

Vision of the Department

Providing quality education to enable the generation of socially conscious software engineers who can contribute to the advancement in the field of computer science and engineering.

Mission of the Department

- To equip the graduates with the knowledge and skills required to enable them to be industry ready.
- To train socially responsible, disciplined engineers who work with good leadership skills and can contribute for nation building.
- To make our graduates proficient in cutting edge technologies through student centric teaching-learning process and empower them to contribute significantly to the software industry.
- To shape the department into a centre of academic and research excellence.

Program Educational Objectives

PEO-1

To provide the graduates with solid foundation in Computer Science and Engineering along with the fundamentals of Mathematics and Sciences with a view to impart in them high quality technical skills like modelling, analyzing, designing, programming and implementation with global competence and helps the graduates for life-long learning.

PEO-2

To prepare and motivate graduates with recent technological developments related to core subjects like Programming, Databases, Design of Compilers and Network Security aspects and future technologies so as to contribute effectively for Research & Development by participating in professional activities like publishing and seeking copy rights.

PEO-3

To train graduates to choose a decent career option either in high degree of employability/Entrepreneur or, in higher education by empowering students with ethical administrative acumen, ability to handle critical situations and training to excel in competitive examinations.

PEO-4

To train the graduates to have basic interpersonal skills and sense of social responsibility that paves them a way to become good team members and leaders.

Program Outcomes (POs)

- 1. Engineering knowledge:** apply the knowledge of mathematics, science, engineering fundamentals and an engineering specialization to the solution of complex engineering problems.
- 2. Problem analysis:** identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural science and engineering sciences.
- 3. Design/development of solutions:** design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal and environmental considerations.
- 4. Conduct investigations of complex problems:** use research based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
- 5. Modern tool usage:** create, select and apply appropriate techniques, resources and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.
- 6. The engineer and society:** apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
- 7. Environment sustainability:** understand the impact of the professional engineering solutions in the societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
- 8. Ethics:** apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
- 9. Individual and team work:** function effectively as an individual and as a member or leader in diverse teams, and in multidisciplinary settings.
- 10. Communication:** communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
- 11. Project management and finance:** demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
- 12. Lifelong learning:** recognize the need for, and have the preparation and ability to engage in independent and lifelong learning in the broader context of technological change.

Program Specific Outcomes (PSOs)

PSO-1: Professional Skills: The ability to understand, analyze and develop computer programs in the areas related to algorithms, system software, multimedia, web design, big data analytics, and networking for efficient design of computer based systems of varying complexity.

PSO-2: Successful Career and Entrepreneurship: The ability to employ modern computer languages, environments, and platforms in creating innovative career paths to be an entrepreneur and a zest for higher studies/employability in the field of Computer Science & Engineering.

II- Year I- Semester	Name of the Course	L	T	P	C
	Java Programming Lab	0	0	3	1.5

Course Objectives:

1. To write programs using abstract classes.
2. To write programs for solving real world problems using java collection frame work.
3. To write multithreaded programs.
4. To design GUI application using swing controls.
5. To introduce java compiler and eclipse platform
6. To impart hands on experience with java programming.

Course Outcomes: at the end of the lab, the student will be able to

CO1: Implement object oriented programming concepts, and apply them in solving problems. (Apply)

CO2: Experiment the implementation of packages and interfaces. (Apply)

CO3: Experiment the concept of multithreading over single threaded programming. (Analyze)

CO4: Use generic data structures of collection framework to manipulate data. (Apply)

CO5: Test the GUI based network applications among multiple users through network programming. (Analyze)

CO-PO mapping Table

CO/ PO- PSO	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12	PSO 1	PSO 2
CO1	2		2		3	1		2	2	2			2	2

CO2	2		2		3	1		2	2	2			2	2
CO3	2	2	2		3	1		2	2	2			2	2
CO4	2		2		3	1		2	3	2		2	2	2
CO5	2	2	2		3	1		2	3	2		2	2	2

Note:

Mandatory to follow test driven development with Eclipse IDE empowered JUnit testing framework and code coverage plugin.

The list suggests the minimum program set. Hence, the concerned staff is requested to add more problems to the list as needed.

List of Experiments

1. Create a class called Invoice that a hardware store might use to represent an invoice for an item sold at the store. An Invoice should include four pieces of information as instance variables-a part number (type String),a part description(type String),a quantity of the item being purchased (type int) and a price per item (double). Your class should have a constructor that initializes the four instance variables. Provide a set and a get method for each instance variable. In addition, provide a method named `getInvoiceAmount()` that calculates the invoice amount (i.e., multiplies the quantity by the price per item), then returns the amount as a double value. If the quantity is not positive, it should be set to 0. If the price per item is not positive, it should be set to 0.0. Write a test application named `InvoiceTest` that demonstrates class Invoice's capabilities. [CO1]

2. Develop a Java application to generate Electricity bill. Create a class with the following members: Consumer no., consumer name, previous month reading, current month reading, and type of EB connection (i.e. domestic or commercial). Compute the bill amount using the following tariff. [CO1]

If the type of the EB connection is domestic, calculate the amount to be paid as follows:

1. First 100 units - Rs. 1 per unit
2. 101-200units - Rs. 2.50 per unit

- 3. 201 -500 units - Rs. 4 per unit
- 4. >501 units - Rs. 6 per unit

If the type of the EB connection is commercial, calculate the amount to be paid as follows:

- 1. First 100 units - Rs. 2 per unit
- 2. 101-200units - Rs. 4.50 per unit
- 3. 201 -500 units - Rs. 6 per unit
- 4. >501 units - Rs. 7 per unit

3. Create class Savings Account. Use a static variable `annualInterestRate` to store the annual interest rate for all account holders. Each object of the class contains a private instance variable `savingsBalance` indicating the amount the saver currently has on deposit. Provide method `calculateMonthlyInterest` to calculate the monthly interest by multiplying the `savingsBalance` by `annualInterestRate` divided by 12 this interest should be added to `savings Balance`. Provide a static method `modifyInterestRate` that sets the `annualInterestRate` to a new value. Write a program to test class `SavingsAccount`. Instantiate two `savingsAccount` objects, `saver1` and `saver2`, with balances of \$2000.00 and \$3000.00, respectively. Set `annualConcentration Rate` to 4%, then calculate the monthly interest and print the new balances for both savers. Then set the `annualInterestRate` to 5%, calculate the next month's interest and print the new balances for both savers. [CO1]
4. Create a class called `Book` to represent a book. A `Book` should include four pieces of information as instance variables; a book name, an ISBN number, an author name and a publisher. Your class should have a constructor that initializes the four instance variables. Provide a mutator method and accessor method (query method) for each instance variable. In addition, provide a method named `getBookInfo` that returns the description of the book as a `String` (the description should include all the information about the book). You should use this keyword in member methods and constructor. Write a test application named `BookTest` to create an array of object for 30 elements for class `Book` to demonstrate the class `Book`'s capabilities. [CO1].
5. Write a JAVA program to search for an element in a given list of elements using binary search mechanism. [CO1]
6. Write a Java program that implements Merge sort algorithm for sorting and also shows the number of interchanges occurred for the given set of integers. [CO1]

7. Write a java program to make rolling a pair of dice 10,000 times and counts the number of times doubles of are rolled for each different pair of doubles. Hint: Math.random() [CO1].
8. Develop a java application with Employee class with Emp_name, Emp_id, Address, Mail_id, Mobile_no as members. Inherit the classes, Programmer, Assistant Professor, Associate Professor and Professor from employee class. Add Basic Pay (BP) as the member of all the inherited classes with 97% of BP as DA, 10 % of BP as HRA, 12% of BP as PF, 0.1% of BP for staff club fund. Generate pay slips for the employees with their gross and net salary. [CO1]
9. Write a Java Program to create an abstract class named Shape that contains two integers and an empty method named print Area(). Provide three classes named Rectangle, Triangle and Circle such that each one of the classes extends the class Shape. Each one of the classes contains only the method print Area () that prints the area of the given shape.[CO2]
10. Develop a java application to implement currencyconverter(DollartoINR, EURO toINR,YentoINR and vice versa), distance converter (meter to KM, miles to KM and vice versa), timeconverter (hours to minutes, seconds and vice versa) using packages. [CO1]
11. Write a Java Program to Handle Arithmetic Exceptions and InputMismatchExceptions. [CO1]
12. Write a multi-threaded Java program to print all numbers below 100,000 that are both prime and Fibonacci number (some examples are 2, 3, 5, 13, etc.). Design a thread that generates prime numbers below 100,000 and writes them into a pipe. Design another thread that generates Fibonacci numbers and writes them to another pipe. The main thread should read both the pipes to identify numbers common to both. [CO3].
13. Write a java program that implements a multi-threaded application that has three threads. First thread generates a random integer every 1 second and if the value is even, second thread computes the square of the number and prints. If the value is odd, the third thread will print the value of cube of the number. [CO3].

14. Write a Java program that correctly implements the producer – consumer problem using the concept of inter-thread communication. [CO3].
15. Write a Java program that reads a file name from the user, displays information about whether the file exists, whether the file is readable, or writable, the type of file and the length of the file inbytes. [CO1].
16. Write a Java program to build a Calculator in Swings. [CO4]
17. Write a Java program to implement JMenu to draw all basic shapes using Graphics. [CO4]
18. Write a Java program to implement JTable and JTree. [CO4]
19. Write a Java program to implement JTabbedPane. [CO4]
20. Write a Java Program that implements a simple client/server application. The client sends data to a server. The server receives the data, uses it to produce a result and then sends the result back to the client. The client displays the result on the console. For ex: The data sent from the client is the radius of a circle and the result produced by the server is the area of the circle. [CO5]

List of Additional Experiments

1. Demonstrate the usage of methods in String, StringBuffer class.
2. Write a java program to demonstrate Stack implementation.
3. Write a java program to demonstrate Queue implementation.
4. Write a java program to implement Queues.
5. Write a java program to demonstrate the usage of ByteStream classes.
6. Write a java program to demonstrate the usage of CharacterStream classes.
7. Write a java program to demonstrate Serialization and Deserialization.

CERTIFICATE

Name of the Lab : JAVA PROGRAMMING

Name of the Student : **BELLA VENKATA PRAVEEN KUMAR**

Student Regd. No. : 20BQ1A0520

CLASS : II B.TECH. I SEM CSE – A

GIT HUB LINK:

INDEX

S. No.	Experiment Number	DATE OF COMPLETION	DATE OF SUBMISSION	PAGE NO	Remarks	Signature of Faculty
1	Experiment -1	12-10-2021	18-10-2021	11-15		
2	Experiment -2	12-10-2021	18-10-2021	16-20		
3	Experiment -3	26-10-2021	28-10-2021	21-23		
4	Experiment -4	26-10-2021	28-10-2021	24-28		
5	Experiment -5	02-11-2021	05-11-2021	29-31		
6	Experiment -6	02-11-2021	05-11-2021	32-34		
7	Experiment -7	09-11-2021	11-11-2021	35-36		
8	Experiment -8	09-11-2021	11-11-2021	37-44		
9	Experiment -9	16-11-2021	19-11-2021	45-48		
10	Experiment -10	16-11-2021	19-11-2021	49-67		
11	Experiment -11	23-11-2021	27-11-2021	68-70		
12	Experiment -12	23-11-2021	27-11-2021	71-75		
13	Experiment -13	30-11-2021	13-12-2021	76-78		
14	Experiment -14	30-11-2021	13-12-2021	79-82		
15	Experiment -15	30-11-2021	13-12-2021	83-85		
16	Experiment -16	14-12-2021	28-12-2021	86-96		
17	Experiment -17	14-12-2021	28-12-2021	97-117		
18	Experiment -18	14-12-2021	28-12-2021	118-126		

19	Experiment -19	04-01-2022	06-01-2022	118-126		
20	Experiment -20	04-01-2022	06-01-2022	127-130		
Additional Experiments						
21	Experiment -1	16-11-2021	19-11-2021	131-135		
22	Experiment -2	16-11-2021	19-11-2021	136-137		
23	Experiment -3	16-11-2021	19-11-2021	138-137		
24	Experiment -4	30-11-2021	13-12-2021	138-140		
25	Experiment -5	30-11-2021	13-12-2021	141-144		
26	Experiment-6	30-11-2021	13-12-2021	145-147		
27	Experiment-7	30-11-2021	13-12-2021	148-149		

EXPERIMENT NO: 1

AIM:

Create a class called Invoice that a hardware store might use to represent an invoice for an item sold at the store. An Invoice should include four pieces of information as instance variables-a part number (type String),a part description(type String),a quantity of the item being purchased (type int) and a price per item (double). Your class should have a constructor that initializes the four instance variables. Provide a set and a get method for each instance variable. In addition, provide a method named getInvoiceAmount() that calculates the invoice amount (i.e., multiplies the quantity by the price per item), then returns the amount as a double value. If the quantity is not positive, it should be set to 0. If the price per item is not positive, it should be set to 0.0. Write a test application named InvoiceTest that demonstrates class Invoice's capabilities.

DESCRIPTION:

If variables or methods that are declared as private as access modifiers then to assign values for variables we can use get and set methods.

Getters and setters:

Getter and Setter are methods used to protect your data and make your code more secure.

Getters:

Getter returns the value (accessors), it returns the value of data type int, String, double, float, etc. For the convenience of the program, getter starts with the word "get" followed by the variable name return type of getters depends on the variable datatype.

Setters:

Setter sets or updates the value (mutators). It sets the value for any variable which is used in the programs of a class and starts with the word "set" followed by the variable name. Getter and Setter make the programmer convenient in setting and getting the value for a particular data type. Return type of setters is void. In both getter and setter, the first letter of the variable should be capital.

SYNTAX:

```
private datatype variable_name;  
// to assign value to variable_name  
public returntype getVariable_name()  
{  
    return variable_name; //return value  
}  
public void setVariable_name(datatype variable_name)  
{
```

```
this.variable_name = variable_name;
```

```
}
```

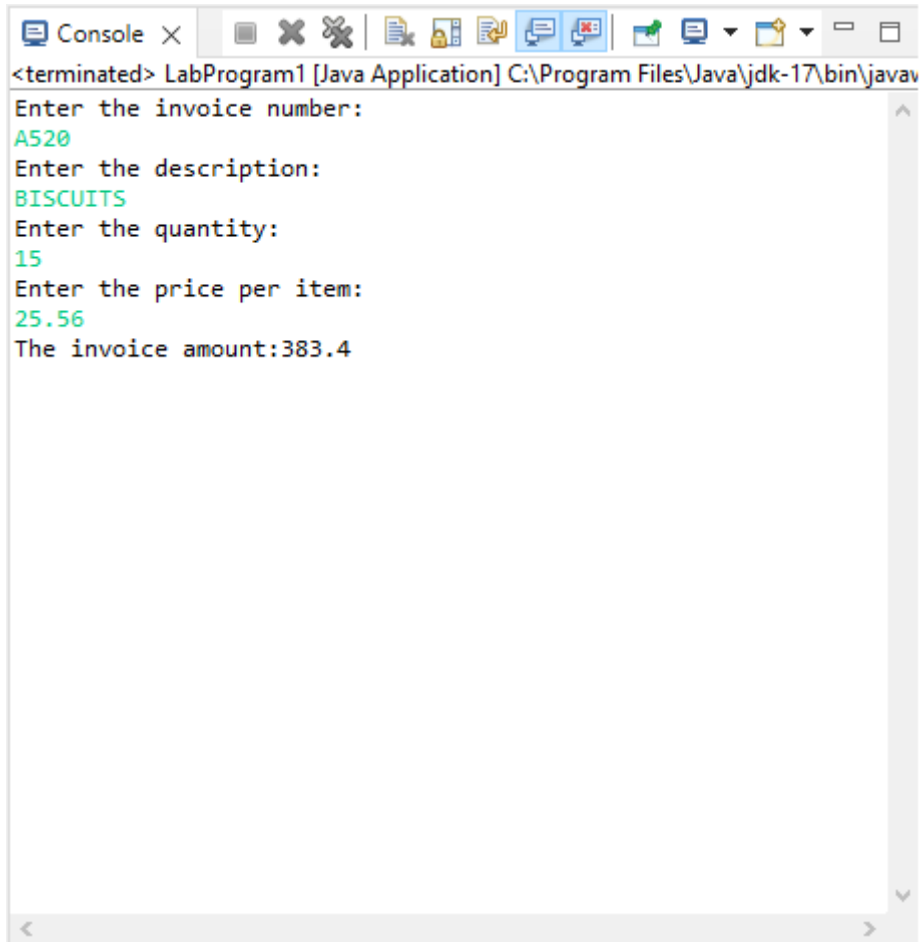
PROGRAM:

```
import java.util.Scanner;
class Invoice{
    private String number;
    private String description;
    private int quantity;
    private double price;
    void assignNum(String number) {
        this.number=number;
    }
    void assigndes(String description) {
        this.description=description;
    }
    void assignqua(int quantity) {
        if(quantity>=0)
            this.quantity=quantity;
        else
            this.quantity=0;
    }
    void assignpri(double pri) {
        if(price>=0)
            this.price=pri;
        else
            this.price=0.0;
    }
    String getnumber() {
        return number;
    }
    String getdescription() {
        return description;
    }
    int getquantity() {
        return quantity;
    }
    double getprice() {
        return price;
    }
    double getInvoiceAmount() {
        return (getprice()*getquantity());
    }
}
public class LabProgram1 {

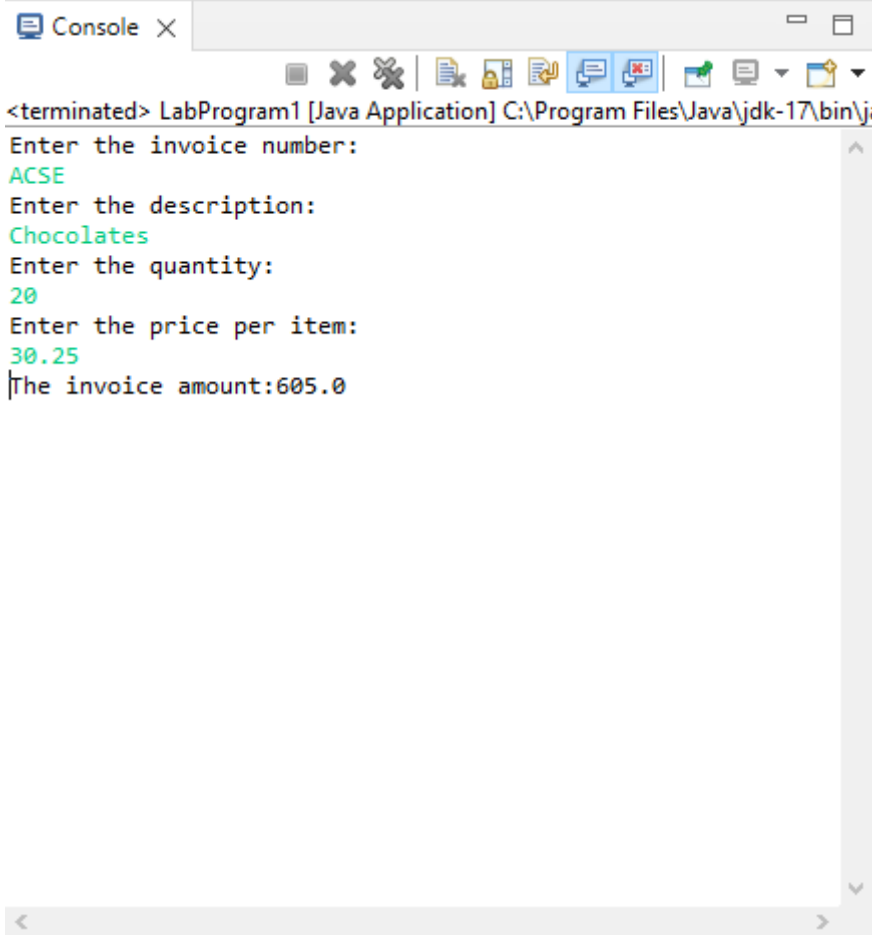
    public static void main(String[] args) {
        // TODO Auto-generated method stub
```

```
Scanner sc= new Scanner(System.in);
Invoice v1=new Invoice();
System.out.println("Enter the invoice number;");
String invoi=sc.next();
sc.nextLine();
System.out.println("Enter the description:");
String des=sc.nextLine();
System.out.println("Enter the quantity:");
int quan=sc.nextInt();
System.out.println("Enter the price per item:");
double price=sc.nextDouble();
v1.assignNum(invoi);
v1.assigndes(des);
v1.assignqua(quan);
v1.assignpri(price);
System.out.println("The invoice amount:"+v1.getInvoiceAmount());
sc.close();
}
```

OUTPUT:



```
<terminated> LabProgram1 [Java Application] C:\Program Files\Java\jdk-17\bin\javaw
Enter the invoice number:
A520
Enter the description:
BISCUITS
Enter the quantity:
15
Enter the price per item:
25.56
The invoice amount:383.4
```

```
<terminated> LabProgram1 [Java Application] C:\Program Files\Java\jdk-17\bin\j  
Enter the invoice number:  
ACSE  
Enter the description:  
Chocolates  
Enter the quantity:  
20  
Enter the price per item:  
30.25  
The invoice amount:605.0
```

EXPERIMENT NO: 2

AIM:

Develop a Java application to generate Electricity bill. Create a class with the following members: Consumer no., consumer name, previous month reading, current month reading, and type of EB connection (i.e. domestic or commercial). Compute the bill amount using the following tariff. [CO1]

If the type of the EB connection is domestic, calculate the amount to be paid as follows:

1. First 100 units - Rs. 1 per unit
2. 101-200units - Rs. 2.50 per unit
3. 201 -500 units - Rs. 4 per unit
4. >501 units - Rs. 6 per unit

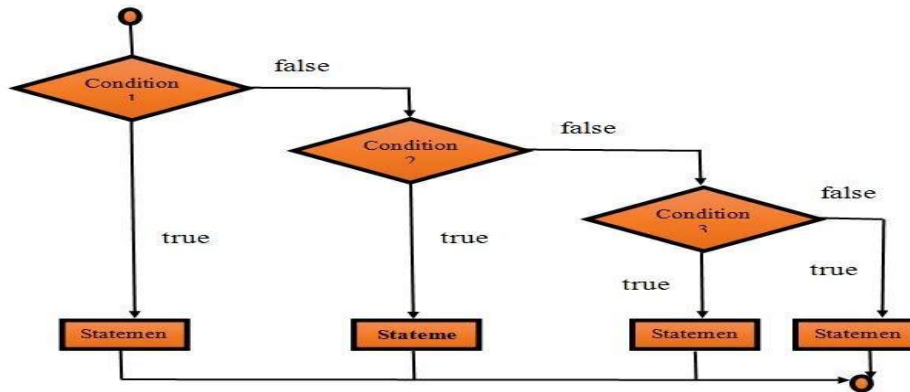
If the type of the EB connection is commercial, calculate the amount to be paid as follows:

1. First 100 units - Rs. 2 per unit
2. 101-200units - Rs. 4.50 per unit
3. 201 -500 units - Rs. 6 per unit
4. >501 units - Rs. 7 per unit

DESCRIPTION:

Conditional Statements:

else if statement:



SYNTAX:

```

if (condition1) {
    // block of code to be executed if condition1 is true
} else if (condition2) {
    // block of code to be executed if the condition1 is false and condition2 is true
} else {
    // block of code to be executed if the condition1 is false and condition2 is false
}
  
```

PROGRAM:

```

import java.util.Scanner;
class Electricity{
    long counsumer_number;
    String consumer_name;
    long previous_reading;
    long current_reading;
    String eb_connection;
    Electricity(long no, String name, long pr, long cr,String eb){
        counsumer_number=no;
        consumer_name=name;
        previous_reading=pr;
        current_reading=cr;
        eb_connection=eb;
    }
    double rate() {
  
```

```

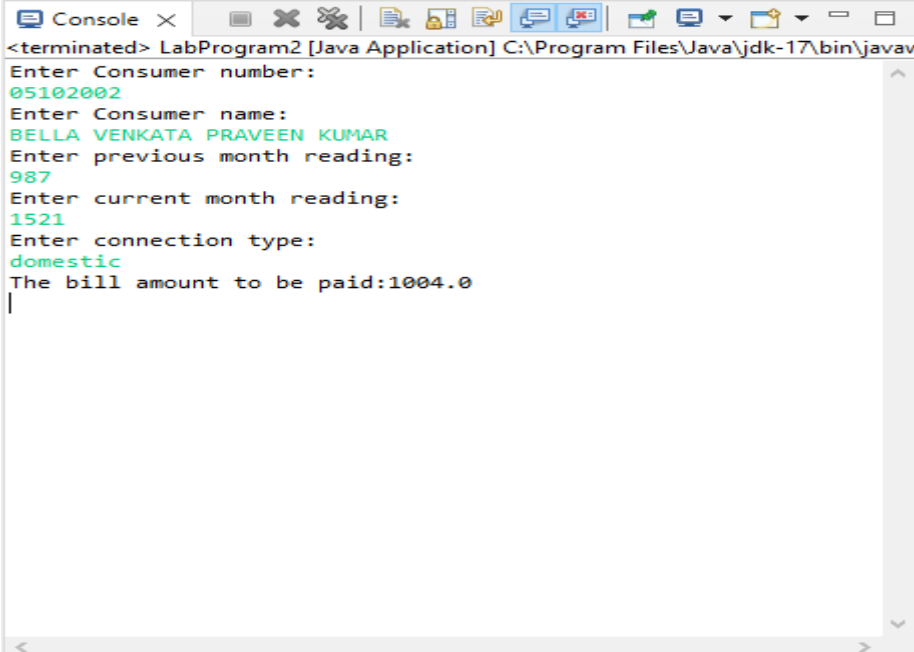
double p=0.0;
double c=current_reading-previous_reading;
if(eb_connection.equals("domestic"))
{
    if((c)<=100)
        p= 100*1;
    else
        if((c)<=200)
            p= 1*(100)+(c-100)*2.50;
        else
            if((c)<=500)
                p= 1*(100)+(200*2.50)+(c-500)*4;
            else
                p= (1*(100))+(200*2.50)+(500*4)+((c-800)*6);
}
else
{
    if((c)<=100)
        p=100*2;
    else
        if((c)<=200)
            p= (100*2)+((c-100)*4.50);
        else
            if((c)<=500)
                p=(100*2)+(200*4.50)+((c-500)*6);
            else
                if((c)>501)
                    p=(100*2)+(200*4.50)+(500*6)+((c-800)*7);
}
return p;
}
}
public class LabProgram2 {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter Consumer number:");
        long number=sc.nextLong();
        sc.nextLine();
        System.out.println("Enter Consumer name:");
        String name=sc.nextLine();
        System.out.println("Enter previous month reading:");
        long pr=sc.nextLong();
        System.out.println("Enter current month reading:");
        long cr=sc.nextLong();
        sc.nextLine();
        System.out.println("Enter connection type:");
        String eb=sc.nextLine();
    }
}

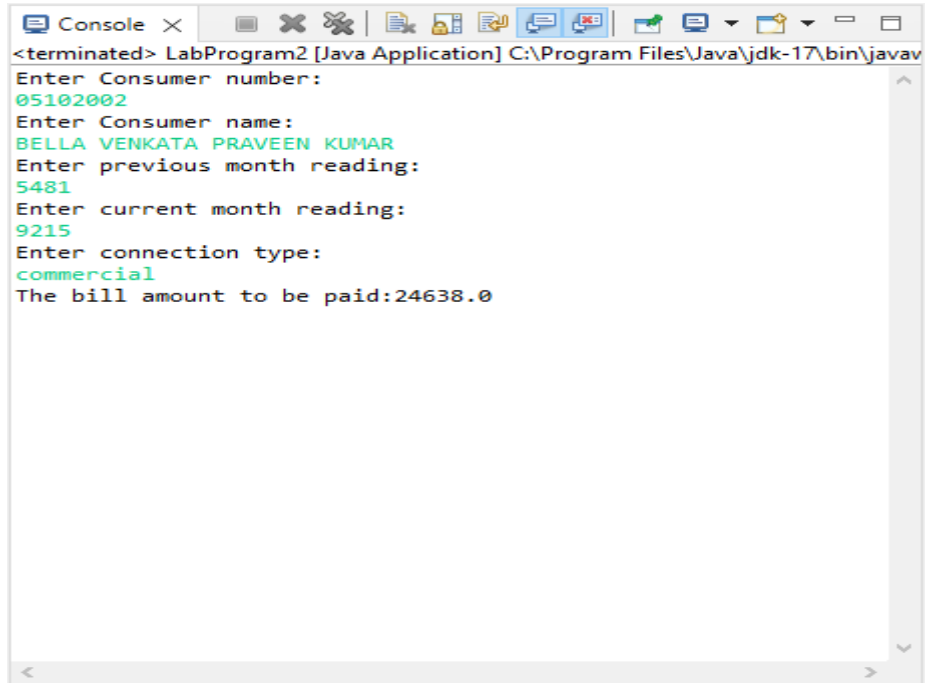
```

```
Electricity e= new Electricity(number,name,pr,cr,eb);  
System.out.println("The bill amount to be paid:"+e.rate());  
sc.close();  
}  
}
```

OUTPUT:



```
<terminated> LabProgram2 [Java Application] C:\Program Files\Java\jdk-17\bin\javaw  
Enter Consumer number:  
05102002  
Enter Consumer name:  
BELLA VENKATA PRAVEEN KUMAR  
Enter previous month reading:  
987  
Enter current month reading:  
1521  
Enter connection type:  
domestic  
The bill amount to be paid:1004.0  
|
```



```
<terminated> LabProgram2 [Java Application] C:\Program Files\Java\jdk-17\bin\javav
Enter Consumer number:
05102002
Enter Consumer name:
BELLA VENKATA PRAVEEN KUMAR
Enter previous month reading:
5481
Enter current month reading:
9215
Enter connection type:
commercial
The bill amount to be paid:24638.0
```


EXPERIMENT NO: 3

AIM:

Create class Savings Account. Use a static variable annualInterestRate to store the annual interest rate for all account holders. Each object of the class contains a private instance variable savingsBalance indicating the amount the saver currently has on deposit. Provide method calculateMonthlyInterest to calculate the monthly interest by multiplying the savingsBalance by annualInterestRate divided by 12 this interest should be added to savingsBalance. Provide a static method modifyInterestRate that sets the annualInterestRate to a new value. Write a program to test class SavingsAccount. Instantiate two savingsAccount objects, saver1 and saver2, with balances of \$2000.00 and \$3000.00, respectively. Set annualConcentration Rate to 4%, then calculate the monthly interest and print the new balances for both savers. Then set the annualInterestRate to 5%, calculate the next month's interest and print the new balances for both savers.

DESCRIPTION:

Static variable:

Static variable in Java is variable which belongs to the class and initialized only once at the start of the execution. It is a variable which belongs to the class and not to object (instance).

1. A single copy to be shared by all instances of the class
2. A static variable can be accessed directly by the class name and doesn't need any object

Static Method:

Static method in Java is a method which belongs to the class and not to the object. A static method can access only static data. It is a method which belongs to the class and not to the object (instance).

