are: ");
    String[] res = getWords(para, startsWith );
    for (int i = 0; i < res.length; i++) {
        if (res[i] == null) break;
        System.out.println(res[i]);
    }
}
}

```

```

portkey@Taruni:/mnt/c/Users/Taruni/Desktop/javaprogs/lab8$ javac Paragraph.java
portkey@Taruni:/mnt/c/Users/Taruni/Desktop/javaprogs/lab8$ java Paragraph
Enter a para:
She sells, sea shells, at the sea shore. What are you doing here? Shouldn't be at work? Yes, I'm going now.
Enter words for start with:
sh
Words starting with sh are:
she
shells
shore
shouldn't
portkey@Taruni:/mnt/c/Users/Taruni/Desktop/javaprogs/lab8$

```



```
portkey@Taruni:/mnt/c/Users/Taruni/Desktop/javaprogs/lab8$ java Paragraph
Enter a para:
An apple a day keeps the doctor away. Anna, please go have an apple I bought from the apple store
Enter words for start with:
a
Words starting with a are:
an
apple
a
away
anna
an
apple
apple
portkey@Taruni:/mnt/c/Users/Taruni/Desktop/javaprogs/lab8$
```

```
portkey@Taruni:/mnt/c/Users/Taruni/Desktop/javaprogs/lab8$ java Paragraph
Enter a para:
This is totally unacepatable. What were you doing all night? You need to figure out a way to fix all this before friday night. Make sure u do it a
s fast as possible.
Enter words for start with:
f
Words starting with f are:
figure
fix
friday
fast
portkey@Taruni:/mnt/c/Users/Taruni/Desktop/javaprogs/lab8$
```

Program3:

```
import java.util.*;
```

```
//N Taruni
```

```
//IT-B
```

```
//1602-19-737-120
```

```
public class CalendarDate {
    public static void main(String args[]) {
        Date today = new Date();
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter date, month, year: ");
        Calendar curr = Calendar.getInstance();
        today.setDate(sc.nextInt());
        today.setMonth(sc.nextInt() - 1);
        today.setYear(sc.nextInt() - 1900);
        curr.set(Calendar.SECOND, today.getSeconds());
        curr.set(Calendar.MINUTE, today.getMinutes());
        curr.set(Calendar.MONTH, today.getMonth());
        curr.set(Calendar.DAY_OF_MONTH, today.getDate());
        curr.set(Calendar.YEAR, today.getYear());
        int dd = today.getDate();
        int maxdd = curr.getActualMaximum(Calendar.DAY_OF_MONTH);
        System.out.println("Today : " + today);
    }
}
```

```

        today.setDate(maxdd);
        System.out.println("Last day of month is : " + today);
        System.out.println(maxdd == dd);
    }
}

```

```

portkey@Taruni:/mnt/c/Users/Taruni/Desktop/javaprogs/lab8$ java CalendarDate
Enter date, month, year:
31
1
2020
Today : Fri Jan 31 19:48:18 IST 2020
Last day of month is : Fri Jan 31 19:48:18 IST 2020
true
portkey@Taruni:/mnt/c/Users/Taruni/Desktop/javaprogs/lab8$

```

```

portkey@Taruni:/mnt/c/Users/Taruni/Desktop/javaprogs/lab8$ java CalendarDate
Enter date, month, year:
4
5
2016
Today : Wed May 04 19:48:50 IST 2016
Last day of month is : Tue May 31 19:48:50 IST 2016
false
portkey@Taruni:/mnt/c/Users/Taruni/Desktop/javaprogs/lab8$

```

Prelab Questions:

1. The string tokenizer class allows an application to break a string into token.... The StringTokenizer methods do not distinguish among identifiers, numbers, and quoted strings. A simple text scanner which can parse primitive types and strings using regular expressions. So Scanner unlike StringTokenizer have methods like nextInt, nextBoolean etc. While Scanner is useful in some cases when you need to parse user input containing numbers, StringTokenizer in most cases can be replaced with org.apache.commons.lang.StringUtils.split - which doesn't use regular expressions and is pretty fast.
2. split(String regex, int limit) method splits this string around matches of the given regular expression. The array returned by this method contains each substring of this string that is terminated by another substring that matches the given expression or is terminated by the end of the string.
3. Boolean hasNext() -Returns true if this scanner has another token in its input.
 next() Finds and returns the next complete token from this scanner.
 nextByte() Scans the next token of the input as a byte.
 toString() Returns the string representation of this Scanner.
 nextDouble() Scans the next token of the input as a double
 nextFloat() Scans the next token of the input as a float.
 nextInt() Scans the next token of the input as an int.

`nextLine()` Advances this scanner past the current line and returns the input that was skipped

`nextLong()` Scans the next token of the input as a long.

`nextShort()` Scans the next token of the input as a short

4. `boolean hasMoreTokens()` - checks if there is more tokens available.

`String nextToken()` - returns the next token from the `StringTokenizer` object.

`String nextToken(String delim)` - returns the next token based on the delimiter.

`boolean hasMoreElements()` - same as `hasMoreTokens()` method.

`Object nextElement()` - same as `nextToken()` but its return type is `Object`.

`int countTokens()` - returns the total number of tokens.

5. The difference between `Date` and `Calendar` is that `Date` class operates with specific instant in time and `Calendar` operates with difference between two dates. The `Calendar` class gives you the possibility for converting between a specific instant in time and a set of calendar fields such as `HOUR`, `YEAR`, `MONTH`, `DAY_OF_MONTH`.
6. `Calendar.getInstance()` gives the `Calendar` based on the current time in the default time zone with the default locale, and `Calendar.getInstance().getTime()` returns a `Date` object representing this `Calendar`'s time value.
7. Yes. When we need to perform date/time calculations, or to format dates for displaying them to the user.
8. `TimerTask` class implements `Runnable` interface.
9. `public void cancel()` : Terminates this timer, discarding any currently scheduled tasks.
`public int purge()` : Removes all cancelled tasks from this timer's task queue.
`public final void setTime(Date date)` : Sets this `Calendar`'s time with the given `Date`.
returns the number of tasks removed from the queue.
10. We can achieve thread safety by 'synchronized' keyword, using wrapper classes, using collection classes, Use of locks from `java.util.concurrent.locks` package, or using 'volatile' keyword with variables.

Lab9:

Program1:

