**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 | PO1 | PO2 | PO3 | PO4 | PO5 | PO6 | PO7 | PO8 | PO9 | PO10 | PO 11 | PO 12 | PSO1 | PSO2 |
| 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.

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% ofBP 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.

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

2. Write a java program to implement stacks.

3. Write a java program to implement queues.

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

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

6. Write a java program to demonstrate Serialization and Deserialization.

CERTIFICATE

Name of the Lab : JAVA PROGRAMMING

Name of the Student :

Student Regd. No. :

CLASS :

GIT HUB LINK:

INDEX

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| S. No. | Experiment Number | DATE OF COMPLETION | DATE OF SUBMISSION | PAGE NO | Remarks | Signature of Faculty |
| 1 | Experiment -1 |  |  |  |  |  |
| 2 | Experiment -2 |  |  |  |  |  |
| 3 | Experiment -3 |  |  |  |  |  |
| 4 | Experiment -4 |  |  |  |  |  |
| 5 | Experiment -5 |  |  |  |  |  |
| 6 | Experiment -6 |  |  |  |  |  |
| 7 | Experiment -7 |  |  |  |  |  |
| 8 | Experiment -8 |  |  |  |  |  |
| 9 | Experiment -9 |  |  |  |  |  |
| 10 | Experiment -10 |  |  |  |  |  |
| 11 | Experiment -11 |  |  |  |  |  |
| 12 | Experiment -12 |  |  |  |  |  |
| 13 | Experiment -13 |  |  |  |  |  |
| 14 | Experiment -14 |  |  |  |  |  |
| 15 | Experiment -15 |  |  |  |  |  |
| 16 | Experiment -16 |  |  |  |  |  |
| 17 | Experiment -17 |  |  |  |  |  |
| 18 | Experiment -18 |  |  |  |  |  |
| 19 | Experiment -19 |  |  |  |  |  |
| 20 | Experiment -20 |  |  |  |  |  |
| **Additional Experiments** | | | | | | |
| 21 | Experiment -1 |  |  |  |  |  |
| 22 | Experiment -2 |  |  |  |  |  |
| 23 | Experiment -3 |  |  |  |  |  |
| 24 | Experiment -4 |  |  |  |  |  |
| 25 | Experiment -5 |  |  |  |  |  |
| 26 | Experiment -6 |  |  |  |  |  |
| 27 | Experiment -7 |  |  |  |  |  |

**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:**

GETTER AND SETTER :

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

While 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. In both getter and setter, the first letter of the variable should be capital.

**SYNTAX:**

class GetSet {

private String name;

public String getName() {

return name;

}

public void setName(String N)

{ this.name = N;

}

}

**PROGRAM:**

import java.util.Scanner;

class Invoice{

String part\_number;

String part\_description;

int quantity;

double price\_item;

public double total=0;

Invoice(){

String part\_number="";

String part\_description="";

int quantity=0;

double price\_item=0.0;

}//writing getters and setters to initiate class variables and to retrieve values easily

public String getPart\_number() {

return part\_number;

}

public void setPart\_number(String part\_number) {

this.part\_number = part\_number;

}

public String getPart\_description() {

return part\_description;

}

public void setPart\_description(String part\_description) {

this.part\_description = part\_description;

}

public int getQuantity() {

return quantity;

}

public void setQuantity(int quantity) {

if(quantity<0)

quantity=0;

else

this.quantity = quantity;

}

public double getPrice\_item() {

return price\_item;

}

public void setPrice\_item(double price\_item) {

if(price\_item<0)

price\_item=0.0;

else

this.price\_item = price\_item;

}

double getInvoiceAmount() {

totalprice();

return total;

}//adding the price of all items

void totalprice(){

total=total+getQuantity()\*getPrice\_item();

}

}

public class Main {

public static void main(String[] args) {

Scanner sc=new Scanner(System.in);

Invoice in=new Invoice();

System.out.print("Enter part number :");

in.setPart\_number(sc.nextLine());

System.out.print("Enter part description :");

in.setPart\_description(sc.nextLine());

System.out.print("Enter no. of items purchased :");

in.setQuantity(sc.nextInt());

System.out.print("Enter price per item :");

in.setPrice\_item(sc.nextDouble());

System.out.print(" Item Details \n");

System.out.print("Part number :" + in.getPart\_number());

System.out.print("\nPart description :" + in.getPart\_description());

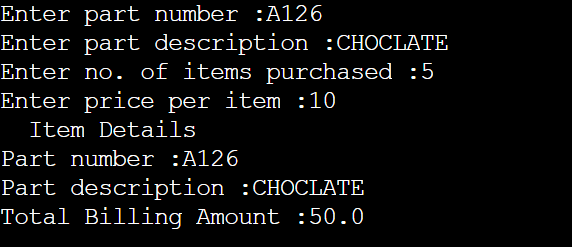
System.out.print("\nTotal Billing Amount :" + in.getInvoiceAmount());

}//main ending

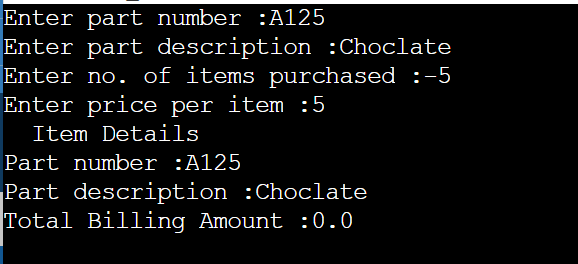
}//class ending

**OUTPUT:**

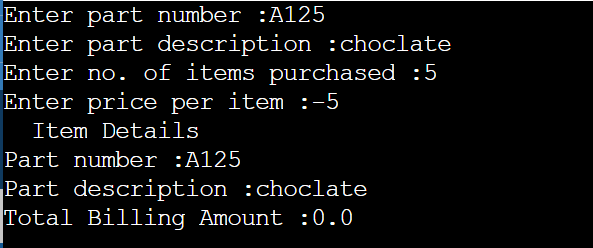
**Output 1**



Output 2

****

**Output 3**

****

**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

1. 101-200units - Rs. 2.50 per unit
2. 201 -500 units - Rs. 4 per unit
3. >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:**

# Java String compareTo()

The **Java String class compareTo()** method compares the given string with the current string lexicographically. It returns a positive number, negative number, or 0.

It compares strings on the basis of the Unicode value of each character in the strings.

If the first string is lexicographically greater than the second string, it returns a positive number (difference of character value). If the first string is less than the second string lexicographically, it returns a negative number, and if the first string is lexicographically equal to the second string, it returns 0.

1. **if** s1 > s2, it returns positive number
2. **if** s1 < s2, it returns negative number
3. **if** s1 == s2, it returns 0

**Syntax:**

**public** **int** compareTo(String anotherString)

The method accepts a parameter of type String that is to be compared with the current string.

It returns an integer value.

**PROGRAM:**

import java.util.Scanner;

class ElectricityBill{

Scanner sc=new Scanner(System.in);

Scanner scs=new Scanner(System.in);

String consumer\_number;

String consumer\_name;

double previous\_month\_reading;

double current\_month\_reading;

double amount;

String type;

ElectricityBill(){

String consumer\_number="";

String consumer\_name="";

double previous\_month\_reading=0.0;

double current\_month\_reading=0.0;

String type="";

}

void getBillDetails() {

System.out.println("enter consumer number:");

consumer\_number=scs.next();

System.out.println("enter consumer name:");

consumer\_name=scs.next();

System.out.println("enter previous month reading:");

previous\_month\_reading=sc.nextDouble();

System.out.println("enter current month reading:");

current\_month\_reading=sc.nextDouble();

System.out.println("enter type:");

type=scs.next();

}

void calculate\_amount() {

double units;

units=current\_month\_reading-previous\_month\_reading;

if(type.equals("d")){

if(units<=100)

amount=units\*1;

else if (units<=200)

amount=100+(units-100)\*2.50;

else if(units<=500)

amount=100+250+(units-200)\*4;

else

amount=units\*6;

}

else {

if(units<=100)

amount=units\*2;

else if(units<=200)

amount=200+(units-100)\*4.50;

else if(units<=500)

amount=200+450+(units-200)\*6;

else

amount=units\*7;

} }

void showBill() {

System.out.println("Consumer name:"+consumer\_name);

System.out.println("consumer number:"+consumer\_number);

if(type.equals("d"))

System.out.println("type=domestic");

else

System.out.println("type=commercial");

System.out.println("previous month reading"+previous\_month\_reading);

System.out.println("current month reading:"+current\_month\_reading);

System.out.println("electricityBill:"+amount);

}}

public class Lab2 {

public static void main(String[] args) {

ElectricityBill e=new ElectricityBill();

e.getBillDetails();

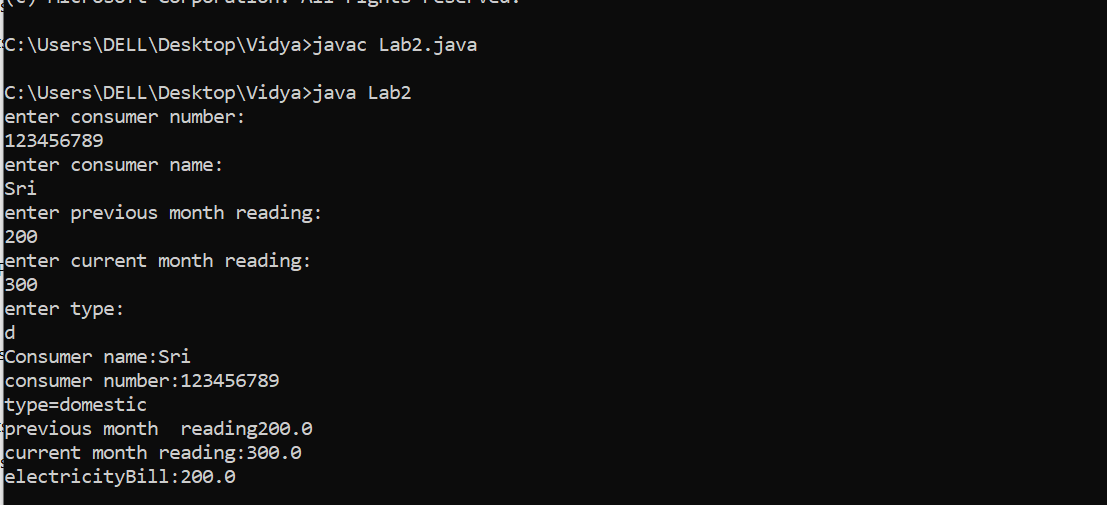
e.calculate\_amount();

e.showBill();

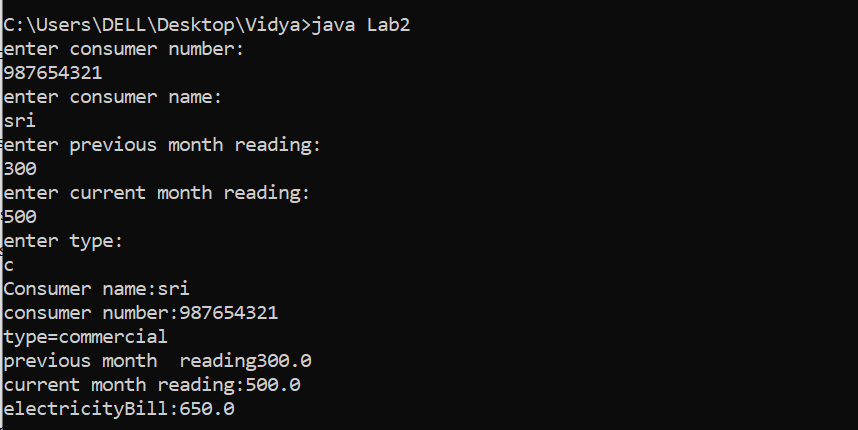
}

}

**Output 1**



**Output 2**

****

**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 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.

**DESCRIPTION:**

**JAVA STATIC KEYWORD:**

The **static keyword** in [Java](https://www.javatpoint.com/java-tutorial) is used for memory management mainly. We can apply static keyword with [variables](https://www.javatpoint.com/java-variables), methods, blocks and [nested classes](https://www.javatpoint.com/java-inner-class). The static keyword belongs to the class than an instance of the class.

## **1) Java static variable**

If you declare any variable as static, it is known as a static variable.

* The static variable can be used to refer to the common property of all objects (which is not unique for each object), for example, the company name of employees, college name of students, etc.
* The static variable gets memory only once in the class area at the time of class loading.

### **Advantages of static variable**

It makes your program **memory efficient** (i.e., it saves memory).

## **2) Java static method**

If you apply static keyword with any method, it is known as static method.

* A static method belongs to the class rather than the object of a class.
* A static method can be invoked without the need for creating an instance of a class.
* A static method can access static data member and can change the value of it.

## **3) Java static block**

* Is used to initialize the static data member.
* It is executed before the main method at the time of classloading.