1. A static method can call only other static methods and can not call a non-static method from it.
2. A static method can be accessed directly by the class name and doesn't need any object
3. A static method cannot refer to "this" or "super" keywords in anyway

SYNTAX:

```
static datatype variable_name;  
public static return_type function_name(parameters)  
{  
    // static block  
  
}
```

PROGRAM:

```
import java.util.Scanner;
class SavingsAccount{
    double annual_interest;
    private long savings_balance;
    long getSavings_balance() {
        return savings_balance;
    }
    void setSavings_balance(long savings_balance) {
        this.savings_balance = savings_balance;
    }
    void modifyInterestRate(double s) {
        annual_interest=s/100.0;
    }

    double calculateMonthlyInterest() {
        double p= ((annual_interest*getSavings_balance())/12.0);
        return p;
    }
    double savings_balance() {
        return getSavings_balance()+calculateMonthlyInterest();
    }
}

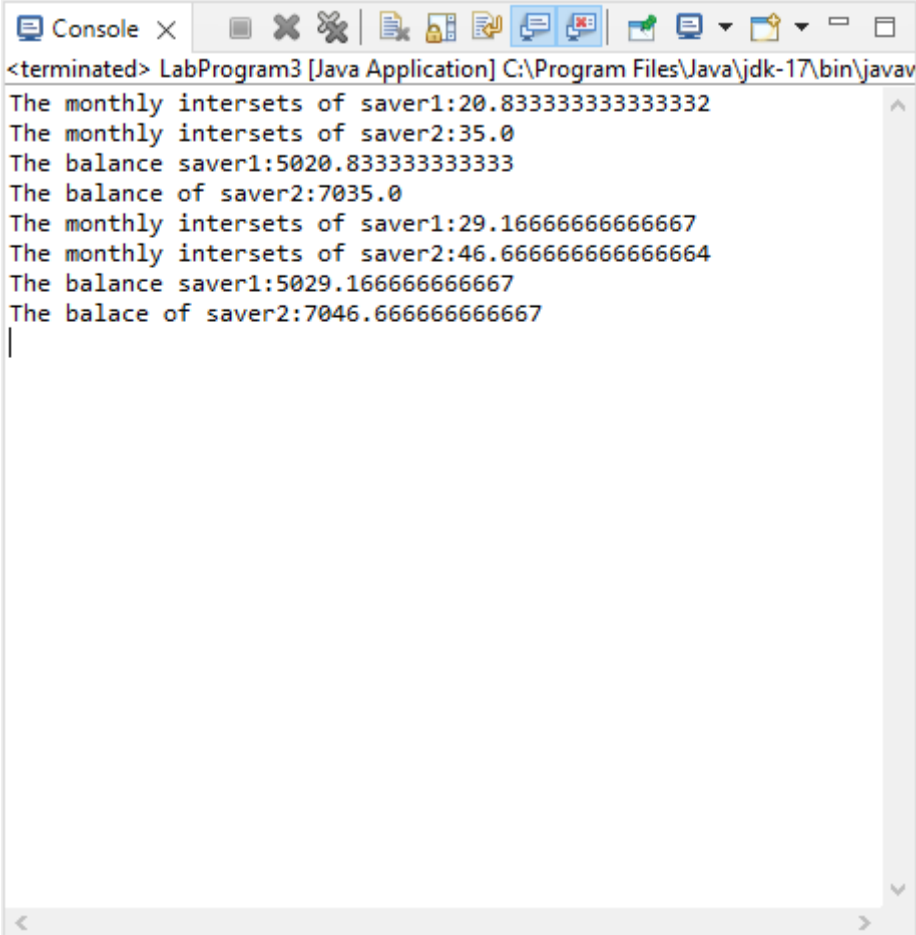
public class LabProgram3 {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Scanner sc=new Scanner(System.in);
        SavingsAccount saver1=new SavingsAccount();
        SavingsAccount saver2=new SavingsAccount();
        saver1.setSavings_balance(5000);
        saver2.setSavings_balance(7000);
        saver1.modifyInterestRate(5);
        saver2.modifyInterestRate(6);
        System.out.println("The monthly intersets of
saver1:"+saver1.calculateMonthlyInterest());
        System.out.println("The monthly intersets of
saver2:"+saver2.calculateMonthlyInterest());

        System.out.println("The balance saver1:"+saver1.savings_balance());
        System.out.println("The balance of
saver2:"+saver2.savings_balance());
        saver1.modifyInterestRate(7);
        saver2.modifyInterestRate(8);
        System.out.println("The monthly intersets of
saver1:"+saver1.calculateMonthlyInterest());
```

```
        System.out.println("The monthly intersets of  
saver2:"+saver2.calculateMonthlyInterest());  
  
        System.out.println("The balance saver1:"+saver1.savings_balance());  
        System.out.println("The balace of  
saver2:"+saver2.savings_balance());  
        sc.close();  
    }  
}
```

OUTPUT:



```
<terminated> LabProgram3 [Java Application] C:\Program Files\Java\jdk-17\bin\javav  
The monthly intersets of saver1:20.833333333333332  
The monthly intersets of saver2:35.0  
The balance saver1:5020.833333333333  
The balance of saver2:7035.0  
The monthly intersets of saver1:29.166666666666667  
The monthly intersets of saver2:46.666666666666664  
The balance saver1:5029.1666666666667  
The balace of saver2:7046.6666666666667  
|
```

EXPERIMENT NO: 4

AIM:

Create a class called Book to represent a book. A Book should include four pieces of information as instance variables-a book name, an ISBN number, an author name and a publisher. Your class should have a constructor that initializes the four instance variables. Provide a mutator method and accessor method (query method) for each instance variable. In addition, provide a method named getBookInfo that returns the description of the book as a String (the description should include all the information about the book). You should use this keyword in member methods and constructor. Write a test application named BookTest to create an array of object for 30 elements for class Book to demonstrate the class Book's capabilities.

DESCRIPTION:

Array of objects:

The array of Objects the name itself suggests that it stores an array of objects. Unlike the traditional array stores values like String, integer, Boolean, etc an Array of Objects stores objects that mean objects are stored as elements of an array. Note that when we say Array of Objects it is not the object itself that is stored in the array but the reference of the object.

Creating an array of objects:

An Array of Objects is created using the object class, and we know Object class is the root class of all Classes.

We use the Class_Name followed by a square bracket [] then object reference name to create an Array of Objects.

SYNTAX:

Single dimensional Array:

```
class_Name obj[ ]= new class_Name[Array_Length];
```

```
class_Name []obj = new class_Name[Array_Length];
```

Array initialisation:

```
Obj[i] = new class_Name(parameters);
```

PROGRAM:

```
import java.util.Scanner;  
class Book{  
    private String book_name;  
    private long ISBN_Number;  
    private String author_name;  
    private String publisher;  
  
    String getBook_name() {  
        return book_name;  
    }  
}
```

```

    }
    void setBook_name(String book_name) {
        this.book_name = book_name;
    }
    long getISBN_Number() {
        return ISBN_Number;
    }
    void setISBN_Number(long ISBN_Number) {
        this.ISBN_Number = ISBN_Number;
    }
    String getAuthor_name() {
        return author_name;
    }
    void setAuthor_name(String author_name) {
        this.author_name = author_name;
    }
    String getPublisher() {
        return publisher;
    }
    void setPublisher(String publisher) {
        this.publisher = publisher;
    }
    void description() {
        System.out.println("The
description:\n"+"BookName:"+getBook_name()+"\nISBN
number:"+getISBN_Number()+"\nAuthor name:"+getAuthor_name()+"\n"
+ "Publisher:"+getPublisher());
    }
}
public class LabProgram4 {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Book b[]=new Book[30];
        Scanner sc=new Scanner(System.in);
        int n,i,isbn;
        String author,publisher,name;
        System.out.println("Enter the number of books:");
        n=sc.nextInt();
        for(i=0;i<n;i++) {
            b[i]=new Book();
            System.out.println("Book Details-"+(i+1));
            if(i==0)
                sc.nextLine();
            System.out.println("Enter the name of book:");
            name=sc.nextLine();
            b[i].setBook_name(name);
            System.out.println("Enter the ISBN of book:");
            isbn=sc.nextInt();
            b[i].setISBN_Number(isbn);
            sc.nextLine();
        }
    }
}

```

```
        System.out.println("Enter the name of author:");
        author=sc.nextLine();
        b[i].setAuthor_name(author);

        System.out.println("Enter the publisher:");
        publisher=sc.nextLine();
        b[i].setPublisher(publisher);
    }
    for(i=0;i<n;i++) {
        System.out.println("Book-"+(i+1));
        b[i].description();
    }
    sc.close();
}
}
```


OUTPUT:

```
Console X
<terminated> LabProgram4 [Java Application] C:\Program Files\Ja
Enter the number of books:
3
Book Details-1
Enter the name of book:
MOTIVATION
Enter the ISBN of book:
159753
Enter the name of author:
A.PRASHANTH
Enter the publisher:
AP PUBLISHERS
Book Details-2
Enter the name of book:
INSPIRING
Enter the ISBN of book:
753951
Enter the name of author:
K.RAMA KISHORE
Enter the publisher:
A.P PUBLISHERS
Book Details-3
Enter the name of book:
CHOOSE RIGHT WAY FROM YOUR MENTOR
Enter the ISBN of book:
852741
Enter the name of author:
GEETHA LALITHA
Enter the publisher:
AP PUBLICATIONS
Book-1
The description:
BookName:MOTIVATION
ISBN number:159753
Author name:A.PRASHANTH
Publisher:AP PUBLISHERS
Book-2
The description:
BookName:INSPIRING
<
```

```
Console X
<terminated> LabProgram4 [Java Application] C:\Program
AP PUBLISHERS
Book Details-2
Enter the name of book:
INSPIRING
Enter the ISBN of book:
753951
Enter the name of author:
K.RAMA KISHORE
Enter the publisher:
A.P PUBLISHERS
Book Details-3
Enter the name of book:
CHOOSE RIGHT WAY FROM YOUR MENTOR
Enter the ISBN of book:
852741
Enter the name of author:
GEETHA LALITHA
Enter the publisher:
AP PUBLICATIONS
Book-1
The description:
BookName:MOTIVATION
ISBN number:159753
Author name:A.PRASHANTH
Publisher:AP PUBLISHERS
Book-2
The description:
BookName:INSPIRING
ISBN number:753951
Author name:K.RAMA KISHORE
Publisher:A.P PUBLISHERS
Book-3
The description:
BookName:CHOOSE RIGHT WAY FROM YOUR MENTOR
ISBN number:852741
Author name:GEETHA LALITHA
Publisher:AP PUBLICATIONS
|
<
```

EXPERIMENT NO: 5

AIM:

Write a JAVA program to search for an element in a given list of elements using binary search mechanism.

DESCRIPTION:

BINARY SEARCH:

It is used to Search element in a sorted array by repeatedly dividing the search interval in half. Begin with an interval covering the whole array. If the value of the search key is less than the item in the middle of the interval, narrow the interval to the lower half. Otherwise, narrow it to the upper half. Repeatedly check until the value is found or the interval is empty.

Algorithm:

1. x is the element to be searched in the given sorted array
2. Compare x with the middle element.
3. If x matches with the middle element, we return the mid index.
4. Else If x is greater than the mid element, then x can only lie in the right half subarray after the mid element. So we recur for the right half.
5. Else (x is smaller) recur for the left half.

Static Method:

Static method in Java is a method which belongs to the class and not to the object. A static method can access only static data. It is a method which belongs to the class and not to the object (instance).

A static method can call only other static methods and cannot call a non-static method from it.

A static method can be accessed directly by the class name and doesn't need any

object A static method cannot refer to "this" or "super" keywords in anyway

SYNTAX:

```
for(initialization; condition; increment/decrement){
```

//statement or code to be executed

}

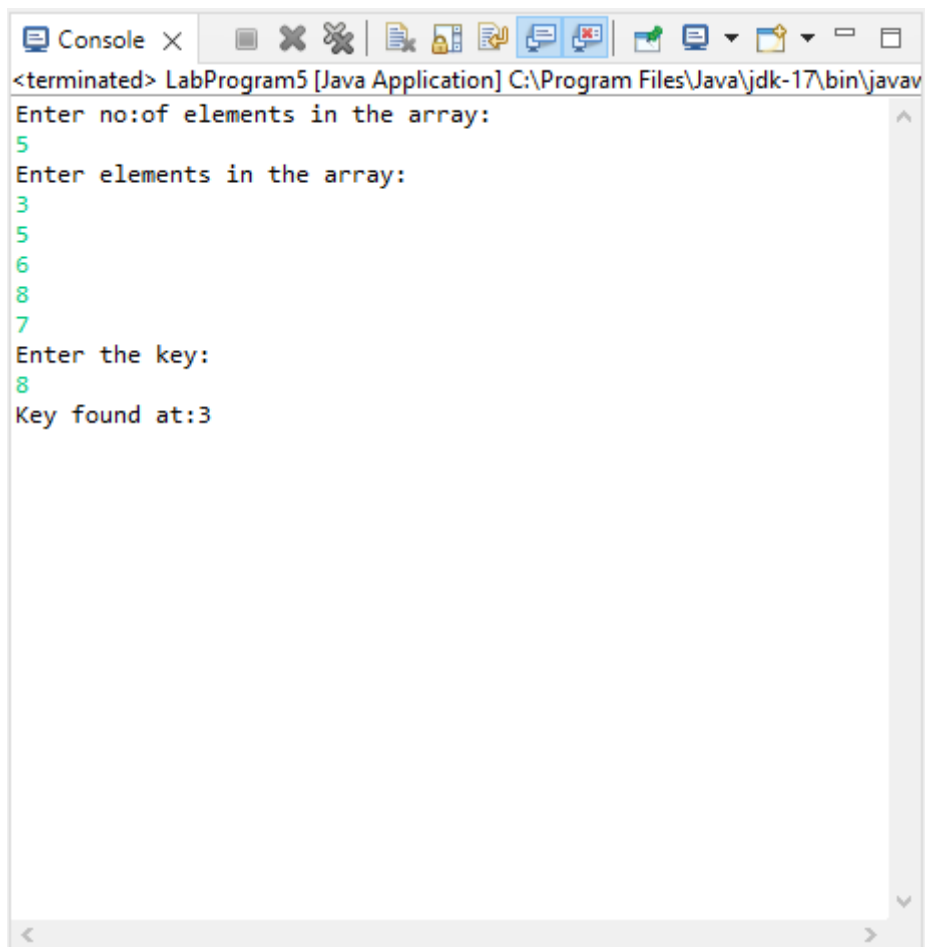
PROGRAM:

```
import java.util.Scanner;
class BinarySearch{
    int a[]=new int[100];
    int key,n;
    BinarySearch(int a[],int b,int c){
        this.a=a;
        n=b;
        key=c;
    }
    void binarysearch() {
        int mid,lb=0,ub=n-1;
        while(lb<=ub) {
            mid=(ub+lb)/2;
            if(key==a[mid]) {
                System.out.println("Key found at:"+mid);
                System.exit(0);
            }
            else
                if(a[mid]>key)
                    ub=mid-1;
            else
                lb=mid+1;
        }
        System.out.println("Key not found");
    }
}
public class LabProgram5 {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Scanner sc=new Scanner(System.in);
        int a[]=new int[100];
        int n,i,key,p;
        System.out.println("Enter no:of elements in the array:");
        n=sc.nextInt();
        System.out.println("Enter elements in the array:");
        for(i=0;i<n;i++) {
            a[i]=sc.nextInt();
        }
        System.out.println("Enter the key:");
        key=sc.nextInt();
        BinarySearch b1=new BinarySearch(a,n,key);
        b1.binarysearch();
    }
}
```

```
        sc.close();  
    }  
}
```

OUTPUT:



```
<terminated> LabProgram5 [Java Application] C:\Program Files\Java\jdk-17\bin\javaw  
Enter no:of elements in the array:  
5  
Enter elements in the array:  
3  
5  
6  
8  
7  
Enter the key:  
8  
Key found at:3
```

EXPERIMENT NO: 6

AIM:

Write a Java program that implements Merge sort algorithm for sorting and also shows the number of interchanges occurred for the given set of integers.

DESCRIPTION:

The Merge Sort algorithm and its implementation in Java:

Merge sort is one of the most efficient sorting techniques and it's based on the **DIVIDE AND CONQUER PARADIGM**

The algorithm can be described as the following 2 step process:

1.Divide: In this step, we divide the input array into 2 halves, the pivot being the midpoint of the array. This step is carried out recursively for all the half arrays until there are no more half arrays to divide.

2.Conquer: In this step, we sort and merge the divided arrays from bottom to top and get the sorted array.

For the implementation, we'll write a mergeSort function which takes in the input array and its length as the parameters. This will be a recursive function so we need the base and the recursive conditions.

The base condition checks if the array length is 1 and it will just return. For the rest of the cases, the recursive call will be executed.

For the recursive case, we get the middle index and create two temporary arrays l[] and r[]. The mergeSort function is then called recursively for both the sub-arrays:

We then call the merge function which takes in the input and both the sub-arrays and the starting and end indices of both the sub arrays.

The merge function compares the elements of both sub-arrays one by one and places the smaller element into the input array.

The time complexity will come to **$O(n \log n)$** .

SYNTAX:

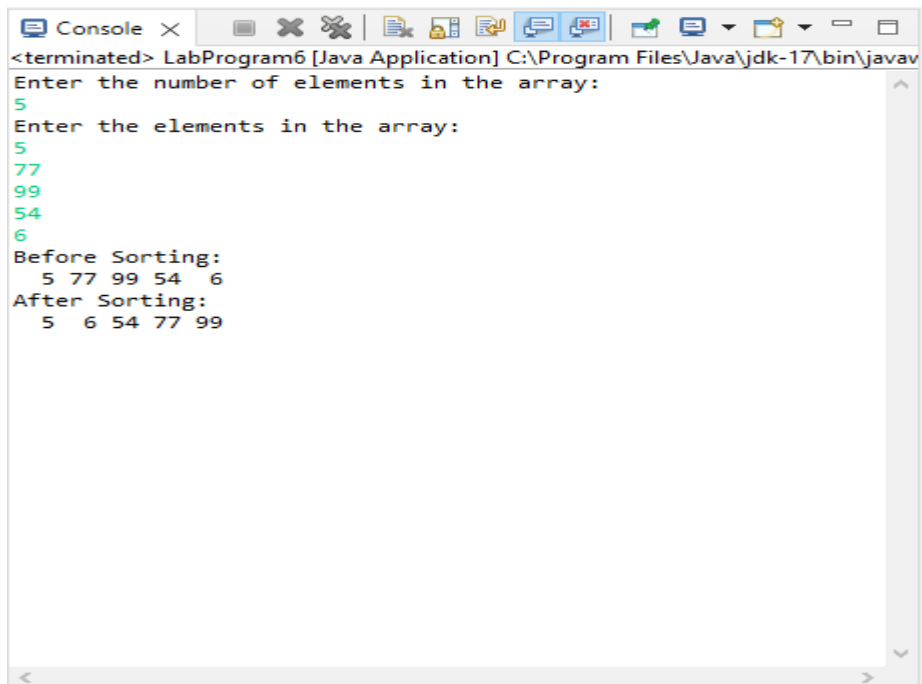
```
//recursive    function    returntype  
  
function_name(parameters)  
{  
  
if(condition)//terminating condition  
{  
  
    function_name(parameters);  
  
}  
  
}
```

PROGRAM:

```
import java.util.Scanner;  
public class LabProgram6 {  
    static void mergePass(int b[],int lb,33upees) {  
        int mid;  
        if(lb!=ub) {  
            mid=(lb+ub)/2;  
            mergePass(b,lb,mid);  
            mergePass(b,mid+1,ub);  
            mergeSort(b,lb,mid,ub);  
        }  
    }  
    static void mergeSort(int c[],int lb,int mid,33upees) {  
        int i,j,k;  
        int temp[]=new int[20];  
        i=lb;j=mid+1; k=lb;  
        while((i<=mid)&&(j<=ub)) {  
            if(c[i]<c[j])  
                temp[k++]=c[i++];  
            else  
                temp[k++]=c[j++];  
        }  
        while(i<=mid)  
            temp[k++]=c[i++];  
        while(j<=ub)  
            temp[k++]=c[j++];  
        for(i=lb;i<=ub;i++)
```

```
        c[i]=temp[i];
    }
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Scanner sc=new Scanner(System.in);
        int a[]= new int[100];
        int n;
        System.out.println("Enter the number of elements in the array:");
        n=sc.nextInt();
        System.out.println("Enter the elements in the array:");
        for(int i=0;i<n;i++)
            a[i]=sc.nextInt();
        System.out.println("Before Sorting:");
        for(int i=0;i<n;i++)
            System.out.printf("%3d",a[i]);
        mergePass(a,0,n-1);
        System.out.println("\nAfter Sorting:");
        for(int i=0;i<n;i++)
            System.out.printf("%3d",a[i]);
        sc.close();
    }
}
```

OUTPUT:



```
<terminated> LabProgram6 [Java Application] C:\Program Files\Java\jdk-17\bin\javaw
Enter the number of elements in the array:
5
Enter the elements in the array:
5
77
99
54
6
Before Sorting:
 5 77 99 54  6
After Sorting:
 5  6 54 77 99
```

EXPERIMENT NO: 7

AIM:

Write a java program to make rolling a pair of dice 10,000 times and counts the number of times doubles of are rolled for each different pair of doubles.

DESCRIPTION:

Java Math.random() method:

The **java.lang.Math.random()** is used to return a pseudorandom double type number greater than or equal to 0.0 and less than 1.0. The default random number always generated between 0 and 1. If you want to specific range of values, you have to multiply the returned value with the magnitude of the range.

For example: if you want to get the random number between 0 to 20, the resultant address has to be multiplied by 20 to get the desired result.

It returns a pseudorandom double value greater than or equal to 0.0 and less than 1.0.

SYNTAX:

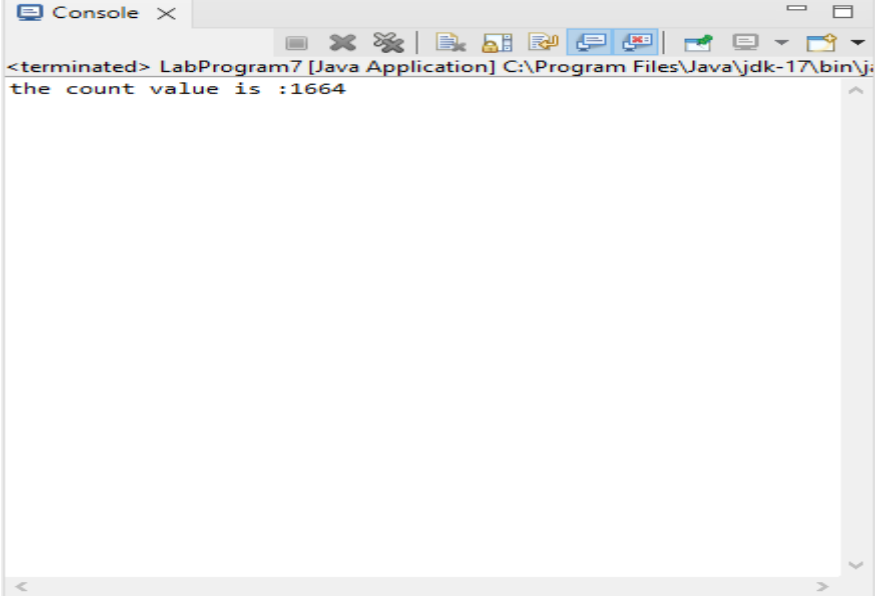
Math.random() //generate random double value

PROGRAM:

```
public class LabProgram7 {  
    int facevalue;  
    public int roll() {  
        facevalue = (int)((Math.random()*6)+1);  
        return facevalue;  
    }  
    public static void main(String[] args) {  
        LabProgram7 d1 = new LabProgram7();  
        LabProgram7 d2 = new LabProgram7();  
        int count = 0;  
        for(int i = 0; i<10000; i++){  
            int f1 = d1.roll();  
            int f2 = d2.roll();  
            if(f1 == f2){  
                count = count+1;  
            }  
        }  
        System.out.println("the count value is :" +count);  
    }  
}
```

}

OUTPUT:



```
<terminated> LabProgram7 [Java Application] C:\Program Files\Java\jdk-17\bin\j  
the count value is :1664
```

EXPERIMENT NO: 8

AIM:

Develop a java application with Employee class with Emp_name, Emp_id, Address, Mail_id, Mobile_no as members. Inherit the classes, Programmer, Assistant Professor, Associate Professor and Professor from employee class. Add Basic Pay (BP) as the member of all the inherited classes with 97% of BP as DA, 10 % of BP as HRA, 12% of BP as PF, 0.1% of BP for staff club fund. Generate pay slips for the employees with their gross and net salary.

DESCRIPTION:

Inheritance:

Inheritance in Java is a mechanism in which one object acquires all the properties and behaviors of a parent object. It is an important part of OOPs (Object Oriented programming system).

Terms used in Inheritance

Class: A class is a group of objects which have common properties. It is a template or blueprint from which objects are created.

Sub Class/Child Class: Subclass is a class which inherits the other class. It is also called a derived class, extended class, or child class.

Super Class/Parent Class: Superclass is the class from where a subclass inherits the features. It is also called a base class or a parent class.

Reusability: As the name specifies, reusability is a mechanism which facilitates you to reuse the fields and methods of the existing class when you create a new class. You can use the same fields and methods already defined in the previous class.

The **extends** keyword indicates that you are making a new class that derives from an existing class. The meaning of "extends" is to increase the functionality.

In the terminology of Java, a class which is inherited is called a parent or superclass, and the new class is called child or subclass.

SYNTAX:

class Subclass-name extends Superclass-name

```
{  
  
    //methods and fields  
  
}
```

PROGRAM:

```
import java.util.Scanner;  
class Employees{  
    String emp_name;  
    long emp_id;  
    String address;  
    String Mail_id;  
    long Mobile_no;  
    double hra,da,basicpay,pf,staffclub,gross,net;  
    public Employees(String name, long id, String add, String mail, long  
mobile) {  
        // TODO Auto-generated constructor stub  
        emp_name=name;  
        emp_id=id;  
        address=add;  
        Mail_id=mail;  
        Mobile_no=mobile;  
    }  
    void display() {  
        System.out.println("EMP NAME:"+emp_name);  
        System.out.println("EMP ID:"+emp_id);  
        System.out.println("EMP ADDRESS:"+address);  
        System.out.println("EMP MAIL ID:"+Mail_id);  
        System.out.println("EMP MOBILE NUMBER:"+Mobile_no);  
    }  
    void see() {  
        System.out.println("BASIC PAY:"+basicpay);  
        System.out.println("DA:"+da);  
        System.out.println("HRA:"+hra);  
        System.out.println("GROSS SALARY:"+gross);  
        System.out.println("NET SALARY:"+net);  
    }  
}  
class Programmer extends Employees{  
    Programmer(String name, long id,String add,String mail,long mobile){  
        super(name,id,add,mail,mobile);  
    }  
    void setBasicPay(double p) {  
        basicpay=p;  
    }  
    void bill() {  
        da=basicpay*0.97;  
        hra=basicpay*0.1;  
    }  
}
```

```

        pf=basicpay*0.12;
        staffclub=basicpay*(0.1/100);
        gross=basicpay+da+hra+pf;
        net=gross-pf-staffclub;
        display();
        see();
    } }

class Assistant_Professor extends Employees{
    Assistant_Professor(String name, long id,String add,String mail,long
mobile){
        super(name,id,add,mail,mobile);
    }
    void setBasicPay(double p) {
        basicpay=p;
    }
    void bill() {
        da=basicpay*0.94;
        hra=basicpay*0.09;
        pf=basicpay*0.13;
        staffclub=basicpay*(0.09/100);
        gross=basicpay+da+hra+pf;
        net=gross-pf-staffclub;
        display();
        see();
    }
}

class Associate_Professor extends Employees{
    Associate_Professor(String name, long id,String add,String mail,long
mobile){
        super(name,id,add,mail,mobile);
    }
    void setBasicPay(double p) {
        basicpay=p;
    }
    void bill() {
        da=basicpay*0.96;
        hra=basicpay*0.2;
        pf=basicpay*0.11;
        staffclub=basicpay*(0.1/100);
        gross=basicpay+da+hra+pf;
        net=gross-pf-staffclub;
        display();
        see();
    }
}

class Professor extends Employees{
    Professor(String name, long id,String add,String mail,long mobile){
        super(name,id,add,mail,mobile);
    }
    void setBasicPay(double p) {
        basicpay=p;
    }
}

```



```

    }
    void bill() {
        da=basicpay*0.97;
        hra=basicpay*0.1;
        pf=basicpay*0.12;
        staffclub=basicpay*(0.1/100);
        gross=basicpay+da+hra+pf;
        net=gross-pf-staffclub;
        display();
        see();
    }
}

public class LabProgram8 {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Scanner sc=new Scanner(System.in);
        System.out.println("ENTER THE NAME:");
        String name=sc.nextLine();
        System.out.println("ENTER THE ID:");
        long id=sc.nextLong();
        sc.nextLine();
        System.out.println("ENTER THE ADDRESS:");
        String add=sc.nextLine();
        System.out.println("ENTER THE MAIL ID:");
        String mail=sc.nextLine();
        System.out.println("ENTER MOBILE NO:");
        long mob=sc.nextLong();
        System.out.println("EMPLOYEE TYPE");
        System.out.println("1.PROGRAMMER");
        System.out.println("2.ASSISTANT PROFESSOR");
        System.out.println("3.ASSOCIATE PROFESSOR");
        System.out.println("4.PROFESSOR");
        int choice;
        System.out.println("\nEnter YOUR CHOICE");
        choice=sc.nextInt();
        switch(choice) {
            case 1: Programmer e1=new Programmer(name,id,add,mail,mob);
                System.out.println("ENTER THE BASIC PAY:");
                double m=sc.nextDouble();
                e1.setBasicPay(m);
                e1.bill();
                break;
            case 2: Assistant_Professor e2= new
Assistant_Professor(name,id,add,mail,mob);
                System.out.println("ENTER THE BASIC PAY:");
                double m1=sc.nextDouble();
                e2.setBasicPay(m1);
                e2.bill();
                break;
            case 3: Associate_Professor e3=new
Associate_Professor(name,id,add,mail,mob);

```

```

        System.out.println("ENTER THE BASIC PAY:");
        double m2=sc.nextDouble();
        e3.setBasicPay(m2);
        e3.bill();
        break;
    case 4:Professor e4=new Professor(name,id,add,mail,mob);
        System.out.println("ENTER THE BASIC PAY:");
        double m3=sc.nextDouble();
        e4.setBasicPay(m3);
        e4.bill();
        break;
    }
}
}

```

OUTPUT:

```
<terminated> LabProgram8 [Java Application] C:\Program
ENTER THE NAME:
BELLA VENKATA PRAVEEN KUMAR
ENTER THE ID:
20021005
ENTER THE ADDRESS:
SANGADIGUNTA,GUNTUR
ENTER THE MAIL ID:
praveen@mail.com
ENTER MOBILE NO:
35486668865
EMPLOYEE TYPE
1.PROGRAMMER
2.ASSISTANT PROFESSOR
3.ASSOCIATE PROFESSOR
4.PROFESSOR

ENTER YOUR CHOICE
1
ENTER THE BASIC PAY:
150000
EMP NAME:BELLA VENKATA PRAVEEN KUMAR
EMP ID:20021005
EMP ADDRESS:SANGADIGUNTA,GUNTUR
EMP MAIL ID:praveen@mail.com
EMP MOBILE NUMBER:35486668865
BASIC PAY:150000.0
DA:145500.0
HRA:15000.0
GROSS SALARY:328500.0
NET SALARY:310350.0
```

```
Console X
<terminated> LabProgram8 [Java Application] C:\Progr
ENTER THE NAME:
SRI LAKSHMI BELLA
ENTER THE ID:
321486786
ENTER THE ADDRESS:
PRAKASM,AP
ENTER THE MAIL ID:
SRILAKSHMI@MAIL.COM
ENTER MOBILE NO:
1654563548
EMPLOYEE TYPE
1.PROGRAMMER
2.ASSISTANT PROFESSOR
3.ASSOCIATE PROFESSOR
4.PROFESSOR

ENTER YOUR CHOICE
2
ENTER THE BASIC PAY:
159663
EMP NAME:SRI LAKSHMI BELLA
EMP ID:321486786
EMP ADDRESS:PRAKASM,AP
EMP MAIL ID:SRILAKSHMI@MAIL.COM
EMP MOBILE NUMBER:1654563548
BASIC PAY:159663.0
DA:150083.22
HRA:14369.67
GROSS SALARY:344872.07999999996
NET SALARY:323972.1933
```

```
Console X
<terminated> LabProgram8 [Java Application] C:\Program Files\Ja
ENTER THE NAME:
POOJI
ENTER THE ID:
2658754
ENTER THE ADDRESS:
ANDAMAN&NICOBAR
ENTER THE MAIL ID:
pooji@mail.com
ENTER MOBILE NO:
254867545687
EMPLOYEE TYPE
1.PROGRAMMER
2.ASSISTANT PROFESSOR
3.ASSOCIATE PROFESSOR
4.PROFESSOR

ENTER YOUR CHOICE
3
ENTER THE BASIC PAY:
154786
EMP NAME:POOJI
EMP ID:2658754
EMP ADDRESS:ANDAMAN&NICOBAR
EMP MAIL ID:pooji@mail.com
EMP MOBILE NUMBER:254867545687
BASIC PAY:154786.0
DA:148594.56
HRA:30957.2
GROSS SALARY:351364.22000000003
NET SALARY:334182.974
```

```
Console X
<terminated> LabProgram8 [Java Application] C:\Program Files\
ENTER THE NAME:
NARAYAN
ENTER THE ID:
346875
ENTER THE ADDRESS:
INDIA
ENTER THE MAIL ID:
narayan@mail.com
ENTER MOBILE NO:
23468535
EMPLOYEE TYPE
1.PROGRAMMER
2.ASSISTANT PROFESSOR
3.ASSOCIATE PROFESSOR
4.PROFESSOR

ENTER YOUR CHOICE
4
ENTER THE BASIC PAY:
234867
EMP NAME:NARAYAN
EMP ID:346875
EMP ADDRESS:INDIA
EMP MAIL ID:narayan@mail.com
EMP MOBILE NUMBER:23468535
BASIC PAY:234867.0
DA:227820.99
HRA:23486.7
GROSS SALARY:514358.73
NET SALARY:485939.823
```

EXPERIMENT NO: 9

AIM:

Write a Java Program to create an abstract class named Shape that contains two integers and an empty method named print Area(). Provide three classes named Rectangle, Triangle and Circle such that each one of the classes extends the class Shape. Each one of the classes contains only the method print Area () that prints the area of the givenshape.

DESCRIPTION:

ABSTRACT CLASS:

A class which is declared with the abstract keyword is known as an abstract class in Java It can have abstract and non-abstract methods (method with the body).