```
import java.io.*;
import java.util.ArrayList;
import java.util.concurrent.ThreadLocalRandom;

//N Taruni
//1602-19-737-120
//IT-B

public class RandomnLines {
    public static void main(String args[]) throws IOException, FileNotFoundException {
        String filename = args[0]; //taking file name as command line argument
        File ptr = new File(filename);
        int numOfLinesToBePrinted = Integer.parseInt(args[1]); //taking number of lines to be printed
        as command line argument
        int avgLineSize = 30; //assuming average line size
        int numOfLines = (int) (ptr.length() / avgLineSize); //calculating total number of lines
        BufferedReader fptr = new BufferedReader(new FileReader(ptr));
        ArrayList <String> lines = new ArrayList <String> (numOfLines); //declaring arraylist of string
        type
        for (int i = 0; i < numOfLines; i++) {
            try {
                lines.add(fptr.readLine()); //storing each line into the arraylist
            }
            catch (IOException e) {}
        }
        fptr.close(); //closing the bufferedReader stream
        for (int i = 0; i < numOfLinesToBePrinted; i++) {
            int randomNum = ThreadLocalRandom.current().nextInt(0, numOfLines); //generating a
            randomn number between for printing randomn lines
            System.out.println(lines.get(randomNum)); //printing a randomn line from file
        }
    }
}
```

```
portkey@Taruni:/mnt/c/Users/Taruni/Desktop/javaprogs/lab9$ javac RandomnLines.java
portkey@Taruni:/mnt/c/Users/Taruni/Desktop/javaprogs/lab9$ java RandomnLines input.txt 2
Ram always comes on time.
They say, "You reap what you sow".
```

```
portkey@Taruni:/mnt/c/Users/Taruni/Desktop/javaprogs/lab9$ java RandomnLines input.txt 2
testing the file and number of lines.
testing the file and number of lines.
```

```
portkey@Taruni:/mnt/c/Users/Taruni/Desktop/javaprogs/lab9$ java RandomnLines input.txt 3
this is the sixth line of the file.
She is so annoying. She doesn't like fruits.
This is the last line of this file.
```