**Syntax:**

**Static block:**

**class** A2{

**static**{System.out.println("static block is invoked");}

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

   //Statements

}

}

Static method:

**class** Calculate{

**static** **int** cube(**int** x){

**//Statements;**

  }

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

**int** result=Calculate.cube(5);

  //Statements;

  }

}

Static Variable:

**class** Counter2{

**static** **int** count=0;//will get memory only once and retain its value

Counter2(){

count++;//incrementing the value of static variable

System.out.println(count);

}

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

//creating objects

Counter2 c1=**new** Counter2();

Counter2 c2=**new** Counter2();

Counter2 c3=**new** Counter2();

}

}

**PROGRAM:**

import java.util.Scanner;

class SavingsAccount{

private static double a\_i;

private double s\_b;

public SavingsAccount() {

s\_b=0; //s\_b=savings balance

a\_i=0; //annual interest rate

}

public SavingsAccount(double balance) {

s\_b=balance;

a\_i=0;

}

public void calculateMonthlyInterestRate() {

System.out.println("current balance:"+s\_b);

do­uble mi;

mi=(s\_b\*a\_i)/12;

s\_b=s\_b+mi;

System.out.println("new balance:"+s\_b);

}

double getbalance() {

return s\_b;

}

static void modifyInterest(double new\_interest) {

a\_i=new\_interest;

}

}­­

class Lab3{

public static void main(String args[]) {

SavingsAccount s1=new SavingsAccount(2000);

SavingsAccount s2=new SavingsAccount(3000);

s1.modifyInterest(0.04);

s1.calculateMonthlyInterestRate();

s2.modifyInterest(0.04);

s2.calculateMonthlyInterestRate();

System.out.println("after changing interest rate from 4% to 5%");

s1.modifyInterest(0.05);

s1.calculateMonthlyInterestRate();

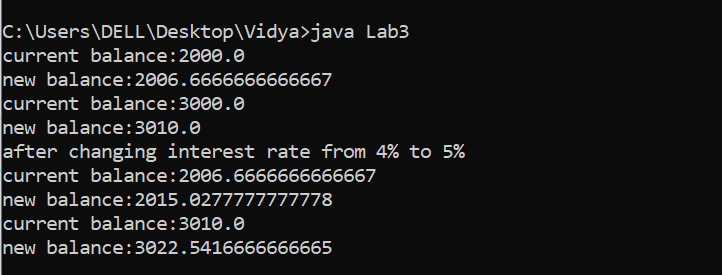
s2.modifyInterest(0.05);

s2.calculateMonthlyInterestRate();

}

}

**OUTPUT:**

****

**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:**

## **Java Simple for Loop**

A simple for loop is the same as [C](https://www.javatpoint.com/c-programming-language-tutorial)/[C++](https://www.javatpoint.com/cpp-tutorial). We can initialize the [variable](https://www.javatpoint.com/java-variables), check condition and increment/decrement value. It consists of four parts:

1. **Initialization**: It is the initial condition which is executed once when the loop starts. Here, we can initialize the variable, or we can use an already initialized variable. It is an optional condition.
2. **Condition**: It is the second condition which is executed each time to test the condition of the loop. It continues execution until the condition is false. It must return boolean value either true or false. It is an optional condition.
3. **Increment/Decrement**: It increments or decrements the variable value. It is an optional condition.
4. **Statement**: The statement of the loop is executed each time until the second condition is false.

**Syntax:**

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

//statement or code to be executed

}

**PROGRAM:**

class Book{

String bookname, authorname, Publisher;

long ISBNnumber;

Book(){

String bookname= authorname= publisher=" ";

long ISBNnumber=0;

}

Book(String a,int b,String c,String d){

bookname=a;

ISBNnumber=b;

authorname=c;

publisher=d;

}

public String getBookname() {

return bookname;

}

public void setBookname(String bookname) {

this.bookname = bookname;

}

public long getISBNnumber() {

return ISBNnumber;

}

public void setISBNnumber(long iSBNnumber) {

ISBNnumber = iSBNnumber;

}

public String getAuthorname() {

return authorname;

}

public void setAuthorname(String authorname) {

this.authorname = authorname;

}

public String getPublisher() {

return publisher;

}

public void setPublisher(String publisher) {

this.publisher = publisher;

}

void getBookInfo() {

System.out.println(bookname+" "+ISBNnumber+" "+authorname+" "+publisher);

}}

public class Lab4 {

public static void main(String[] args) {

Book a[]=new Book[30];

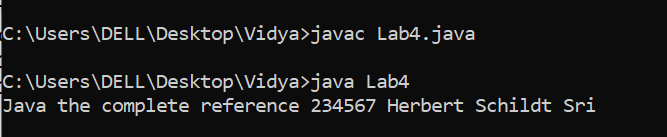
a[1]=new Book("Java the complete reference",234567,"Herbert Schildt","Sri");

a[1].getBookInfo();

}

}

**OUTPUT:**

****

**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 Algorithm

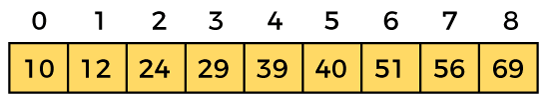
In this article, we will discuss the Binary Search Algorithm. Searching is the process of finding some particular element in the list. If the element is present in the list, then the process is called successful, and the process returns the location of that element. Otherwise, the search is called unsuccessful .Binary search is the search technique that works efficiently on sorted lists. Hence, to search an element into some list using the binary search technique, we must ensure that the list is sorted. Binary search follows the divide and conquer approach in which the list is divided into two halves, and the item is compared with the middle element of the list. If the match is found then, the location of the middle element is returned. Otherwise, we search into either of the halves depending upon the result produced through the match.

## **Working of Binary search**

Now, let's see the working of the Binary Search Algorithm. To understand the working of the Binary search algorithm, let's take a sorted array. It will be easy to understand the working of Binary search with an example. There are two methods to implement the binary search algorithm -

* Iterative method
* Recursive method

The recursive method of binary search follows the divide and conquer approach.Let the elements of array are -



Let the element to search is, **K = 56**

We have to use the below formula to calculate the **mid** of the array -

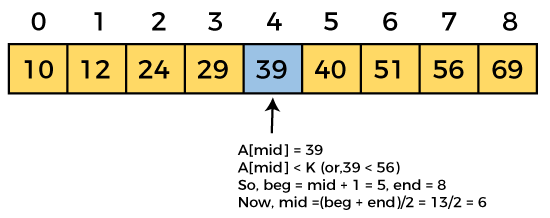
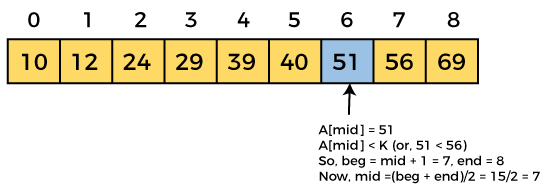
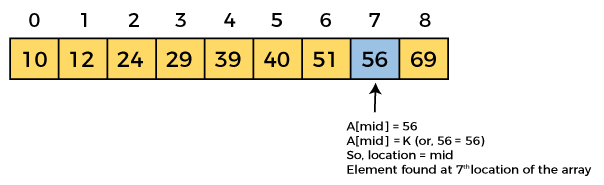
1. mid = (beg + end)/2

So, in the given array -

**beg** = 0

**end** = 8

**mid** = (0 + 8)/2 = 4. So, 4 is the mid of the array.

**PROGRAM:**

import java.util.Scanner;

class Binary{

int b[]=new int[10];

int n,key;

Binary(int b[],int n,int key){

this.b=b;

this.n=n;

this.key=key;

}

void binary() {

int lb=0,ub=n-1,mid;

while(lb<=ub){

mid=(lb+ub)/2;

if(key==b[mid]) {

System.out.println("key is found at:"+mid);

System.exit(0);

}

else {

if(b[mid]<key)

lb=mid+1;

else

ub=mid-1;

}}

System.out.println("our searching element is not found!!");}}

public class Lab5 {

public static void main(String[] args) {

Scanner sc=new Scanner(System.in);

int b[]=new int[10];

int i,j,key,n;

System.out.println("enter total number of elements:");

n=sc.nextInt();

System.out.println("enter the elements:");

for(i=0;i<n;i++)

b[i]=sc.nextInt();

System.out.println("enter your searching element:");

key=sc.nextInt();

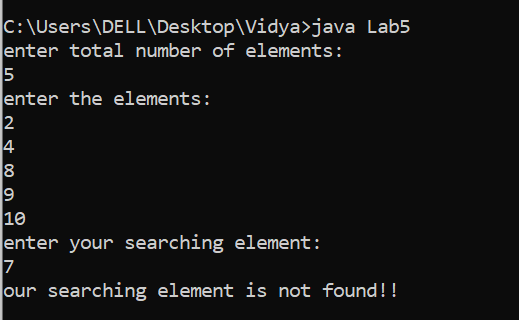
Binary c=new Binary(b,n,key);

c.binary();

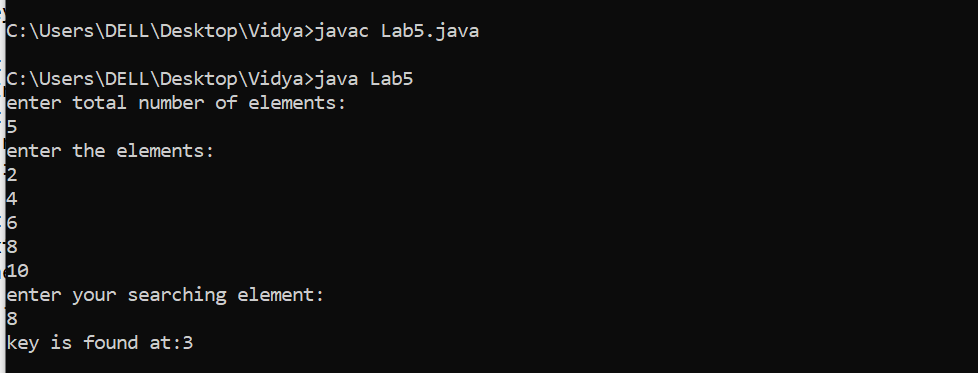
}

}

**Output1**

****

**OUTPUT 2**

****

**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:**

# Data Structures - Merge Sort Algorithm

Merge sort is a sorting technique based on divide and conquer technique. With worst-case time complexity being Ο(n log n), it is one of the most respected algorithms.

Merge sort first divides the array into equal halves and then combines them in a sorted manner.

## **How Merge Sort Works?**

To understand merge sort, we take an unsorted array as the following −

Unsorted Array

We know that merge sort first divides the whole array iteratively into equal halves unless the atomic values are achieved. We see here that an array of 8 items is divided into two arrays of size 4.

Merge Sort Division

This does not change the sequence of appearance of items in the original. Now we divide these two arrays into halves.

Merge Sort Division

We further divide these arrays and we achieve atomic value which can no more be divided.

Merge Sort Division

Now, we combine them in exactly the same manner as they were broken down. Please note the color codes given to these lists.

We first compare the element for each list and then combine them into another list in a sorted manner. We see that 14 and 33 are in sorted positions. We compare 27 and 10 and in the target list of 2 values we put 10 first, followed by 27. We change the order of 19 and 35 whereas 42 and 44 are placed sequentially.

Merge Sort Combine

In the next iteration of the combining phase, we compare lists of two data values, and merge them into a list of found data values placing all in a sorted order.

Merge Sort Combine

After the final merging, the list should look like this −

Merge Sort

**PROGRAM:**

import java.util.Scanner;

class Merge{

void mergepass(int a[],int lb,int ub){

int mid;

if(lb!=ub){

mid=(lb+ub)/2;

mergepass(a,lb,mid);

mergepass(a,mid+1,ub);

mergesort(a,lb,mid,ub);

}

}

void mergesort(int a[],int lb,int mid,int ub){

int i=lb,j=mid+1,k=lb;

int[] temp=new int[100];

while((i<=mid)&&(j<=ub)){

if(a[i]<a[j])

temp[k++]=a[i++];

else

temp[k++]=a[j++];

}

while(i<=mid)

temp[k++]=a[i++];

while(j<=ub)

temp[k++]=a[j++];

for(i=lb;i<=ub;i++)

a[i]=temp[i];

}

}

public class Lab6{

public static void main(String[] args) {

int i,n;

int[] a=new int[100];

Merge m=new Merge();

System.out.println("enter total number of elements:");

Scanner sc=new Scanner(System.in);

n=sc.nextInt();

System.out.println("enter the elements:");

for(i=0;i<n;i++)

a[i]=sc.nextInt();

System.out.println("before sorting:");

for(i=0;i<n;i++)

System.out.println(a[i]);

m.mergepass(a,0,n-1);

System.out.println("after sorting:");

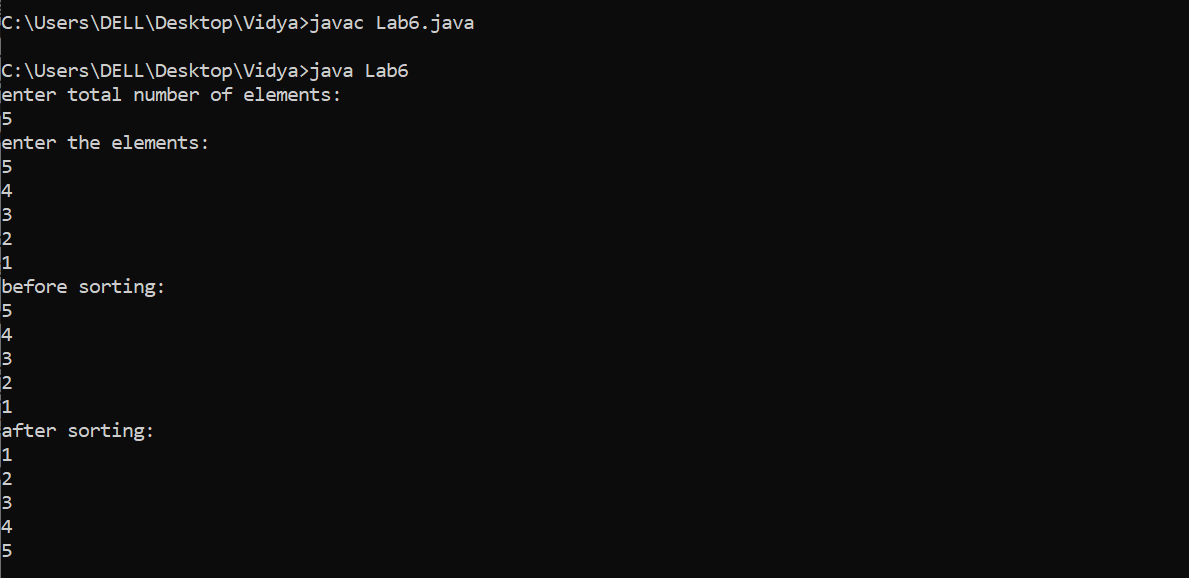
for(i=0;i<n;i++)

System.out.println(a[i]);

}

}

**OUTPUT:**

****

**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:**

The **java.util.Random** class instance is used to generate a stream of pseudorandom numbers.Following are the important points about Random −

* The class uses a 48-bit seed, which is modified using a linear congruential formula.
* The algorithms implemented by class Random use a protected utility method that on each invocation can supply up to 32 pseudorandomly generated bits.