- An abstract class must be declared with an abstract keyword.
- It can have abstract and non-abstract methods.
- It cannot be instantiated.
- It can have constructors and static methods also.
- It can have final methods which will force the subclass not to change the body of the method.

SYNTAX:

```
abstract class A{ }
```

PROGRAM:

```
import java.util.Scanner;
abstract class Shape{
double dim1,dim2;
    Shape(double a,double b){
        dim1=a;
        dim2=b;
    }
    abstract void Area();
}
class Rectangle extends Shape{
    Rectangle(double a,double b){
        super(a,b);
    }
    void Area() {
        System.out.println("The area of the Rectangle:"+(dim1*dim2));
    }
}
class Triangle extends Shape{
```

```

Triangle(double a,double b){
    super(a,b);
}
void Area() {
    System.out.println("The area of the tringle:"+(0.5*dim1*dim2));
}
}
class Circle extends Shape{
    Circle(double a,double b){
        super(a,b);
    }
    void Area() {
        System.out.println("The area of the circle :"+(3.14*dim1*dim2));
    }
}
public class LabProgram9 {

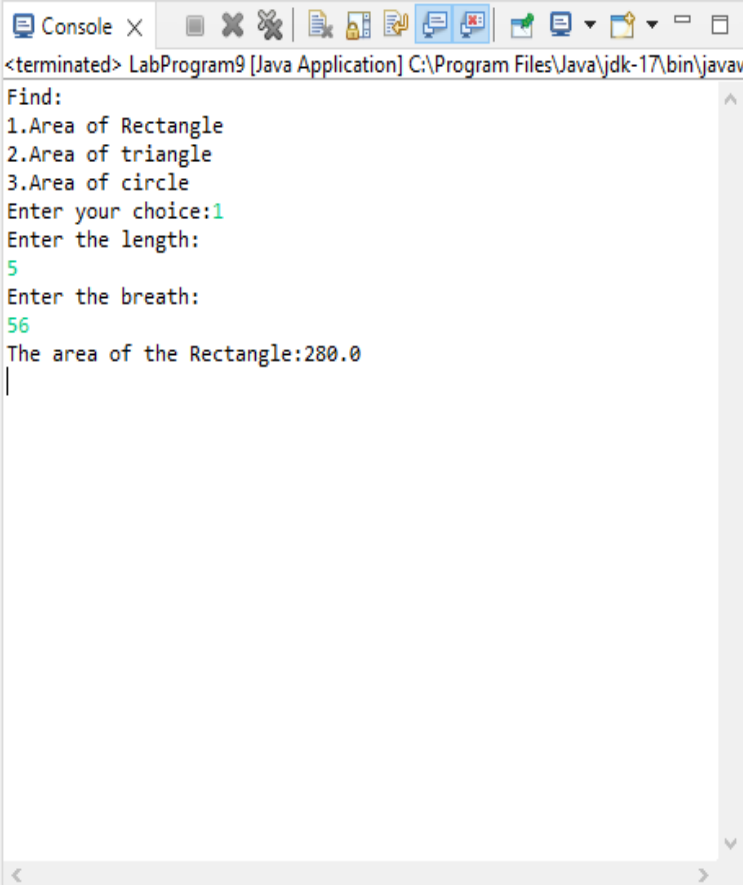
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Scanner sc=new Scanner(System.in);
        System.out.println("Find:\n1.Area of Rectangle\n2.Area of triangle"
            + "\n3.Area of circle");
        System.out.print("Enter your choice:");
        int choice;
        double a,b;
        choice=sc.nextInt();
        switch(choice) {
            case 1:
                System.out.println("Enter the length:");
                a=sc.nextDouble();
                System.out.println("Enter the breath:");
                b=sc.nextDouble();
                Rectangle r=new Rectangle(a,b);
                r.Area();
                break;
            case 2:
                System.out.println("Enter the base:");
                a=sc.nextDouble();
                System.out.println("Enter the height:");
                b=sc.nextDouble();
                Triangle t=new Triangle(a,b);
                t.Area();
                break;
            case 3:
                System.out.println("Enter the radius of the circle:");
                a=sc.nextDouble();
                Circle c=new Circle(a,a);
                c.Area();
                break;
            default:

```



```
        System.out.println("Wrong Choice");  
        break;  
    }  
    sc.close();  
}  
}
```

OUTPUT:



```
<terminated> LabProgram9 [Java Application] C:\Program Files\Java\jdk-17\bin\javav  
Find:  
1.Area of Rectangle  
2.Area of triangle  
3.Area of circle  
Enter your choice:1  
Enter the length:  
5  
Enter the breath:  
56  
The area of the Rectangle:280.0  
|
```

```
<terminated> LabProgram9 [Java Application] C:\Program Files\Java\jdk-17\bin\javav
Find:
1.Area of Rectangle
2.Area of triangle
3.Area of circle
Enter your choice:2
Enter the base:
20
Enter the height:
30
The area of the tringle:300.0
|
```

```
<terminated> LabProgram9 [Java Application] C:\Program Files\Java\jdk-17\bin\javaw
Find:
1.Area of Rectangle
2.Area of triangle
3.Area of circle
Enter your choice:3
Enter the radius of the circle:
20
The area of the circle :1256.0
```

EXPERIMENT NO: 10

AIM:

Develop a java application to implement currencyconverter(DollartoINR, EURO toINR,YentoINR and vice versa), distance converter (meter to KM, miles to KM and vice versa), timeconverter (hours to minutes, seconds and vice versa) using packages.

DESCRIPTION:

Package in Java is a mechanism to encapsulate a group of classes, sub packages and interfaces.

Advantage of Java Package

- 1) Java package is used to categorize the classes and interfaces so that they can be easily maintained.
- 2) Java package provides access protection.
- 3) Java package removes naming collision.

SYNTAX:

import package_name.*;\\for all classes import package_name.class_name;\\for some particular class package package_name;

PROGRAM:

PACKAGES:

CURRENCY CONVERTER:

```
package CurrencyConverter;

public class CurrencyConverterP {
    public double DollarIndian(double dollar) {
        return dollar*74.36;
    }
    public double IndianDollar(double ruppees) {
        return ruppees/74.36;
    }
    public double EuroIndian(double euro) {
        return euro*84.57;
    }
    public double IndianEuro(double 49upees) {
        return 49upees/84.57;
    }
    public double YentoIndian(double yen) {
```

```
        return yen*0.65;
    }
    public double IndiantoYen(double rupees) {
        return rupees/0.65;
    }
}
```

DISTANCE CONVERTER:

```
package DistanceConverter;

public class Distance {
    public double MetertoKm(double meter) {
        return meter/1000;
    }
    public double KmToMeter(double km) {
        return km*1000;
    }
    public double MilesToKm(double miles) {
        return miles*1.609;
    }
    public double KmToMiles(double km) {
        return km/1.609;
    }
}
```

TIME CONVERTER:

```
package DistanceConverter;

public class Distance {
    public double MetertoKm(double meter) {
        return meter/1000;
    }
    public double KmToMeter(double km) {
        return km*1000;
    }
    public double MilesToKm(double miles) {
        return miles*1.609;
    }
    public double KmToMiles(double km) {
        return km/1.609;
    }
}
```

```
import CurrencyConverter.*;
import DistanceConverter.*;
import TimeConverter.*;
import java.util.*;
public class LabProgram10 {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        CurrencyConverterP c=new CurrencyConverterP();
        Distance d=new Distance();
        Time t=new Time();
        Scanner sc =new Scanner(System.in);
        System.out.println("*****Conversions Menu*****");
        System.out.println("1.Currency Convertor");
        System.out.println("2.Distance Convertor");
        System.out.println("3.Time Convertor");
        System.out.print("Enter your choice:");
        int choice1=sc.nextInt();
        switch(choice1) {
            case 1: System.out.println("1.DOLLAR TO RUPEES");
                    System.out.println("2.RUPEES TO DOLLAR");
                    System.out.println("3.EURO TO RUPEES");
                    System.out.println("4.RUPEES TO EURO");
                    System.out.println("5.YEN TO RUPEES");
                    System.out.println("6.RUPEES TO YEN");
                    System.out.print("Enter your choice:");
                    int choice2=sc.nextInt();
                    switch(choice2) {

                        case 1:System.out.println("Enter the money in
dollars:");
                                double m=sc.nextDouble();
                                System.out.println("IN
INR:"+c.DollarIndian(m));
                                break;
                        case 2:System.out.println("Enter the money in
INR:");
                                double k=sc.nextDouble();
                                System.out.println("IN
DOLLARS:"+c.IndianDollar(k));
                                break;
                        case 3:System.out.println("Enter the money in
EURO:");
                                double e=sc.nextDouble();
                                System.out.println("IN
INR:"+c.EuroIndianI);
                                break;
                        case 4:System.out.println("Enter the money in
INR:");
                                double i=sc.nextDouble();
```

```

        System.out.println("IN
EURO:" + c.IndianEuro(i));
        break;
    case 5: System.out.println("Enter the money in
YEN:");
        double y = sc.nextDouble();
        System.out.println("IN
INR:" + c.YentoIndian(y));
        break;
    case 6: System.out.println("Enter the money in
INR:");
        double q = sc.nextDouble();
        System.out.println("IN
YEN:" + c.IndiantoYen(q));
        break;
    }
    break;
    case 2: System.out.println("1.Meter To KM");
        System.out.println("2.KM To Meters");
        System.out.println("3.Miles To KM");
        System.out.println("4.KM To Miles");
        System.out.print("Enter your choice:");
        int choice3 = sc.nextInt();
        switch(choice3) {
            case 1: System.out.println("Enter the distance in
METERS:");
                double d1 = sc.nextDouble();
                System.out.println("IN
KILOMETERS:" + d.MetertoKm(d1));
                break;
            case 2: System.out.println("Enter the distance in
KILOMETERS:");
                double d2 = sc.nextDouble();
                System.out.println("IN
METERS:" + d.KmToMeter(d2));
                break;
            case 3: System.out.println("Enter the distance in
MILES:");
                double d3 = sc.nextDouble();
                System.out.println("IN
KILOMETERS:" + d.MilesToKm(d3));
                break;
            case 4: System.out.println("Enter the distance in
KILOMETERS:");
                double d4 = sc.nextDouble();
                System.out.println("IN
MILES:" + d.KmToMiles(d4));
                break;
        }
    }
    break;
    case 3: System.out.println("1.HOUR TO MINUTE");

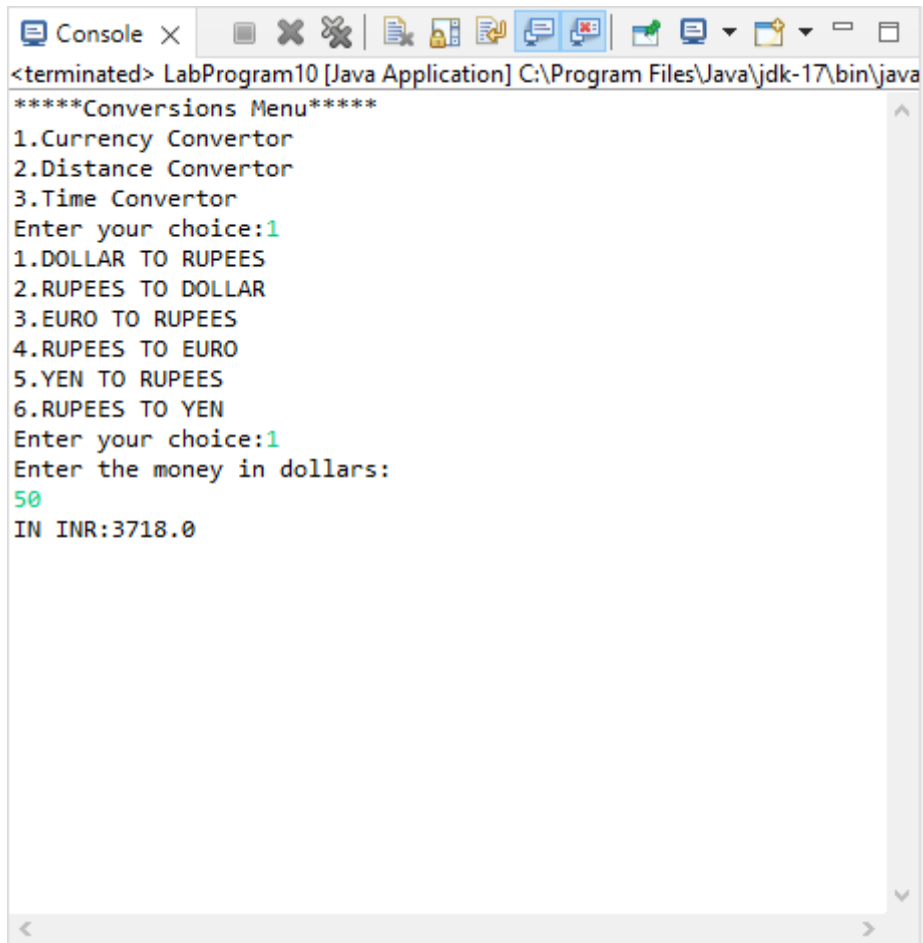
```

```

        System.out.println("2.MINUTE TO HOUR");
        System.out.println("3.HOUR TO SECONDS");
        System.out.println("4.SECONDS TO HOUR");
        System.out.print("Enter your choice:");
        int choice4=sc.nextInt();
        switch(choice4){
        case 1: System.out.println("Enter the time in
HOURS:");
                double t1=sc.nextDouble();
                System.out.println("IN
MINUTES:"+t.HourToMinute(t1));
                break;
        case 2: System.out.println("Enter the time in
MINUTES:");
                double t2=sc.nextDouble();
                System.out.println("IN
HOURS:"+t.MinuteToHour(t2));
                break;
        case 3: System.out.println("Enter the time in HOURS:");
                double t3=sc.nextDouble();
                System.out.println("IN
SECONDS:"+t.HourToSeconds(t3));
                break;
        case 4: System.out.println("Enter the time in
SECONDS:");
                double t4=sc.nextDouble();
                System.out.println("IN
HOUR:"+t.SecondsToHour(t4));
                break;
        }
        break;
    }
}

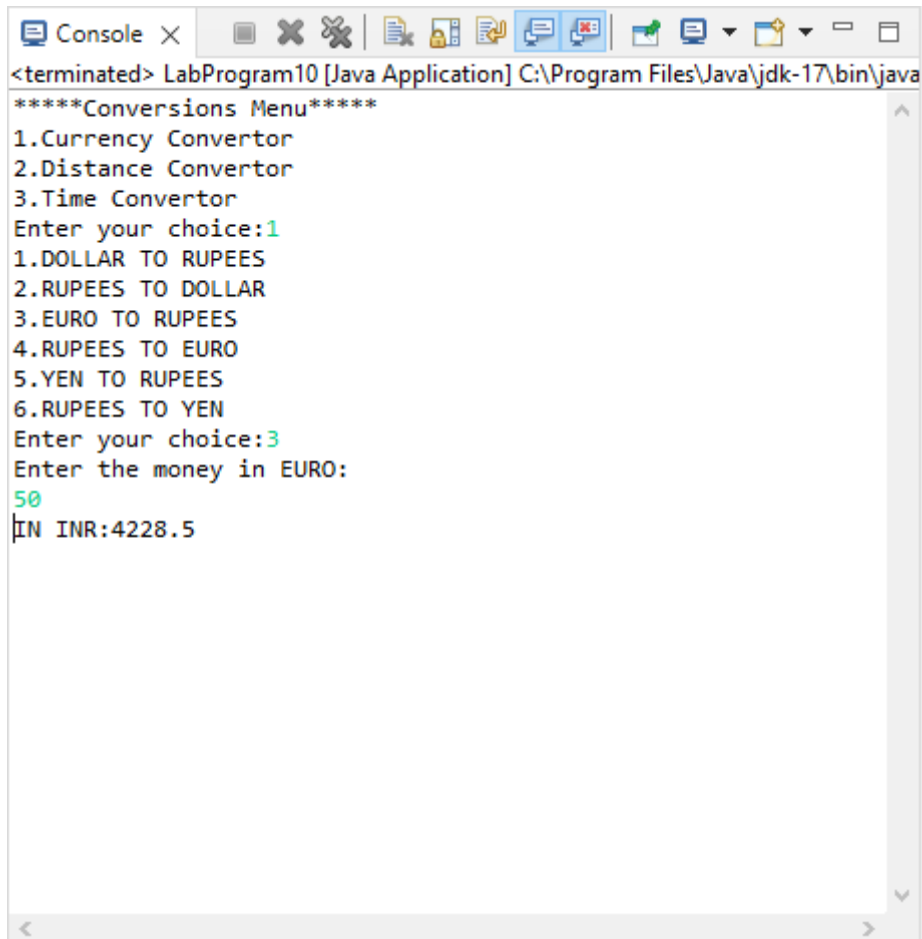
```


OUTPUT:

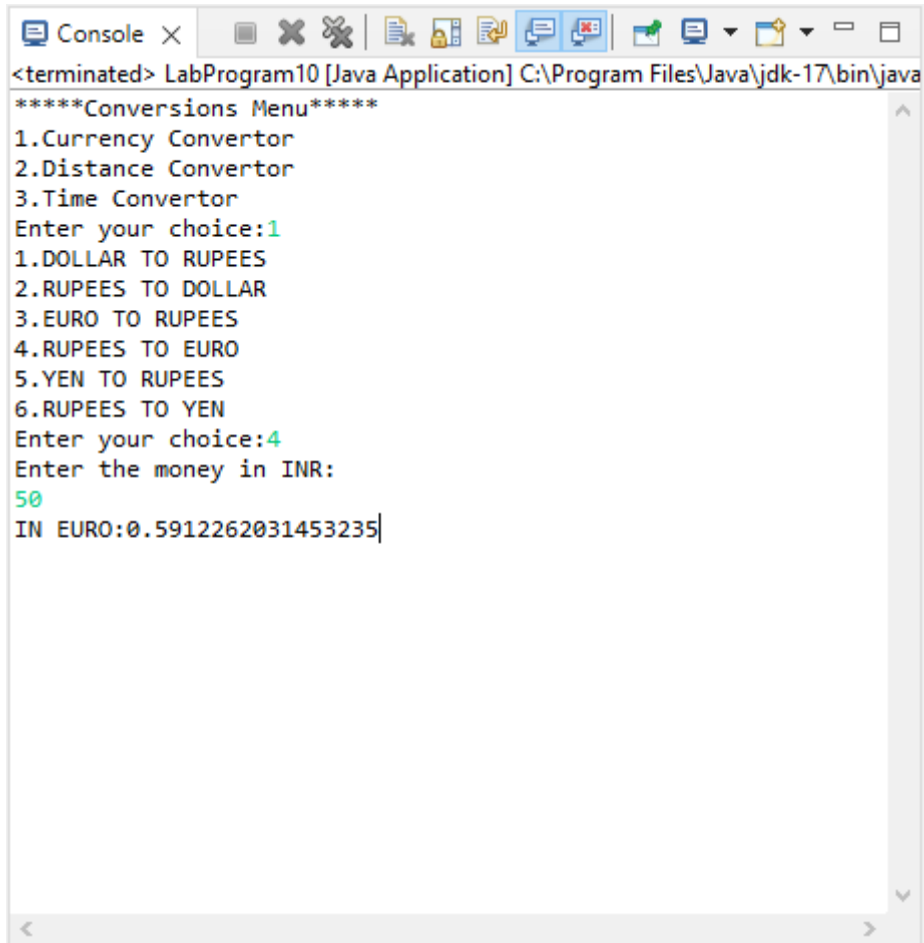


```
<terminated> LabProgram10 [Java Application] C:\Program Files\Java\jdk-17\bin\java
*****Conversions Menu*****
1.Currency Convertor
2.Distance Convertor
3.Time Convertor
Enter your choice:1
1.DOLLAR TO RUPEES
2.RUPEES TO DOLLAR
3.EURO TO RUPEES
4.RUPEES TO EURO
5.YEN TO RUPEES
6.RUPEES TO YEN
Enter your choice:1
Enter the money in dollars:
50
IN INR:3718.0
```

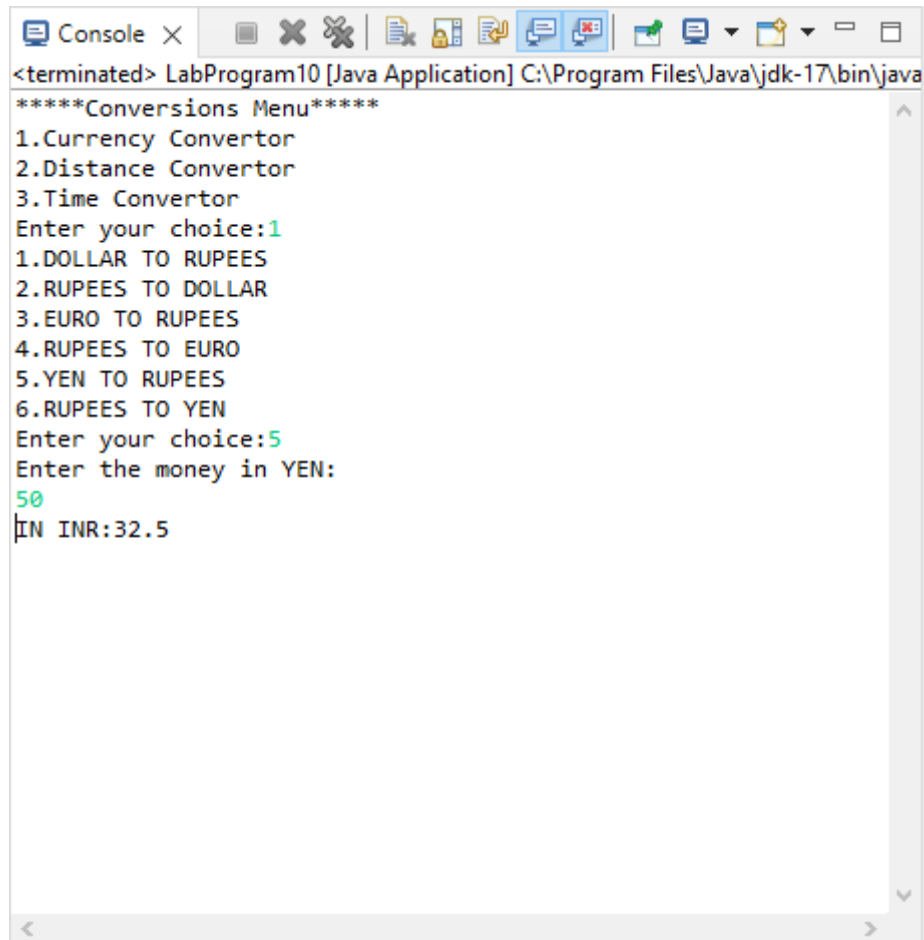
```
Console X [Icons]
<terminated> LabProgram10 [Java Application] C:\Program Files\Java\jdk-17\bin\java
*****Conversions Menu*****
1.Currency Convertor
2.Distance Convertor
3.Time Convertor
Enter your choice:1
1.DOLLAR TO RUPEES
2.RUPEES TO DOLLAR
3.EURO TO RUPEES
4.RUPEES TO EURO
5.YEN TO RUPEES
6.RUPEES TO YEN
Enter your choice:2
Enter the money in INR:
50
IN DOLLARS:0.6724045185583647
```



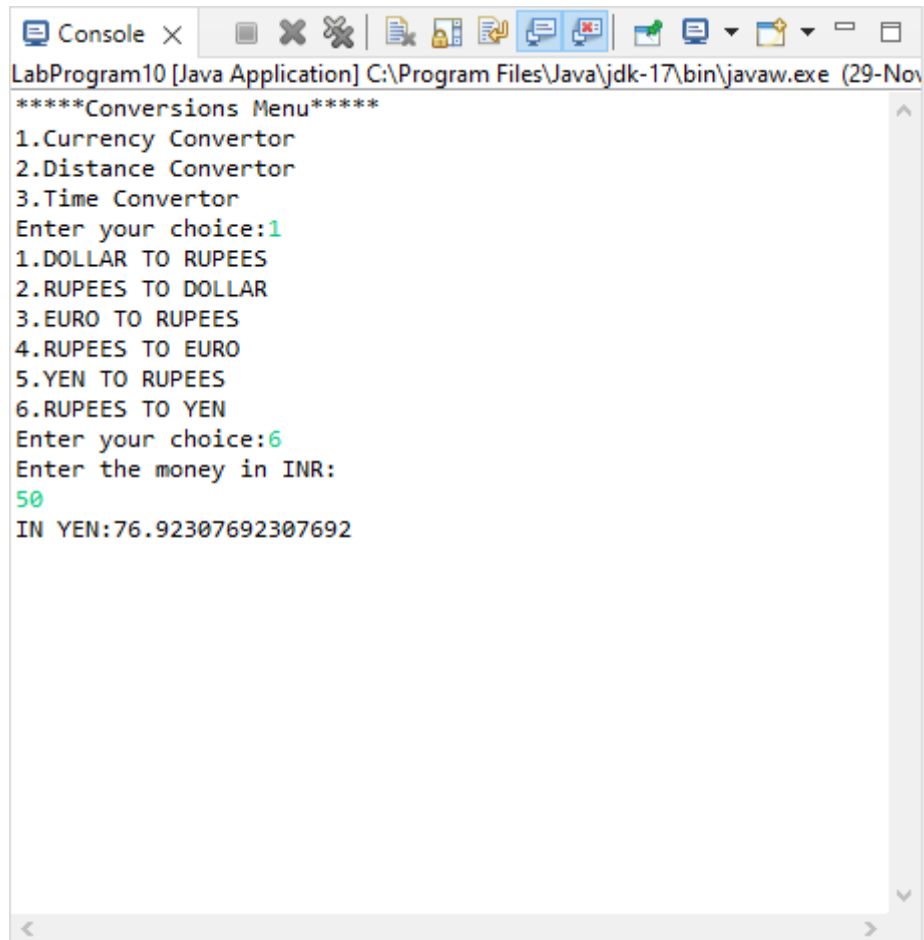
```
<terminated> LabProgram10 [Java Application] C:\Program Files\Java\jdk-17\bin\java
*****Conversions Menu*****
1.Currency Converter
2.Distance Converter
3.Time Converter
Enter your choice:1
1.DOLLAR TO RUPEES
2.RUPEES TO DOLLAR
3.EURO TO RUPEES
4.RUPEES TO EURO
5.YEN TO RUPEES
6.RUPEES TO YEN
Enter your choice:3
Enter the money in EURO:
50
IN INR:4228.5
```



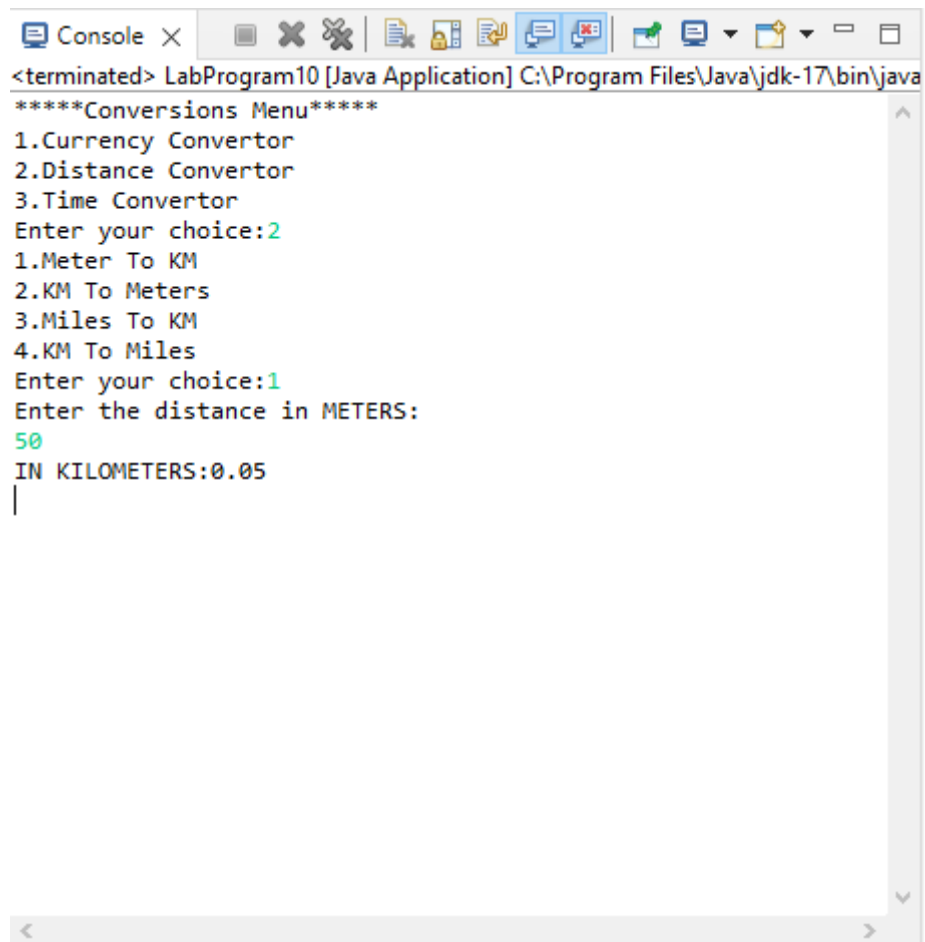
```
<terminated> LabProgram10 [Java Application] C:\Program Files\Java\jdk-17\bin\java
*****Conversions Menu*****
1.Currency Convertor
2.Distance Convertor
3.Time Convertor
Enter your choice:1
1.DOLLAR TO RUPEES
2.RUPEES TO DOLLAR
3.EURO TO RUPEES
4.RUPEES TO EURO
5.YEN TO RUPEES
6.RUPEES TO YEN
Enter your choice:4
Enter the money in INR:
50
IN EURO:0.5912262031453235|
```



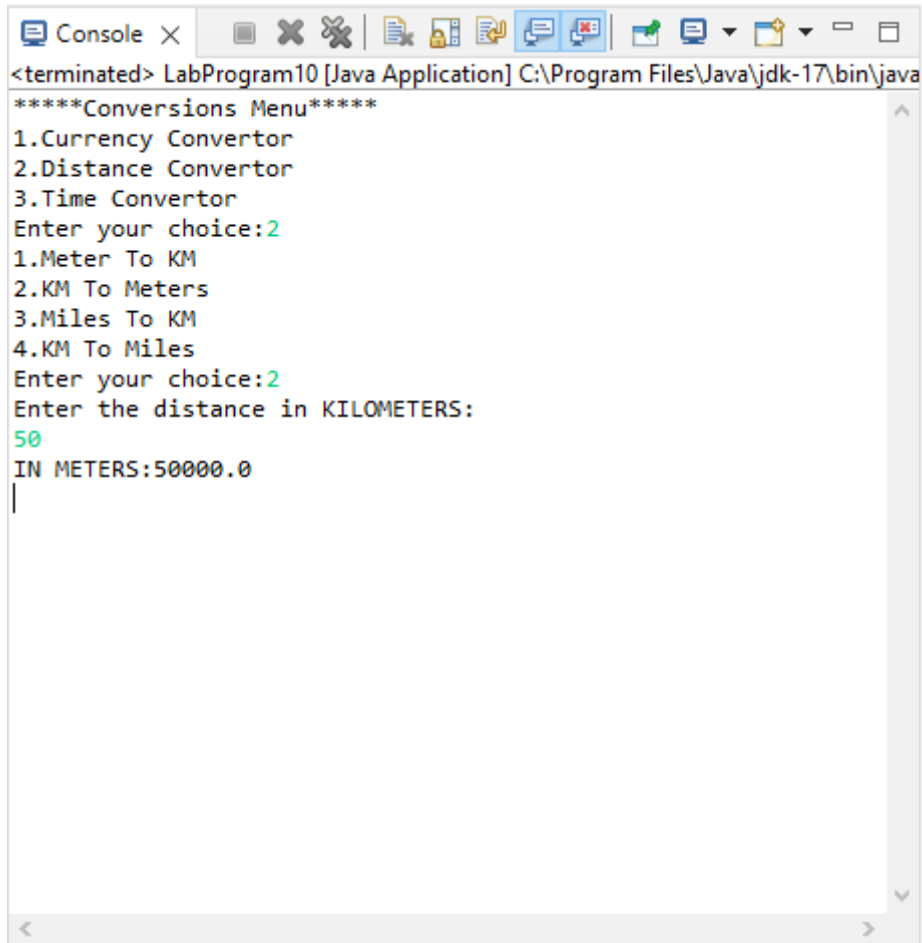
```
<terminated> LabProgram10 [Java Application] C:\Program Files\Java\jdk-17\bin\java
*****Conversions Menu*****
1.Currency Convertor
2.Distance Convertor
3.Time Convertor
Enter your choice:1
1.DOLLAR TO RUPEES
2.RUPEES TO DOLLAR
3.EURO TO RUPEES
4.RUPEES TO EURO
5.YEN TO RUPEES
6.RUPEES TO YEN
Enter your choice:5
Enter the money in YEN:
50
IN INR:32.5
```