```
portkey@Taruni:/mnt/c/Users/Taruni/Desktop/javaprogs/lab9$ java RandomnLines input.txt 1
She is so annoying. She doesn't like fruits.
```

Pre lab Questions:

1. The Java collections framework is a set of classes and interfaces that implement commonly reusable collection data structures. The Java collections framework gives the programmer access to prepackaged data structures as well as to algorithms for manipulating them. A collection is an object that can hold references to other objects. The collection interfaces declare the operations that can be performed on each type of collection.
2. Generics in Collection framework provides type safety, and thereby we can know the errors at the compile time itself. By using generics, we don't even need to typecast objects/variables.
3. We can use the iterator/listIterator or we can use a for each loop to iterate through a list.
4. Iterator is used for traversing List and Set both, whereas, ListIterator can be used to traverse List only, we cannot traverse Set. We can traverse in only forward direction using Iterator. Using ListIterator, we can traverse both forward and Backward.
5. An Iterator is universal and can be applied to all collection classes, whereas enumeration can only be applied to legacy classes. An iterator can make modifications like remove() whereas an enumerator is more of a read only type. Iterator can be used for the traversal of HashMap, LinkedList, ArrayList, HashSet, TreeMap, TreeSet. Enumeration is a legacy interface which is used for traversing Vector, Hashtable.
6. An ArrayList is a collection class that can be used to store objects, whereas an array can store objects or primitive data types. Array has a fixed size unlike ArrayList which can expand its size. Accessing of array elements can be done using [index] while in ArrayList, elements are accessed using ArrayList.get(index). Array doesn't contain any special methods but since ArrayList is a class, it has some predefined methods that can be used.
7. HashMap allows a single null key and multiple null values. TreeMap does not allow null keys but can have multiple null values. HashMap allows heterogeneous elements because it does not perform sorting on keys. TreeMap allows homogeneous values as a key because of sorting. HashMap is a general purpose Map implementation. It provides a performance of $O(1)$, while TreeMap provides a performance of $O(\log(n))$ to add,

search, and remove items. Hence, HashMap is usually faster. ... Use a TreeMap if you need to keep all entries in natural order.

8. ArrayList, HashMap, TreeMap, Hashtable, and Vector classes provide random access to its elements.
9. Comparable provides a single sorting sequence. In other words, we can sort the collection on the basis of a single element. Comparator provides multiple sorting sequences. In other words, we can sort the collection on the basis of multiple elements. Comparable affects the original class, i.e., the actual class is modified. Comparator doesn't affect the original class, i.e., the actual class is not modified. Comparable provides compareTo() method to sort elements. Comparator provides compare() method to sort elements. Comparable is present in java.lang package. Comparator is present in the java.util package. We can sort the list elements of Comparable type by Collections.sort(List) method. We can sort the list elements of Comparator type by Collections.sort(List, Comparator) method.
10. Sorting. Shuffling. Routine Data Manipulation. Searching. Composition. Finding Extreme Values.

Lab10:

Program1:

```
package lab10;

public class InvalidTemperature extends Exception{
    private static final long serialVersionUID = 1L;

    public InvalidTemperature() {}

    public String toString() {
        return "Invalid Input";
    }
}

package lab10;

import javax.swing.*.*;
import java.io.*.*;
import java.awt.event.*.*;
//import java.text.*.*;

public class TempConvertor {
    public JLabel l1,l2;
    public JTextField t1,t2;
    public JFrame f;
    public JMenuBar mb;
    public JMenu menu;
    public JMenuItem open, exit;

    public TempConvertor() { //default constructor
        f = new JFrame("Temperature Convertor");
        l1 = new JLabel("Fahrenheit");
        l1.setBounds(100,50,100,30);
        l2 = new JLabel("Celsius");
        l2.setBounds(100,80,100,30);
        t1 = new JTextField();
        t1.setBounds(200,50,100,30);
        t2 = new JTextField();
        t2.setBounds(200,80,100,30);
        mb = new JMenuBar();
        menu = new JMenu("File");
```

```

        open = new JMenuItem("Open");
        exit = new JMenuItem("Exit");
        reactToEvents();
        addComponents();
    }

    public static void checkException(String s) throws InvalidTemperature {
        try {
            double d = Double.parseDouble(s);
        }
        catch (NumberFormatException e) {
            throw new InvalidTemperature();
        }
    }

    public void reactToEvents() {
        t1.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                try {
                    checkException(t1.getText());
                    Double fah = Double.parseDouble(t1.getText());
                    Double cel = ((fah - 32) * (5.0 / 9.0));
                    t2.setText(Double.toString(cel));
                }
                catch (InvalidTemperature err) {
                    JFrame jf= new JFrame("Invalid temperature");
                    JLabel l3;
                    l3 = new JLabel("Try Again!");
                    l3.setBounds(100,50,100,30);
                    jf.add(l3);
                    jf.setSize(300,300);
                    jf.setLayout(null);
                    jf.setVisible(true);
                }
            }
        });
        t2.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                try {
                    checkException(t2.getText());
                    Double cel = Double.parseDouble(t2.getText());
                    Double fah = ((9.0 / 5.0) * cel) + 32.0;
                    t1.setText(Double.toString(fah));
                }
            }
        });
    }

```



```

        catch (InvalidTemperature err) {
            JFrame jf1 = new JFrame("Invalid Temperature");
            JLabel l4;
            l4 = new JLabel("Try Again!");
            l4.setBounds(100,50,100,30);
            jf1.add(l4);
            jf1.setSize(300,300);
            jf1.setLayout(null);
            jf1.setVisible(true);
        }
    }
});
exit.addMouseListener(new MouseAdapter() {
    public void mousePressed(MouseEvent e) {
        ObjectOutputStream ois;
        try {
            ois = new ObjectOutputStream(new
FileOutputStream("data.bin"));
            ois.writeObject(this);
            ois.close();
        }
        catch (Exception err) {}
        finally {
            f.dispose();
        }
    }
});
}

public void addComponents() {
    mb.add(menu);
    menu.add(open);
    menu.add(exit);
    f.setJMenuBar(mb);
    f.add(l1);
    f.add(l2);
    f.add(t1);
    f.add(t2);
    f.setSize(500, 500);
    f.setLayout(null);
    f.setVisible(true);
}
}

```