**Syntax:**

## **Class declaration**

Following is the declaration for **java.util.Random** class −

public class Random

extends Object

implements Serializable

## **Class constructors**

|  |  |
| --- | --- |
| **Sr.No.** | **Constructor & Description** |
| 1 | **Random()**  This creates a new random number generator. |
| 2 | **Random(long seed)**  This creates a new random number generator using a single long seed. |

**PROGRAM:**

import java.util.Random;

public class Lab7{

public static void main( String[ ] args){

Random randomNumbers = new Random( );

int frequency1 = frequency2 = frequency3 = 0;

int frequency4 = frequency5 = frequency6 =0;

int dice1,dice2;

for ( int roll = 1; roll <= 10000; roll++ ){

dice1 = 1 + randomNumbers.nextInt( 6 );

dice2 = 1 + randomNumbers.nextInt( 6 );

if(dice1==dice2){

switch(dice1){

case 1: ++frequency1;

break;

case 2: ++frequency2;

break;

case 3: ++frequency3;

break;

case 4: ++frequency4;

break;

case 5: ++frequency5;

break;

case 6: ++frequency6;

break;

}

}

}

System.out.println( "Doubles\tFrequency" );

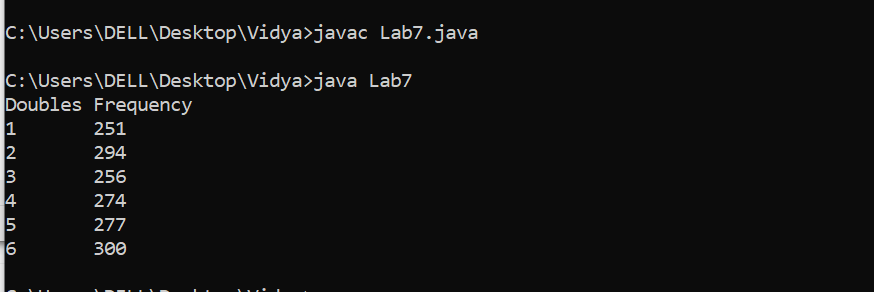
System.out.printf( "1\t%d\n2\t%d\n3\t%d\n4\t%d\n5\t%d\n6\t%d\n",

frequency1, frequency2, frequency3, frequency4,frequency5, frequency6 );

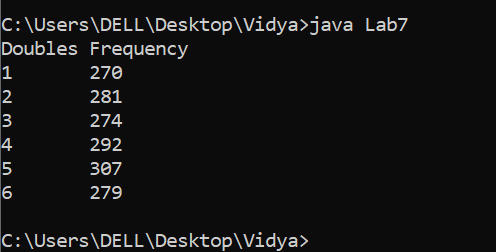
}

}

**Output 1**

****

**Output 2**

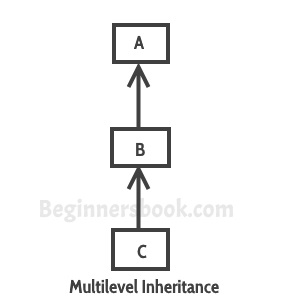
****

**EXPEREMENT 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% ofBP for staff club fund. Generate pay slips for the employees with their gross and net salary.

**DESCRIPTION:**

# Multilevel inheritance in java

When a class extends a class, which extends anther class then this is called **multilevel inheritance**. For example class C extends class B and class B extends class A then this [type of inheritance](https://beginnersbook.com/2013/05/java-inheritance-types/) is known as multilevel inheritance.  
Lets see this in a diagram:  


It’s pretty clear with the diagram that in Multilevel inheritance there is a concept of grand parent class. If we take the example of this diagram, then class C inherits class B and class B inherits class A which means B is a parent class of C and A is a parent class of B. So in this case class C is implicitly inheriting the properties and methods of class A along with class B that’s what is called multilevel inheritance.

**Syntax:**

### Syntax of Multilevel Inheritance in Java

Let us see the syntax of multilevel inheritance in java which is given below:

class A{  
//class A is parent of class B  
//class A is grand parent of class C  
public A(){  
//A constructor  
}  
public void fun1(){  
//function in Parent Class  
}  
}  
class B extends A{  
//class B is a child class of class A  
//class B is a parent class of class C  
public B(){  
//class B constructor  
}  
}  
class C extends B{  
//class C is a child class of class B  
//class C is grand child class of class A  
public C(){  
//Class C constructor  
}  
}  
public class Test{  
public static void main(String[] args){  
C obj = new C();  
}  
}

**PROGRAM:**

import java.util.Scanner;

class Employee{

double bp,da,hra,pf,scf, gs, ns;

Scanner sc=new Scanner(System.in);

Scanner scs=new Scanner(System.in);

String emp\_name,emp\_id,address,mail\_id;

long mobileno;

void getdata(){

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

emp\_name=scs.next();

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

emp\_id=scs.next();

System.out.println("enter employee address:");

address=scs.next();

System.out.println("enter employeee mailid:");

mail\_id=scs.next();

System.out.println("enter employee mobileno:");

mobileno=sc.nextLong();

}

void cal() {

System.out.println("enter basicpay of employee:");

bp=sc.nextDouble();

da=(97/100)\*bp;

hra=(10/100)\*bp;

pf=(12/100)\*bp;

scf=(0.1/100)\*bp;

gs=bp+da+hra+pf;

ns=gs-pf-scf;

}

void payslip1() {

System.out.println("Programmer name:"+emp\_name);

System.out.println("Programmer id:"+emp\_id);

System.out.println("Programmer address:"+address);

System.out.println("Programmer mail id:"+mail\_id);

System.out.println("Programmer mobileno:"+mobileno);

System.out.println("Programmer da:"+da);

System.out.println("Programmer hra:"+hra);

System.out.println("Programmer pf:"+pf);

System.out.println("Programmer scf:"+scf);

System.out.println("Programmer grossalary:"+gs);

System.out.println("Programmer netsalary:"+ns);

}

void payslip2() {

System.out.println("Assistant Professor name:"+emp\_name);

System.out.println("Assistant Professor id:"+emp\_id);

System.out.println("Assistant Professor address:"+address);

System.out.println("Assistant Professor mail id:"+mail\_id);

System.out.println("Assistant Professor mobileno:"+mobileno);

System.out.println("Assistant Professor da:"+da);

System.out.println("Assistant Professor hra:"+hra);

System.out.println("Assistant Professor pf:"+pf);

System.out.println("Assistant Professor scf:"+scf);

System.out.println("Assistant Professor grossalary:"+gs);

System.out.println("Assistant Professor netsalary:"+ns);

}

void payslip3() {

System.out.println("Associate professor name:"+emp\_name);

System.out.println("Associate professor id:"+emp\_id);

System.out.println("Associate professor address:"+address);

System.out.println("Associate professor mail id:"+mail\_id);

System.out.println("Associate professor mobileno:"+mobileno);

System.out.println("Associate professor da:"+da);

System.out.println("Associate professor hra:"+hra);

System.out.println("Associate professor pf:"+pf);

System.out.println("Associate professor scf:"+scf);

System.out.println("Associate professor grossalary:"+gs);

System.out.println("Associate professor netsalary:"+ns);

}

void payslip4(){

System.out.println("Professor name:"+emp\_name);

System.out.println("Professor id:"+emp\_id);

System.out.println("Professor address:"+address);

System.out.println("Professor mail id:"+mail\_id);

System.out.println("Professor mobileno:"+mobileno);

System.out.println("Professor da:"+da);

System.out.println("Professor hra:"+hra);

System.out.println("Professor pf:"+pf);

System.out.println("Professor scf:"+scf);

System.out.println("Professor grossalary:"+gs);

System.out.println("Professor netsalary:"+ns);

}

}

class Programmer extends Employee{

Programmer(){

getdata();

cal();

payslip1();

}

}

class AssistantProfessor extends Employee{

AssistantProfessor(){

getdata();

cal();

payslip2();

}

}

class AssociateProfessor extends Employee{

AssociateProfessor(){

getdata();

cal();

payslip3();

}

}

class Professor extends Employee{

Professor(){

getdata();

cal();

payslip4();

}

}

public class Lab8

{

public static void main(String[] args) {

Scanner sc=new Scanner(System.in);

Employee in=new Employee();

int choice;

do {

System.out.println("###menu for type of employee###");

System.out.println("1.Programmer\n2.Assistant proffesor\n3.Associate professor\n4.Professor");

System.out.println("enter your choice:");

choice=sc.nextInt();

switch(choice) {

case 1:

System.out.println("Programmer details");

Programmer p=new Programmer();

break;

case 2:

System.out.println("Assistant proffesor details");

AssistantProfessor a=new AssistantProfessor();

break;

case 3:

System.out.println("Assosiate professor details");

AssociateProfessor b=new AssociateProfessor();

break;

case 4:

System.out.println("Professor details");

Professor c=new Professor();

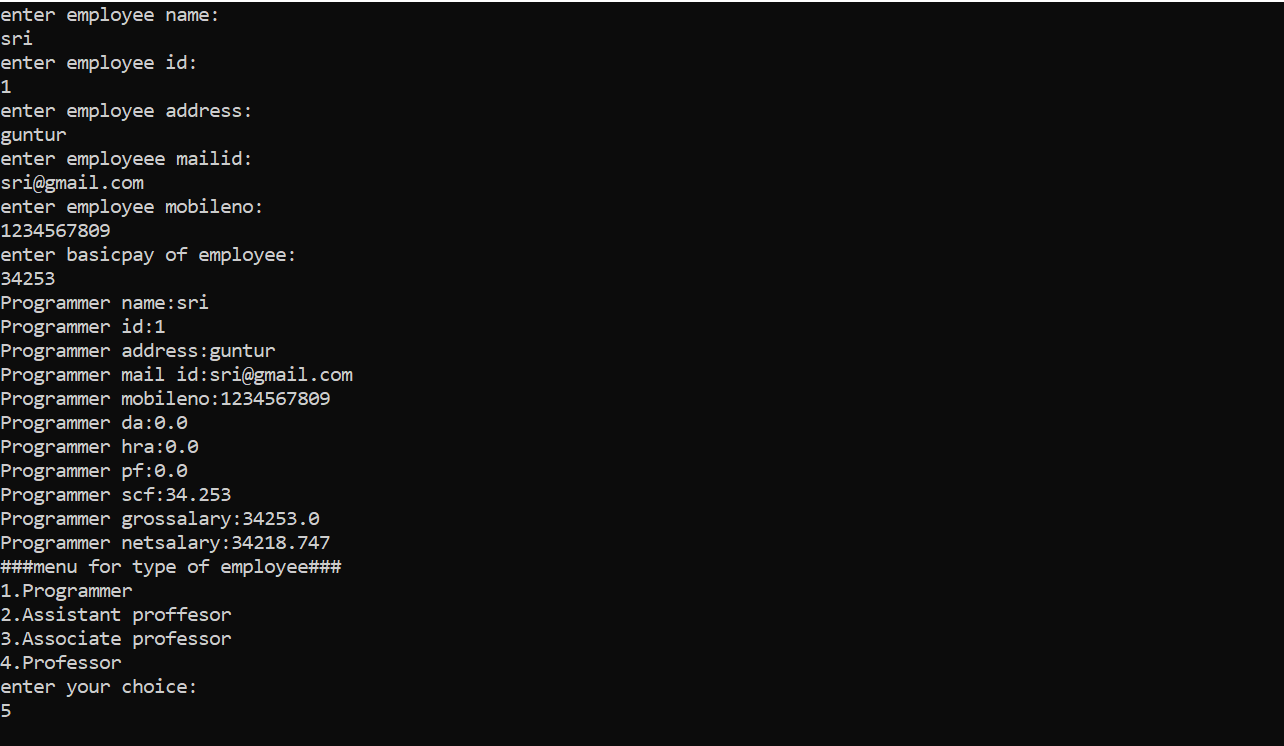
}

}while(choice!=5);