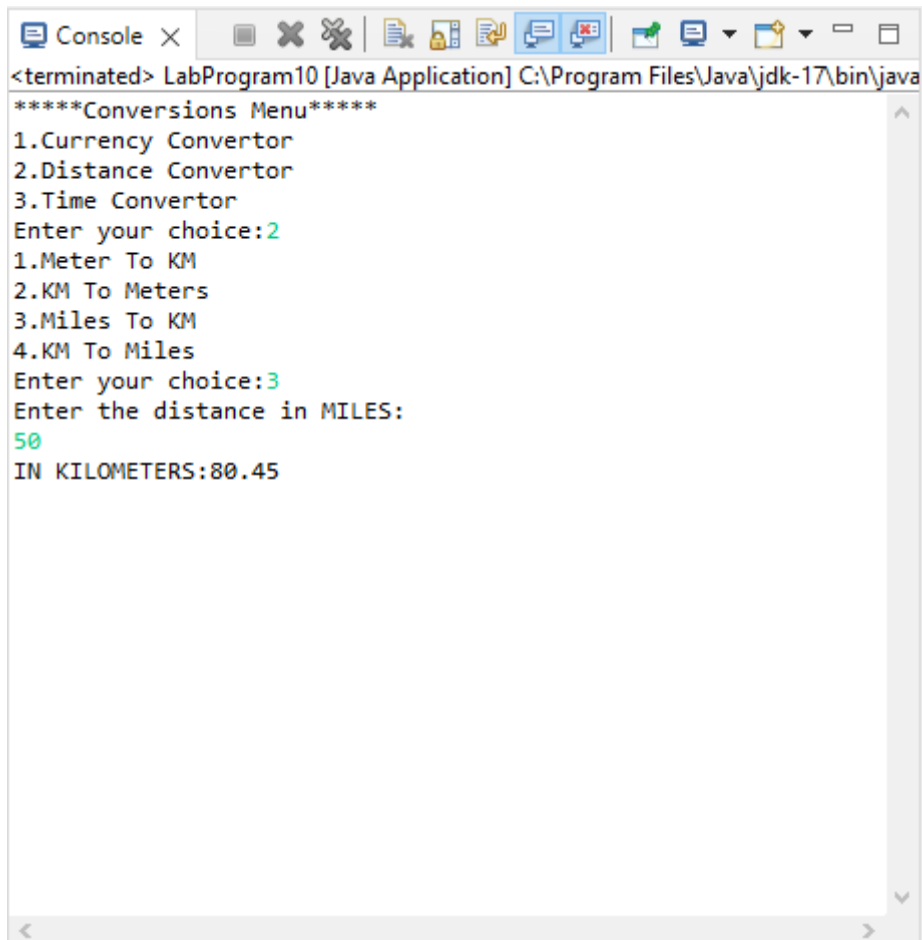
```
LabProgram10 [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe (29-Nov-2023)
*****Conversions Menu*****
1.Currency Convertor
2.Distance Convertor
3.Time Convertor
Enter your choice:1
1.DOLLAR TO RUPEES
2.RUPEES TO DOLLAR
3.EURO TO RUPEES
4.RUPEES TO EURO
5.YEN TO RUPEES
6.RUPEES TO YEN
Enter your choice:6
Enter the money in INR:
50
IN YEN:76.92307692307692
```



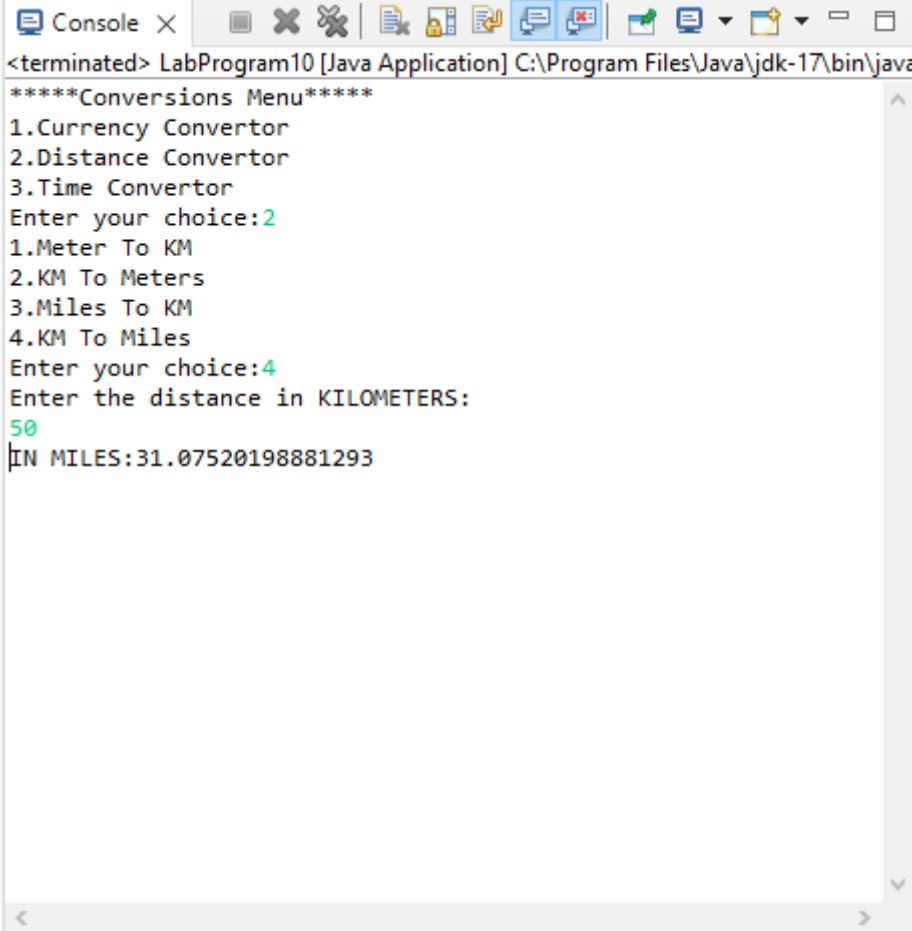
```
<terminated> LabProgram10 [Java Application] C:\Program Files\Java\jdk-17\bin\java
*****Conversions Menu*****
1.Currency Convertor
2.Distance Convertor
3.Time Convertor
Enter your choice:2
1.Meter To KM
2.KM To Meters
3.Miles To KM
4.KM To Miles
Enter your choice:1
Enter the distance in METERS:
50
IN KILOMETERS:0.05
|
```

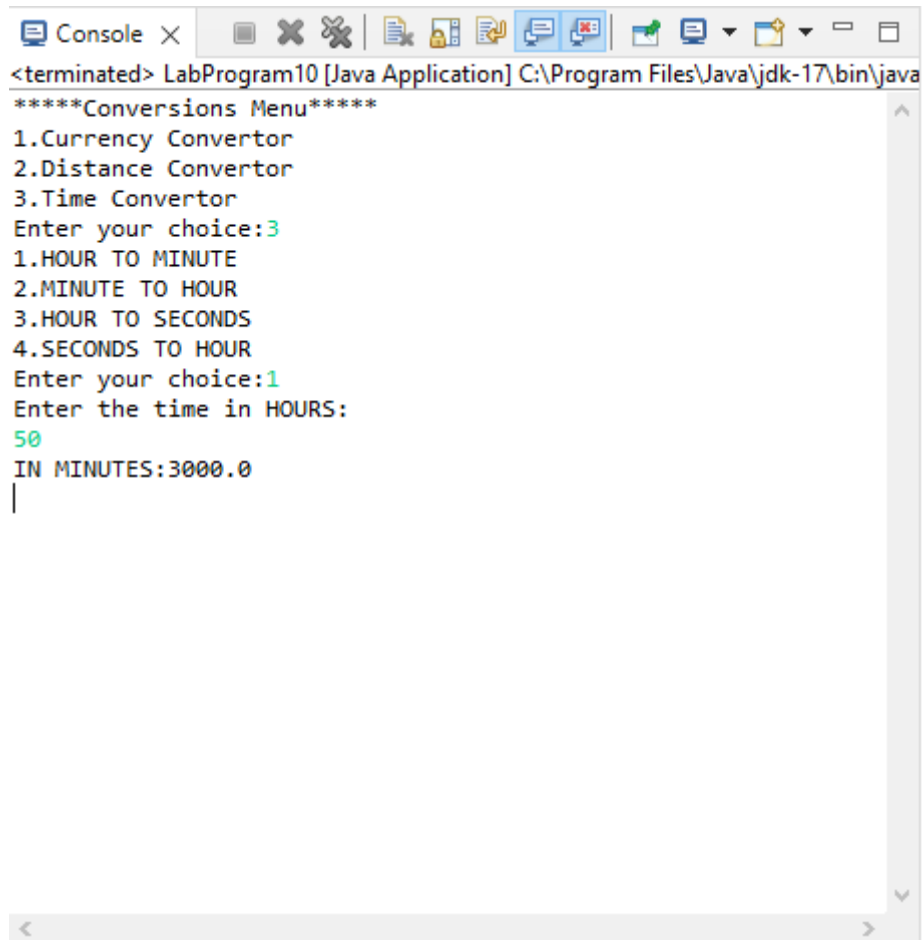
```
<terminated> LabProgram10 [Java Application] C:\Program Files\Java\jdk-17\bin\java
*****Conversions Menu*****
1.Currency Convertor
2.Distance Convertor
3.Time Convertor
Enter your choice:2
1.Meter To KM
2.KM To Meters
3.Miles To KM
4.KM To Miles
Enter your choice:2
Enter the distance in KILOMETERS:
50
IN METERS:50000.0
|
```



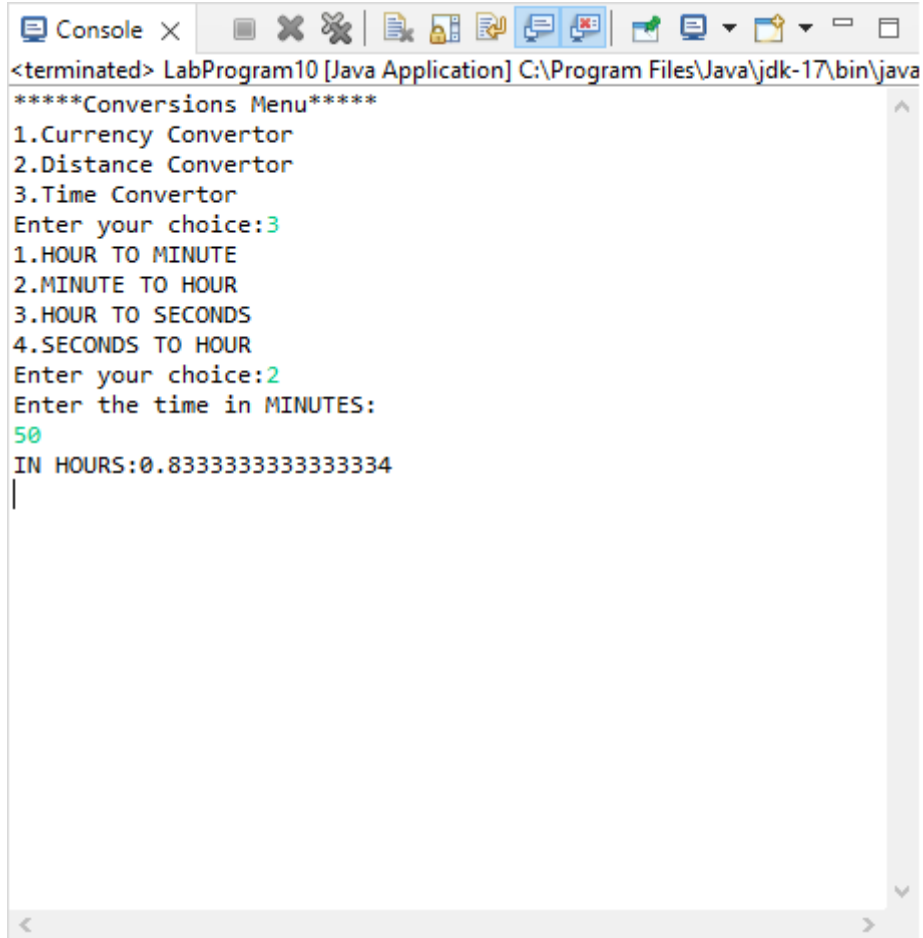
```
<terminated> LabProgram10 [Java Application] C:\Program Files\Java\jdk-17\bin\java
*****Conversions Menu*****
1.Currency Convertor
2.Distance Convertor
3.Time Convertor
Enter your choice:2
1.Meter To KM
2.KM To Meters
3.Miles To KM
4.KM To Miles
Enter your choice:3
Enter the distance in MILES:
50
IN KILOMETERS:80.45
```



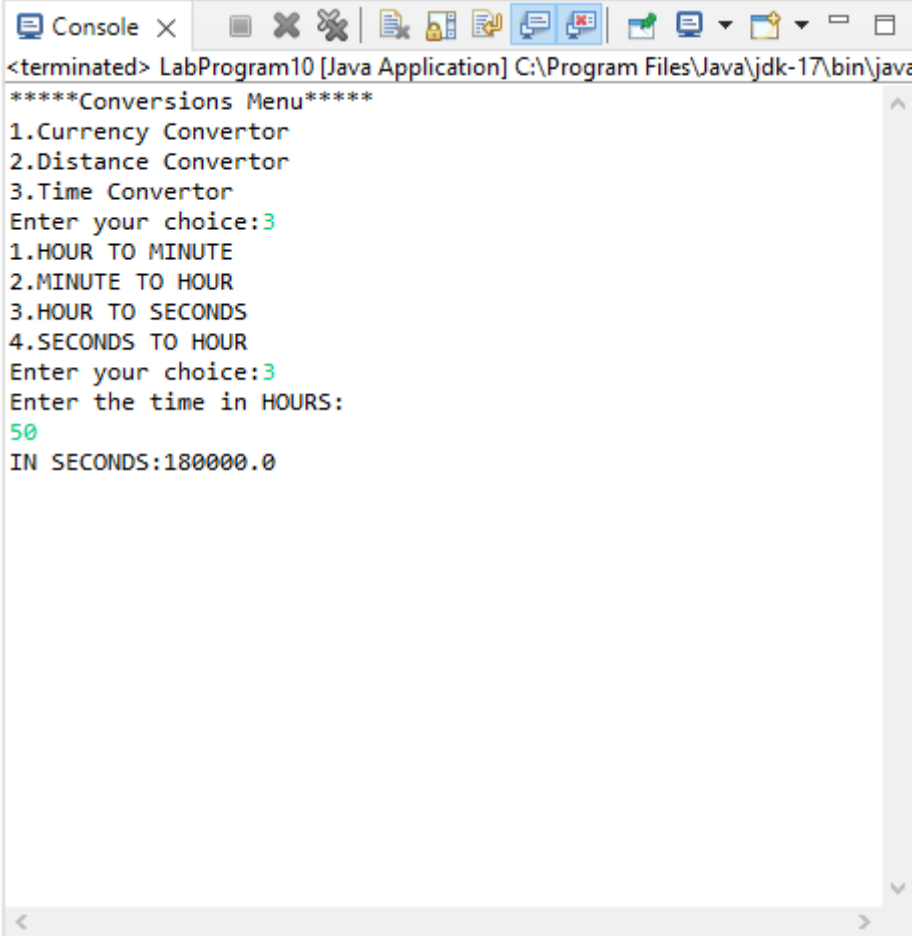
```
<terminated> LabProgram10 [Java Application] C:\Program Files\Java\jdk-17\bin\java
*****Conversions Menu*****
1.Currency Convertor
2.Distance Convertor
3.Time Convertor
Enter your choice:2
1.Meter To KM
2.KM To Meters
3.Miles To KM
4.KM To Miles
Enter your choice:4
Enter the distance in KILOMETERS:
50
IN MILES:31.07520198881293
```



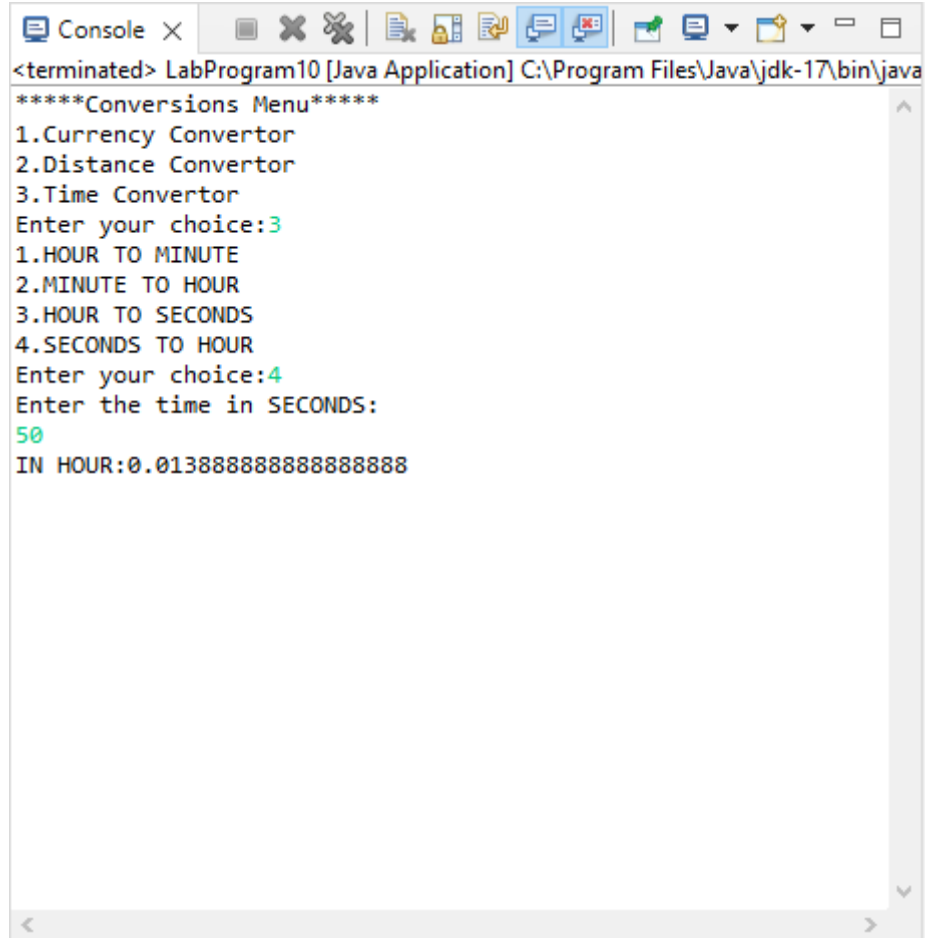
```
<terminated> LabProgram10 [Java Application] C:\Program Files\Java\jdk-17\bin\java
*****Conversions Menu*****
1.Currency Convertor
2.Distance Convertor
3.Time Convertor
Enter your choice:3
1.HOUR TO MINUTE
2.MINUTE TO HOUR
3.HOUR TO SECONDS
4.SECONDS TO HOUR
Enter your choice:1
Enter the time in HOURS:
50
IN MINUTES:3000.0
|
```



```
<terminated> LabProgram10 [Java Application] C:\Program Files\Java\jdk-17\bin\java
*****Conversions Menu*****
1.Currency Convertor
2.Distance Convertor
3.Time Convertor
Enter your choice:3
1.HOUR TO MINUTE
2.MINUTE TO HOUR
3.HOUR TO SECONDS
4.SECONDS TO HOUR
Enter your choice:2
Enter the time in MINUTES:
50
IN HOURS:0.8333333333333334
|
```



```
<terminated> LabProgram10 [Java Application] C:\Program Files\Java\jdk-17\bin\java
*****Conversions Menu*****
1.Currency Convertor
2.Distance Convertor
3.Time Convertor
Enter your choice:3
1.HOUR TO MINUTE
2.MINUTE TO HOUR
3.HOUR TO SECONDS
4.SECONDS TO HOUR
Enter your choice:3
Enter the time in HOURS:
50
IN SECONDS:180000.0
```



```
<terminated> LabProgram10 [Java Application] C:\Program Files\Java\jdk-17\bin\java
*****Conversions Menu*****
1.Currency Convertor
2.Distance Convertor
3.Time Convertor
Enter your choice:3
1.HOUR TO MINUTE
2.MINUTE TO HOUR
3.HOUR TO SECONDS
4.SECONDS TO HOUR
Enter your choice:4
Enter the time in SECONDS:
50
IN HOUR:0.013888888888888888
```


EXPERIMENT NO: 11

AIM:

Write a Java Program to Handle Arithmetic Exceptions and InputMismatchExceptions.

DESCRIPTION:

An exception is an unwanted or unexpected event, which occurs during the execution of a program i.e at run time, that disrupts the normal flow of the program's instructions.

Exceptions can be handled by using try-catch blocks. The try block contains the code which may raise an exception and the catch is used to handle the exception. A code or set of statements can be written in the catch block. The catch block is executed when there is an exception in the try block. There can be multiple catch blocks to handle multiple exceptions or a single catch block to handle multiple exceptions. The catch block is then followed by finally block. The finally block executes even if the program terminates.

SYNTAX:

```
try {  
    //code that may raise an exception  
} catch(<exceptionname> en1) {  
    //code that must be executed incase of an exception  
} catch(<exceptionname> en2) {  
    //code that must be executed incase of an exception  
}  
  
finally {  
    //this block executes even if the program terminates  
}  
To catch multiple exceptions using single catch block  
try { // catch block  
} catch (ExceptionType1 | Exceptiontype2 ex) {  
    // catch block  
}
```

PROGRAM:

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

public class ExceptionHandling {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        int num,dem,res=0;
        boolean loop=true;
        Scanner sc=new Scanner(System.in);
        do
        {
            try
            {
                System.out.print("Enter numerator:");
                num=sc.nextInt();
                System.out.print("Enter denominator:");
                dem=sc.nextInt();
                if(dem==0)
                    res=num/dem;
                System.out.println("Result:"+num/dem);
                loop=false;
            }
            catch(ArithmeticException e)
            {
                System.err.print("Exception: ArithmeticException ");
                System.out.println("Enter another number other than zero");
            }
            catch(InputMismatchException e)
            {
                System.err.print("Exception: InputMismatchException ");
                System.out.println("You must enter numbers only");
                sc.nextLine();
                System.out.print("Enter numerator:");
                num=sc.nextInt();
                System.out.print("Enter denominator:");
                dem=sc.nextInt();
                res=num/dem;
                System.out.println("Result:"+res);
                loop=false;
            }
        }while(loop);
    }
}
```

OUTPUT:

```
Console X
<terminated> ExceptionHandling [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe
Enter numerator:5
Enter denominator:0
Exception: ArithmeticException Enter another number other than zero
Enter numerator:25
Enter denominator:praveen
Exception: InputMismatchException You must enter numbers only
Enter numerator:25
Enter denominator:5
Result:5
```

EXPERIMENT NO: 12

AIM:

Write a multi-threaded Java program to print all numbers below 100,000 that are both prime and Fibonacci number (some examples are 2, 3, 5, 13, etc.). Design a thread that generates prime numbers below 100,000 and writes them into a pipe. Design another thread that generates Fibonacci numbers and writes them to another pipe. The main thread should read both the pipes to identify numbers common to both.

DESCRIPTION:

Java - PipedReader

The PipedReader class is used to read the contents of a pipe as a stream of characters. This class is used generally to read text.

PipedReader class must be connected to the same PipedWriter and are used by different threads.

Java - PipedWriter

The PipedWriter class is used to write java pipe as a stream of characters. This class is used generally for writing text. Generally PipedWriter is connected to a PipedReader and used by different threads.

Java Threads

Threads allows a program to operate more efficiently by doing multiple things at the same time.

Threads can be used to perform complicated tasks in the background without interrupting the main program.

SYNTAX:

```
PipedReader(int pipeSize)
```

```
//constructor
```

```
PipedWriter()//constructor
```

```
public class Main extends
```

```
Thread { public void run() {
```

```
System.out.println("This code is running in a thread");
```

```
}
```

```
}
```

public class Main implements

Runnable {public void run() {

```
System.out.println("This code is running in a thread");
```

```
}
```

```
}
```

PROGRAM:

```
import java.io.*;
import java.io.PipedWriter;
import java.io.PipedReader;
class fibonacci extends Thread
{
    PipedWriter fw=new PipedWriter();
    public PipedWriter getwrite()
    {
        return fw;
    }
    public void run()
    {
        super.run();
        fibo();
    }
    int f(int n)
    {
        if(n<2)
            return n;
        else
            return f(n-1)+f(n-2);
    }
    void fibo()
    {
        for(int i=2,fibv=0;(fibv=f(i))<100000;i++)
        {
            try{
                fw.write(fibv);
            }
            catch(IOException e){
            }
        }
    }
}
```

```

    }
}
class receiver extends Thread
{
    PipedReader fibr,primer;
    public receiver(fibonacci fib,prime pr)throws IOException
    {
        fibr=new PipedReader(fib.getwrite());
        primer=new PipedReader(pr.getwrite());
    }
    public void run()
    {
        int p=0,f=0;
        try{

            p=primer.read();
            f=fibr.read();
        }
        catch(IOException e)
        {
        }
        while(true)
        {
            try
            {
                if(p==f){

                    System.out.println (p);
                    p=primer.read();
                    f=fibr.read();
                }
                else if(f<p)
                    f=fibr.read();
                else
                    p=primer.read();
            }catch(IOException e)
            {System.exit(-1);
            }
        }
    }
}
class prime extends Thread
{
    PipedWriter pw=new PipedWriter();
    public PipedWriter getwrite()
    {
        return pw;
    }
}

```

```

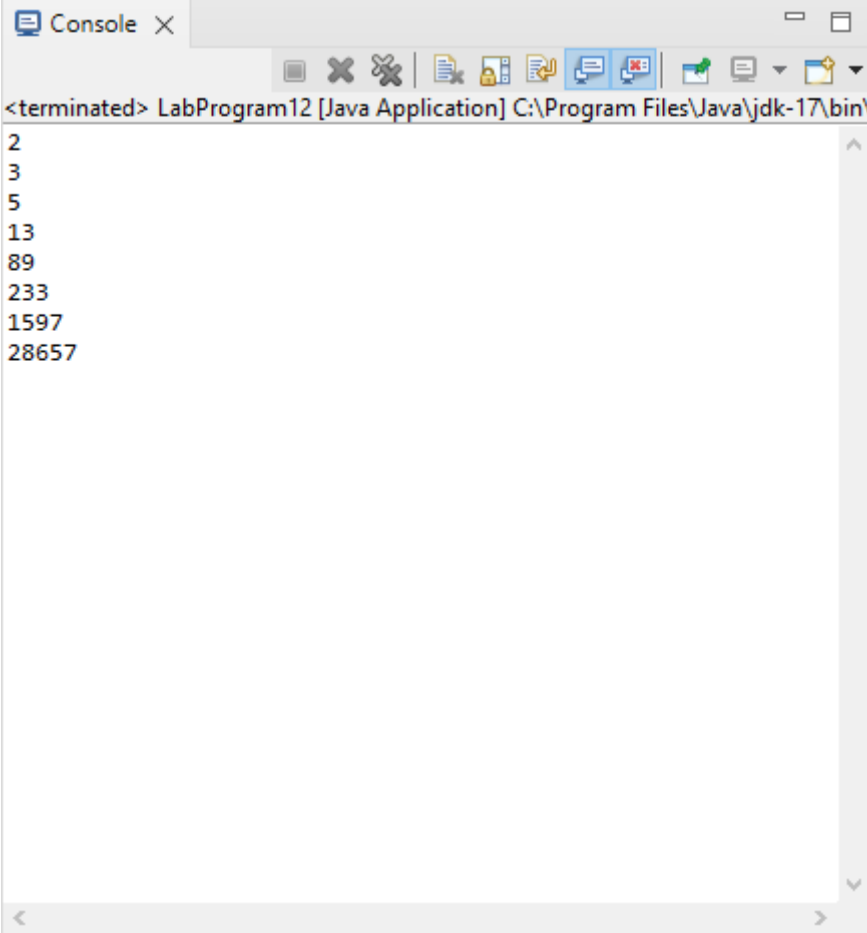
    public void run()
    {
        super.run();
        prim();
    }
    public void prim()
    {
        for(int i=2;i<100000;i++)
        {
            if(isprime(i))
            {
                try{
                    pw.write(i);
                }
                catch(IOException e){
                }
            }
        }
    }
    boolean isprime(int n)
    {
        boolean p=true;
        int s=(int)Math.sqrt(n);
        for(int i=2;i<=s;i++)
        {
            if(n%i==0)
                p=false;
        }
        return p;
    }
}

public class LabProgram12 {

    public static void main(String[] args)throws IOException {
        // TODO Auto-generated method stub
        fibonacci fi=new fibonacci();
        prime pri=new prime();
        receiver r=new receiver(fi,pri);
        fi.start();
        pri.start();
        r.start();
    }
}

```


OUTPUT:



```
<terminated> LabProgram12 [Java Application] C:\Program Files\Java\jdk-17\bin\  
2  
3  
5  
13  
89  
233  
1597  
28657
```

EXPERIMENT NO: 13

AIM:

Write a java program that implements a multi-threaded application that has three threads. First thread generates a random integer every 1 second and if the value is even, second thread computes the square of the number and prints. If the value is odd, the third thread will print the value of cube of the number.

DESCRIPTION:

MultiThreading:

Multithreading in java is a process of executing multiple threads simultaneously. A thread is a lightweight sub-process, the smallest unit of processing. Multiprocessing and multithreading, both are used to achieve multitasking. we use multithreading than multiprocessing because threads use a shared memory area. They don't allocate separate memory area so saves memory, and contextswitching between the threads takes less time than process. Java Multithreading is mostly used in games, animation, etc.

Thread :

A thread is a lightweight subprocess, the smallest unit of processing. It is a separate path of execution. Threads are independent. If there occurs exception in one thread, it doesn't affect other threads. It uses a shared memory area.

Multitasking is a process of executing multiple tasks simultaneously. We use multitasking to utilize the CPU. Multitasking can be achieved in two ways:

Process-based Multitasking (Multiprocessing)

Thread-based Multitasking (Multithreading)

Methods in thread class:

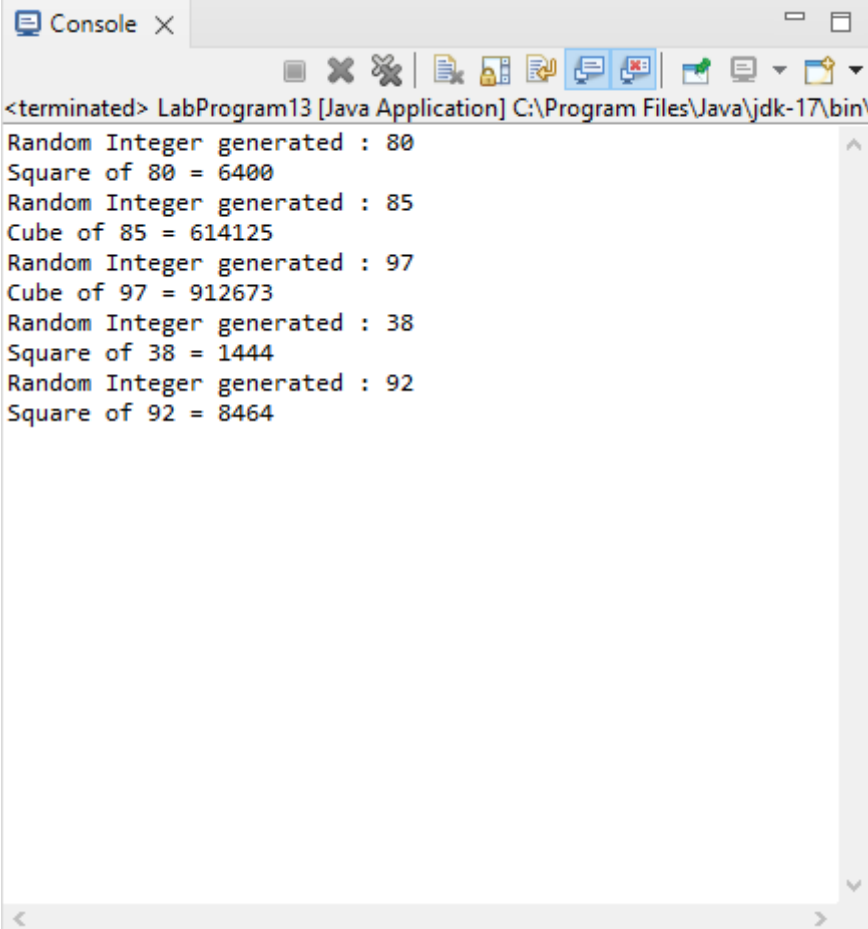
1.start()	2.run()	3.sleep()	4.currentThread()	5.join()
6.setPriority()	7.getPriority()	8.getState()	etc..	

PROGRAM:

```
import java.util.Random;
class Square extends Thread
{
    int x;
    Square(int n)
    {
        x = n;
    }
    public void run()
    {
        int sqr = x * x;
        System.out.println("Square of " + x + " = " + sqr );
    }
}
class Cube extends Thread
{
    int x;
    Cube(int n)
    {x = n;
    }
    public void run()
    {
        int cub = x * x * x;
        System.out.println("Cube of " + x + " = " + cub );
    }
}
class Number extends Thread
{
    public void run()
    {
        Random random = new Random();
        for(int i =0; i<5; i++)
        {
            int randomInteger = random.nextInt(100);
            System.out.println("Random Integer generated : " + randomInteger);
            if(randomInteger%2==0)
            {
                Square s = new Square(randomInteger);
                s.start();
            }
            else
            {
                Cube c = new Cube(randomInteger);
                c.start();
            }
        }
    }
    try {
        Thread.sleep(1000);
    } catch (InterruptedException ex) {
        System.out.println(ex);
    }
}
```

```
}  
}  
}  
public class LabProgram13 {  
  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
        Number n = new Number();  
        n.start();  
    }  
  
}
```

OUTPUT:



```
<terminated> LabProgram13 [Java Application] C:\Program Files\Java\jdk-17\bin\  
Random Integer generated : 80  
Square of 80 = 6400  
Random Integer generated : 85  
Cube of 85 = 614125  
Random Integer generated : 97  
Cube of 97 = 912673  
Random Integer generated : 38  
Square of 38 = 1444  
Random Integer generated : 92  
Square of 92 = 8464
```

EXPERIMENT NO: 14

AIM:

Write a Java program that correctly implements the producer – consumer problem using the concept of inter-thread communication.

DESCRIPTION:

Inter-thread communication or Co-operation is all about allowing synchronized threads to communicate with each other.

Cooperation (Inter-thread communication) is a mechanism in which a thread is paused running in its critical section and another thread is allowed to enter (or lock) in the same critical section to be executed. It is implemented by following methods of Object class:

1. wait()
2. notify()
3. notifyAll()

The wait() method causes current thread to release the lock and wait until either another thread invokes the notify() method or the notifyAll() method for this object, or a specified amount of time has elapsed.

The current thread must own this object's monitor, so it must be called from the synchronized method only otherwise it will throw exception.

The notify() method wakes up a single thread that is waiting on this object's monitor. If any threads are waiting on this object, one of them is chosen to be awakened. The choice is arbitrary and occurs at the discretion of the implementation.

Wakes up all threads that are waiting on this object's monitor.

Java programming language provides a very handy way of creating threads and synchronizing their task by using **synchronized** blocks

SYNTAX:

```
public    final    void  
notify()    public final  
void notifyAll()    try{  
wait();
```

```
}  
  
catch(Exception e)  
  
{  
  
}  
  
synchronized(objectIdentifier) {  
  
    // Access shared variables and other shared resources  
  
}
```

PROGRAM:

```
class Thread1  
{  
    int n;  
    boolean valueset=false;    synchronized int get()  
    {  
        if (!valueset)  
            try  
            {  
                wait();  
            }  
            catch (Exception e)  
            {  
                System.out.println("Exception occur at : "+e);  
            }  
            System.out.println("get" +n);  
            try  
            {  
                Thread.sleep(1000);  
            }  
            catch (Exception e)  
            {  
                System.out.println("Exception occur at :"+e);  
            }  
            valueset=false;  
            notify();                return n;  
        }  
    }  
    synchronized int put(int n)  
    {  
        if (valueset)  
            try  
            {  
                wait();  
            }  
        }  
    }  
}
```

```

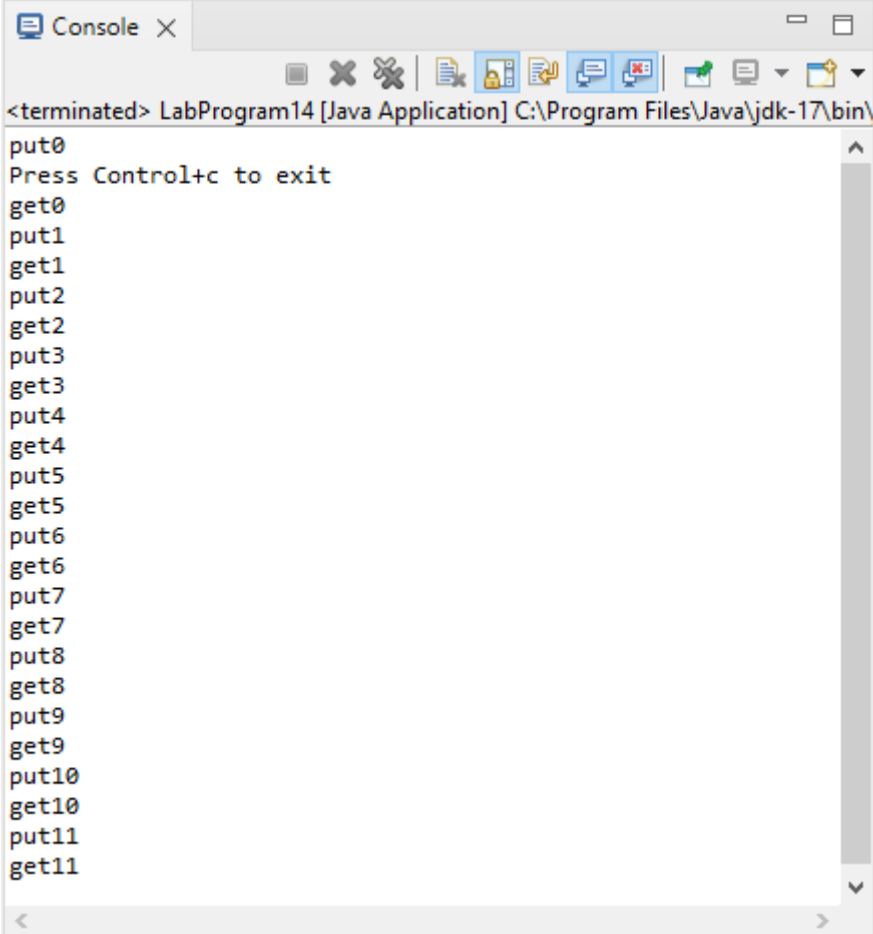
        catch (Exception e)
        {
            System.out.println("Exception occur at : "+e);
        }
        this.n=n;          valueset=true;
        System.out.println("put"+n);
        try
        {
            Thread.sleep(1000);
        }
        catch (Exception e)
        {
            System.out.println("Exception occur at : "+e);
        }
    }
    notify();
    return n;
}
}
class Producer implements Runnable
{
    Thread1 t;
    Producer(Thread1 t)
    {
        this.t=t;
        new Thread(this,"Producer").start();
    }
    public void run()
    {
        int i=0;          while (true)
        {
            t.put(i++);
        }
    }
}
class Consumer implements Runnable
{
    Thread1 t;
    Consumer(Thread1 t)
    {
        this.t=t;
        new Thread(this,"Consumer").start();
    }
    public void run()
    {
        int i=0;          while (true)
        {
            t.get();
        }
    }
}

```



```
public class LabProgram14 {  
  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
        Thread1 t=new Thread1();  
        new Producer(t);  
        new Consumer(t);  
        System.out.println("Press Control+c to exit");  
    }  
}
```

OUTPUT:



```
<terminated> LabProgram14 [Java Application] C:\Program Files\Java\jdk-17\bin\  
put0  
Press Control+c to exit  
get0  
put1  
get1  
put2  
get2  
put3  
get3  
put4  
get4  
put5  
get5  
put6  
get6  
put7  
get7  
put8  
get8  
put9  
get9  
put10  
get10  
put11  
get11
```

EXPERIMENT NO: 15

AIM:

Write a Java program that reads a file name from the user, displays information about whether the file exists, whether the file is readable, or writable, the type of file and the length of the file in bytes.

DESCRIPTION:

Java File Handling

The File class is an abstract representation of file and directory pathname. A pathname can be either absolute or relative.

The File class have several methods for working with directories and files such as creating new directories or files, deleting and renaming directories or files, listing the contents of a directory etc.

Methods:

getParent():

It returns the pathname string of this abstract pathname's parent, or null if this pathname does not name a parent directory. **canWrite():**

It tests whether the application can modify the file denoted by this abstract pathname. **String[]**

canExecute():

It tests whether the application can execute the file denoted by this abstract pathname.

isAbsolute():

It tests whether this abstract pathname is absolute.

Syntax:

File(File parent, String child)// It creates a new File instance from a parent abstract pathname and a child pathname string.

File(String pathname)// It creates a new File instance by converting the given pathname string into an abstract pathname.

File(File parent, String child)// It creates a new File instance from a parent abstract pathname and a child pathname string.

File(String pathname)// It creates a new File instance by converting the given pathname string into an abstract pathname.

File f1=new File(fname);

f1.getName()

f1.canExecute()

f1.canRead()

f1.canWrite()

f1.getParentFile()

f1.exists()

f1.length()

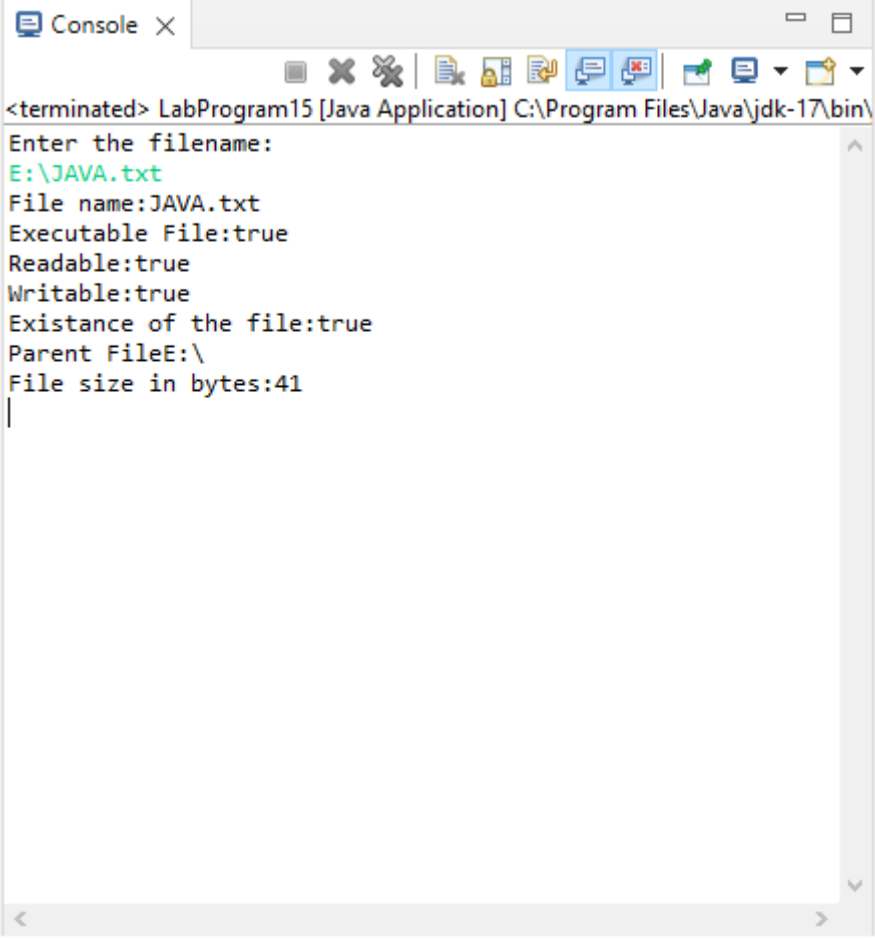
PROGRAM:

```
import java.io.*;
import java.util.*;
public class LabProgram15 {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        String fname;
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter the filename:");
        fname=sc.nextLine();
        File f1=new File(fname);
        System.out.println("File name:"+f1.getName());
        System.out.println("Executable File:"+f1.canExecute());
        System.out.println("Readable:"+f1.canRead());
        System.out.println("Writable:"+f1.canWrite());
        System.out.println("Existance of the file:"+f1.exists());
        System.out.println("Parent File"+f1.getParentFile());
        System.out.println("File size in bytes:"+f1.length());
        sc.close();
    }

}
```

OUTPUT:



```
<terminated> LabProgram15 [Java Application] C:\Program Files\Java\jdk-17\bin\  
Enter the filename:  
E:\JAVA.txt  
File name:JAVA.txt  
Executable File:true  
Readable:true  
Writable:true  
Existence of the file:true  
Parent FileE:\  
File size in bytes:41  
|
```

EXPERIMENT NO: 16

AIM:

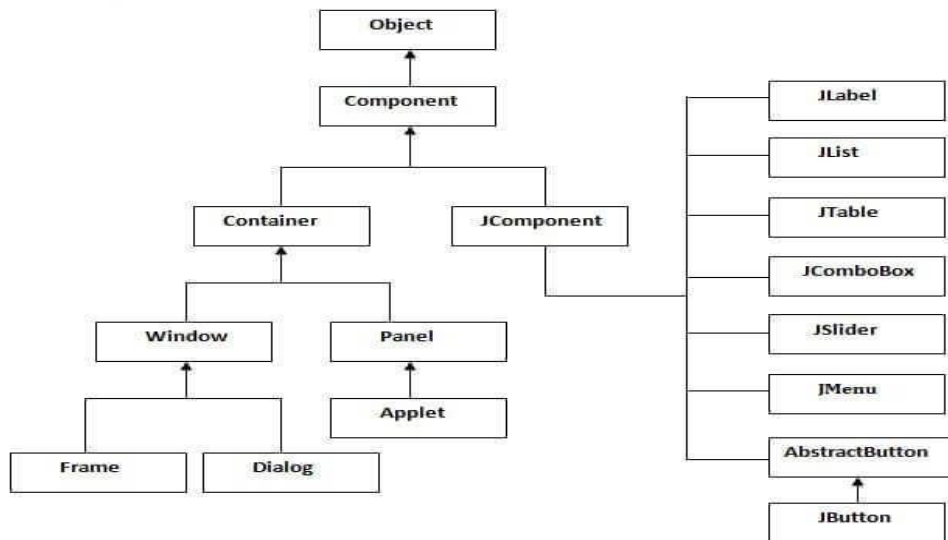
Write a Java program to build a Calculator in Swings

DESCRIPTION:

Java Swing tutorial is a part of Java Foundation Classes (JFC) that is used to create window-based applications. It is built on the top of AWT (Abstract Windowing Toolkit) API and entirely written in java.

Unlike AWT, Java Swing provides platform-independent and lightweight components.

The javax.swing package provides classes for java swing API such as JButton, JTextField, JTextArea, JRadioButton, JCheckbox, JMenu, JColorChooser etc.



SYNTAX:

// for lable:

JLabel jb = new JLabel("content to display");//constructor

// for frame:

JFrame jfrm = new Jframe("frame title");//constructor

//for button:

JButton jbt = new JButton("button content");//constructor

//for text field:

JtextField jtf = new JTextFeild(22)//constructor specify number of letters to fit in text field

addWindowListener//interface
public interface WindowListener extends EventListener
addWindowAdapter//class
WindowAdapter() //to be implemented using windowlistener by anonymous function

PROGRAM:

```
import java.awt.*;
import java.awt.event.*;
/*****/

public class MyCalculator extends Frame
{

    public boolean setClear=true;
    double number, memValue;
    char op;

    String digitButtonText[] = {"7", "8", "9", "4", "5", "6", "1", "2", "3", "0", "+/-", "." };
    String operatorButtonText[] = {"/", "sqrt", "*", "%", "-", "1/X", "+", "=" };
    String memoryButtonText[] = {"MC", "MR", "MS", "M+" };
    String specialButtonText[] = {"Backspc", "C", "CE" };

    MyDigitButton digitButton[]=new MyDigitButton[digitButtonText.length];
    MyOperatorButton operatorButton[]=new MyOperatorButton[operatorButtonText.length];
    MyMemoryButton memoryButton[]=new MyMemoryButton[memoryButtonText.length];
    MySpecialButton specialButton[]=new MySpecialButton[specialButtonText.length];

    Label displayLabel=new Label("0",Label.RIGHT);
    Label memLabel=new Label(" ",Label.RIGHT);

    final int FRAME_WIDTH=325,FRAME_HEIGHT=325;
    final int HEIGHT=30, WIDTH=30, H_SPACE=10,V_SPACE=10;
    final int TOPX=30, TOPY=50;
    ///////////////////////////////////
    MyCalculator(String frameText)//constructor
    {
        super(frameText);

        int tempX=TOPX, y=TOPY;
        displayLabel.setBounds(tempX,y,240,HEIGHT);
        displayLabel.setBackground(Color.BLUE);
        displayLabel.setForeground(Color.WHITE);
        add(displayLabel);

        memLabel.setBounds(TOPX, TOPY+HEIGHT+ V_SPACE,WIDTH, HEIGHT);
        add(memLabel);
```

```
// set Co-ordinates for Memory Buttons
tempX=TOPX;
y=TOPY+2*(HEIGHT+V_SPACE);
for(int i=0; i<memoryButton.length; i++)
{
memoryButton[i]=new MyMemoryButton(tempX,y,WIDTH,HEIGHT,memoryButtonText[i],
this);
memoryButton[i].setForeground(Color.RED);
y+=HEIGHT+V_SPACE;
}

//set Co-ordinates for Special Buttons
tempX=TOPX+1*(WIDTH+H_SPACE); y=TOPY+1*(HEIGHT+V_SPACE);
for(int i=0;i<specialButton.length;i++)
{
specialButton[i]=new MySpecialButton(tempX,y,WIDTH*2,HEIGHT,specialButtonText[i],
this);
specialButton[i].setForeground(Color.RED);
tempX=tempX+2*WIDTH+H_SPACE;
}

//set Co-ordinates for Digit Buttons
int digitX=TOPX+WIDTH+H_SPACE;
int digitY=TOPY+2*(HEIGHT+V_SPACE);
tempX=digitX; y=digitY;
for(int i=0;i<digitButton.length;i++)
{
digitButton[i]=new MyDigitButton(tempX,y,WIDTH,HEIGHT,digitButtonText[i], this);
digitButton[i].setForeground(Color.BLUE);
tempX+=WIDTH+H_SPACE;
if((i+1)%3==0){tempX=digitX; y+=HEIGHT+V_SPACE;}
}

//set Co-ordinates for Operator Buttons
int opsX=digitX+2*(WIDTH+H_SPACE)+H_SPACE;
int opsY=digitY;
tempX=opsX; y=opsY;
for(int i=0;i<operatorButton.length;i++)
{
tempX+=WIDTH+H_SPACE;
operatorButton[i]=new MyOperatorButton(tempX,y,WIDTH,HEIGHT,operatorButtonText[i],
this);
operatorButton[i].setForeground(Color.RED);
if((i+1)%2==0){tempX=opsX; y+=HEIGHT+V_SPACE;}
}

addWindowListener(new WindowAdapter()
{
public void windowClosing(WindowEvent ev)
{System.exit(0);}
});
```



```

});

setLayout(null);
setSize(FRAME_WIDTH,FRAME_HEIGHT);
setVisible(true);
}
////////////////////
static String getFormattedText(double temp)
{
String resText="" +temp;
if(resText.lastIndexOf(".0")>0)
    resText=resText.substring(0,resText.length()-2);
return resText;
}
////////////////////
public static void main(String []args)
{
new MyCalculator("Calculator - JavaTpoint");
}
}

/*****/

class MyDigitButton extends Button implements ActionListener
{
MyCalculator cl;

////////////////////
MyDigitButton(int x,int y, int width,int height,String cap, MyCalculator clc)
{
super(cap);
setBounds(x,y,width,height);
this.cl=clc;
this.cl.add(this);
addActionListener(this);
}
////////////////////
static boolean isInString(String s, char ch)
{
for(int i=0; i<s.length();i++) if(s.charAt(i)==ch) return true;
return false;
}
////////////////////
public void actionPerformed(ActionEvent ev)
{
String tempText=((MyDigitButton)ev.getSource()).getLabel();

if(tempText.equals("."))
{
    if(cl.setClear)
        {cl.displayLabel.setText("0.");cl.setClear=false;}
}

```

```

else if(!isInString(cl.displayLabel.getText(), '.'))
    cl.displayLabel.setText(cl.displayLabel.getText()+".");
return;
}

int index=0;
try{
    index=Integer.parseInt(tempText);
}catch(NumberFormatException e){return;}

if (index==0 && cl.displayLabel.getText().equals("0")) return;

if(cl.setClear)
    {cl.displayLabel.setText(""+index);cl.setClear=false;}
else
    cl.displayLabel.setText(cl.displayLabel.getText()+index);
} //actionPerformed
} //class defination

/*****/

class MyOperatorButton extends Button implements ActionListener
{
    MyCalculator cl;

    MyOperatorButton(int x,int y, int width,int height,String cap, MyCalculator clc)
    {
        super(cap);
        setBounds(x,y,width,height);
        this.cl=clc;
        this.cl.add(this);
        addActionListener(this);
    }
    ///////////////
    public void actionPerformed(ActionEvent ev)
    {
        String opText=((MyOperatorButton)ev.getSource()).getLabel();

        cl.setClear=true;
        double temp=Double.parseDouble(cl.displayLabel.getText());

        if(opText.equals("1/x"))
        {
            try
            {
                double tempd=1/(double)temp;
                cl.displayLabel.setText(MyCalculator.getFormattedText(tempd));
            }
            catch(ArithmeticException excp)
            {
                cl.displayLabel.setText("Divide by 0.");
            }
            return;
        }
        if(opText.equals("sqrt"))

```

```

    {
    try
    {double tempd=Math.sqrt(temp);
    cl.displayLabel.setText(MyCalculator.getFormattedText(tempd));}
    catch(ArithmeticException excp)
    {cl.displayLabel.setText("Divide by 0.");}

    return;
    }
if(!opText.equals("="))
{
    cl.number=temp;
    cl.op=opText.charAt(0);
    return;
}
// process = button pressed
switch(cl.op)
{
case '+':
    temp+=cl.number;break;
case '-':
    temp=cl.number-temp;break;
case '*':
    temp*=cl.number;break;
case '%':
    try{temp=cl.number%temp;}
    catch(ArithmeticException excp)
    {cl.displayLabel.setText("Divide by 0."); return;}
    break;
case '/':
    try{temp=cl.number/temp;}
    catch(ArithmeticException excp)
    {cl.displayLabel.setText("Divide by 0."); return;}
    break;
}
}
cl.displayLabel.setText(MyCalculator.getFormattedText(temp));
//cl.number=temp;
}
}
}

/*****/

class MyMemoryButton extends Button implements ActionListener
{
    MyCalculator cl;

    ///////////////////////////////////
    MyMemoryButton(int x,int y, int width,int height,String cap, MyCalculator clc)
    {
        super(cap);
        setBounds(x,y,width,height);
    }
}

```

```

this.cl=clc;
this.cl.add(this);
addActionListener(this);
}
////////////////////////////////////
public void actionPerformed(ActionEvent ev)
{
char memop=((MyMemoryButton)ev.getSource()).getLabel().charAt(1);

cl.setClear=true;
double temp=Double.parseDouble(cl.displayLabel.getText());

switch(memop)
{
case 'C':
    cl.memLabel.setText(" ");cl.memValue=0.0;break;
case 'R':
    cl.displayLabel.setText(MyCalculator.getFormattedText(cl.memValue));break;
case 'S':
    cl.memValue=0.0;
case '+':
    cl.memValue+=Double.parseDouble(cl.displayLabel.getText());
    if(cl.displayLabel.getText().equals("0") ||
cl.displayLabel.getText().equals("0.0") )
        cl.memLabel.setText(" ");
    else
        cl.memLabel.setText("M");
    break;
}
}
}
}

/*****/

class MySpecialButton extends Button implements ActionListener
{
MyCalculator cl;

MySpecialButton(int x,int y, int width,int height,String cap, MyCalculator clc)
{
super(cap);
setBounds(x,y,width,height);
this.cl=clc;
this.cl.add(this);
addActionListener(this);
}
}
////////////////////////////////////
static String backSpace(String s)
{
String Res="";
for(int i=0; i<s.length()-1; i++) Res+=s.charAt(i);

```

```

return Res;
}

////////////////////////////////////
public void actionPerformed(ActionEvent ev)
{
String opText=((MySpecialButton)ev.getSource()).getLabel();
//check for backspace button
if(opText.equals("Backspc"))
{
String tempText=backSpace(cl.displayLabel.getText());
if(tempText.equals(""))
    cl.displayLabel.setText("0");
else
    cl.displayLabel.setText(tempText);
return;
}
//check for "C" button i.e. Reset
if(opText.equals("C"))
{
cl.number=0.0; cl.op=' '; cl.memValue=0.0;
cl.memLabel.setText(" ");
}

//it must be CE button pressed
cl.displayLabel.setText("0");cl.setClear=true;
} //actionPerformed
} //class

/*****
Features not implemented and few bugs

i) No coding done for "+/-" button.
ii) Menubar is not included.
iii)Not for Scientific calculation
iv)Some of the computation may lead to unexpected result
    due to the representation of Floating point numbers in computer
    is an approximation to the given value that can be stored
    physically in memory.
*****/

```

OUTPUT:

Calculator - JavaTpoint

109					
Backspc		C		CE	
MC	7	8	9	/	sqrt
MR	4	5	6	*	%
MS	1	2	3	-	1/X
M+	0	+/-	.	+	=

Calculator - JavaTpoint

46					
Backspc		C		CE	
MC	7	8	9	/	sqrt
MR	4	5	6	*	%
MS	1	2	3	-	1/X
M+	0	+/-	.	+	=

Calculator - JavaTpoint

63					
Backspc		C		CE	
MC	7	8	9	/	sqrt
MR	4	5	6	*	%
MS	1	2	3	-	1/X
M+	0	+/-	.	+	=

Calculator - JavaTpoint

12					
Backspc		C		CE	
MC	7	8	9	/	sqrt
MR	4	5	6	*	%
MS	1	2	3	-	1/X
M+	0	+/-	.	+	=

Calculator - JavaTpoint

1308					
Backspc		C		CE	
MC	7	8	9	/	sqrt
MR	4	5	6	*	%
MS	1	2	3	-	1/X
M+	0	+/-	.	+	=

EXPERIMENT NO: 17

AIM:

Write a Java program to implement JMenu to draw all basic shapes using Graphics.

DESCRIPTION:

Class Graphics

[java.lang.Object](#)

java.awt.Graphics

public abstract class Graphics

extends [Object](#)

The Graphics class is the abstract base class for all graphics contexts that allow an application to draw onto components that are realized on various devices, as well as onto off-screen images.

```
public Graphics create(int x,  
    int y,  
    int width,  
    int height)
```

Creates a new Graphics object based on this Graphics object, but with a new translation and clip area. The new Graphics object has its origin translated to the specified point (x, y). Its clip area is determined by the intersection of the original clip area with the specified rectangle. The arguments are all interpreted in the coordinate system of the original Graphics object. The new graphics context is identical to the original, except in two respects:

The new graphics context is translated by (x, y). That is to say, the point (0, 0) in the new graphics context is the same as (x, y) in the original graphics context.

The new graphics context has an additional clipping rectangle, in addition to whatever (translated) clipping rectangle it inherited from the original graphics context. The origin of the new clipping rectangle is at (0, 0), and its size is specified by the width and height arguments.

Parameters:

x - the x coordinate.

y - the y coordinate.

width - the width of the clipping rectangle.

height - the height of the clipping rectangle.

SYNTAX:

```
Graphics create(int x, int y, int width, int height)
```

```
abstract void drawPolygon(int[] xPoints, int[] yPoints, int nPoints)
```

```
//Draws a closed polygon defined by arrays of x and y coordinates.
```

```
void drawPolygon(Polygon p)
```

```
//Draws the outline of a polygon defined by the specified Polygon object.
```

```
abstract void drawPolyline(int[] xPoints, int[] yPoints, int nPoints)
```

//Draws a sequence of connected lines defined by arrays of x and y coordinates.

void drawRect(int x, int y, int width, int height)

PROGRAM:

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

public class Menu extends JFrame {

    public static void main(String[] args) {
        // A main routine that allows this class to be run
        // as a stand-alone application. It just opens a frame.
        new Menu();
    }

    JRadioButtonMenuItem black, red, green, blue, cyan, magenta,
        yellow, white, custom;
    // Items for the "Color" menu, which controls the drawing color.
    // They form a group in which only one item can be selected.
    // When the user starts drawing, the color is determined by
    // checking to see which of the items is selected.

    JRadioButtonMenuItem curve, straightLine, rectangle, oval,
        roundRect, filledRectangle, filledOval, filledRoundRect;
    // Items for the "Shape" menu, which determine the shape to be drawn.

    JRadioButtonMenuItem noSymmetry, twoWay, fourWay, eightWay;
    // Items for the "Symmetry" menu, which determine which
    // reflections of the basic figure should be drawn.

    public boolean standAlone = true;
    // If a frame is created by an applet, the applet should
    // set this variable to false. Otherwise, an error will
    // be generated when the user selects the "Quit" command,
    // since that command will call System.exit() if standalone
    // is true. The applet should also call the frame's
    // setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE).

    public Menu() { // replaces init() method.
        // Constructor creates a drawing area and uses it as its
        // content pane. It also sets up the menu bar.

        super("Graphics Menu"); // Set a title for the window.
```

```
Display canvas = new Display(); // The drawing area.
setContentPane(canvas);

// Create menu bar and menus.

JMenuBar menubar = new JMenuBar();
JMenu controlMenu = new JMenu("Control", true);
menubar.add(controlMenu);
JMenu colorMenu = new JMenu("Color", true);
menubar.add(colorMenu);
JMenu shapeMenu = new JMenu("Shape", true);
menubar.add(shapeMenu);
JMenu symmetryMenu = new JMenu("Symmetry", true);
menubar.add(symmetryMenu);
setJMenuBar(menubar);

// Set up the "Control" menu, and set the canvas to respond
// to commands from this menu. Add accelerators for some
// of the commands.

controlMenu.add("Fill with Black").addActionListener(canvas);
controlMenu.add("Fill with Red").addActionListener(canvas);
controlMenu.add("Fill with Green").addActionListener(canvas);
controlMenu.add("Fill with Blue").addActionListener(canvas);
controlMenu.add("Fill with Cyan").addActionListener(canvas);
controlMenu.add("Fill with Magenta").addActionListener(canvas);
controlMenu.add("Fill with Yellow").addActionListener(canvas);
controlMenu.add("Fill with White").addActionListener(canvas);
controlMenu.add("Fill with Custom").addActionListener(canvas);
controlMenu.addSeparator();
JMenuItem customItem = new JMenuItem("Set Custom Color...");
customItem.addActionListener(canvas);
customItem.setAccelerator( KeyStroke.getKeyStroke("ctrl T") );
controlMenu.add(customItem);
JMenuItem clearItem = new JMenuItem("Clear");
clearItem.addActionListener(canvas);
clearItem.setAccelerator( KeyStroke.getKeyStroke("ctrl K") );
controlMenu.add(clearItem);
JMenuItem undoItem = new JMenuItem("Undo");
undoItem.addActionListener(canvas);
undoItem.setAccelerator( KeyStroke.getKeyStroke("ctrl Z") );
controlMenu.add(undoItem);
JMenuItem quitItem = new JMenuItem("Quit");
quitItem.setAccelerator( KeyStroke.getKeyStroke("ctrl Q") );
quitItem.addActionListener(canvas);
controlMenu.add(quitItem);

// Set up the "Color" menu, with all the items in a button group.

ButtonGroup colorGroup = new ButtonGroup();
black = new JRadioButtonMenuItem("Black");
```

```
colorGroup.add(black);
colorMenu.add(black);
red = new JRadioButtonMenuItem("Red");
colorGroup.add(red);
colorMenu.add(red);
green = new JRadioButtonMenuItem("Green");
colorGroup.add(green);
colorMenu.add(green);
blue = new JRadioButtonMenuItem("Blue");
colorGroup.add(blue);
colorMenu.add(blue);
cyan = new JRadioButtonMenuItem("Cyan");
colorGroup.add(cyan);
colorMenu.add(cyan);
magenta = new JRadioButtonMenuItem("Magenta");
colorGroup.add(magenta);
colorMenu.add(magenta);
yellow = new JRadioButtonMenuItem("Yellow");
colorGroup.add(yellow);
colorMenu.add(yellow);
white = new JRadioButtonMenuItem("White");
colorGroup.add(white);
colorMenu.add(white);
custom = new JRadioButtonMenuItem("Custom Color");
colorGroup.add(custom);
colorMenu.add(custom);
black.setSelected(true);

// Set up the "Shape" menu.

ButtonGroup shapeGroup = new ButtonGroup();
curve = new JRadioButtonMenuItem("Curve");
shapeGroup.add(curve);
shapeMenu.add(curve);
straightLine = new JRadioButtonMenuItem("Straight Line");
shapeGroup.add(straightLine);
shapeMenu.add(straightLine);
rectangle = new JRadioButtonMenuItem("Rectangle");
shapeGroup.add(rectangle);
shapeMenu.add(rectangle);
oval = new JRadioButtonMenuItem("Oval");
shapeGroup.add(oval);
shapeMenu.add(oval);
roundRect = new JRadioButtonMenuItem("RoundRect");
shapeGroup.add(roundRect);
shapeMenu.add(roundRect);
filledRectangle = new JRadioButtonMenuItem("Filled Rectangle");
shapeGroup.add(filledRectangle);
shapeMenu.add(filledRectangle);
filledOval = new JRadioButtonMenuItem("Filled Oval");
shapeGroup.add(filledOval);
```

```

shapeMenu.add(filledOval);
filledRoundRect = new JRadioButtonMenuItem("Filled RoundRect");
shapeGroup.add(filledRoundRect);
shapeMenu.add(filledRoundRect);
curve.setSelected(true);

// Set up the "Symmetry" menu.

ButtonGroup symmetryGroup = new ButtonGroup();
noSymmetry = new JRadioButtonMenuItem("None");
noSymmetry.setAccelerator( KeyStroke.getKeyStroke("ctrl 0") );
symmetryGroup.add(noSymmetry);
symmetryMenu.add(noSymmetry);
twoWay = new JRadioButtonMenuItem("Two-way");
twoWay.setAccelerator( KeyStroke.getKeyStroke("ctrl 2") );
symmetryGroup.add(twoWay);
symmetryMenu.add(twoWay);
fourWay = new JRadioButtonMenuItem("Four-way");
fourWay.setAccelerator( KeyStroke.getKeyStroke("ctrl 4") );
symmetryGroup.add(fourWay);
symmetryMenu.add(fourWay);
eightWay = new JRadioButtonMenuItem("Eight-way");
eightWay.setAccelerator( KeyStroke.getKeyStroke("ctrl 8") );
symmetryGroup.add(eightWay);
symmetryMenu.add(eightWay);
noSymmetry.setSelected(true);

// Set size, etc., of frame and make it visible.

pack();
setLocation(75,50);
setResizable(false);
setDefaultCloseOperation(EXIT_ON_CLOSE);
show();
} // end constructor

private class Display extends JPanel
    implements MouseListener, MouseMotionListener, ActionListener {

    // Nested class Display represents the drawing surface of the
    // applet. It lets the user use the mouse to draw colored curves
    // and shapes. The current color is specified by the pop-up menu
    // colorChoice. The current shape is specified by another pop-up menu,
    // figureChoice. (These are instance variables in the main class.)
    // The panel also listens for action events from buttons
    // named "Clear" and "Set Background". The "Clear" button fills
    // the panel with the current background color. The "Set Background"
    // button sets the background color to the current drawing color and