```
package lab10;

public class TestTempConvertor {
    public static void main(String args[]) {
        new TempConvertor();
    }
}
```

Pre Lab Question:

1) Component and Container are standard classes in Java. The main difference between them is that a Container is a subclass of Component which can contain other components and containers.

Frame, Panel and Applet are subclasses of Container. A Container helps to create groupings of objects on the screen.

2) To set the color of the background of an applet window, setBackground () method is used
void setBackground(mycolor)

Similarly, to set the foreground color to a specific color, that is, the color of text, setForeground () method is used.

The general form of the setForeground () method is
void setForeground(mycolor)

3) The getLabel() method retrieves the current text of the label on the Button and returns it as a String.

4) Text Component Class is the super class of textfield and textarea.

5)

1. Java AWT is an API to develop GUI applications in Java Swing is a part of Java Foundation Classes and is used to create various applications.

2. The components of Java AWT are heavily weighted. The components of Java Swing are light weighted.

3. Java AWT has comparatively less functionality as compared to Swing. Java Swing has more functionality as compared to AWT.

4. The execution time of AWT is more than Swing. The execution time of Swing is less than AWT.

5. The components of Java AWT are platform dependent. The components of Java Swing are platform independent.

6. MVC pattern is not supported by AWT. MVC pattern is supported by Swing.

7. AWT provides comparatively less powerful components. Swing provides more powerful components.

6) Steps involved in event handling:

The User clicks the button and the event is generated. Now the object of the concerned event class is created automatically and information about the source and the event get populated within the same object. Event object is forwarded to the method of registered listener class. The method now gets executed and returns.

7) ActionEvent, AdjustmentEvent, ComponentEvent, ItemEvent, TextEvent, ContainerEvent, FocusEvent, InputEvent, paintEvent, WindowEvent, KeyEvent, MouseEvent, MouseWheelEvent.

8) init(), start(), stop(), paint(), destroy().

The method execution sequence when an applet is executed is: init() start() paint()

The method execution sequence when an applet is closed is: stop() destroy()

9) The getDocumentBase() method is used to return the URL of the directory in which the document resides. URL getCodeBase(): Gets the base URL. URL getDocumentBase(): Gets the URL of the document in which the applet is embedded.

```
10) import java.awt.*;
    import java.applet.*;
    public class DisplayImage extends Applet {
        Image picture;
    public void init() {
        picture = getImage(getDocumentBase(),"pic.jpg");
    }
    public void paint(Graphics g) {
        g.drawImage(picture, 30,30, this);
    }
    }
```

11) The bottom of the applet window is known as status bar.

12) obj.drawString(str); -- to display text in applet window
 obj.showStatus(str) --- to display text on status bar

Lab11:

Program1:

```
package lab11;

import javax.swing.*.*;
import java.awt.*.*;
import java.awt.event.*.*;

class SwingRegistration implements ActionListener{
    public JLabel name, title, pwrld, number, languages, gender, state;
    public JTextField tname, tnumber;
    public JPasswordField tpwrld;
    public TextArea warnings;
    public JFrame frame;
    public JButton submit;
    public Checkbox male, female, hindi, english, telugu;
    public CheckboxGroup cbg;
    public Choice c;

    public SwingRegistration() { // default constructor
        frame = new JFrame("Registration");
        frame.setSize(480, 500);

        title = new JLabel("STUDENT REGISTRATION FORM");
        title.setBounds(125, 10, 200, 10);

        name = new JLabel("Name:");
        name.setBounds(35, 40, 50, 10);
        tname = new JTextField();
        tname.setBounds(130, 40, 100, 18);

        pwrld = new JLabel("Password:");
        pwrld.setBounds(35, 70, 80, 10);
        tpwrld = new JPasswordField();
        tpwrld.setEchoChar('*');
        tpwrld.setBounds(130, 70, 100, 18);

        number = new JLabel("Phone:");
        number.setBounds(35, 100, 80, 10);
        tnumber = new JTextField();
        tnumber.setBounds(130, 100, 100, 18);
```

```
languages = new JLabel("Select Languages Known:");
languages.setBounds(35, 130, 150, 18);
hindi = new Checkbox("Hindi");
hindi.setBounds(200, 135, 60, 10);
english = new Checkbox("English");
english.setBounds(270, 135, 60, 15);
telugu = new Checkbox("Telugu");
telugu.setBounds(350, 135, 60, 15);
```

```
gender = new JLabel("Select Gender:");
gender.setBounds(35, 160, 150, 10);
cbg = new CheckboxGroup();
male = new Checkbox("Male", cbg, false);
male.setBounds(200, 165, 50, 15);
female = new Checkbox("Female", cbg, false);
female.setBounds(270, 165, 60, 15);
```