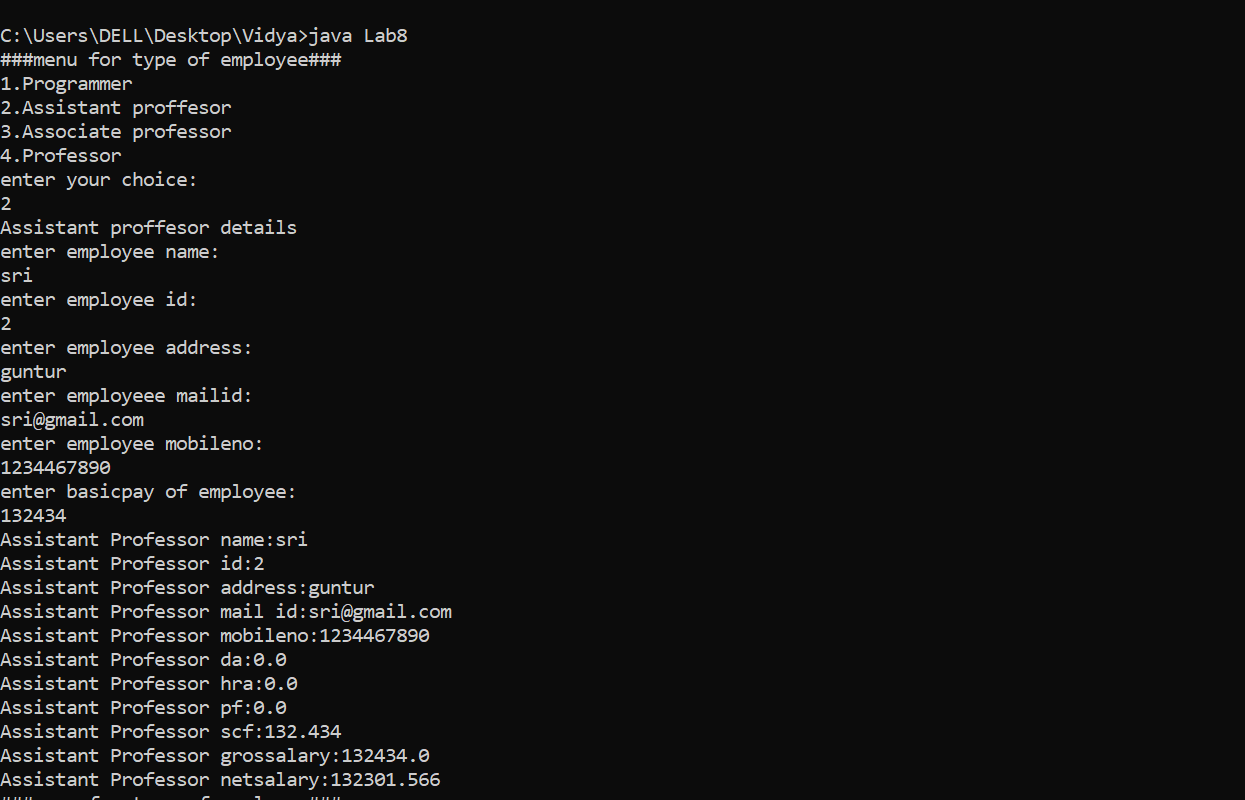
}

}  
OUTPUT:

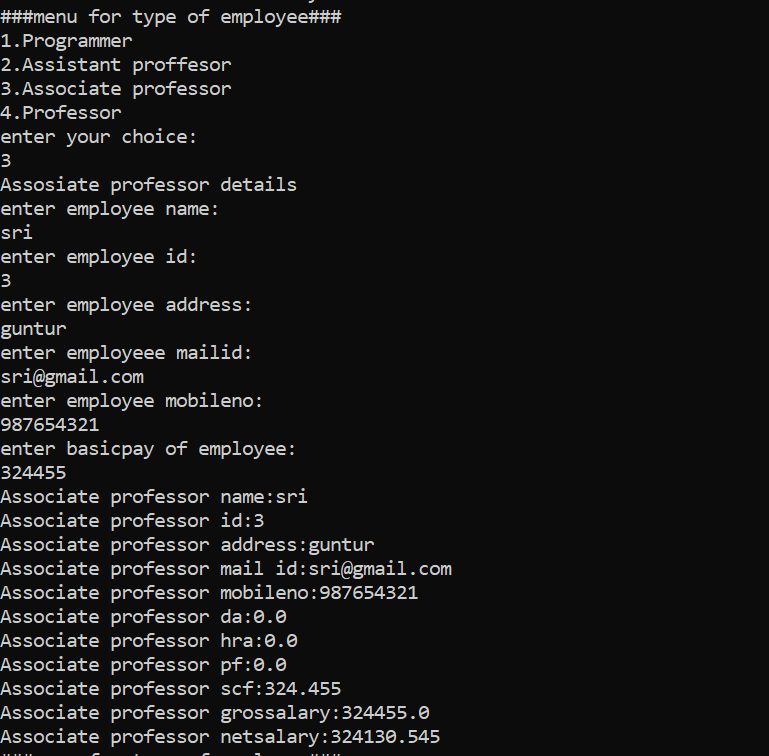
Output1

****

**Output2**

****

**Output3**

****

**EXPERIMENT 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 given shape.

**DESCRIPTION:**

# Abstract class in Java

A class which is declared with the abstract keyword is known as an abstract class in [Java](https://www.javatpoint.com/java-tutorial). It can have abstract and non-abstract methods (method with the body).

Before learning the Java abstract class, let's understand the abstraction in Java first.

### **Abstraction in Java**

**Abstraction** is a process of hiding the implementation details and showing only functionality to the user.

Another way, it shows only essential things to the user and hides the internal details, for example, sending SMS where you type the text and send the message. You don't know the internal processing about the message delivery.

Abstraction lets you focus on what the [object](https://www.javatpoint.com/object-and-class-in-java) does instead of how it does it.

### **Ways to achieve Abstraction**

There are two ways to achieve abstraction in java

1. Abstract class (0 to 100%)
2. Interface (100%)

### **Abstract class in Java**

A class which is declared as abstract is known as an **abstract class**. It can have abstract and non-abstract methods. It needs to be extended and its method implemented. It cannot be instantiated.

#### **Points to Remember**

* 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](https://www.javatpoint.com/java-constructor) and static methods also.
* It can have final methods which will force the subclass not to change the body of the method.

### **Abstract Method in Java**

A method which is declared as abstract and does not have implementation is known as an abstract method.

**Syntax:**

**abstract** **class** A{

**abstract** **void** printStatus();//no method body and abstract

}

**PROGRAM:**

abstract class Shape{

int dim1, dim2;

Shape (int dim1, int dim2) {

this.dim1 = dim1;

this.dim2 = dim2;

}

Shape (int dim1){

this.dim1 = dim1;

}

abstract void Area ();

}

class Rectangle extends Shape{

Rectangle (int x, int y){

super (x, y);

}

void Area () {

System.out.println ("Area of the rectangle is:" + dim1 \* dim2);

}

}

class Triangle extends Shape{

Triangle (int x, int y){

super (x, y);

}

void Area () {

System.out.println ("Area of the triangle is:" + 0.5 \* dim1 \* dim2);

}

}

class Circle extends Shape{

Circle (int x){

super (x);

}

void Area (){

System.out.println ("Area of the circle is:" + 3.14 \* dim1 \* dim1);

}}

class Lab9{

public static void main (String args[]){

Rectangle r1 = new Rectangle (4, 5);

r1.Area ();

Triangle t1 = new Triangle (3, 4);

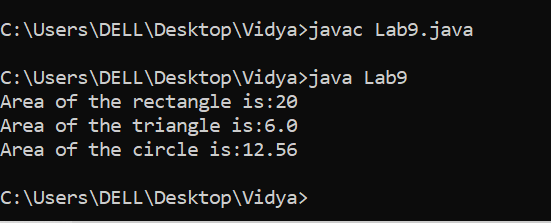
t1.Area ();

Circle c1 = new Circle (2);

c1.Area ();

}

}  
**OUTPUT:**

****

**EXPERIMENT 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:**

# Java Package

A **java package** is a group of similar types of classes, interfaces and sub-packages.

Package in java can be categorized in two form, built-in package and user-defined package.

There are many built-in packages such as java, lang, awt, javax, swing, net, io, util, sql etc.

## **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.



## How to Create a package?

Creating a package is a simple task as follows

* Choose the name of the package
* Include the package command as the first line of code in your Java Source File.
* The Source file contains the classes, interfaces, etc you want to include in the package
* Compile to create the Java packages

**Syntax:**

package nameOfPackage;

PROGRAM:

Package 1:

package currencyconverter;

import java.util.\*;

public class Currencyconversion{

double inr,dollar;

double euro,yen;

Scanner sc=new Scanner(System.in);

public void dollartorupee(){

System.out.println("Enter dollars:");

dollar=sc.nextInt();

inr=dollar\*67;

System.out.println("Dollar to rupees is:"+inr);

}

public void rupeetodollar(){

System.out.println("Enter Rupee:");

inr=sc.nextInt();

dollar=inr/67;

System.out.println("Rupee to dollars is:"+dollar);

}

public void eurotorupee(){

System.out.println("Enter euro:");

euro=sc.nextInt();

inr=euro\*79.50;

System.out.println("Euro to rupee is:"+inr);

}

public void rupeetoeuro(){

System.out.println("Enter Rupee:");

inr=sc.nextInt();

euro=(inr/79.50);

System.out.println("Rupee to euro is:"+euro);

}

public void yentorupee(){

System.out.println("Enter yen:");

yen=sc.nextInt();

inr=yen\*0.61;

System.out.println("YEN to rupee is:"+inr);

}

public void rupeetoyen(){

System.out.println("Enter Rupees:");

inr=sc.nextInt();

yen=(inr/0.61);

System.out.println("rupees to yen is:"+yen);

}}

Package 2:

package distanceconverter;

import java.util.\*;

public class Distanceconversion

{

double km,m,miles;

Scanner sc = new Scanner(System.in);

public void kmtom(){

System.out.print("Enter km:");

km=sc.nextDouble();

m=(km\*1000);

System.out.println("kilometer to meter is:"+m);

}

public void mtokm(){

System.out.print("Ente meter:");

m=sc.nextDouble();

km=(m/1000);

System.out.println("meter to klometer is:"+km);

}

public void milestokm(){

System.out.print("Enter miles:");

miles=sc.nextDouble();

km=(miles\*1.60934);

System.out.println("miles to kilometer is:"+km);

}

public void kmtomiles(){

System.out.print("Enter km:");

km=sc.nextDouble();

miles=(km\*0.621371);

System.out.println("kilometer to miles is:"+miles);

}}

Package 3:

package timeconverter;

import java.util.\*;

public class Timeconversion{

int hours,seconds,minutes;

Scanner sc = new Scanner(System.in);

public void secondstohours(){

System.out.print("Enter the number of seconds:");

seconds= sc.nextInt();

hours =seconds/3600;

System.out.println("Seconds to hours is:"+hours);

}

public void minutestohours(){

System.out.print("Enter the number of minutes:");

minutes=sc.nextInt();

hours=minutes/60;

System.out.println("Minutes to seconds is:"+hours);

}

public void hourstominutes(){

System.out.println("enter the no of hours");

hours=sc.nextInt();

minutes=(hours\*60);

System.out.println("hours to minutes is:"+minutes);

}

public void hourstoseconds(){

System.out.println("enter the no of hours");

hours=sc.nextInt();

seconds=(hours\*3600);

System.out.println("hours to seconds is:"+seconds);

}}

Package 4:

package mypack;

import java.util.\*;

import currencyconverter.\*;

import distanceconverter.\*;import timeconverter.\*;

class Lab10{

public static void main(String args[]){

Scanner sc=new Scanner(System.in);

int choice;

Currencyconversion c=new Currencyconversion();

Distanceconversion d=new Distanceconversion();

Timeconversion t=new Timeconversion();

System.out.println("###menu for operations###");

System.out.println("\n1.dollar to rupee \n2.rupee to dollar\n3.Euro to rupee\n4.rupee to Euro\n5.Yen to rupee\n6.Rupee to Yen\n7.Meter to kilometer\n8.kilometer to meter\n9.Miles to kilometer\n10.kilometer to miles\n11.Hours to Minutes\n12.Hours to Seconds\n13.Seconds to Hours\n14.Minutes to Hours");

do{ System.out.println("Enter ur choice");

choice=sc.nextInt();

switch(choice){

case 1: c.dollartorupee();

break;

case 2: c.rupeetodollar();

break;

case 3:c.eurotorupee();

break;

case 4: c.rupeetoeuro();

break;

case 5: c.yentorupee();

break;

case 6 : c.rupeetoyen();

break;

case 7 :d.mtokm();

break;

case 8 :d.kmtom();

break;

case 9 :d.milestokm();

break;

case 10 :d.kmtomiles();

break;

case 11 :t.hourstominutes();

break;

case 12 :t.hourstoseconds();

break;

case 13 :t.secondstohours();

break;

case 14 : t.minutestohours();

break;

}

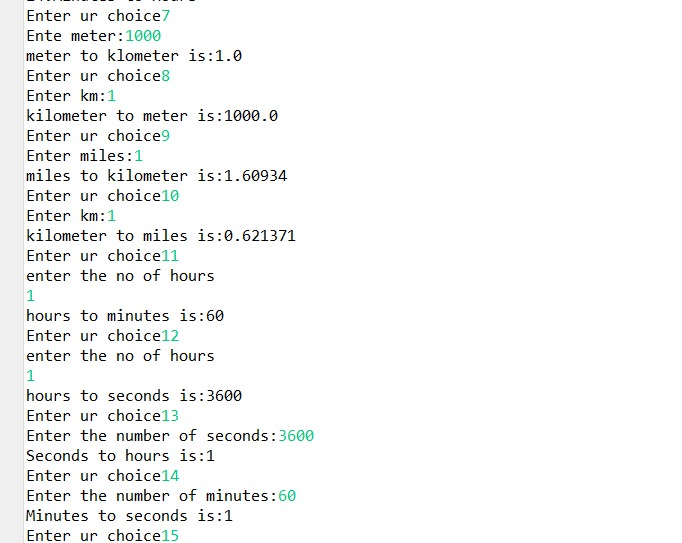
}while(choice!=15);

}

}

OUTPUT:

****

****

**EXPERIMENT 11**

**AIM:** Write a Java Program to Handle Arithmetic Exceptions and InputMisMatchExceptions.

**DESCRIPTION:**

# Exception Handling in Java

The **Exception Handling in Java** is one of the powerful mechanism to handle the runtime errors so that the normal flow of the application can be maintained.