```



```
// then clears. These buttons are set up in the main class.
```

```
private final static int
```

```
    CURVE = 0,  
    LINE = 1,  
    RECT = 2,           // Some constants that code  
    OVAL = 3,           // for the different types of  
    ROUNDRECT = 4,      // figure the program can draw.  
    FILLED_RECT = 5,  
    FILLED_OVAL = 6,  
    FILLED_ROUNDRECT = 7;
```

```
private final static int
```

```
    NO_SYMMETRY = 0,    // Some constants that code for  
    SYMMETRY_2 = 1,     // the different symmetry styles.  
    SYMMETRY_4 = 2,  
    SYMMETRY_8 = 3;
```

```
Color customColor = Color.gray; // The custom color that is used  
                                // when the user selects "Custom Color"  
                                // as the drawing color or "Fill with
```

```
Custom"
```

```
                                // from the "Control" menu. This color  
                                // is changed when the user selects the  
                                // "Set Custom Color..." command.
```

```
/* Some variables used for backing up the contents of the panel. */
```

```
Image OSI; // The off-screen image (created in checkOSI()).
```

```
int widthOfOSI, heightOfOSI; // Current width and height of OSI. These  
                              // are checked against the size of the applet,  
                              // to detect any change in the panel's size.  
                              // If the size has changed, a new OSI is
```

```
created.
```

```
                                // The picture in the off-screen image is lost  
                                // when that happens.
```

```
Image undoBuffer; // An off-screen image that is used to implement  
                  // the undo operation. When the user begins  
                  // a drawing operation, the OSI is copied to  
                  // undoBuffer. If the user selects the "Undo"  
                  // command, the OSI and the undoBuffer are swapped  
                  // and the panel is repainted to show the previous image.
```

```
/* The following variables are used when the user is sketching a  
   curve while dragging a mouse. */
```



```

private int mouseX, mouseY;    // The location of the mouse.

private int prevX, prevY;      // The previous location of the mouse.

private int startX, startY;    // The starting position of the mouse.
                                // (Not used for drawing curves.)

private boolean dragging;      // This is set to true when the user is
drawing.

private int figure;            // What type of figure is being drawn. This is
                                // specified by the figureChoice menu.

private int symmetry;          // What type of symmetry style is being used. This
is                               // specified by the symmetryChoice menu.

private Graphics dragGraphics; // A graphics context for the off-screen
image,                           // to be used while a drag is in progress.

private Color dragColor;       // The color that is used for the figure that is
                                // being drawn.

Display() {
    // Constructor. When this component is first created, it is set to
    // listen for mouse events and mouse motion events from
    // itself. The initial background color is white.
    addMouseListener(this);
    addMouseMotionListener(this);
    setBackground(Color.white);
    setPreferredSize( new Dimension(450,450) );
}

private Color getSelectedColor() {
    // Check the "Color" menu and return the color
    // that is currently selected.
    if (black.isSelected())
        return Color.black;
    else if (red.isSelected())
        return Color.red;
    else if (green.isSelected())
        return Color.green;
    else if (blue.isSelected())
        return Color.blue;
    else if (cyan.isSelected())
        return Color.cyan;
    else if (magenta.isSelected())
        return Color.magenta;
    else if (yellow.isSelected())

```

```

        return Color.yellow;
    else if (white.isSelected())
        return Color.white;
    else
        return customColor;
}

private int getSelectedShape() {
    // Check the "Shape" menu and return the code
    // for the shape that is currently selected.
    if (curve.isSelected())
        return CURVE;
    else if (straightLine.isSelected())
        return LINE;
    else if (rectangle.isSelected())
        return RECT;
    else if (oval.isSelected())
        return OVAL;
    else if (roundRect.isSelected())
        return ROUNRECT;
    else if (filledRectangle.isSelected())
        return FILLED_RECT;
    else if (filledOval.isSelected())
        return FILLED_OVAL;
    else
        return FILLED_ROUNDRECT;
}

private int getSelectedSymmetry() {
    // Check the "Symmetry" menu and return the code
    // for the type of symmetry that is currently selected.
    if (noSymmetry.isSelected())
        return NO_SYMMETRY;
    else if (twoWay.isSelected())
        return SYMMETRY_2;
    else if (fourWay.isSelected())
        return SYMMETRY_4;
    else
        return SYMMETRY_8;
}

private void drawFigure(Graphics g, int shape, int x1, int y1, int x2, int
y2) {
    // This method is called to do ALL drawing in this applet!
    // Draws a shape in the graphics context g.
    // The shape parameter tells what kind of shape to draw. This
    // can be LINE, RECT, OVAL, ROUNRECT, FILLED_RECT,
    // FILLED_OVAL, or FILLED_ROUNDRECT. (Note that a CURVE is

```

```

// drawn by drawing multiple LINES, so the shape parameter is
// never equal to CURVE.) For a LINE, a line is drawn from
// the point (x1,y1) to (x2,y2). For other shapes, the
// points (x1,y1) and (x2,y2) give two corners of the shape
// (or of a rectangle that contains the shape).
if (shape == LINE) {
    // For a line, just draw the line between the two points.
    g.drawLine(x1,y1,x2,y2);
    return;
}
int x, y; // Top left corner of rectangle that contains the figure.
int w, h; // Width and height of rectangle that contains the figure.
if (x1 >= x2) { // x2 is left edge
    x = x2;
    w = x1 - x2;
}
else { // x1 is left edge
    x = x1;
    w = x2 - x1;
}
if (y1 >= y2) { // y2 is top edge
    y = y2;
    h = y1 - y2;
}
else { // y1 is top edge.
    y = y1;
    h = y2 - y1;
}
switch (shape) { // Draw the appropriate figure.
    case RECT:
        g.drawRect(x, y, w, h);
        break;
    case OVAL:
        g.drawOval(x, y, w, h);
        break;
    case ROUNDRECT:
        g.drawRoundRect(x, y, w, h, 20, 20);
        break;
    case FILLED_RECT:
        g.fillRect(x, y, w, h);
        break;
    case FILLED_OVAL:
        g.fillOval(x, y, w, h);
        break;
    case FILLED_ROUNDRECT:
        g.fillRoundRect(x, y, w, h, 20, 20);
        break;
}
}

```

```

private void putMultiFigure(Graphics g, int shape, int x1, int y1, int x2,
int y2) {
    // Draws the shape and possibly some of its reflections.
    // The reflections that are drawn depend on the selected
    // item in symmetryChoice. The shapes are drawn by calling
    // the drawFigure method.

    int width = getWidth();
    int height = getHeight();

    drawFigure(g, shape, x1, y1, x2, y2); // Draw the basic figure

    if (symmetry >= SYMMETRY_2) { // Draw the horizontal reflection.
        drawFigure(g, shape, width - x1, y1, width - x2, y2);
    }

    if (symmetry >= SYMMETRY_4) { // Draw the two vertical reflections.
        drawFigure(g, shape, x1, height - y1, x2, height - y2);
        drawFigure(g, shape, width - x1, height - y1, width - x2, height -
y2);
    }

    if (symmetry == SYMMETRY_8) { // Draw the four diagonal reflections.
        int a1 = (int)((double)y1 / height) * width;
        int b1 = (int)((double)x1 / width) * height;
        int a2 = (int)((double)y2 / height) * width;
        int b2 = (int)((double)x2 / width) * height;
        drawFigure(g, shape, a1, b1, a2, b2);
        drawFigure(g, shape, width - a1, b1, width - a2, b2);
        drawFigure(g, shape, a1, height - b1, a2, height - b2);
        drawFigure(g, shape, width - a1, height - b1, width - a2, height -
b2);
    }
}

private void repaintRect(int x1, int y1, int x2, int y2) {
    // Call repaint on a rectangle that contains the points (x1,y1)
    // and (x2,y2). (Add a 1-pixel border along right and bottom
    // edges to allow for the pen overhang when drawing a line.)
    int x, y; // top left corner of rectangle that contains the figure
    int w, h; // width and height of rectangle that contains the figure
    if (x2 >= x1) { // x1 is left edge
        x = x1;
        w = x2 - x1;
    }
    else { // x2 is left edge
        x = x2;
        w = x1 - x2;
    }
}

```

```

if (y2 >= y1) { // y1 is top edge
    y = y1;
    h = y2 - y1;
}
else { // y2 is top edge.
    y = y2;
    h = y1 - y2;
}
repaint(x,y,w+1,h+1);
}

```

```

private void repaintMultiRect(int x1, int y1, int x2, int y2) {
    // Call repaint on a rectangle that contains the points (x1,y1)
    // and (x2,y2). Also call repaint on reflections of this
    // rectangle, depending on the type of symmetry. The
    // rects are repainted by calling repaintRect().
    int width = getWidth();
    int height = getHeight();
    repaintRect(x1,y1,x2,y2); // repaint the original rect
    if (symmetry >= SYMMETRY_2) { // repaint the horizontal reflection.
        repaintRect(width - x1, y1, width - x2, y2);
    }
    if (symmetry >= SYMMETRY_4) { // repaint the two vertical reflections.
        repaintRect(x1, height - y1, x2, height - y2);
        repaintRect(width - x1, height - y1, width - x2, height - y2);
    }
    if (symmetry == SYMMETRY_8) { // repaint the four diagonal reflections.
        int a1 = (int)((double)y1 / height) * width;
        int b1 = (int)((double)x1 / width) * height;
        int a2 = (int)((double)y2 / height) * width;
        int b2 = (int)((double)x2 / width) * height;
        repaintRect(a1, b1, a2, b2);
        repaintRect(width - a1, b1, width - a2, b2);
        repaintRect(a1, height - b1, a2, height - b2);
        repaintRect(width - a1, height - b1, width - a2, height - b2);
    }
}
}

```

```

private void checkOSI() {
    // This method is responsible for creating the off-screen image.
    // It should be called before using the OSI. It will make a new OSI if
    // the size of the panel changes.
    if (OSI == null || widthOfOSI != getSize().width || heightOfOSI !=
    getSize().height) {
        // Create the OSI, or make a new one if panel size has changed.
        OSI = null; // (If OSI already exists, this frees up the memory.)
        undoBuffer = null; // (Free memory.)
        widthOfOSI = getWidth();
        heightOfOSI = getHeight();
    }
}

```

```

        OSI = createImage(widthOfOSI,heightOfOSI);
        Graphics OSG = OSI.getGraphics(); // Graphics context for drawing to
OSI.
        OSG.setColor(getBackground());
        OSG.fillRect(0, 0, widthOfOSI, heightOfOSI);
        OSG.dispose();
        undoBuffer = createImage(widthOfOSI,heightOfOSI);
        OSG = undoBuffer.getGraphics(); // Graphics context for drawing to
undoBuffer
        OSG.setColor(getBackground());
        OSG.fillRect(0, 0, widthOfOSI, heightOfOSI);
        OSG.dispose();
    }
}

public void paintComponent(Graphics g) {
    // Copy the off-screen image to the screen,
    // after checking to make sure it exists. Then,
    // if a shape other than CURVE is being drawn,
    // draw it on top of the image from the OSI.
    checkOSI();
    g.drawImage(OSI, 0, 0, this);
    if (dragging && figure != CURVE) {
        g.setColor(dragColor);
        putMultiFigure(g,figure,startX,startY,mouseX,mouseY);
    }
}

public void actionPerformed(ActionEvent evt) {
    // Respond when the user selects an item from the "Control" menu.
    String command = evt.getActionCommand();
    checkOSI();
    if (command.equals("Fill with Black"))
        clear(Color.black);
    else if (command.equals("Fill with Red"))
        clear(Color.red);
    else if (command.equals("Fill with Green"))
        clear(Color.green);
    else if (command.equals("Fill with Blue"))
        clear(Color.blue);
    else if (command.equals("Fill with Cyan"))
        clear(Color.cyan);
    else if (command.equals("Fill with Magenta"))
        clear(Color.magenta);
    else if (command.equals("Fill with Yellow"))
        clear(Color.yellow);
    else if (command.equals("Fill with White"))
        clear(Color.white);
    else if (command.equals("Fill with Custom"))

```



```

        clear(customColor);
    else if (command.equals("Set Custom Color...")) {
        Color c = JColorChooser.showDialog(this, "Select Custom
Color", customColor);
        if (c != null) {
            // Change the custom color and select it for use as
            // the drawing color.
            customColor = c;
            custom.setSelected(true);
        }
    }
    else if (command.equals("Clear")) {
        // Clear to current background color.
        Graphics g = OSI.getGraphics();
        g.setColor(getBackground());
        g.fillRect(0,0,getSize().width,getSize().height);
        g.dispose();
        repaint();
    }
    else if (command.equals("Undo")) {
        // Undo the most recent drawing operation
        // by swapping OSI with undoBuffer.
        Image temp = OSI;
        OSI = undoBuffer;
        undoBuffer = temp;
        repaint();
    }
    else if (command.equals("Quit")) {
        // Close the window and exit. Note: The
        // exit command will cause an error when
        // the frame is opened from an applet.
        // An applet should set the frame's standAlone
        // variable to false after creating the frame.
        dispose();
        if (standAlone)
            System.exit(0);
    }
}

private void clear(Color background) {
    // Fill with the specified color. If the
    // color is equal to the current drawing color, then
    // the current drawing color is changed, so that
    // drawing operations will not be invisible.
    setBackground(background);
    if (background.equals(getSelectedColor())) {
        if (background.equals(Color.black))
            white.setSelected(true); // On a black background, draw in white.
        else
            black.setSelected(true); // On other backgrounds, use black.
    }
}

```



```

Graphics g = OSI.getGraphics();
g.setColor(getBackground());
g.fillRect(0,0,getSize().width,getSize().height);
g.dispose();
repaint();
}

public void mousePressed(MouseEvent evt) {
    // This is called when the user presses the mouse on the
    // panel. This begins a draw operation in which the user
    // sketches a curve or draws a shape. (Note that curves
    // are handled differently from other shapes. For CURVE,
    // a new segment of the curve is drawn each time the user
    // moves the mouse. For the other shapes, a "rubber band
    // cursor" is used. That is, the figure is drawn between
    // the starting point and the current mouse location.)

    if (dragging == true) // Ignore mouse presses that occur
        return;          // when user is already drawing a curve.
                          // (This can happen if the user presses
                          // two mouse buttons at the same time.)

    prevX = startX = evt.getX(); // Save mouse coordinates.
    prevY = startY = evt.getY();

    figure = getSelectedShape(); // Get data from menus for drawing.
    symmetry = getSelectedSymmetry();
    dragColor = getSelectedColor();

    checkOSI();

    Graphics undoGraphics = undoBuffer.getGraphics();
    undoGraphics.drawImage(OSI,0,0,null); // Remember the current image,
                                          // for "Undo" operations,
                                          // before changing the image.

    undoGraphics.dispose();

    dragGraphics = OSI.getGraphics();
    dragGraphics.setColor(dragColor);

    dragging = true; // Start drawing.
} // end mousePressed()

public void mouseReleased(MouseEvent evt) {
    // Called whenever the user releases the mouse button.
    // If the user was drawing a shape, we make the shape
    // permanent by drawing it to the off-screen image.
    if (dragging == false)

```

```

        return; // Nothing to do because the user isn't drawing.
    dragging = false;
    mouseX = evt.getX();
    mouseY = evt.getY();
    if (figure == CURVE) {
        // A CURVE is drawn as a series of LINES
        putMultiFigure(dragGraphics, LINE, prevX, prevY, mouseX, mouseY);
        repaintMultiRect(prevX, prevY, mouseX, mouseY);
    }
    else if (figure == LINE) {
        repaintMultiRect(startX, startY, prevX, prevY);
        if (mouseX != startX || mouseY != startY) {
            // Draw the line only if it has non-zero length.
            putMultiFigure(dragGraphics, figure, startX, startY, mouseX, mouseY);
            repaintMultiRect(startX, startY, mouseX, mouseY);
        }
    }
    else {
        repaintMultiRect(startX, startY, prevX, prevY);
        if (mouseX != startX && mouseY != startY) {
            // Draw the shape only if both its height
            // and width are both non-zero.
            putMultiFigure(dragGraphics, figure, startX, startY, mouseX, mouseY);
            repaintMultiRect(startX, startY, mouseX, mouseY);
        }
    }
    dragGraphics.dispose();
    dragGraphics = null;
}

public void mouseDragged(MouseEvent evt) {
    // Called whenever the user moves the mouse while a mouse button
    // is down. If the user is drawing a curve, draw a segment of
    // the curve on the off-screen image, and repaint the part
    // of the panel that contains the new line segment. Otherwise,
    // just call repaint and let paintComponent() draw the shape on
    // top of the picture in the off-screen image.

    if (dragging == false)
        return; // Nothing to do because the user isn't drawing.

    mouseX = evt.getX(); // x-coordinate of mouse.
    mouseY = evt.getY(); // y=coordinate of mouse.

    if (figure == CURVE) {
        // A CURVE is drawn as a series of LINES.
        putMultiFigure(dragGraphics, LINE, prevX, prevY, mouseX, mouseY);
        repaintMultiRect(prevX, prevY, mouseX, mouseY);
    }
    else {

```

```
        // Repaint two rectangles: The one that contains the previous
        // version of the figure, and the one that will contain the
        // new version. The first repaint is necessary to restore
        // the picture from the off-screen image in that rectangle.
        repaintMultiRect(startX,startY,prevX,prevY);
        repaintMultiRect(startX,startY,mouseX,mouseY);
    }

    prevX = mouseX; // Save coords for the next call to mouseDragged or
mouseReleased.
    prevY = mouseY;

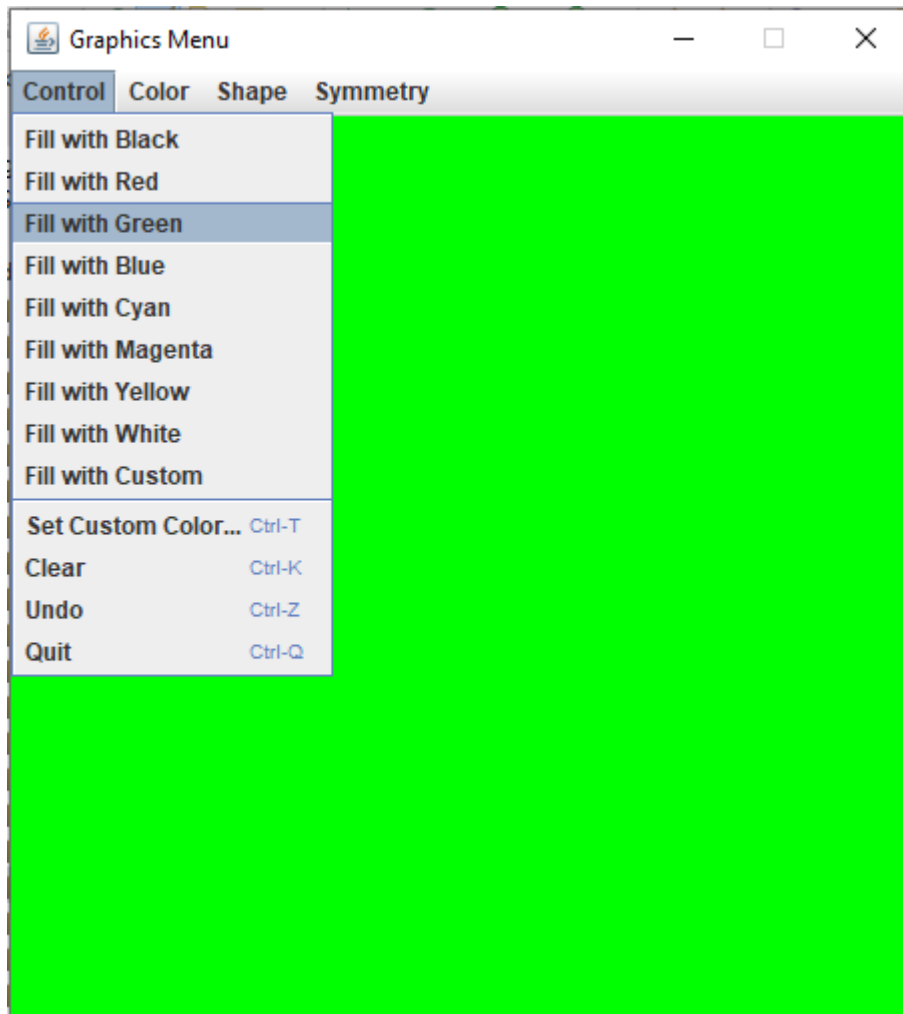
} // end mouseDragged.

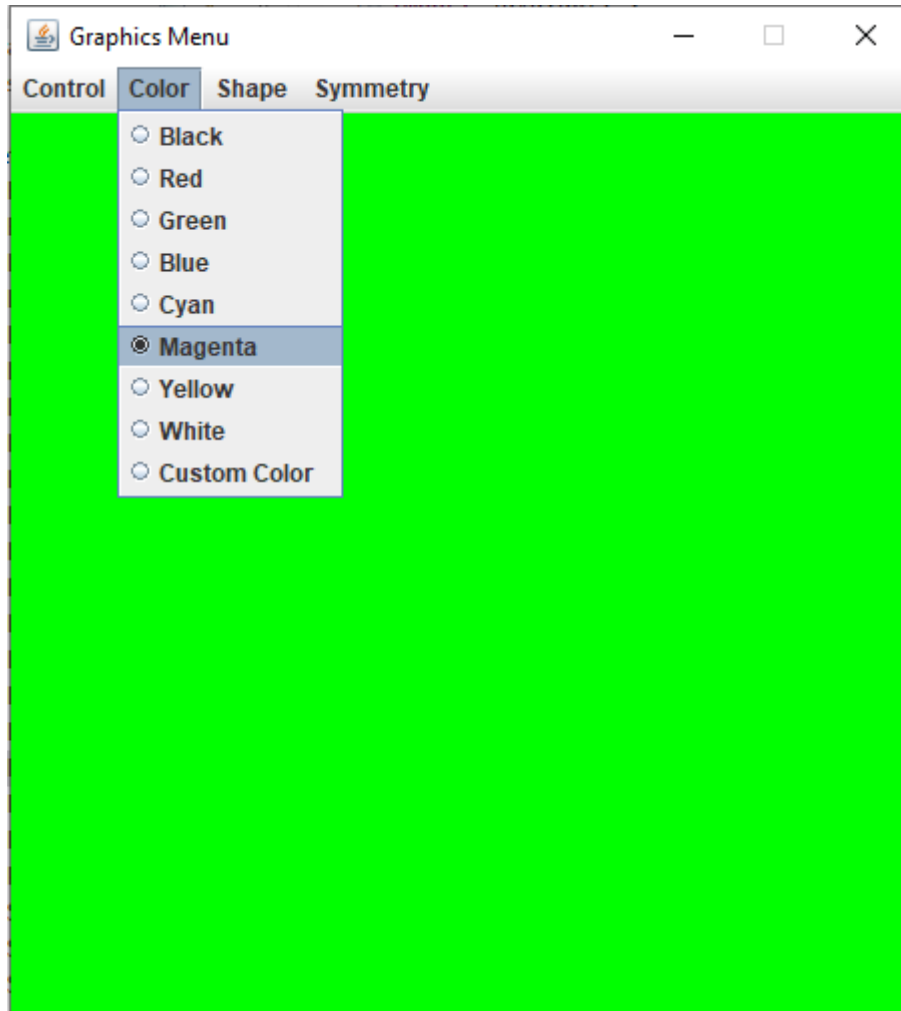
    public void mouseEntered(MouseEvent evt) { } // Some empty routines.
    public void mouseExited(MouseEvent evt) { } // (Required by the
MouseListener
    public void mouseClicked(MouseEvent evt) { } // and MouseMotionListener
    public void mouseMoved(MouseEvent evt) { } // interfaces).

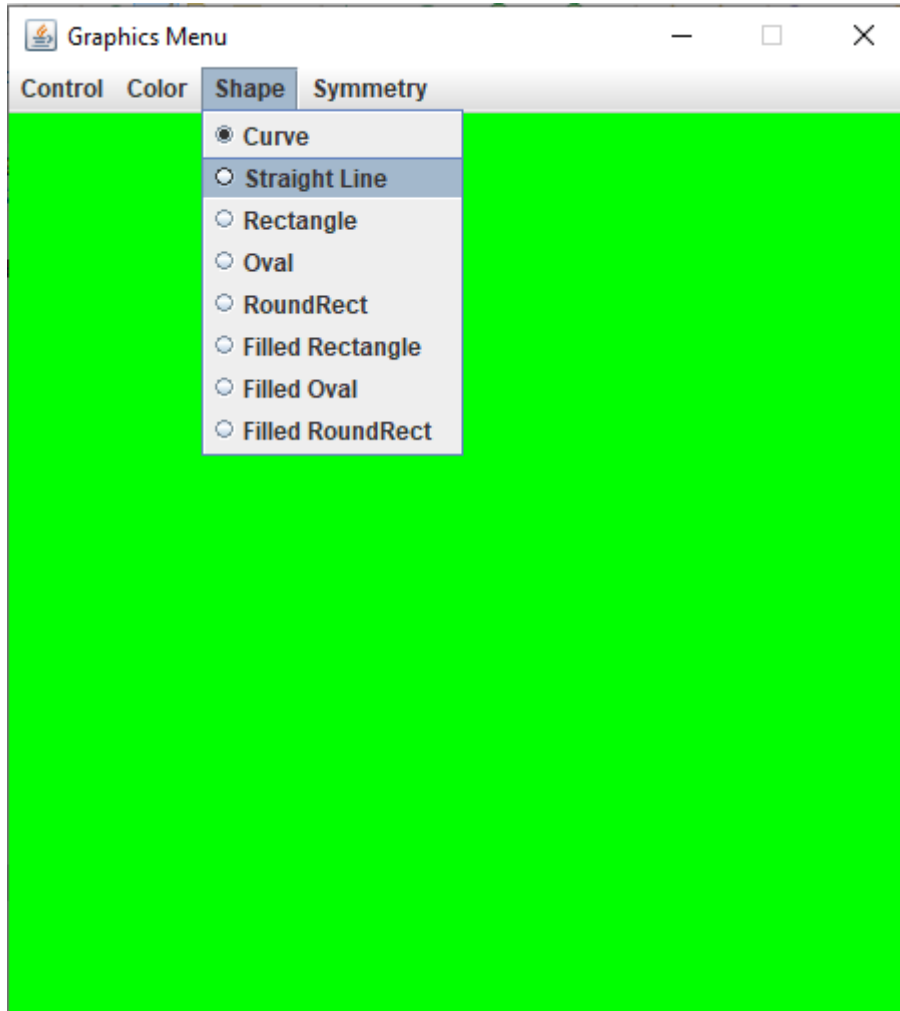
} // end nested class Display

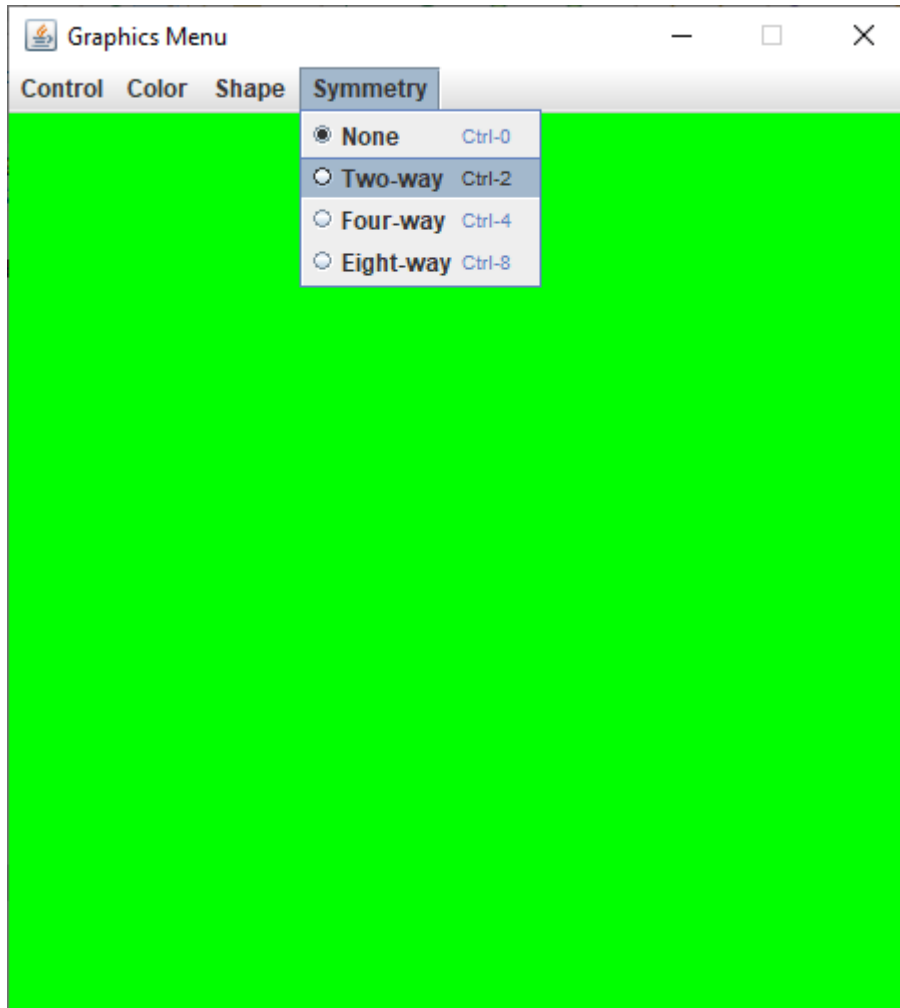
} // end class Menu
```

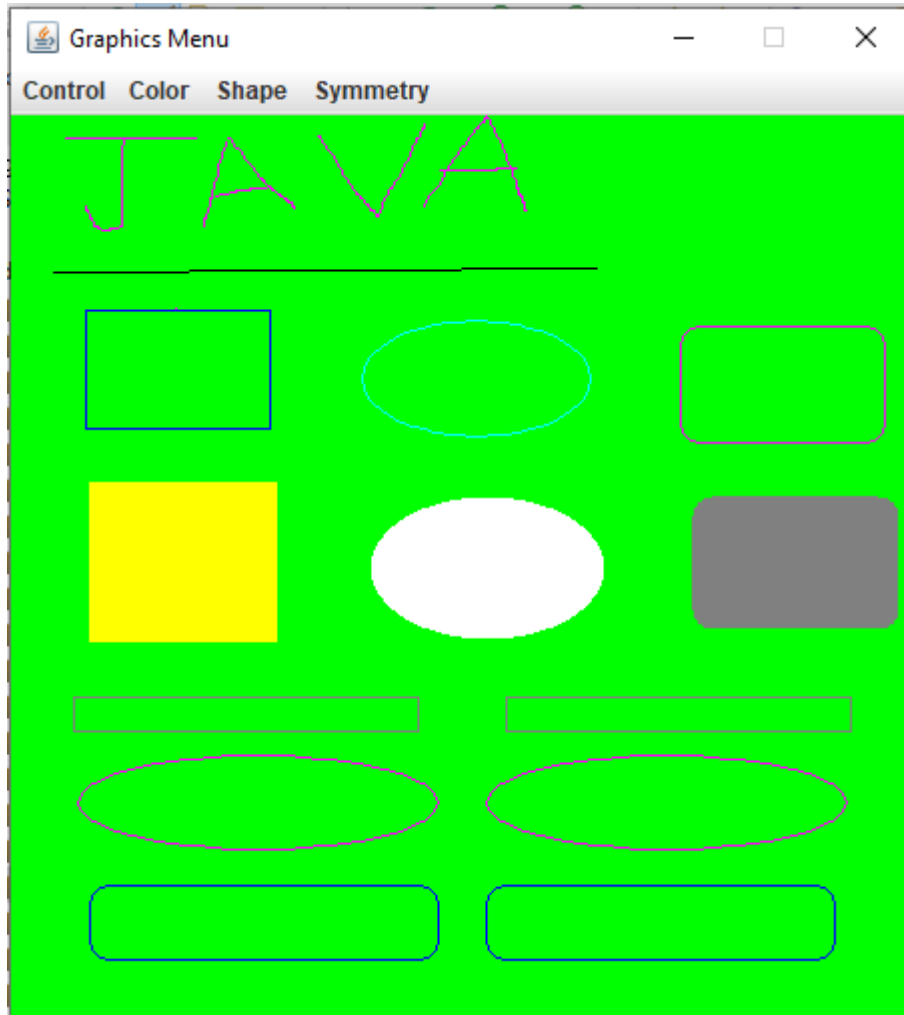
OUTPUT:











EXPERIMENT NO: 18,19

AIM:

Write a Java program to implement JTable and JTree.