```
state = new JLabel("Select State:");
state.setBounds(35, 190, 150, 10);
c = new Choice();
c.setBounds(200, 190, 150, 50);
c.add("Andhra Pradesh");
c.add("Arunachal Pradesh");
c.add("Assam");
c.add("Bihar");
c.add("Chhattisgarh");
c.add("Goa");
c.add("Haryana");
c.add("Himachal Pradesh");
c.add("Jammu and Kashmir");
c.add("Jharkhand");
c.add("Karnataka");
c.add("Kerala");
c.add("Madhya Pradesh");
c.add("Maharashtra");
c.add("Manipur");
c.add("Meghalaya");
c.add("Mizoram");
c.add("Nagaland");
c.add("Orissa");
c.add("Punjab");
c.add("Rajasthan");
c.add("Sikkim");
c.add("Tamil Nadu");
```

```

c.add("Telangana");
c.add("Tripura");
c.add("Uttarakhand");
c.add("Uttar Pradesh");
c.add("West Bengal");
    submit = new JButton("Submit");
    submit.setBounds(200, 240, 100, 20);
    submit.addActionListener(this);
    warnings = new TextArea();
    warnings.setBounds(50, 280, 360, 140);
    warnings.setEditable(false);
    warnings.setBackground(Color.WHITE);
    addComponents();

```

```

}

```

```

public int numOfDigits( String num) { //finding num of digits in phone number
    int number = Integer.parseInt(num);
    int digits = 0;
    while (number > 0) {
        number = number / 10;
        ++ digits;
    }
    return digits;
}

```

```

public String validatePhone(String number) { // checking if phone number entered is valid
    for (int i = 0; i < number.length(); i++) {
        if (!Character.isDigit(number.charAt(i))) {
            return "Phone numbers should have only numbers.\n";
        }
    }
    if (numOfDigits(number) != 10) {
        return "Phone number should have 10 digits.\n";
    }
    return "";
}

```

```

public String validatePassword(char[] pwr) { //checking if password entered is valid
    if (pwr.length < 8) return "Password should have atleast 8 characters.\n";
    return "";
}

```

```

public void actionPerformed(ActionEvent e ) {

```

```

String a = validatePhone(tnumber.getText());
String b = validatePassword(tpwrd.getPassword());
if (!(a + b).equals("")) {
    warnings.setText(a + b);
}
else {
    String message = "Name = " + tname.getText() + "\nLanguages
known:\n";

    if (hindi.getState()) message += hindi.getLabel() + "\n";
    if (english.getState()) message += english.getLabel() + "\n";
    if (telugu.getState()) message += telugu.getLabel() + "\n";
    message += "Phone = " + tnumber.getText() + "\n";
    message += "Gender = " + ((male.getState()) ? male.getLabel() :
female.getLabel()) + "\n";
    message += "State = " + c.getSelectedItem() + "\n";
    warnings.setText(message);

}
}

public void addComponents() { //adding components to the frame
    frame.add(title);
    frame.add(name);
    frame.add(pwrd);
    frame.add(number);
    frame.add(languages);
    frame.add(gender);
    frame.add(state);
    frame.add(submit);
    frame.add(tname);
    frame.add(tpwrd);
    frame.add(tnumber);
    frame.add(hindi);
    frame.add(english);
    frame.add(telugu);
    frame.add(male);
    frame.add(female);
    frame.add(c);
    frame.add(warnings);
    frame.setLayout(null);
    frame.setVisible(true);
}
}

```

```
package lab11;
```

```
public class TestSwingRegistration {  
    public static void main(String args[]) {  
        new SwingRegistration();  
    }  
}
```

Registration

STUDENT REGISTRATION FORM

Name:

Password:

Phone:

Select Languages Known: ☒ Hindi ☐ English ☐ Telugu

Select Gender: ☐ Male ☒ Female

Select State:

Phone numbers should have only numbers.
Password should have atleast 8 characters.

Registration

STUDENT REGISTRATION FORM

Name:

Password:

Phone:

Select Languages Known: ☒ Hindi ☐ English ☐ Telugu

Select Gender: ☐ Male ☒ Female

Select State:

Phone number should have 10 digits.

Registration

STUDENT REGISTRATION FORM

Name:

Password:

Phone:

Select Languages Known: ☒ Hindi ☐ English ☐ Telugu

Select Gender: ☐ Male ☒ Female

Select State:

Name = abc
Languages known:
Hindi
Phone = 1235667890
Gender = Female
State = Goa

Registration

STUDENT REGISTRATION FORM

Name:

Password:

Phone:

Select Languages Known: ☒ Hindi ☒ English ☒ Telugu

Select Gender: ☐ Male ☒ Female

Select State:

Password should have atleast 8 characters.

Pre Lab Questions:

1. AWT components are called Heavyweight components. Swings are called light weight components because swing components sit on the top of AWT components and do the work. AWT components are platform dependent. Swing components are made in purely java and they are platform independent.
2. The Java Swing components can only be accessed from the Event Dispatch Thread (EDT) once a component is available for painting on screen. ... Only thread-safe methods can be safely invoked from any thread. Since most of the Swing object methods are not thread-safe, they can be invoked from a single thread, the EDT.
3. In Java Swing, the layer that is used to hold objects is called the content pane. Objects are added to the content pane layer of the container. The getContentPane() method retrieves the content pane layer so that you can add an object to it. The content pane is an object created by the Java runtime environment.
4. UIManager. getCrossPlatformLookAndFeelClassName() – To get the Look and Feel of Java.
UIManager. setLookAndFeel() – To set the look and feel of UI components.
JFrame. setDefaultCloseOperation(true); – To change the look and feel of the frame.
5. Swing event-handling and painting code executes in a single thread, called the event-dispatching thread. This ensures that each event handler finishes executing before the next one executes and that painting isn't interrupted by events.
6. The JScrollBar sends out AdjustmentEvent s every time something happens; the JSlider fires off ChangeEvent s when its value changes.
7. No, both awt and swings use common packages for event handling.
8. UIManager.setLookAndFeel("com.plaf.motif.MotifLookAndFeel");
9. It sizes the JFrame so that all its contents are at or above their preferred sizes.
10. We can use the TextField component and set the echo character to a space or empty character by setEchoChar() method. (OR) we can use the JPasswordField component.

Lab12:

Program1:

```
package lab12;

import java.applet.*;

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

public class Calculator extends Applet implements ActionListener
{
    private static final long serialVersionUID = 1L;
    TextField t1;
    int i;
    public void init()
    {
        setBackground(Color.white);
        setForeground(Color.blue);
        setLayout(null);

        Frame title=(Frame)this.getParent().getParent();
        title.setTitle("Calculator");

        t1=new TextField();
        t1.setBounds(20,20,240,40);
        this.add(t1);

        Button b1=new Button("1");
        b1.setBounds(20,80,40,40);
        add(b1);
        b1.setVisible(true);
        b1.addActionListener(this);

        Button b2=new Button("2");
        b2.setBounds(100,80,40,40);
        add(b2);
        b2.addActionListener(this);

        Button b3=new Button("3");
        b3.setBounds(180,80,40,40);
        add(b3);
    }
}
```

```
b3.addActionListener(this);
```

```
Button b4=new Button("*");  
b4.setBounds(260,80,40,40);  
add(b4);  
b4.addActionListener(this);
```

```
Button b5=new Button("4");  
b5.setBounds(20,140,40,40);  
add(b5);  
b5.addActionListener(this);
```

```
Button b6=new Button("5");  
b6.setBounds(100,140,40,40);  
add(b6);  
b6.addActionListener(this);
```

```
Button b7=new Button("6");  
b7.setBounds(180,140,40,40);  
add(b7);  
b7.addActionListener(this);
```

```
Button b8=new Button("/");  
b8.setBounds(260,140,40,40);  
add(b8);  
b8.addActionListener(this);
```

```
Button b9=new Button("7");  
b9.setBounds(20,200,40,40);  
add(b9);  
b9.addActionListener(this);
```

```
Button b10=new Button("8");  
b10.setBounds(100,200,40,40);  
add(b10);  
b10.addActionListener(this);
```

```
Button b11=new Button("9");  
b11.setBounds(180,200,40,40);  
add(b11);  
b11.addActionListener(this);
```

```
Button b12=new Button("-");  
b12.setBounds(260,200,40,40);
```

```
add(b12);
b12.addActionListener(this);
```

```
Button b13=new Button("0");
b13.setBounds(20,260,40,40);
add(b13);
b13.addActionListener(this);
```

```
Button b14=new Button(".");
b14.setBounds(100,260,40,40);
add(b14);
b14.addActionListener(this);
```

```
Button b15=new Button("=");
b15.setBounds(180,260,40,40);
add(b15);
b15.addActionListener(this);
```

```
Button b16=new Button("+");
b16.setBounds(260,260,40,40);
add(b16);
b16.addActionListener(this);
```

```
Button clr=new Button("Clear All");
clr.setBounds(80,320,150,50);
this.add(clr);
clr.addActionListener(this);
```

```
}
String num1="";
String op="";
String num2="";
```

```
public void actionPerformed(ActionEvent e)//Function to calculate arithmetic expressions
{
```

```
    String button = e.getActionCommand();
    char ch = button.charAt(0);
    if(ch>='0' && ch<='9' || ch=='.')
    {
        if (!op.equals(""))
            num2 = num2 + button;
        else
            num1 = num1 + button;
```