In this tutorial, we will learn about Java exceptions, it's types, and the difference between checked and unchecked exceptions.

## **What is Exception in Java?**

**Dictionary Meaning:** Exception is an abnormal condition.

In Java, an exception is an event that disrupts the normal flow of the program. It is an object which is thrown at runtime.

### **Advantage of Exception Handling**

The core advantage of exception handling is **to maintain the normal flow of the application**. An exception normally disrupts the normal flow of the application; that is why we need to handle exceptions.

Let us handle the **ArithmeticException**using **try and catch** blocks.

* Surround the statements that can throw **ArithmeticException**with **try and catch** blocks.
* We can **Catch**the **ArithmeticException**
* Take necessary action for our program, as the execution **doesn’t abort**.

## InputMismatchException is an Unchecked Exception

The **java.util** package provides **InputMismatchException** class that inherits NoSuchElementException class. NoSuchElementException inherits RuntimeException. So, InputMismatchException is a runtime exception. Hence, it is an **[unchecked exception](https://javahungry.blogspot.com/2019/08/difference-checked-and-unchecked-exception.html" \t "_blank)**.

**PROGRAM:**

import java.util.\*;

public class Lab11{

public static void main(String[] args) {

Scanner sc=new Scanner(System.in);

try {

System.out.println("enter integer value:");

int i=sc.nextInt();

System.out.println(i);

int a;

a=5/(i-5);

System.out.println(a+"value is");

}

catch(InputMismatchException e){

System.out.println(e);

}

catch(ArithmeticException e){

System.out.println(e);

}

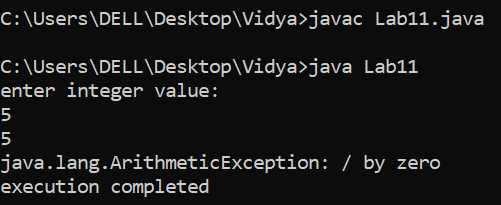
finally {

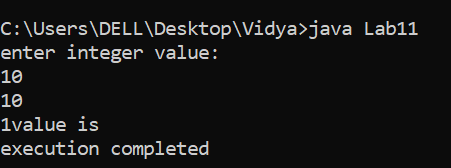
System.out.println("execution completed");

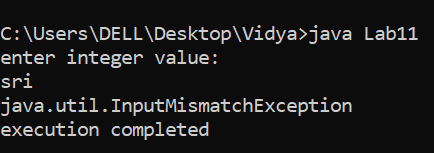
}

}}

**OUTPUT:**

****

****

****

**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:

# Piped Readers and Writers

Piped readers and writers do for character streams what piped input and output streams do for byte streams: they allow two threads to communicate. Character output from one thread becomes character input for the other thread:

public class PipedWriter extends Writer

public class PipedReader extends Reader

The PipedWriter class has two constructors. The first constructs an unconnected PipedWriter object. The second constructs one that’s connected to the PipedReader object sink:

public PipedWriter()

public PipedWriter(PipedReader sink) throws IOException

**PROGRAM**

import java.io.\*;

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 Lab12{

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

fibonacci fi=new fibonacci();

prime pri=new prime();

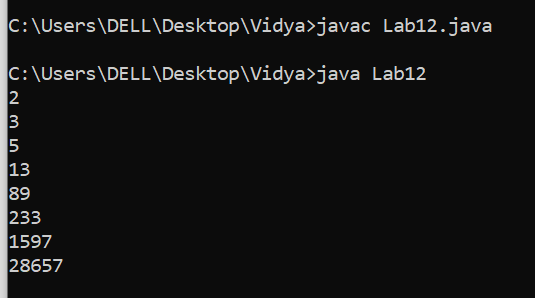
receiver r=new receiver(fi,pri);

fi.start();

pri.start();

r.start();

}}  
**OUTPUT:**

****

**EXPERIMENT 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 is a Java feature that allows concurrent execution of two or more parts of a program for maximum utilization of CPU. Each part of such program is called a thread. So, threads are light-weight processes within a process.

Threads can be created by using two mechanisms :

1. Extending the Thread class
2. Implementing the Runnable Interface

**Thread creation by extending the Thread class**  
We create a class that extends the **java.lang.Thread** class. This class overrides the run() method available in the Thread class. A thread begins its life inside run() method. We create an object of our new class and call start() method to start the execution of a thread. Start() invokes the run() method on the Thread object.

**Thread creation by implementing the Runnable Interface**  
We create a new class which implements java.lang.Runnable interface and override run() method. Then we instantiate a Thread object and call start() method on this object.

**SYNTAX:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.N.** | **Modifier and Type** | **Method** | **Description** |
| 1) | Void | [start()](https://www.javatpoint.com/java-thread-start-method) | It is used to start the execution of the thread. |
| 2) | Void | [run()](https://www.javatpoint.com/java-thread-run-method) | It is used to do an action for a thread. |

**PROGRAM:**

import java.util.Random;

class Thread1 extends Thread{

public void run() {

Random r=new Random();

int n=r.nextInt(10);

System.out.println("number generated+"+n);

if(n%2==0){

Thread2 t2=new Thread2(n);

t2.start();

}

else if(n%2!=0) {

Thread3 t3=new Thread3(n);

t3.start();

}

}

}

class Thread2 extends Thread{

int x;

Thread2(int n) {

x=n;

}

public void run(){

System.out.println("square of random int generated is:"+(x\*x));

}

}

class Thread3 extends Thread {

int x;

Thread3(int n) {

x=n;

}

public void run() {

System.out.println("cube of random int generated is:"+(x\*x\*x));

}

}

public class Main{

public static void main(String[] args) {

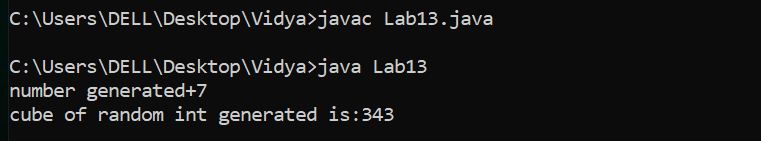
Thread1 t1=new Thread1();

t1.start();

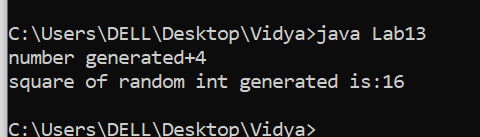
}

}  
OUTPUT:

Output1

****

**Output2**

****

**EXPEREMENT 14**

**AIM**: Write a java program that correctly implements the producer-consumer problem using the concept of inter-thread communication.

**DESCRIPTION:**

# Inter-thread Communication in Java

**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() method**

### 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.

### **2) notify() method**

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.

### **3) notifyAll() method**

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

**SYNTAX:**

|  |  |  |
| --- | --- | --- |
| **MODIFIER AND TYPE** | | **METHOD** |
| 1. | static void | [sleep()](https://www.javatpoint.com/java-thread-sleep-method) |
| 2. | public final void | notifyAll() |
| 3. | public final void | notify() |
| 4. | public final void | wait() |

**Program:**

class Q{

int n;

boolean val=false;

synchronized int get() {

while(!val)

try {

Thread.sleep(1000);

wait();

}

catch(InterruptedException e) {

System.out.println("Interrupted"+e);

}

System.out.println("got:"+n);

val=false;

notify();

return n;

}

synchronized void put(int n) {

while(val)

try {

Thread.sleep(1000);

wait();

}

catch(InterruptedException e) {

System.out.println("Interrupted"+e);

}

this.n=n;

val=true;

System.out.println("put:"+n);

notify();

}

}

class Producer implements Runnable{

Q q;

Producer(Q q){

this.q=q;

new Thread(this,"Producer").start();

}

public void run() {

int i=0;

while(true) {

q.put(i++);

}

}

}

class Consumer implements Runnable{

Q q;

Consumer(Q q){

this.q=q;

new Thread(this,"Consumer").start();

}

public void run() {

while(true) {

q.get();

}

}

}

public class Main {

public static void main(String args[]) {

Q q=new Q();

new Producer(q);

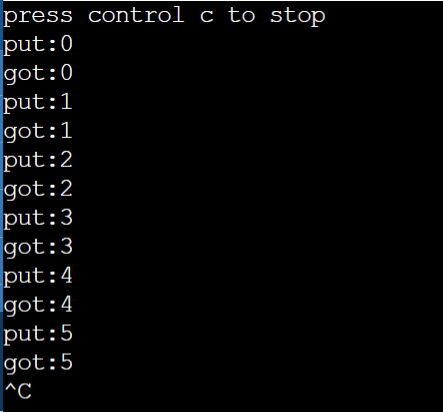
new Consumer(q);

System.out.println("press control c to stop");

}

}

**Output:**

****

**EXPERIMENT 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 inbytes.

**DESCRIPTION:**

The java.io package contains nearly every class you might ever need to perform input and output (I/O) in Java. All these streams represent an input source and an output destination. The stream in the java.io package supports many data such as primitives, object, localized characters, etc.

## **Stream**

A stream can be defined as a sequence of data. There are two kinds of Streams −

* **InPutStream** − The InputStream is used to read data from a source.
* **OutPutStream** − The OutputStream is used for writing data to a destination.

**SYNTAX:**

|  |  |  |
| --- | --- | --- |
| **Method** | **Type** | **Description** |
| canRead() | Boolean | Tests whether the file is readable or not |
| canWrite() | Boolean | Tests whether the file is writable or not |
| createNewFile() | Boolean | Creates an empty file |
| delete() | Boolean | Deletes a file |
| exists() | Boolean | Tests whether the file exists |
| getName() | String | Returns the name of the file |

PROGRAM:

import java.io.File;

import java.util.Scanner;

class Main{

public static void main(String args[]){

String s;

Scanner sc =new Scanner(System.in);

System.out.println("enter file name:");

s=sc.next();

File f1=new File(s);

System.out.println(f1.exists()?"exists":"does not exist");

System.out.println(f1.canWrite()?"is writable":"is not writable");

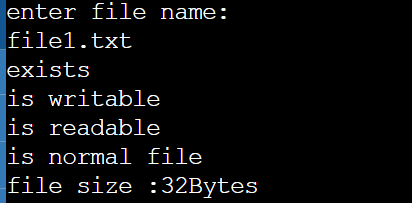
System.out.println(f1.canRead()?"is readable":"is not readable");

System.out.println(f1.isFile()?"is normal file":"might be a named pipe");

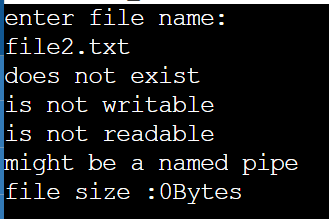
System.out.println("file size :"+f1.length()+"Bytes");

}}

**OUTPUT:**

****

**Output 2:**

****

**EXPEREMENT NO: 16**

**AIM:** Write a Java program to build a Calculator in Swings.

**DESCRIPTION:**

Java Swing is a GUI (graphical user Interface) widget toolkit for Java. Java Swing is a part of Oracle’s Java foundation classes . Java Swing is an API for providing graphical user interface elements to Java Programs.Swing was created to provide more powerful and flexible components than Java AWT (Abstract Window Toolkit).  
In this article we will use Java Swing components to create a simple calculator with only +, -, /, \* operations.

**SYNTAX:**

**methods used :** 

1. **add(Component c)** : adds component to container.
2. **addActionListenerListener(ActionListener d)**: add actionListener for specified component
3. **setBackground(Color c)** : sets the background color of the specified container
4. **setSize(int a, int b)**: sets the size of container to specified dimensions.
5. **setText(String s)**: sets the text of the label to s.
6. **getText()** : returns the text of the label.