DESCRIPTION:

Class TreeSelectionEvent

java.lang.Object

java.util.EventObject

javax.swing.event.TreeSelectionE

vent

An event that characterizes a change in the current selection. The change is based on any number of paths. TreeSelectionListeners will generally query the source of the event for the new selected status of each potentially changed row.

To detect when the user selects a node in a tree, you need to register a tree selection listener. Here is an example, taken from the TreeDemo example discussed in Responding to Node Selection, of detecting node selection in a tree that can have at most one node selected at a time

The JTabbedPane class is used to switch between a group of components by clicking on a tab with a given title or icon. It inherits JComponent class.

SYNTAX:

public class TreeSelectionEvent extends EventObject public class JTabbedPane extends

JComponent implements Serializable, Accessible, SwingConstants

JTabbedPane()Creates an empty TabbedPane with a default tab placement of
JTabbedPane.Top.

JTabbedPane(int tabPlacement) Creates an empty TabbedPane with a specified tab
placement.

JTabbedPane(int tabPlacement, int tabLayoutPolicy) Creates an empty TabbedPane
with a specified tab placement and tab layout policy.

PROGRAM:

```
//Demonstrate JTabbedPane.
import javax.swing.*;
import javax.swing.event.TreeSelectionEvent;
import javax.swing.event.TreeSelectionListener;
import javax.swing.tree.DefaultMutableTreeNode;

import java.awt.*;

public class JTabbedPaneDemo {

    public JTabbedPaneDemo() {

        // Set up the JFrame.
        JFrame jfrm = new JFrame("JTabbedPaneDemo");
        jfrm.setLayout(new FlowLayout());
        jfrm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        jfrm.setSize(1000, 1000);

        // Create the tabbed pane.
        JTabbedPane jtp = new JTabbedPane();
        jtp.addTab("Tree", new TreePanel());
        jtp.addTab("Table", new TablePanel());

        jfrm.add(jtp);

        // Display the frame.
        jfrm.setVisible(true);
    }

    public static void main(String[] args) {

        // Create the frame on the event dispatching thread.
        SwingUtilities.invokeLater(
            new Runnable() {
                public void run() {
                    new JTabbedPaneDemo();
                }
            }
        );
    }

    //Make the panels that will be added to the tabbed pane.
    class TreePanel extends JPanel {

        public TreePanel() {

            // Create top node of tree.
            DefaultMutableTreeNode top = new DefaultMutableTreeNode("Options");
```

```
// Create subtree of "A".
DefaultMutableTreeNode a = new DefaultMutableTreeNode("A");
top.add(a);
DefaultMutableTreeNode a1 = new DefaultMutableTreeNode("A1");
a.add(a1);
DefaultMutableTreeNode a2 = new DefaultMutableTreeNode("A2");
a.add(a2);

// Create subtree of "B".
DefaultMutableTreeNode b = new DefaultMutableTreeNode("B");
top.add(b);
DefaultMutableTreeNode b1 = new DefaultMutableTreeNode("B1");
b.add(b1);
DefaultMutableTreeNode b2 = new DefaultMutableTreeNode("B2");
b.add(b2);
DefaultMutableTreeNode b3 = new DefaultMutableTreeNode("B3");
b.add(b3);

// Create the tree.
JTree tree = new JTree(top);

// Add the tree to a scroll pane.
JScrollPane jsp = new JScrollPane(tree);

// Add the scroll pane to the content pane.
add(jsp);

// Add the label to the content pane.
JLabel jlab = new JLabel();
add(jlab, BorderLayout.SOUTH);

// Handle tree selection events.
tree.addTreeSelectionListener(new TreeSelectionListener() {
    public void valueChanged(TreeSelectionEvent tse) {
        jlab.setText("Selection is " + tse.getPath());
    }
});
}
}

class TablePanel extends JPanel {

    public TablePanel() {

        // Initialize column headings.
        String[] colHeads = { "Name", "Extension", "ID#" };

        // Initialize data.
        Object[][] data = {
            { "Gail", "4567", "865" },

```

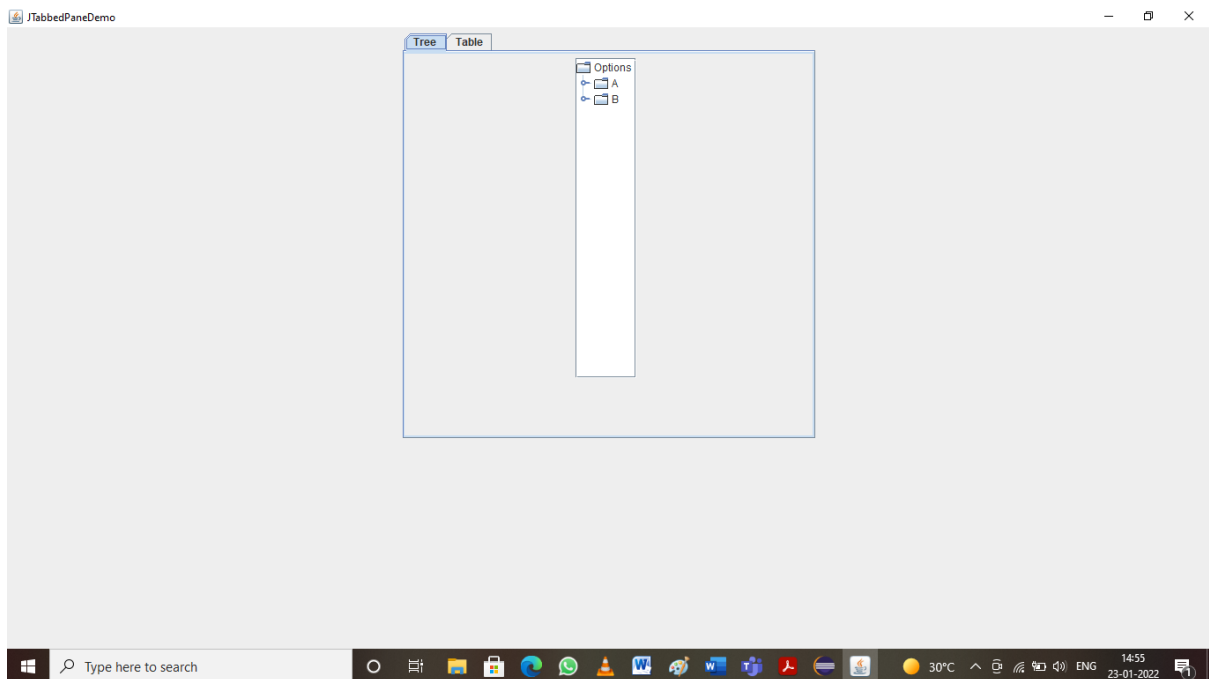
```
{ "Ken", "7566", "555" },
{ "Viviane", "5634", "587" },
{ "Melanie", "7345", "922" },
{ "Anne", "1237", "333" },
{ "John", "5656", "314" },
{ "Matt", "5672", "217" },
{ "Claire", "6741", "444" },
{ "Erwin", "9023", "519" },
{ "Ellen", "1134", "532" },
{ "Jennifer", "5689", "112" },
{ "Ed", "9030", "133" },
{ "Helen", "6751", "145" }
};

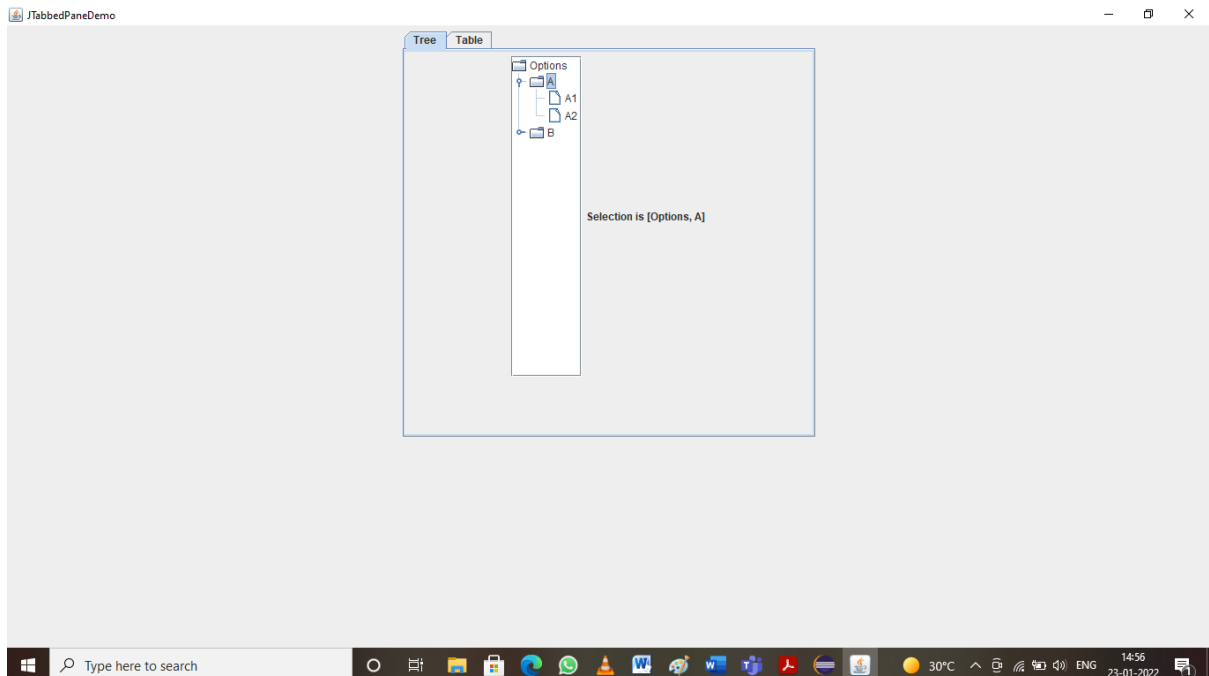
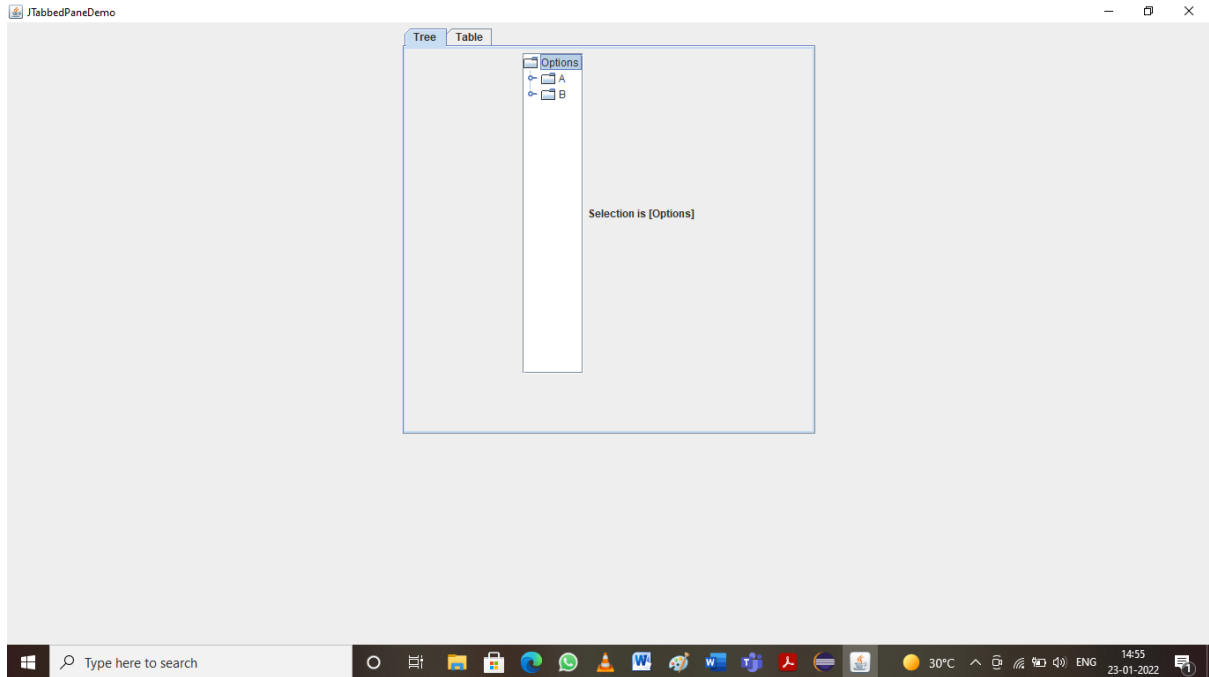
// Create the table.
JTable table = new JTable(data, colHeads);

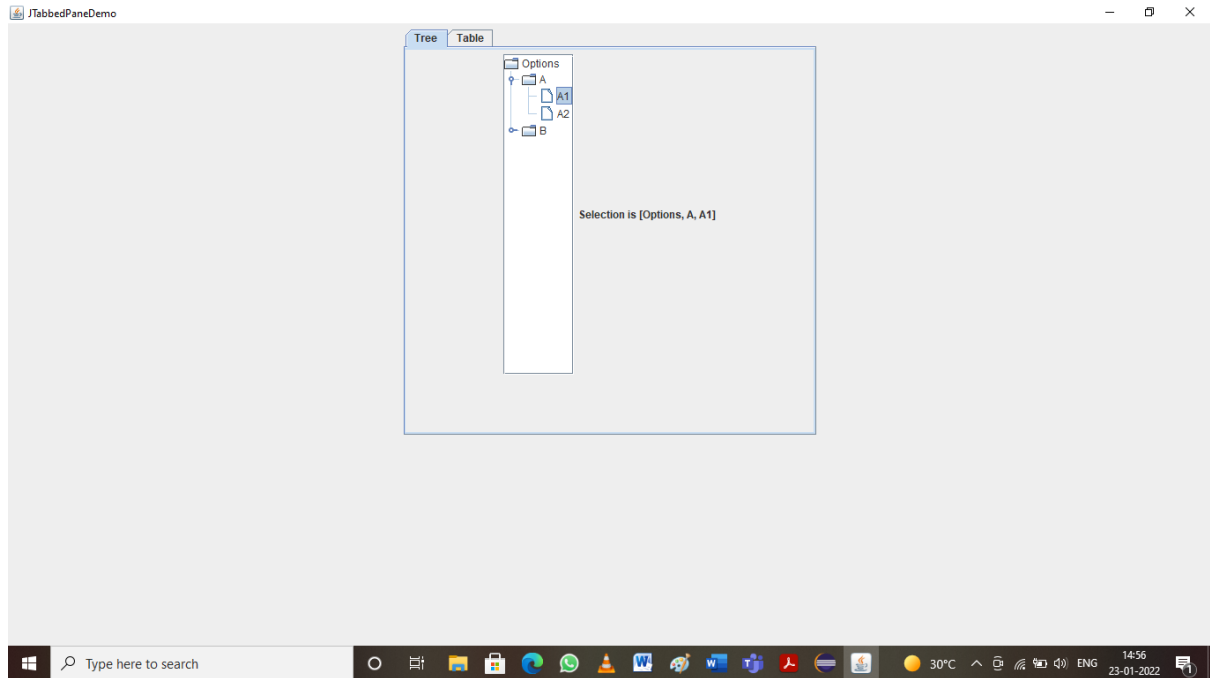
// Add the table to a scroll pane.
JScrollPane jsp = new JScrollPane(table);

// Add the scroll pane to the content pane.
add(jsp);
}
}
```

OUTPUT:







The image shows two sequential screenshots of a Java Swing application titled "JTabbedPaneDemo". The application features two tabs: "Tree" and "Table". The "Tree" tab is active in both screenshots. It displays a tree structure with a root node "Options", which has three children: "A", "A1", and "A2". Node "A" has a child "B". In the first screenshot, "A2" is selected, and the text "Selection is [Options, A, A2]" is displayed. In the second screenshot, "B" is selected, and the text "Selection is [Options, B]" is displayed. The application is running on a Windows 10 desktop with a taskbar showing various icons and system information like temperature (31°C) and date (23-01-2022).

JTabbedPaneDemo

Tree Table

Options

- A
 - A1
 - A2
- B
 - B1
 - B2
 - B3

Selection is [Options, B, B1]

Type here to search

31°C 14:57 23-01-2022

JTabbedPaneDemo

Tree Table

Options

- A
 - A1
 - A2
- B
 - B1
 - B2
 - B3

Selection is [Options, B, B2]

Type here to search

31°C 14:57 23-01-2022

JTabbedPaneDemo

Tree Table

Options

- A
 - A1
 - A2
- B
 - B1
 - B2
 - B3

Selection is [Options, B, B3]

Type here to search

31°C 14:57 23-01-2022

JTabbedPaneDemo

Tree Table

Name	Extension	ID#
Gail	4567	865
Ken	7566	555
Viviane	5634	587
Melanie	7345	922
Anne	1237	333
John	5656	314
Matt	5672	217
Claire	6741	444
Erwin	9023	519
Ellen	1134	532
Jennifer	5689	112
Ed	9030	133
Helen	6751	145

Type here to search

31°C 15:12 23-01-2022

EXPERIMENT NO: 20

AIM:

Write a Java Program that implements a simple client/server application. The client sends data to a server. The server receives the data, uses it to produce a result and then sends the result back to the client. The client displays the result on the console. For ex: The data sent from the client is the radius of a circle and the result produced by the server is the area of the circle.

DESCRIPTION:

Socket Programming Socket programming is used to make a connection between two nodes namely server and client on a network. By using this we can create a two-way connection between multiple nodes.

Logic

- 1) Firstly we will use sockets to request a connection between the nodes by passing the port number and keeping the host as localhost.
- 2) Once the server accepts the connection, we will implement a Runnable interface and override its methods to display the messages between the nodes.
- 3) We have used ExecutorService to create a thread pool and to connect multiple clients with the server at a time.
- 4) We will be using Threads to handle multiple messages from clients at a time.
- 5) Once the message is sent by any node our program will stop.

SYNTAX:

Public Socket accept() // returns socket and establish connection between server and client

Public synchronized void close() //closes the sever socket

PROGRAM:

```
import java.io.*;
import java.net.*;
public class Server {

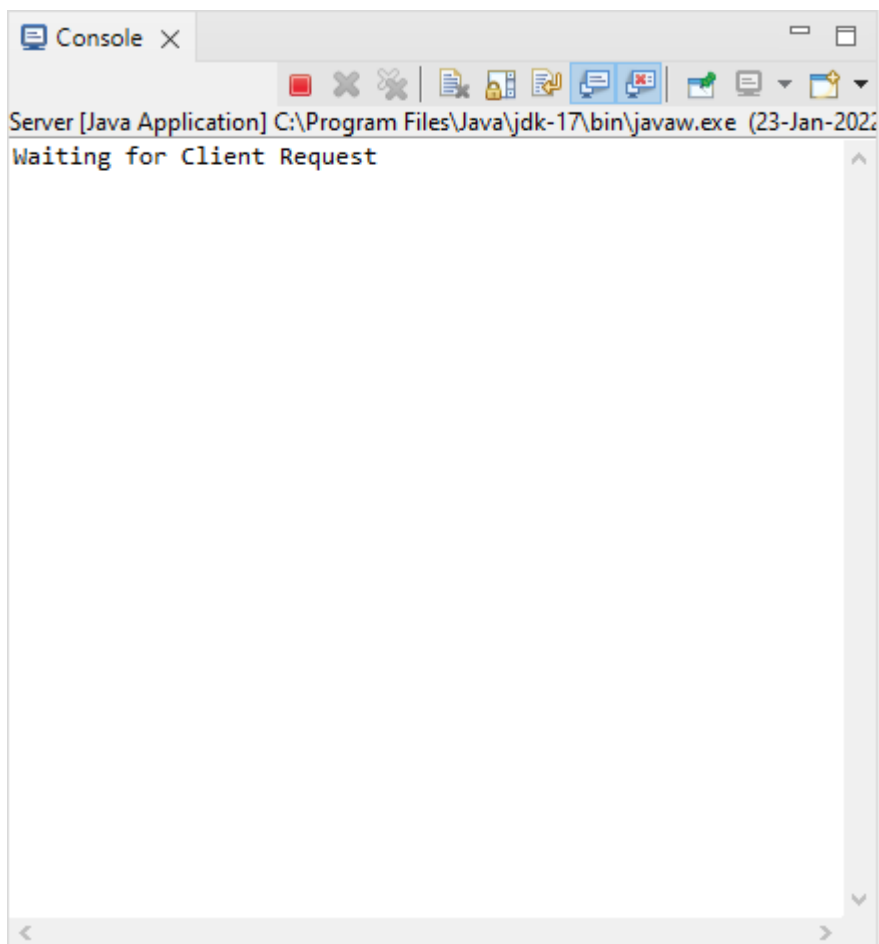
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        try
        {
            ServerSocket ss=new ServerSocket(1064);
            System.out.println("Waiting for Client Request");
            Socket s=ss.accept();
            BufferedReader br;
            PrintStream ps;
            String str;
            br=new BufferedReader(new
InputStreamReader(s.getInputStream()));
            str=br.readLine();
            System.out.println("Received radius");
            double r=Double.parseDouble(str);
            double area=3.14*r*r;
            ps=new PrintStream(s.getOutputStream());
            ps.println(String.valueOf(area));
            br.close();
            ps.close();
            s.close();
            ss.close();
        }
        catch(Exception e)
        {
            System.out.println(e);
        }
    }
}
```

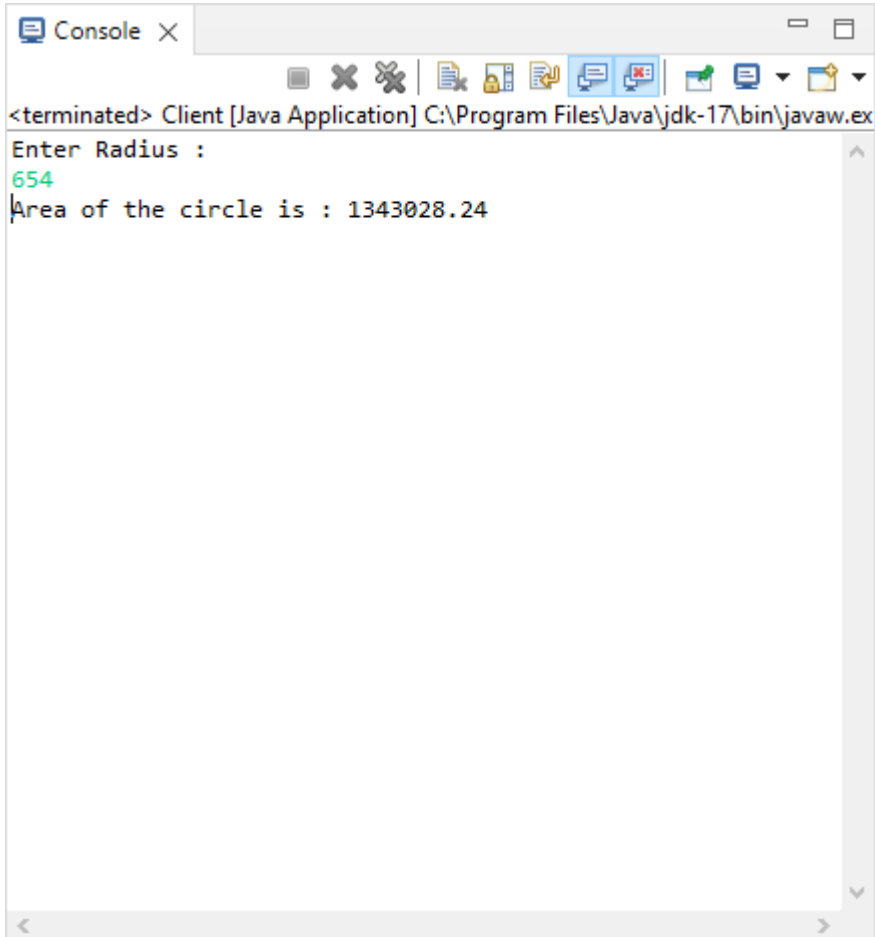
```
import java.io.*;
import java.net.*;
public class Client {

    public static void main(String[] args)throws IOException {
        // TODO Auto-generated method stub
        Socket s=new Socket(InetAddress.getLocalHost(),1064);
        BufferedReader br;
        PrintStream ps;
        String str;
        System.out.println("Enter Radius :");
```

```
        br=new BufferedReader(new InputStreamReader(System.in));
        ps=new PrintStream(s.getOutputStream());
        ps.println(br.readLine());
        br=new BufferedReader(new InputStreamReader(s.getInputStream()));
        str=br.readLine();
        System.out.println("Area of the circle is : "+str);
        br.close();
        ps.close();
    }
}
```

OUTPUT:





```
<terminated> Client [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe
Enter Radius :
654
Area of the circle is : 1343028.24
```


EXPERIMENT NO: 1

AIM:

Write a java program to implement the functions in String class.

DESCRIPTION:

In Java, string is basically an object that represents sequence of char values. An array of characters works same as Java string

Java String class provides a lot of methods to perform operations on strings such as compare(), concat(), equals(), split(), length(), replace(), compareTo(), intern(), substring() etc.

The java.lang.String class implements Serializable, Comparable and CharSequence

interfaces.No.

Method Description

- | | | |
|---|--|---|
| 1 | char charAt(int index) | It returns char value for the particular index |
| 2 | int length() | It returns string length |
| 3 | static String format(String format, Object... args) | It returns a formatted string. |
| 4 | String substring(int beginIndex)
index. | It returns substring for given begin index. |
| 5 | String substring(int beginIndex, int endIndex)
index | It returns substring for given begin index and end index. |
| 6 | String concat(String str) | It concatenates the specified string. |
| 7 | String replace(char old, char new) | It replaces all occurrences of the specified char value. |
| 8 | String replace(CharSequence old, CharSequence new)
of the | It replaces all occurrences of the |

specified CharSequence.

- | | | |
|----|--|--|
| 9 | static String equalsIgnoreCase(String another) | It compares another string. It doesn't |
| | | check case. |
| 10 | String[] split(String regex) | It returns a split string matching regex. |
| 11 | int indexOf(int ch) | It returns the specified char value index. |
| 12 | int indexOf(String substring) | It returns the specified substring index. |
| 13 | String toLowerCase() | It returns a string in lowercase. |
| 14 | String toUpperCase() | It returns a string in uppercase. |
| 15 | String trim() | It removes beginning and ending spaces of this string. |

PROGRAM:

```
public class Strings {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        String s=new String("Welcome to JAVA");//Constructor declaration
        System.out.println(s);
        byte ascii[]={67,69,87,99};
        String s1=new String(ascii);
        System.out.println(s1);
        char c[]={ 'P', 'R', 'A', 'V', 'E', 'E', 'N' };
        String s3=new String(c);
        System.out.println(s3);
        System.out.println(s3.length());//length of string
        System.out.println(s+" "+s3);//concatenation of strings
        System.out.println(s3+" "+2+"-"+"CSE-A");//concatenation of strings
        of various data types
        System.out.println("Addition of 31+541:"+(31+541));//concatenation
        of strings of various data types
        String s4="Hello";
        System.out.println(s4.charAt(2));//charAt() method
```

```

char target[]=new char[5-2];
s4.getChars(2,5,target,0);
System.out.println(target);//getchars() method
s="abcde";
s1="ABCDE";
String s2=new String("Welcome to JAVA");
System.out.println(s2.hashCode()+"\n"+s.hashCode());
byte baa[]=s.getBytes();//getBytes() method
for(int i=0;i<s.length();i++)
{
System.out.println(baa[i]);
}
char c1[]=s.toCharArray();//toCharArray() method
c1[1]='c';
System.out.println(c1);
System.out.println(s.equals(s4));//equals() case
System.out.println(s.equalsIgnoreCase(s1));//equalsIgnoreCase()

method
System.out.println(s.regionMatches(0,s1,0,5));//regionMathes()

method
System.out.println(s.regionMatches(true,0,s1,0,5));//regionMatchesIgnoreCas
e() method
System.out.println(s.startsWith("abc"));//startswith() function
System.out.println(s.endsWith("cde"));//endswith() function
Strings S= new Strings();
System.out.println(S);
String a="praveen";
String A=new String(a);
System.out.println(a.equals(A));//equals()
System.out.println(a==A);//==
System.out.println(a.compareTo(s1));//compareTo() method is in
interface comparable<T>

System.out.println(a.compareToIgnoreCase(s1));//compareToIgnoreCase()
method
}

}

public class Strings1 {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        String s="Praveen";
        System.out.println(s.indexOf("e"));//indexOf() method
        System.out.println(s.lastIndexOf("e"));//lastIndexOf() method

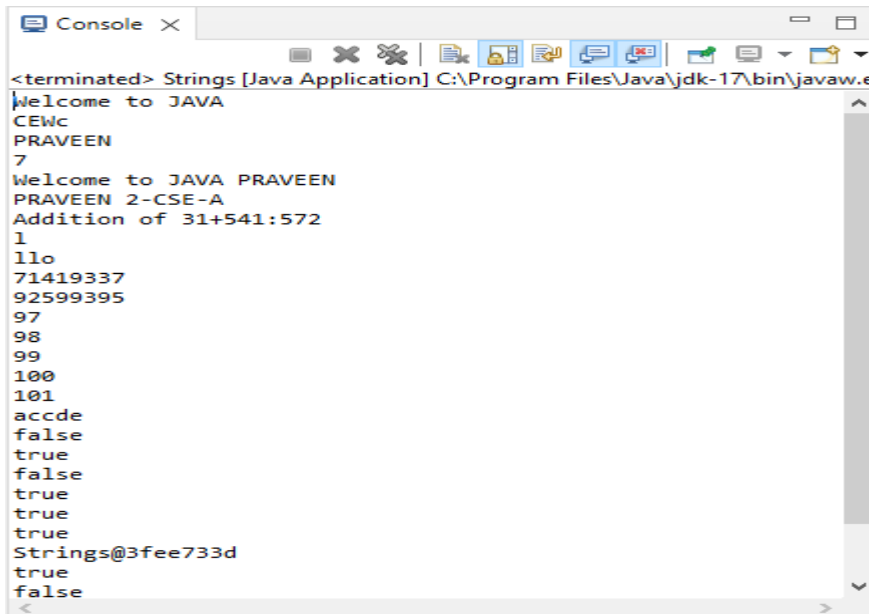
```

```

        System.out.println(s.indexOf("v",2)); //indexOf() method using
        decalration index
        String s1=s.substring(2,7); //substring() method
        System.out.println(s1);
        System.out.println(s.concat(s1)); //concat() method
        System.out.println(s.replace('P','p')); //repalce() method
        String s3=" Pandu ";
        System.out.println(s.trim()); //trim() method
        System.out.println(s.strip()); //strip() method
        System.out.println(s3.stripLeading()); //stripLeading() method
        System.out.println(s3.stripTrailing()); //StripTrailing() method
        System.out.println(s.getClass().getSimpleName()); //determines the
        data type of a string
        System.out.println(String.valueOf(10)); //valueOf() method
        System.out.println(s.toLowerCase()); //toLowerCase() method
        System.out.println(s.toUpperCase()); //toUpperCase() method
        System.out.println(String.join(" ",s,s1,s3)); //join() method uses
        to join the set of strings
    }
}

```

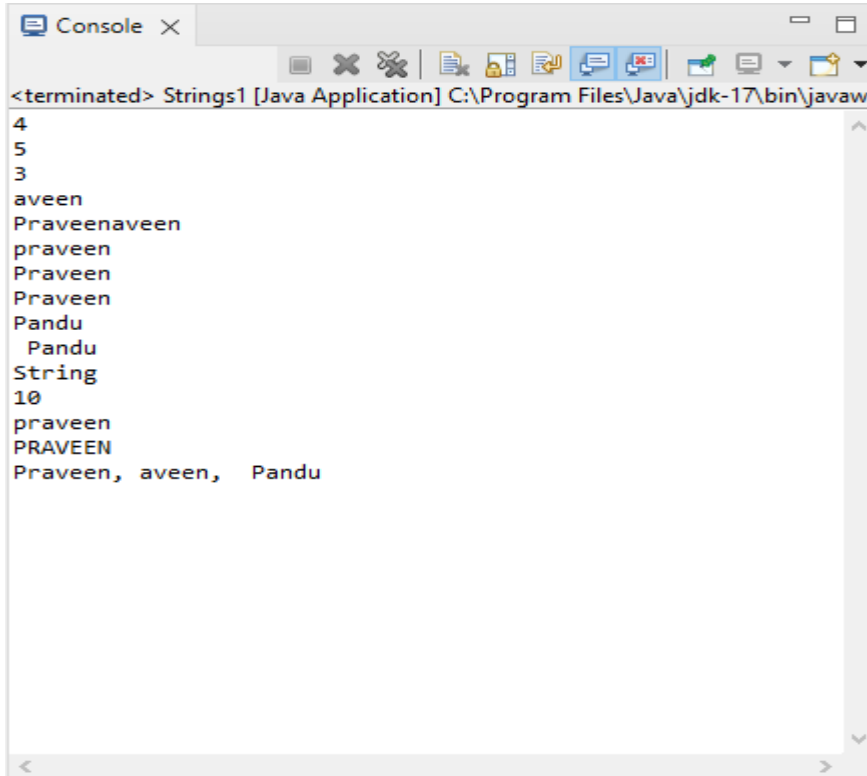
OUTPUT:



```

<terminated> Strings [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.e
Welcome to JAVA
CEWc
PRAVEEN
7
Welcome to JAVA PRAVEEN
PRAVEEN 2-CSE-A
Addition of 31+541:572
1
11o
71419337
92599395
97
98
99
100
101
accde
false
true
false
true
true
true
Strings@3fee733d
true
false
<

```



```
<terminated> Strings1 [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.  
4  
5  
3  
aveen  
Praveenaveen  
praveen  
Praveen  
Praveen  
Pandu  
Pandu  
String  
10  
praveen  
PRAVEEN  
Praveen, aveen, Pandu
```

EXPERIMENT NO: 2

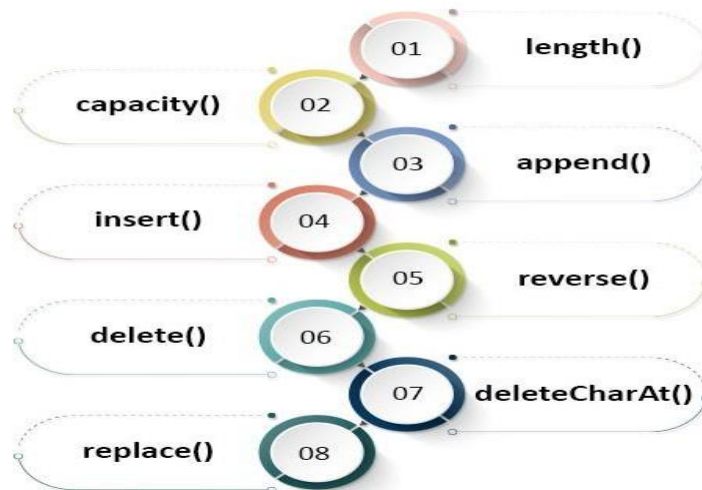
AIM:

Write a java program to implement the functions in StringBuffer class.