```

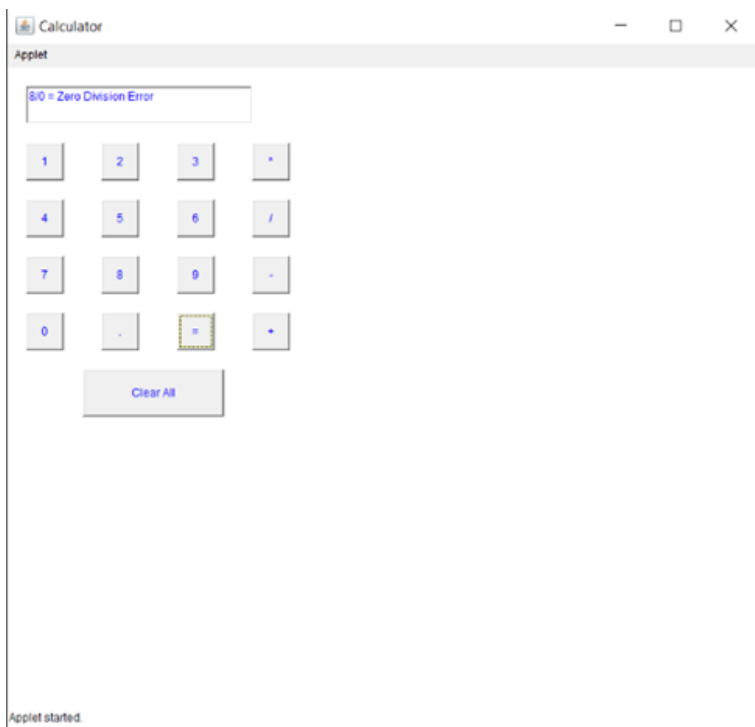
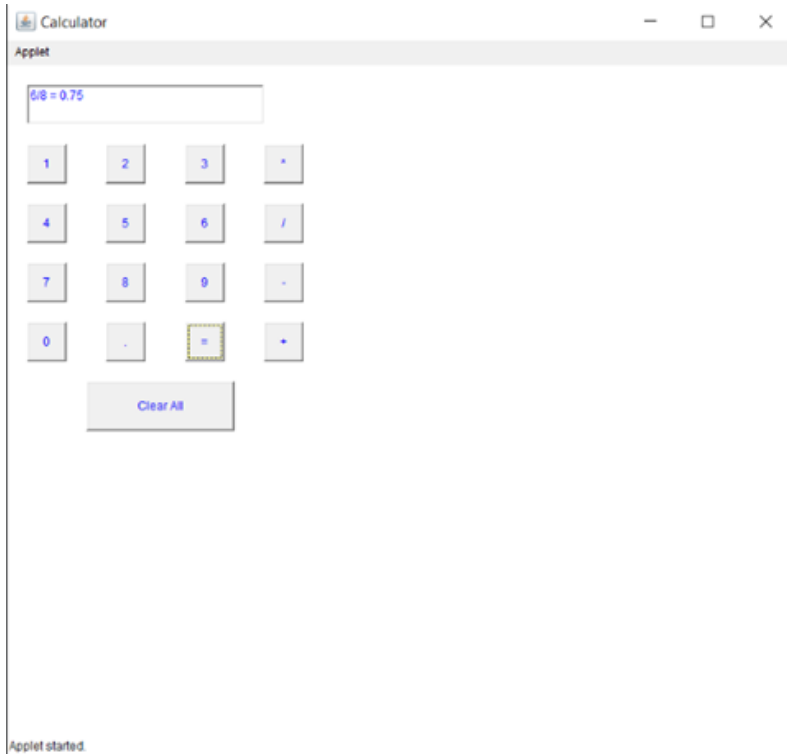
        t1.setText(num1+op+num2);
    }
    else if(ch=='C')
    {
        num1 = op = num2 = "";
        t1.setText("");
    }
    else if (ch == '=')
    {
        if(!num1.equals("") && !num2.equals(""))
        {
            double temp;
            double n1=Double.parseDouble(num1);
            double n2=Double.parseDouble(num2);
            if(n2==0 && op.equals("/"))
            {
                t1.setText(num1+op+num2+" = Zero Division Error");
                num1 = op = num2 = "";
            }
            else
            {
                if (op.equals("+"))
                    temp = n1 + n2;
                else if (op.equals("-"))
                    temp = n1 - n2;
                else if (op.equals("/"))
                    temp = n1/n2;
                else
                    temp = n1*n2;
                t1.setText(num1+op+num2+" = "+temp);
                num1 = Double.toString(temp);
                op = num2 = "";
            }
        }
    }
    else
    {
        num1 = op = num2 = "";
        t1.setText("");
    }
}
else
{
    if (op.equals("") || num2.equals(""))
        op = button;
}