**PROGRAM:**

import java.awt.\*;

import java.awt.event.\*;

public class LabProgram16 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;

LabProgram16(String frameText){

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);

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;

}

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; }

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;}

}

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 LabProgram16("Calculator - JavaTpoint");

} }

class MyDigitButton extends Button implements ActionListener {

LabProgram16 cl;

MyDigitButton(int x,int y, int width,int height,String cap, LabProgram16 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

{

LabProgram16 cl;

MyOperatorButton(int x,int y, int width,int height,String cap, LabProgram16 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(LabProgram16.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(LabProgram16.getFormattedText(tempd));}

catch(ArithmeticException excp)

{cl.displayLabel.setText("Divide by 0.");}

return;

}

if(!opText.equals("="))

{

cl.number=temp;

cl.op=opText.charAt(0);

return;

}

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;

}//switch

cl.displayLabel.setText(LabProgram16.getFormattedText(temp));

}//actionPerformed

}//class

class MyMemoryButton extends Button implements ActionListener

{

LabProgram16 cl;

MyMemoryButton(int x,int y, int width,int height,String cap, LabProgram16 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(LabProgram16.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;

}//switch

}//actionPerformed

}//class

class MySpecialButton extends Button implements ActionListener {

LabProgram16 cl;

MySpecialButton(int x,int y, int width,int height,String cap, LabProgram16 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();

if(opText.equals("Backspc")) {

String tempText=backSpace(cl.displayLabel.getText());

if(tempText.equals(""))

cl.displayLabel.setText("0");

else

cl.displayLabel.setText(tempText);

return; }

if(opText.equals("C")) {

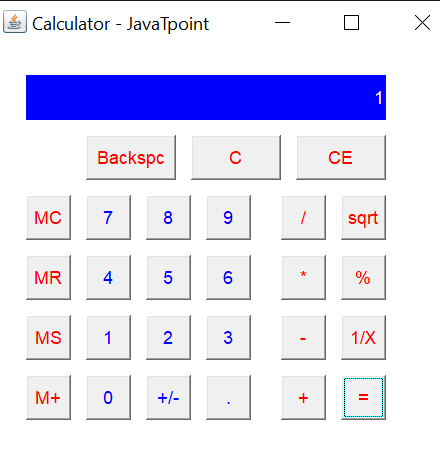
cl.number=0.0; cl.op=' '; cl.memValue=0.0;

cl.memLabel.setText(" "); }

cl.displayLabel.setText("0");cl.setClear=true;

}//actionPerformed

} **OUTPUT:**

****

**EXPEREMENT NO: 17**

**AIM:** . Write a Java program to implement JMenu to draw all basic shapes using Graphics. **DESCRIPTION AND SYNTAX:**

|  |
| --- |
| 1. **public abstract void drawString(String str, int x, int y):** is used to draw the specified string. 2. **public void drawRect(int x, int y, int width, int height):** draws a rectangle with the specified width and height. 3. **public abstract void fillRect(int x, int y, int width, int height):** is used to fill rectangle with the default color and specified width and height. 4. **public abstract void drawOval(int x, int y, int width, int height):** is used to draw oval with the specified width and height. 5. **public abstract void fillOval(int x, int y, int width, int height):** is used to fill oval with the default color and specified width and height. 6. **public abstract void drawLine(int x1, int y1, int x2, int y2):** is used to draw line between the points(x1, y1) and (x2, y2). 7. **public abstract boolean drawImage(Image img, int x, int y, ImageObserver observer):** is used draw the specified image. 8. **public abstract void drawArc(int x, int y, int width, int height, int startAngle, int arcAngle):** is used draw a circular or elliptical arc. 9. **public abstract void fillArc(int x, int y, int width, int height, int startAngle, int arcAngle):** is used to fill a circular or elliptical arc. 10. **public abstract void setColor(Color c):** is used to set the graphics current color to the specified color. 11. **public abstract void setFont(Font font):** is used to set the graphics current font to the specified font. |

**PROGRAM:**

import java.awt.\*;

import java.awt.event.\*;

import javax.swing.\*;

public class Menu extends JFrame {

public static void main(String[] args) {

new Menu(); }

JRadioButtonMenuItem black, red, green, blue, cyan, magenta, yellow, white, custom;

JRadioButtonMenuItem curve, straightLine, rectangle, oval, roundRect, filledRectangle, filledOval, filledRoundRect;

JRadioButtonMenuItem noSymmetry, twoWay, fourWay, eightWay;

public boolean standAlone = true;

public Menu() {

super("Graphics Menu");

Display canvas = new Display();

setContentPane(canvas);

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);

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);

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);

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);

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);

pack();

setLocation(75,50);

setResizable(false);

setDefaultCloseOperation(EXIT\_ON\_CLOSE);

show();

}

private class Display extends JPanel

implements MouseListener, MouseMotionListener, ActionListener {

private final static int

CURVE = 0,

LINE = 1,

RECT = 2,

OVAL = 3,

ROUNDRECT = 4,

FILLED\_RECT = 5,

FILLED\_OVAL = 6,

FILLED\_ROUNDRECT = 7;

private final static int

NO\_SYMMETRY = 0,

SYMMETRY\_2 = 1,

SYMMETRY\_4 = 2,

SYMMETRY\_8 = 3;

Color customColor = Color.gray;

Image OSI;

int widthOfOSI, heightOfOSI;

Image undoBuffer;

private int mouseX, mouseY;

private int prevX, prevY;

private int startX, startY;

private boolean dragging;

private int figure;

private int symmetry;

private Graphics dragGraphics;

private Color dragColor;

Display() {

addMouseListener(this);

addMouseMotionListener(this);

setBackground(Color.white);

setPreferredSize( new Dimension(450,450) );

}

private Color getSelectedColor() {

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() {

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 ROUNDRECT;

else if (filledRectangle.isSelected())

return FILLED\_RECT;

else if (filledOval.isSelected())

return FILLED\_OVAL;

else

return FILLED\_ROUNDRECT;

}

private int getSelectedSymmetry() {

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) {

if (shape == LINE) {

g.drawLine(x1,y1,x2,y2);

return;

}

int x, y,w,h;

if (x1 >= x2) {

x = x2;

w = x1 - x2;

}

else {

x = x1;

w = x2 - x1;

}

if (y1 >= y2) {

y = y2;

h = y1 - y2;

}

else {

y = y1;

h = y2 - y1;

}

switch (shape) {

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) {

int width = getWidth();

int height = getHeight();

drawFigure(g,shape,x1,y1,x2,y2);

if (symmetry >= SYMMETRY\_2) {

drawFigure(g, shape, width - x1, y1, width - x2, y2);

}

if (symmetry >= SYMMETRY\_4) {

drawFigure(g, shape, x1, height - y1, x2, height - y2);

drawFigure(g, shape, width - x1, height - y1, width - x2, height - y2);

}

if (symmetry == SYMMETRY\_8) {

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) {

int x, y;

int w, h;

if (x2 >= x1) {

x = x1;

w = x2 - x1;

}

else {

x = x2;

w = x1 - x2;

}

if (y2 >= y1) {

y = y1;

h = y2 - y1;

}

else {

y = y2;

h = y1 - y2;

}

repaint(x,y,w+1,h+1);

}

private void repaintMultiRect(int x1, int y1, int x2, int y2) {

int width = getWidth();

int height = getHeight();

repaintRect(x1,y1,x2,y2);

if (symmetry >= SYMMETRY\_2) {

repaintRect(width - x1, y1, width - x2, y2);

}

if (symmetry >= SYMMETRY\_4) {

repaintRect(x1, height - y1, x2, height - y2);

repaintRect(width - x1, height - y1, width - x2, height - y2);

}

if (symmetry == SYMMETRY\_8) {

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() {

if (OSI == null || widthOfOSI != getSize().width || heightOfOSI != getSize().height) {

OSI = null;

undoBuffer = null;

widthOfOSI = getWidth();

heightOfOSI = getHeight();

OSI = createImage(widthOfOSI,heightOfOSI);

Graphics OSG = OSI.getGraphics();

OSG.setColor(getBackground());

OSG.fillRect(0, 0, widthOfOSI, heightOfOSI);

OSG.dispose();

undoBuffer = createImage(widthOfOSI,heightOfOSI);

OSG = undoBuffer.getGraphics();

OSG.setColor(getBackground());

OSG.fillRect(0, 0, widthOfOSI, heightOfOSI);

OSG.dispose();

}

}

public void paintComponent(Graphics g) {

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) {

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) {

customColor = c;

custom.setSelected(true);

}

}

else if (command.equals("Clear")) {

Graphics g = OSI.getGraphics();

g.setColor(getBackground());

g.fillRect(0,0,getSize().width,getSize().height);

g.dispose();

repaint();

}

else if (command.equals("Undo")) {

Image temp = OSI;

OSI = undoBuffer;

undoBuffer = temp;

repaint();

}

else if (command.equals("Quit")) {

dispose();

if (standAlone)

System.exit(0);

}

}

private void clear(Color background) {

setBackground(background);

if (background.equals(getSelectedColor())) {

if (background.equals(Color.black))

white.setSelected(true);

else

black.setSelected(true);

}

Graphics g = OSI.getGraphics();

g.setColor(getBackground());

g.fillRect(0,0,getSize().width,getSize().height);

g.dispose();

repaint();

}

public void mousePressed(MouseEvent evt) {

if (dragging == true)

return;

prevX = startX = evt.getX();

prevY = startY = evt.getY();

figure = getSelectedShape();

symmetry = getSelectedSymmetry();

dragColor = getSelectedColor();

checkOSI();

Graphics undoGraphics = undoBuffer.getGraphics();

undoGraphics.drawImage(OSI,0,0,null);

undoGraphics.dispose();

dragGraphics = OSI.getGraphics();

dragGraphics.setColor(dragColor);

dragging = true;

}

public void mouseReleased(MouseEvent evt) {

if (dragging == false)

return;

dragging = false;

mouseX = evt.getX();

mouseY = evt.getY();

if (figure == CURVE) {

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) {

putMultiFigure(dragGraphics,figure,startX,startY,mouseX,mouseY);

repaintMultiRect(startX,startY,mouseX,mouseY);

}

}

else {

repaintMultiRect(startX,startY,prevX,prevY);

if (mouseX != startX && mouseY != startY) {

putMultiFigure(dragGraphics,figure,startX,startY,mouseX,mouseY);

repaintMultiRect(startX,startY,mouseX,mouseY);

}

}

dragGraphics.dispose();

dragGraphics = null;

}

public void mouseDragged(MouseEvent evt) {

if (dragging == false)

return;

mouseX = evt.getX();

mouseY = evt.getY();

if (figure == CURVE) {

putMultiFigure(dragGraphics,LINE,prevX,prevY,mouseX,mouseY);

repaintMultiRect(prevX,prevY,mouseX,mouseY);

}

else {

repaintMultiRect(startX,startY,prevX,prevY);

repaintMultiRect(startX,startY,mouseX,mouseY);

}

prevX = mouseX;

prevY = mouseY;

}

public void mouseEntered(MouseEvent evt) { }

public void mouseExited(MouseEvent evt) { }

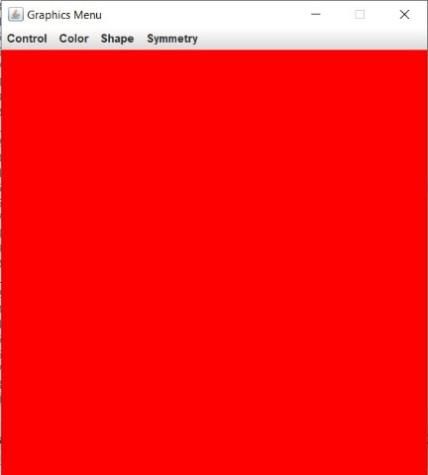
public void mouseClicked(MouseEvent evt) { }

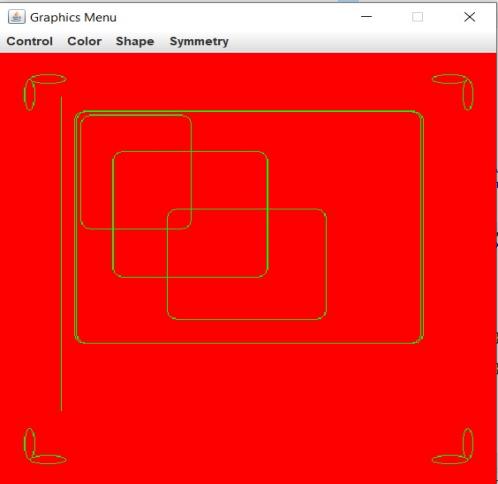
public void mouseMoved(MouseEvent evt) { }

}

}

**OUTPUT:**

****

****

**Experiment No:18,19**

**AIM:** Write a Java program to implement JTable and JTree. Write a Java program to implement JTabbedPane.

**DESCRIPTION :**

**Java JTabbedPane:**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.

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

**JTable:** The JTable class is used to display data in tabular form. It is composed of rows and columns.

**JTree:** The JTree class is used to display the tree structured data or hierarchical data. JTree is a complex component. It has a 'root node' at the top most which is a parent for all nodes in the tree. It inherits JComponent class.