DESCRIPTION:

Java StringBuffer class is used to create mutable (modifiable) String objects. The StringBuffer class in Java is the same as String class except it is mutable i.e. it can be changed.

StringBuffer Methods in Java



SYNTAX:

StringBuffer()

StringBuffer(String str)

StringBuffer(int capacity)

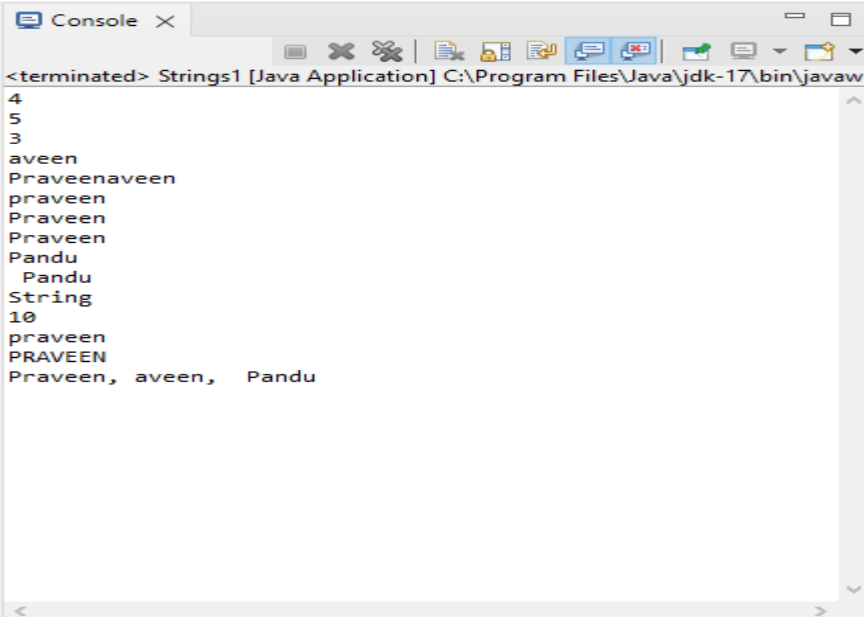
PROGRAM:

```
public class StrBuff {  
  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
        StringBuffer sb =new StringBuffer("Prasanth Sir");  
        StringBuffer s=new StringBuffer("Welcome");  
        System.out.println(sb.length()); //length() function  
        System.out.println(sb.capacity()); //capacity() function  
    }  
}
```



```
s.ensureCapacity(18);//ensureCapacity Function()
System.out.println(s.capacity());//after applied of ensureCapacity() capacity of
the StringBuffer
s.setLength(25);
System.out.println(s.length());//setLength() function
System.out.println(sb.charAt(10));//charAt() function
char ch='p';
sb.setCharAt(0,ch);//setCharAt() function
System.out.println(sb.charAt(0));
System.out.println(sb);
String a="JAVA PROGRAMMING";
char b[]=new char[10];
a.getChars(2,7,b,3);//getChars() function here last parameter targetStart do the
spaces for the final output of the character array
System.out.println(b);
System.out.println(sb.append(" WELCOME"));//append() function
System.out.println(s.insert(8,"to JAVA"));//insert() method
System.out.println(s.reverse());//reverse() function
System.out.println(sb.delete(0,8));//delete() function
System.out.println(sb.deleteCharAt(4));//deleteCharAt() function
System.out.println(sb.replace(0,0,"Prasanth"));//replace() function
System.out.println(sb.substring(14));//substring()
    }
}
```

OUTPUT:



```
<terminated> Strings1 [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.
4
5
3
aveen
Praveenaveen
praveen
Praveen
Praveen
Pandur
Pandur
String
10
praveen
PRAVEEN
Praveen, aveen, Pandur
```


EXPERIMENT NO: 3

AIM:

Write a java program to implement stacks.

DESCRIPTION:

The stack is a linear data structure that is used to store the collection of objects. It is based on Last- In-First-Out (LIFO). Java collection framework provides many interfaces and classes to store the collection of objects. One of them is the Stack class that provides different operations such as push, pop, search, etc.

SYNTAX:

empty() The method checks the stack is empty or not.

pop() The method removes an element from the top of the stack and returns the same element as the value of that function.

peek() The method looks at the top element of the stack without removing it.

PROGRAM:

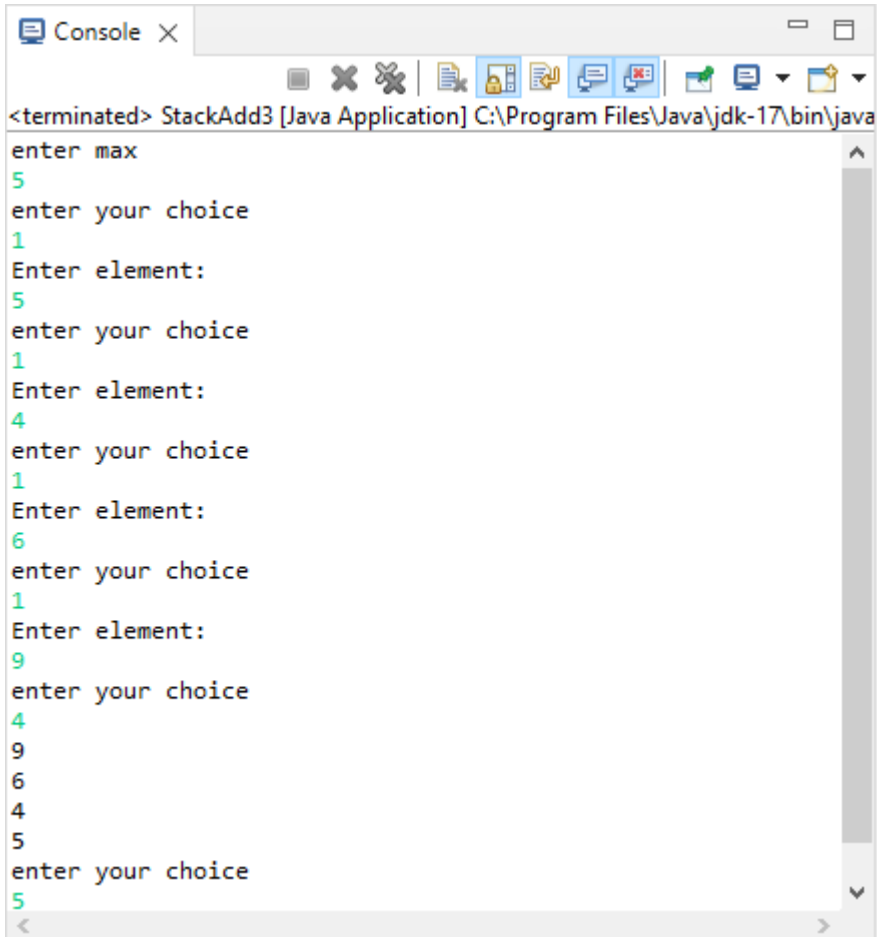
```
import java.util.Scanner;
class StackN{
int max;
int top=-1;
Scanner s=new Scanner(System.in);
int stk[];
StackN()
{
System.out.println("enter max");
max=s.nextInt();
stk=new int[max];
}
void push(int e)
{
if(!isFull())
stk[++top]=e;
else
System.out.println("stackoverflow");
}
void pop()
{
if(!isEmpty())
System.out.println("the element which is deleted from the stack is"
+stk[top--]);
else
```

```
System.out.println("stackunderflow");
}
int peek()
{
if(!isEmpty())
return stk[top];
else
return -1;
}
boolean isEmpty()
{
if(top== -1)
return true;
else
return false;
}
boolean isFull()
{
if(top==max-1)
return true;
else
return false;
}
void display()
{
for(int i=top;i>=0;i--)
System.out.println(stk[i]);
}
}
public class StackAdd3 {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        StackN x=new StackN();
        Scanner s= new Scanner(System.in);
        int ch,ch1;
        while(true)
        {
            System.out.println("enter your choice");
            ch=s.nextInt();
            switch(ch)
            {
                case 1:
                    System.out.println("Enter element:");
                    ch1=s.nextInt();
                    x.push(ch1);
                    break;
                case 2:
                    x.pop();
                    break;
                case 3:
```

```
        System.out.println(x.peek());  
        break;  
        case 4:  
            x.display();  
            break;  
        default:  
            return;  
        }  
    }  
}
```

OUTPUT:



```
<terminated> StackAdd3 [Java Application] C:\Program Files\Java\jdk-17\bin\java  
enter max  
5  
enter your choice  
1  
Enter element:  
5  
enter your choice  
1  
Enter element:  
4  
enter your choice  
1  
Enter element:  
6  
enter your choice  
1  
Enter element:  
9  
enter your choice  
4  
9  
6  
4  
5  
enter your choice  
5
```

EXPERIMENT NO: 4

AIM:

Write a java program to implement queues.

DESCRIPTION:

The Queue interface present in the java.util package and extends the Collection interface is used to hold the elements about to be processed in FIFO(First In First Out) order. It is an ordered list of objects with its use limited to insert elements at the end of the list and deleting elements from the start of the list, (i.e.), it follows the FIFO or the First-In-First-Out principle.

SYNTAX:

Method	Description
add(int index, element)	This method is used to add an element at a particular index in the list. When a single parameter is passed, it simply adds the element at the end of the list.
addAll(int index, Collection collection)	This method is used to add all the elements in the given collection to the list. When a single parameter is passed, it adds all the elements of the given collection at the end of the list.
size()	This method is used to return the size of the list.
clear()	This method is used to remove all the elements in the list. However, the reference of the list created is still stored.
remove(int index)	This method removes an element from the specified index. It shifts subsequent elements(if any) to left and decreases their indexes by 1.
remove(element)	This method is used to remove the first occurrence of the given element in the list.

get(int index)

This method returns elements at the specified index.

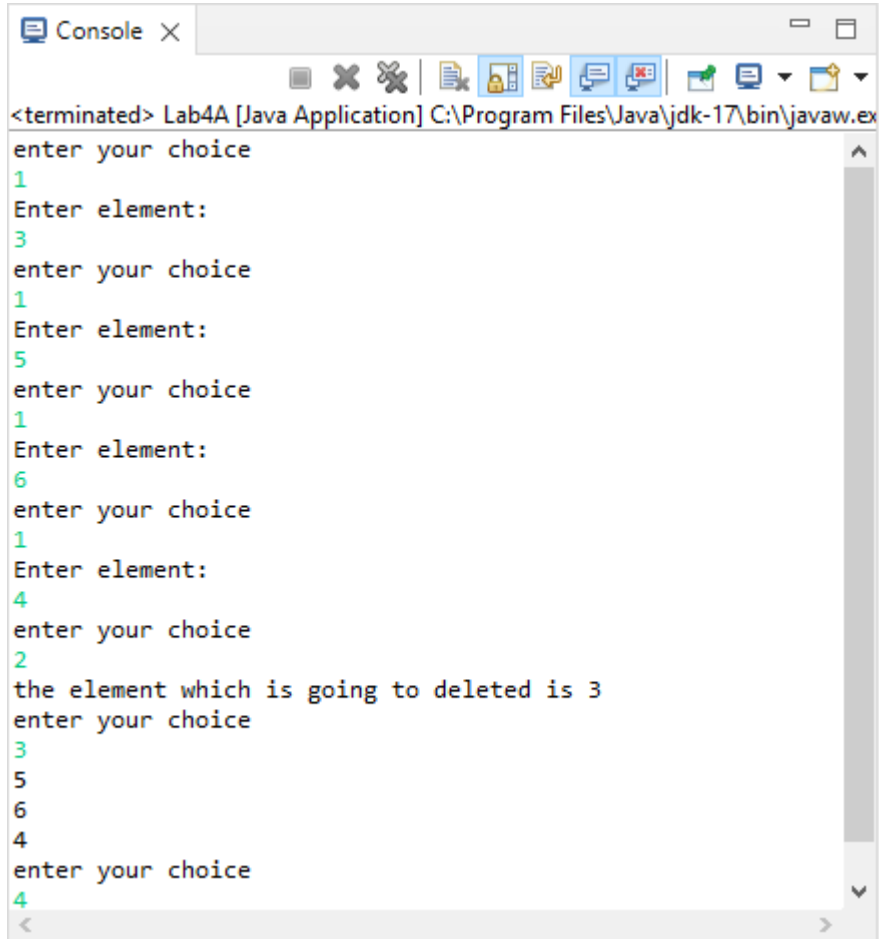
PROGRAM:

```
import java.util.Scanner;
class Queue1
{
    int max,f,r;
    int q[];
    Queue1()
    {
        f=r=-1;
        max=5;
        q=new int[max];
    }
    void enqueue(int e)
    {
        if(!isFull())
        {
            q[++r]=e;
            if(f== -1)
                f++;
        }
        else
        {
            System.out.println("queue is full");
        }
    }
    void dequeue()
    {
        if(!isEmpty())
        {
            System.out.println("the element which is going to deleted is "+q[f]);
            if(f==r)
                f=r=-1;
            else
                f++;
        }
        else
        {
            System.out.println("queue is empty");
        }
    }
    boolean isEmpty()
    {
        if(f== -1)
            return true;
        else
            return false;
    }
}
```

```
boolean isFull()
{
    if(r==max-1)
        return true;
    else
        return false;
}
void display()
{
    if(!isEmpty())
    {
        for (int i=f;i<=r;i++)
        {
            System.out.println(q[i]);
        }
    }
    else
        System.out.println("Queue is empty");
}
}
public class Lab4A {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Queue1 x=new Queue1();
        Scanner s=new Scanner(System.in);
        int ch,ch1;
        while(true)
        {
            System.out.println("enter your choice ");
            ch=s.nextInt();
            switch(ch)
            {
                case 1:
                    System.out.println("Enter element:");
                    ch1=s.nextInt();
                    x.enqueue(ch1);
                    break;
                case 2:
                    x.dequeue();
                    break;
                case 3:
                    x.display();
                    break;
                default:
                    return;
            }
        }
    }
}
```

OUTPUT:



```
<terminated> Lab4A [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe
enter your choice
1
Enter element:
3
enter your choice
1
Enter element:
5
enter your choice
1
Enter element:
6
enter your choice
1
Enter element:
4
enter your choice
2
the element which is going to deleted is 3
enter your choice
3
5
6
4
enter your choice
4
```


EXPERIMENT NO: 5

AIM:

Write a java program to demonstrate the usage of ByteStream classes.

DESCRIPTION:

ByteStream classes are used to read bytes from the input stream and write bytes to the output stream. In other words, we can say that ByteStream classes read/write the data of 8-bits. We can store video, audio, characters, etc., by using ByteStream classes. These classes are part of the java.io package.

The ByteStream classes are divided into two types of classes, i.e., InputStream and OutputStream. These classes are abstract and the super classes of all the Input/Output stream classes.

InputStream Class

The InputStream class provides methods to read bytes from a file, console or memory. It is an abstract class and can't be instantiated; however, various classes inherit the InputStream class and override its methods

SN	Method	Description
1	int read()	This method returns an integer, an integral representation of the next available byte of the input. The integer -1 is returned once the end of the input is encountered.
2	int read (byte buffer [])	This method is used to read the specified buffer length bytes from the input and returns the total number of bytes successfully read. It returns -1 once the end of the input is encountered.
3	int read (byte	This method is used to read the 'nBytes' bytes from the buffer

	buffer [], int loc, int nBytes)	starting at a specified location, 'loc'. It returns the total number of bytes successfully read from the input. It returns -1 once the end of the input is encountered.
4	int available ()	This method returns the number of bytes that are available to read.
5	Void mark(int nBytes)	This method is used to mark the current position in the input stream until the specified nBytes are read.
6	void reset ()	This method is used to reset the input pointer to the previously set mark.
7	long skip (long nBytes)	This method is used to skip the nBytes of the input stream and returns the total number of bytes that are skipped.
8	void close ()	This method is used to close the input source. If an attempt is made to read even after the closing, IOException is thrown by the method.

```
ByteArrayInputStream inputStream = new ByteArrayInputStream(content);
```

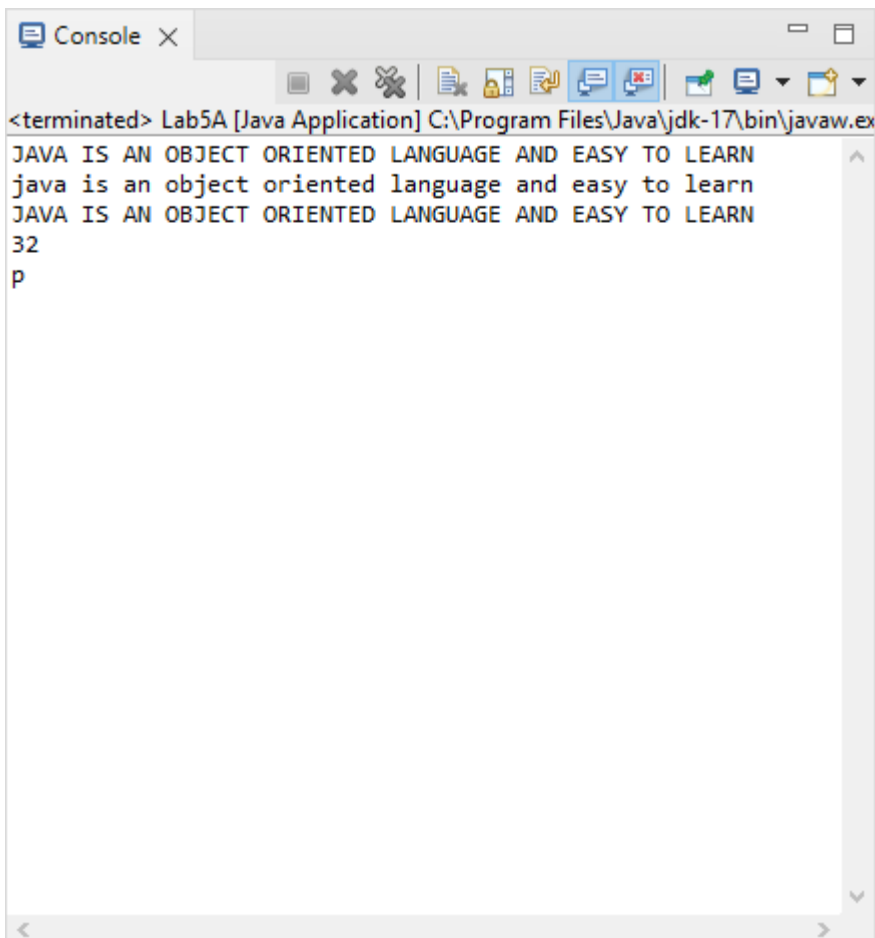
PROGRAM:

```
import java.io.*;
public class Lab5A {

    public static void main(String[] args)throws IOException {
        // TODO Auto-generated method stub
        String s="JAVA IS AN OBJECT ORIENTED LANGUAGE AND EASY TO LEARN";
        byte b1[]=s.getBytes();
        for(int i=0;i<b1.length;i++)
        {
            System.out.print((char)b1[i]);
        }
        ByteArrayInputStream bis=new ByteArrayInputStream(b1);
        int c;
        System.out.println();
        while((c=bis.read())!=-1)
        {
            char ch;
            ch=(char)c;
            System.out.print(Character.toLowerCase(ch));
        }
        System.out.println();
    }
}
```

```
        ByteArrayOutputStream bos=new ByteArrayOutputStream();  
        bos.write(b1);  
        String s1=bos.toString();  
        System.out.printf(s1);  
        DataOutputStream dos=new DataOutputStream(new FileOutputStream(  
            "E:\\JAVA.txt"));  
        dos.writeInt(32);  
        dos.writeChar('p');  
        System.out.println("");  
        DataInputStream dow=new DataInputStream(new FileInputStream(  
            "E:\\JAVA.txt"));  
        System.out.println(dow.readInt());  
        System.out.println(dow.readChar());  
    }  
}
```

OUTPUT:



The screenshot shows a console window titled "Console" with a tab for "Lab5A [Java Application]". The command prompt shows the execution of the Java application. The output consists of three lines: "JAVA IS AN OBJECT ORIENTED LANGUAGE AND EASY TO LEARN", "java is an object oriented language and easy to learn", and "JAVA IS AN OBJECT ORIENTED LANGUAGE AND EASY TO LEARN". Below these lines, the numbers "32" and the character "p" are printed on separate lines.

EXPERIMENT NO: 6

AIM:

Write a java program to demonstrate the usage of CharacterStream classes.

DESCRIPTION:

The java.io package provides CharacterStream classes to overcome the limitations of ByteStream classes, which can only handle the 8-bit bytes and is not compatible to work directly with the Unicode characters. CharacterStream classes are used to work with 16-bit Unicode characters. They can perform operations on characters, char arrays and Strings.

However, the CharacterStream classes are mainly used to read characters from the source and writethem to the destination. For this purpose, the CharacterStream classes are divided into two types ofclasses, I.e., Reader class and Writer class.

READER CLASS

Reader class is used to read the 16-bit characters from the input stream. However, it is an abstractclass and can't be instantiated, but there are various subclasses that inherit the Reader class and override the methods of the Reader class. All methods of the Reader class throw an IOException.

SYNTAX:

int read() returns the integral representation of the next available character of input. It returns -1 when end of file is encountered

int read (char buffer []) attempts to read buffer. length characters into the buffer and returns thetotal number of characters successfully read. It returns -1 when end of file is encountered

PROGRAM:

```
import java.io.*;

public class Lab6A {

    public static void main(String[] args) throws IOException {
        // TODO Auto-generated method stub
        //Creating FileReader object
        File file = new File("E:\\JAVA.txt");
        FileReader reader = new FileReader(file);
        char chars[] = new char[(int) file.length()];
        reader.read(chars);
        File out = new File("E:\\JAVA.txt");
```

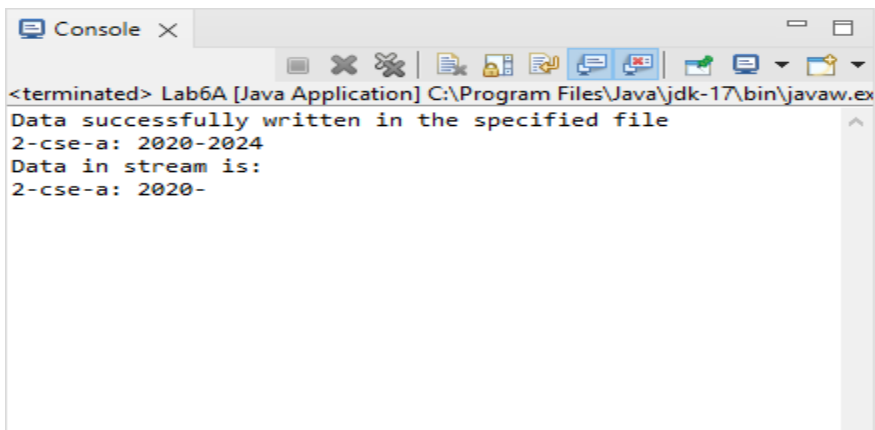
```

        FileWriter writer = new FileWriter(out);
        writer.write(chars);
        System.out.println("Data successfully written in the specified
file");

        writer.close();
        reader.close();
        String s="2-cse-a: 2020-2024";
        char c[]=new char[s.length()];
        s.getChars(0, s.length(), c, 0);
        BufferedReader br=new BufferedReader(new CharArrayReader(c));
        int x;
        while((x=br.read())>=0)
        System.out.print((char)x);
        BufferedWriter bw=new BufferedWriter(new
FileWriter("E:\\JAVA.txt"));
        bw.write(s);
        bw.close();
        br.close();
        System.out.println();
        String s1="VIVA VVIT 2021";
        char c1[]=new char[s.length()];
        s.getChars(0, s1.length(), c1, 0);
        CharArrayWriter cw=new CharArrayWriter();
        cw.write(c1);
        String s2=cw.toString();
        System.out.println("Data in stream is:");
        CharArrayReader cr=new CharArrayReader(c1);
        int x1;
        while((x1=cr.read())>=0)
        System.out.print((char)x1);
        cw.close();
        cr.close();
    }
}

```

OUTPUT:



The screenshot shows a Java application window titled "Lab6A [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe". The console output is as follows:

```

Data successfully written in the specified file
2-cse-a: 2020-2024
Data in stream is:
2-cse-a: 2020-

```

EXPERIMENT NO: 7

AIM:

Write a java program to demonstrate Serialization and Deserialization.

DESCRIPTION:

Serialization is a mechanism of converting the state of an object into a byte stream. Deserialization is the reverse process where the byte stream is used to recreate the actual Java object in memory. This mechanism is used to persist the object.

The byte stream created is platform independent. So, the object serialized on one platform can be deserialized on a different platform.

To make a Java object serializable we implement the java.io.Serializable interface.

The ObjectOutputStream class contains writeObject() method for serializing an Object.

SYNTAX:

```
public final void writeObject(Object obj)
```

throws IOException

PROGRAM:

```
import java.io.*;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
class DemoofSerialization implements java.io.Serializable {
public int a;
public String b;
// Default constructor
public DemoofSerialization(int a, String b)
{
this.a = a;
this.b = b;
}
}

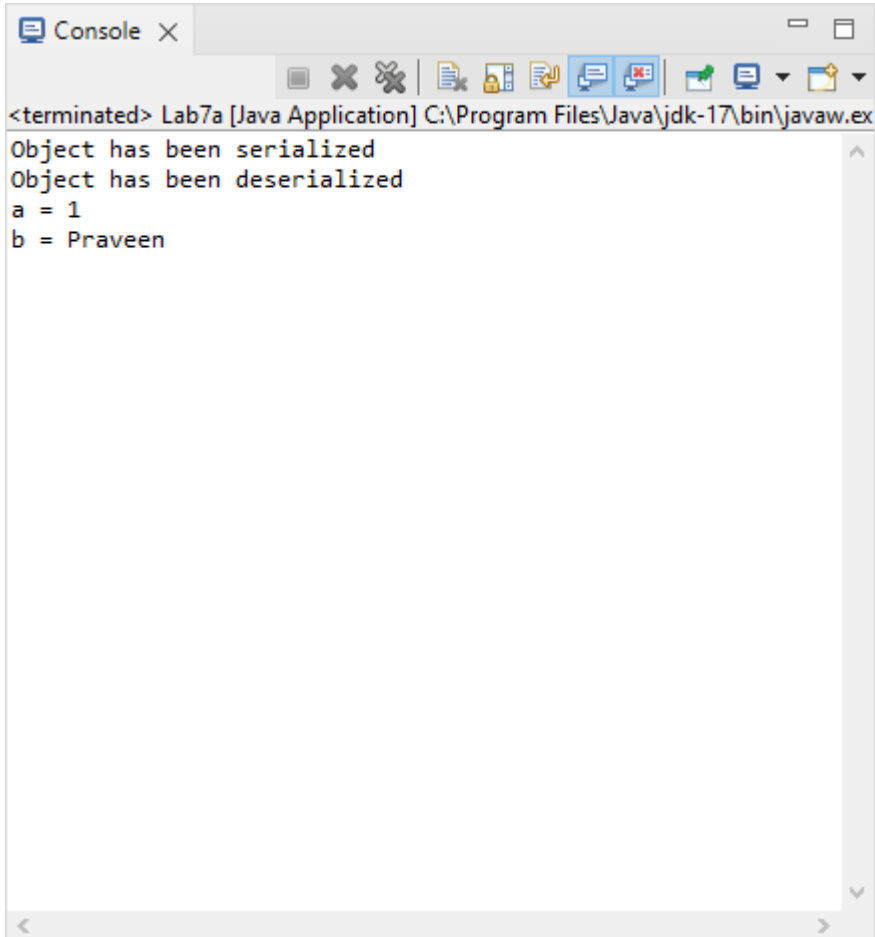
public class Lab7a {

    public static void main(String[] args) {
```



```
// TODO Auto-generated method stub
DemoofSerialization object = new DemoofSerialization(1, "Praveen");
String filename = "E:\\txt";
// Serialization
try
{
    //Saving of object in a file
    FileOutputStream file = new FileOutputStream(filename);
    ObjectOutputStream out = new ObjectOutputStream(file);
    // Method for serialization of object
    out.writeObject(object);
    out.close();
    file.close();
    System.out.println("Object has been serialized");
}
catch(IOException ex)
{
    System.out.println("IOException is caught");
}
DemoofSerialization object1 = null;
// Deserialization
try
{
    // Reading the object from a file
    FileInputStream file = new FileInputStream(filename);
    ObjectInputStream in = new ObjectInputStream(file);
    // Method for deserialization of object
    object1 = (DemoofSerialization)in.readObject();
    in.close();
    file.close();
    System.out.println("Object has been deserialized ");
    System.out.println("a = " + object1.a);
    System.out.println("b = " + object1.b);
}
catch(IOException ex)
{
    System.out.println("IOException is caught");
}
catch(ClassNotFoundException ex)
{
    System.out.println("ClassNotFoundException is caught");
}
}
}
```


OUTPUT:



```
<terminated> Lab7a [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe
Object has been serialized
Object has been deserialized
a = 1
b = Praveen
```