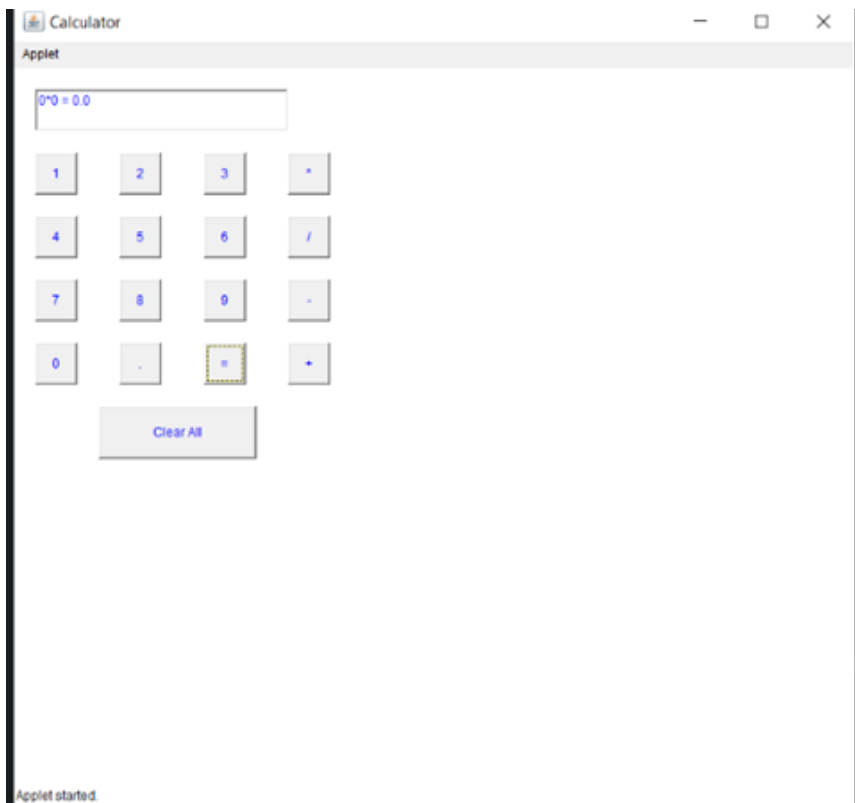
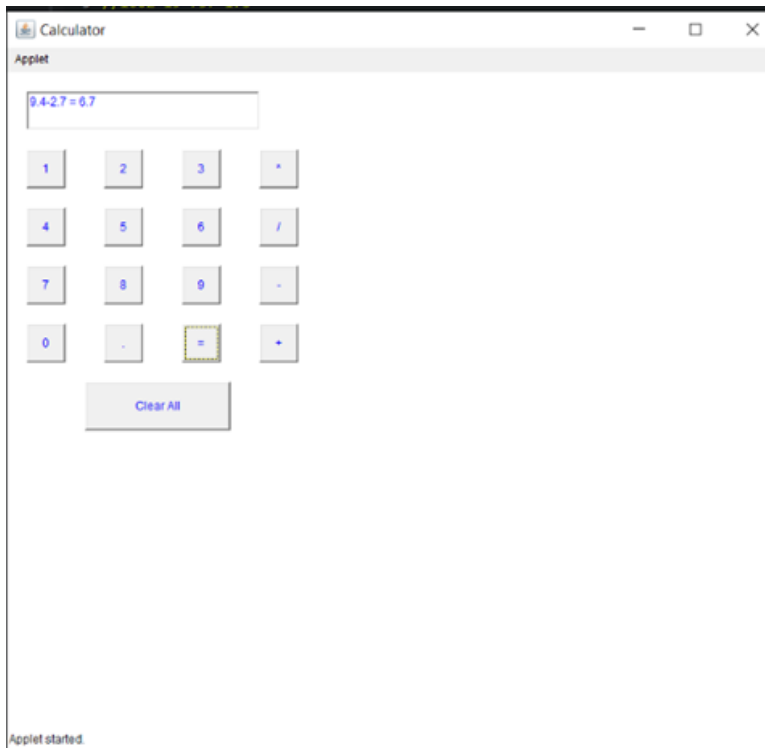
```

```

else
{
    double temp;
    double n1=Double.parseDouble(num1);
    double n2=Double.parseDouble(num2);
    if(n2==0 && op.equals("/"))
    {
        t1.setText(num1+op+num2+" = Zero Division Error");
        num1 = op = num2 = "";
    }
    else
    {
        if (op.equals("+"))
            temp = n1 + n2;
        else if (op.equals("-"))
            temp = n1 - n2;
        else if (op.equals("/"))
            temp = n1/n2;
        else
            temp = n1*n2;
        num1 = Double.toString(temp);
        op = button;
        num2 = "";
    }
}
t1.setText(num1+op+num2);
}
}
}

```





Pre lab Questions:

1. `init()`, `start()`, `stop()`, `paint()`, `destroy()`.
Applet is initialized.
Applet is started.
Applet is painted.
Applet is stopped.
Applet is destroyed.
2. - The `getDocumentBase()` method is used to return the URL of the directory in which the document resides. ... URL `getDocumentBase()`: Gets the URL of the document in which the applet is embedded.
3.

```
import java.awt.*;
import java.applet.*;
public class DisplayImage extends Applet {
    Image picture;
    public void init() {
        picture = getImage(getDocumentBase(),"sonoo.jpg");
    }
    public void paint(Graphics g) {
        g.drawImage(picture, 30,30, this);
    }
}
```
4. The bottom of the applet window is known as status bar. In a Browser or MS-Word, the bottom panel is known as status bar where in the line number where the cursor exists, current page number and total number of pages etc. are displayed. ... The applet can use this status bar to display strings (messages).
5. `obj.drawString(str);` -- to display text in applet window
`obj.showStatus(str)` --- to display text on status bar