**SYNTAX:**

**JTable class declaration, Constructors:**

|  |  |
| --- | --- |
| **Constructor** | **Description** |
| JTable() | Creates a table with empty cells. |
| JTable(Object[][] rows, Object[] columns) | Creates a table with the specified data. |

**JTree constructors:**

|  |  |
| --- | --- |
| **Constructor** | **Description** |
| JTree() | Creates a JTree with a sample model. |
| JTree(Object[] value) | Creates a JTree with every element of the specified array as the child of a new root node. |
| JTree(TreeNode root) | Creates a JTree with the specified TreeNode as its root, which displays the root node. |

**PROGRAM:**

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() {

JFrame jfrm = new JFrame("JTabbedPaneDemo");

jfrm.setLayout(new FlowLayout());

jfrm.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

jfrm.setSize(1000, 1000);

JTabbedPane jtp = new JTabbedPane();

jtp.addTab("Tree", new TreePanel());

jtp.addTab("Table", new TablePanel());

jfrm.add(jtp);

jfrm.setVisible(true);

}

public static void main(String[] args) {

SwingUtilities.invokeLater(

new Runnable() {

public void run() {

new JTabbedPaneDemo();

}});

}}

class TreePanel extends JPanel {

public TreePanel() {

DefaultMutableTreeNode top = new DefaultMutableTreeNode("Options");

DefaultMutableTreeNode a = new DefaultMutableTreeNode("A");

top.add(a);

DefaultMutableTreeNode a1 = new DefaultMutableTreeNode("A1");

a.add(a1);

DefaultMutableTreeNode a2 = new DefaultMutableTreeNode("A2");

a.add(a2);

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);

JTree tree = new JTree(top);

JScrollPane jsp = new JScrollPane(tree);

add(jsp);

JLabel jlab = new JLabel();

add(jlab, BorderLayout.SOUTH);

tree.addTreeSelectionListener(new TreeSelectionListener() {

public void valueChanged(TreeSelectionEvent tse) {

jlab.setText("Selection is " + tse.getPath());

}

});

} }

class TablePanel extends JPanel {

public TablePanel() {

String[] colHeads = { "Name", "Extension", "ID#" };

Object[][] data = { { "Matt", "5672", "217" }, { "Claire", "6741", "444" },

{ "Erwin", "9023", "519" }, { "Ellen", "1134", "532" }, { "Jennifer", "5689", "112" },

{ "Ed", "9030", "133" }, { "Helen", "6751", "145" } };

JTable table = new JTable(data, colHeads);

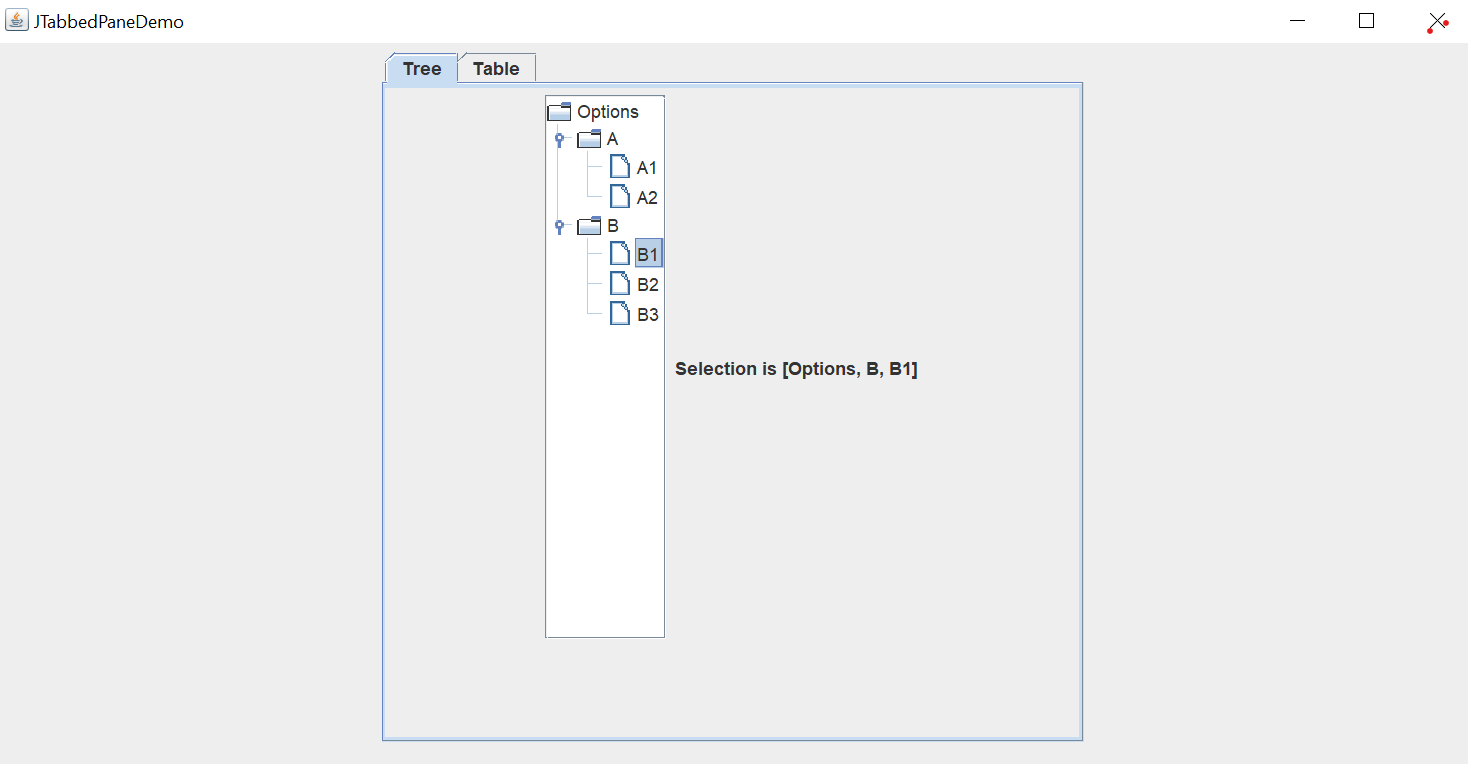
JScrollPane jsp = new JScrollPane(table);

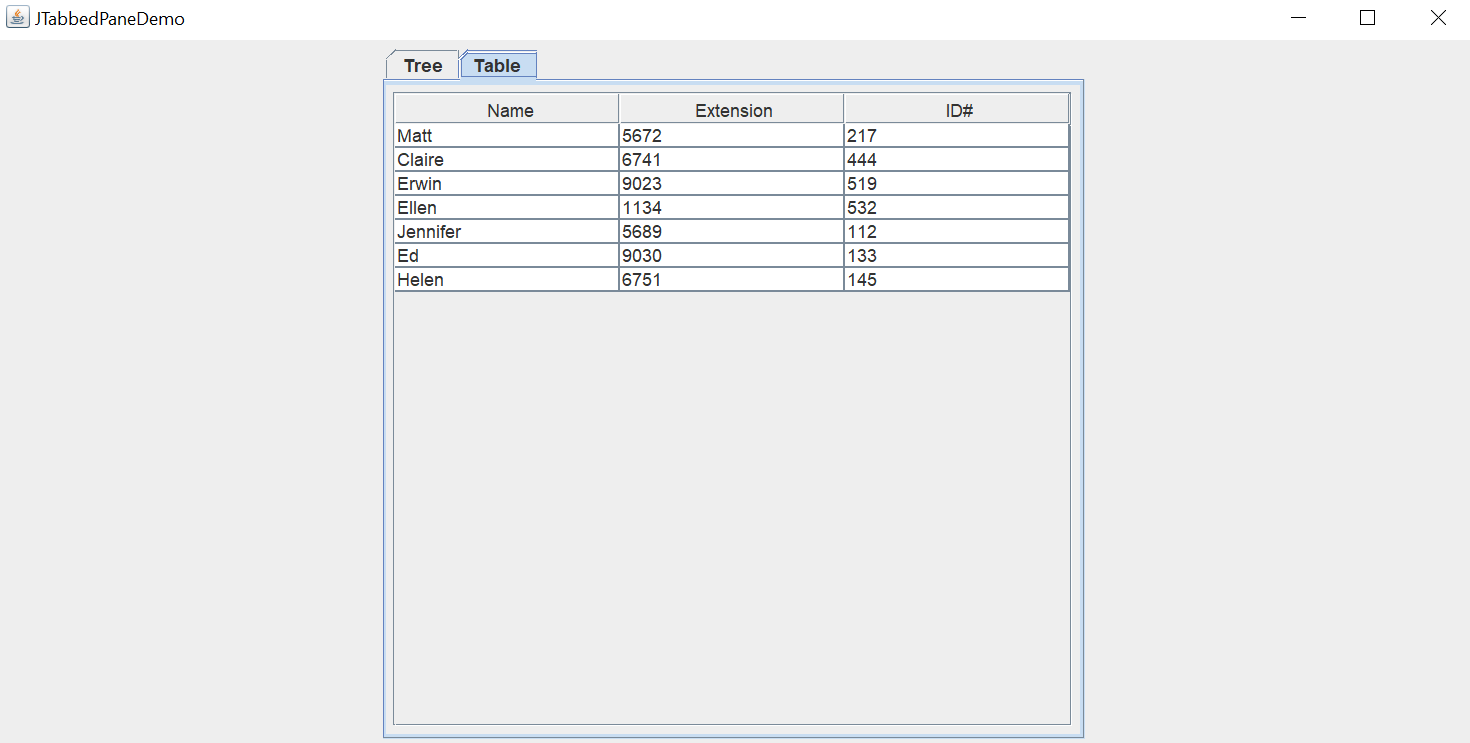
add(jsp);

}

}

OUTPUT:



****

**EXPEREMENT 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:**

### **Important methods**

|  |  |
| --- | --- |
| **Method** | **Description** |
| 1) public Socket accept() | returns the socket and establish a connection between server and client. |
| 2) public synchronized void close() | closes the server socket. |

**PROGRAM:**

**Server**

import java.io.\*;

import java.net.\*;

public class Server{

public static void main(String args[]){

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);

}

}}

Client

import java.io.\*;

import java.net.\*;

public class Client{

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

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);

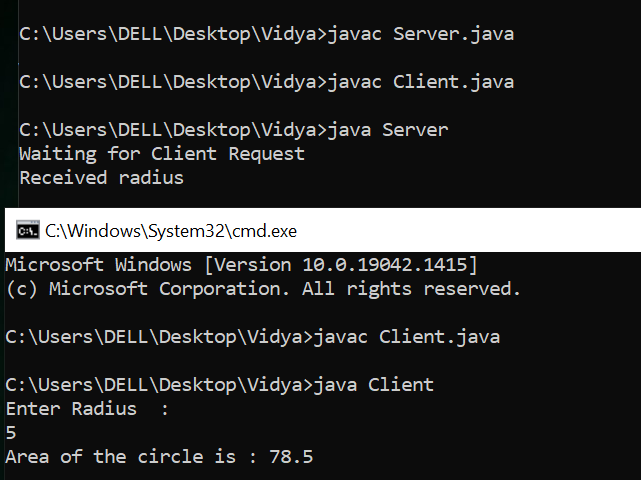
br.close();

ps.close();

}

}

**OUTPUT:**

****

**Additional Experiment 1**

**AIM:** Demonstrate the usage of methods in String, StringBuffer class.

**DESCRIPTION:**

->The Java development environment provides two classes that store and manipulate

character data: String, for constant strings, and StringBuffer, for mutable strings.

->String is a sequence of characters. In java, objects of String are immutable which

means a constant and cannot be changed once created.

->Java StringBuffer class is used to create mutable (modifiable) String objects

**SYNTAX:**

Syntax for using StringBuffer class methods:

|  |  |
| --- | --- |
| **Sr.No.** | **Methods & Description** |
| 1 | [public StringBuffer append(String s)](https://www.tutorialspoint.com/java/stringbuffer_append.htm)  Updates the value of the object that invoked the method. The method takes boolean, char, int, long, Strings, etc. |
| 2 | [public StringBuffer reverse()](https://www.tutorialspoint.com/java/stringbuffer_reverse.htm)  The method reverses the value of the StringBuffer object that invoked the method. |
| 3 | [public delete(int start, int end)](https://www.tutorialspoint.com/java/stringbuffer_delete.htm)  Deletes the string starting from the start index until the end index. |
| 4 | [public insert(int offset, int i)](https://www.tutorialspoint.com/java/stringbuffer_insert.htm)  This method inserts a string **s** at the position mentioned by the offset. |
| 5 | [replace(int start, int end, String str)](https://www.tutorialspoint.com/java/stringbuffer_replace.htm)  This method replaces the characters in a substring of this StringBuffer with characters in the specified String. |

Syntax for using String class methods:

|  |  |
| --- | --- |
| **Sr.No.** | **Method & Description** |
| 1 | [char charAt(int index)](https://www.tutorialspoint.com/java/java_string_charat.htm):Returns the character at the specified index. |
| 2 | [int compareTo(String anotherString)](https://www.tutorialspoint.com/java/java_string_compareto_anotherstring.htm):Compares two strings lexicographically. |
| 3 | [int compareToIgnoreCase(String str)](https://www.tutorialspoint.com/java/java_string_comparetoignorecase.htm):Compares two strings lexicographically, ignoring case differences. |
| 4 | [String concat(String str)](https://www.tutorialspoint.com/java/java_string_concat.htm):Concatenates the specified string to the end of this string. |

**PROGRAM:**

public class Main {

public static void main(String[] args) {

StringBuffer sb = new StringBuffer("Welcome to");

System.out.println("Given stringbuffer is: " + sb);

System.out.println("length of stringbuffer is: " + sb.length());

System.out.println("capacity of stringbuffer is: " + sb.capacity());

System.out.println("character at index 5 is: " + sb.charAt(5));

sb.setCharAt(7,'!');

System.out.println("after using setCharAT() " + sb);

System.out.println("codePointAt index 5 of the stringbuffer is: " + sb.codePointAt(5));

System.out.println("appendind: " + sb.append("java programming......"));

System.out.println("substring of stringbuffer" + sb.substring(10,20));

sb.delete(27,32);

System.out.println("after using deleting:"+sb);

System.out.println("reverse of the stringbuffer is: " + sb.reverse());

System.out.println("USING STRING CLASS");

String s= "Welcome to programming";

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

System.out.println("character at 3rd position is " + s.charAt(3));

System.out.println("substring " + s.substring(3));

System.out.println("substring is" + s.substring(2,5));

String s1 = "java";

String s2 = " programming....";

System.out.println("concatenated string is" +s1.concat(s2));

String s4 = "java programming";

System.out.println("index of Share " + s4.indexOf("programming"));

Boolean out = "Java".equals("java");

System.out.println("checking Equality " + out);

out = "java".equals("java");

System.out.println("Checking Equality " + out);

out = "Java".equalsIgnoreCase("jAVA");

System.out.println("Checking Equality " + out);

int out1 = s1.compareTo(s2);

System.out.println("the difference between s1 and s2="+out1);

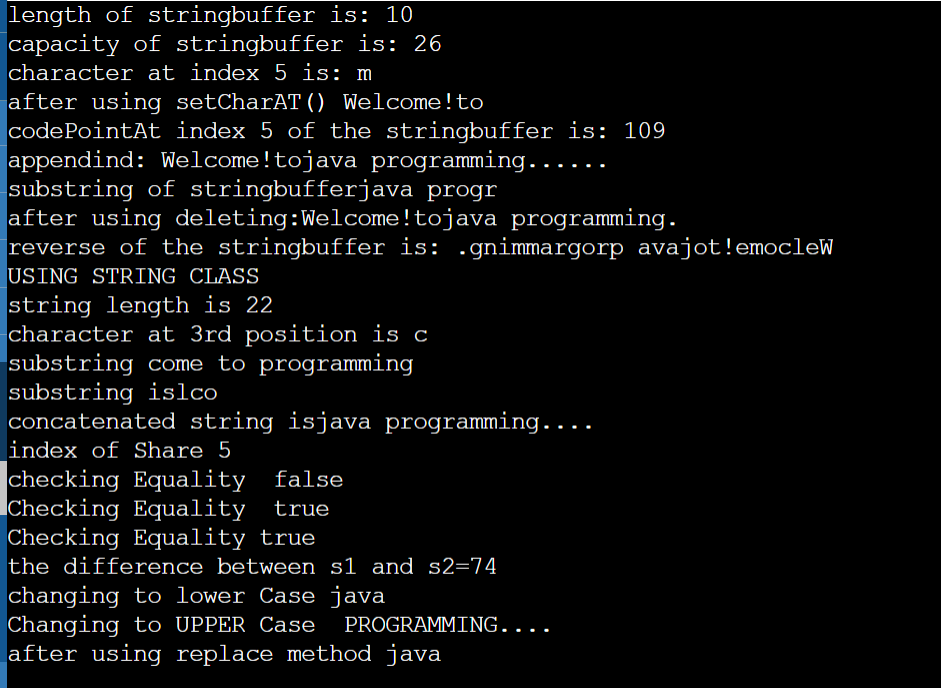
System.out.println("changing to lower Case " +s1.toLowerCase());

System.out.println("Changing to UPPER Case " +s2.toUpperCase());

String str2 = "jeve".replace('e' ,'a') ;

System.out.println("after using replace method " + str2);

}}  
**OUTPUT:**

****

**ADDITIONAL EXPERIMENT 2**

**AIM:** Write a java program to demonstrate Stack implementation.

**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).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.

**push():** When we insert an element in a stack then the operation is known as a push. If the stack is full then the overflow condition occurs.

**pop():** When we delete an element from the stack, the operation is known as a pop. If the stack is empty means that no element exists in the stack, this state is known as an underflow state.

**isEmpty():** It determines whether the stack is empty or not.

**isFull():** It determines whether the stack is full or not.'

**peek():** It returns the element at the given position.

**SYNTAX:**

**Switch case in java:**

switch(*expression*) {

case x:

*// code block*

break;

case y:

*// code block*

break;

default:

*// code block*

}

**PROGRAM:**

import java.util.Scanner;

class Stack1{

int max=5;

int stack[]=new int[max];

int top=-1;

boolean isfull() {

return (top==max-1)?true:false;

}

boolean isempty() {

return (top==-1)?true:false;

}

void push(int e)

{

if(isfull())

System.out.println("stack is full!!");

else

stack[++top]=e;

}

int pop() {

if(isempty()) {

System.out.println("stack is empty!!");

}

return stack[top--];

}

int peek() {

return stack[top];

}

void viewstack() {

if(isempty())

System.out.println("stack is empty!!");

else

System.out.print("the elements in the stack are:");

for(int i=top;i>=0;i--) {

System.out.print(stack[i]);

}

}

}

public class Stacks {

public static void main(String[] args) {

Scanner sc=new Scanner(System.in);

Stac s=new Stac();

int max=5;

int i,j,e,x,y,z,choice;

System.out.println("menu for stack");

System.out.println("\n1.push\n2.pop\n3.peek\n4.isempty\n5.isfull\n6.viewstack\n7.exit");

do{

System.out.println("enter your choice:");

choice=sc.nextInt();

switch(choice)

{

case 1:

System.out.println("enter the elements:");

for(i=0;i<max;i++)

{

e=sc.nextInt();

s.push(e);

}

break;

case 2:

x=s.pop();

System.out.println(" popped element is:"+x);

break;

case 3:

System.out.println(" peek element is :"+s.peek());

break;

case 4:if(s.isempty())

System.out.println("stack is empty!! ");

else

System.out.println("stack is not empty");

break;

case 5: if(s.isfull())

System.out.println("stack is full!!");

else

System.out.println("stack is not full");

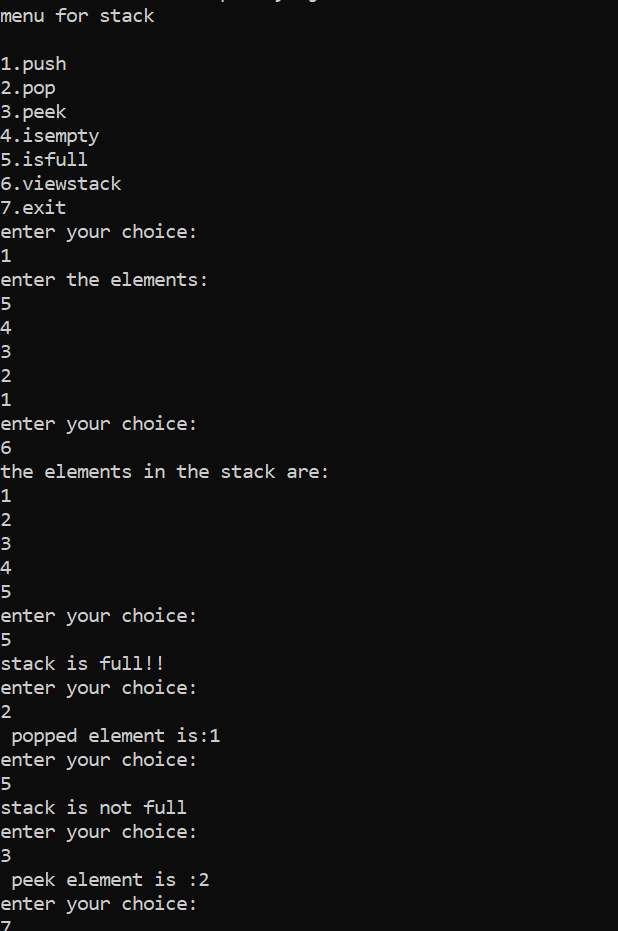
break;

case 6:s.viewstack();

break;

}}while(choice!=7);

}

}  
OUTPUT**:  
**

**ADDITIOMAL EXPEREMENT 3**

**AIM:** Write a java program to demonstrate Queue implementation.

**DESCRIPTION:**

**QUEUE:** A Queue is a linear structure which follows a particular order in which the operations are performed. The order is First In First Out (FIFO)

**enqueue()** − add (store) an item to the queue.

**dequeue()** − remove (access) an item from the queue.

**peek()** − Gets the element at the front of the queue without removing it.

**isfull()** − Checks if the queue is full.

**isempty()** − Checks if the queue is empty.

**SYNTAX:**

switch(*expression*) {

case x:

*// code block*

break;

case y:

*// code block*

break;

default:

*// code block*

}

**PROGRAM:**

import java.util.Scanner;

class Queue1{

int max;

int front,rear;

Queue1(int m)

{

max=m;

front=rear=0;

}

int queue[]=new int[5];

boolean isfull() {

return (rear==max)?true:false;

}

boolean isempty() {

return (front==rear)?true:false;

}

void enqueue(int e){

if(isfull())

System.out.println("queue is full!!");

else{

queue[rear]=e;

rear++;

}

return;

}

int dequeue() {

if(isempty()) {

System.out.println("queue is empty!!");

return -1;

}

return queue[front++];

}

void viewqueue() {

if(isempty())

System.out.println("queue is empty!!");

else

System.out.println("the elements in the queue are:");

for(int i=front;i<rear;i++)

System.out.println(queue[i]);

}

}

public class Queues{

public static void main(String[] args) {

Scanner sc=new Scanner(System.in);

Queue1 s=new Queue1(5);

int i,j,e,x,y,z,choice ,max=5;

System.out.println("menu for queue");

System.out.println("\n1.Eenqueue\n2.Dequeue\n3.viewqueue\n4.isempty\n5.isfull\n6.exit); do{

System.out.println("enter your choice:");

choice=sc.nextInt();

switch(choice){

case 1:System.out.println("enter the elements:");

for(i=0;i<max;i++){

e=sc.nextInt();

s.enqueue(e);

}

break;

case 2:x=s.dequeue();

if(x!=-1)

System.out.println(" removed element is:"+x);

break;

case 3:s.viewqueue();

break;

case 4:if(s.isempty())

System.out.println("queue is empty");

else

System.out.println("queue is not empty");

break;

case 5:if(s.isfull())

System.out.println("queue is full");

else

System.out.println("queue is not full");

break;

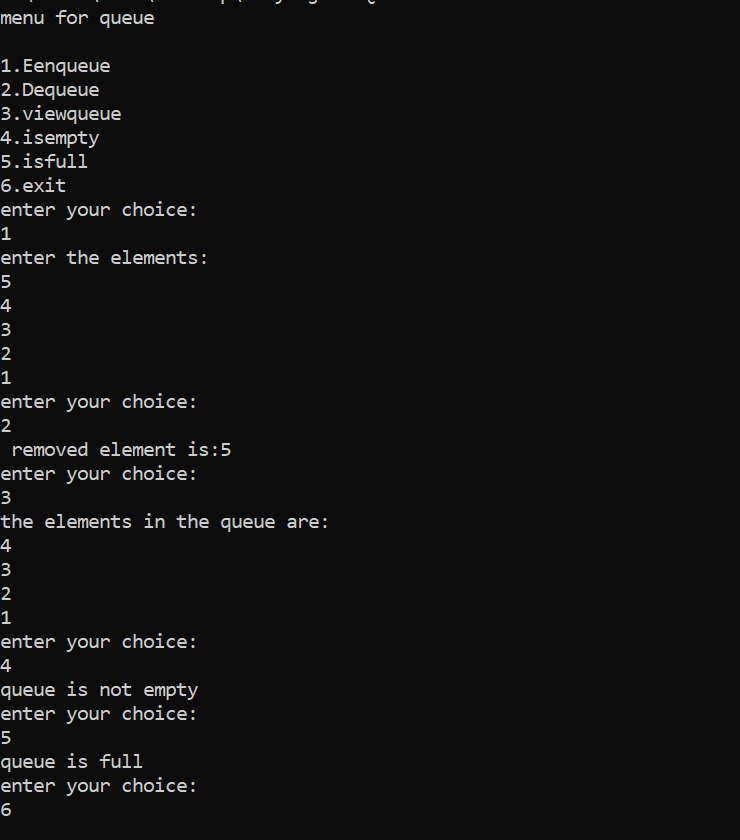
case 6:break;

}

}while(choice!=6);

}**}**

**OUTPUT:**

****

**ADDITIONAL PROGRAM 4**

**AIM**: Write a program to demonstrate byte array streams in java

**DESCRIPTION:**

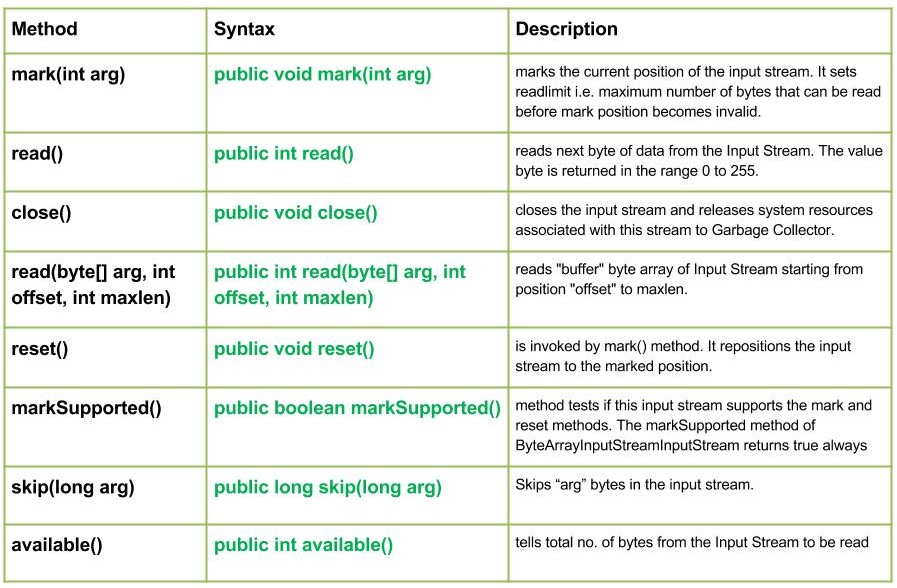
# ByteStream Classes in Java

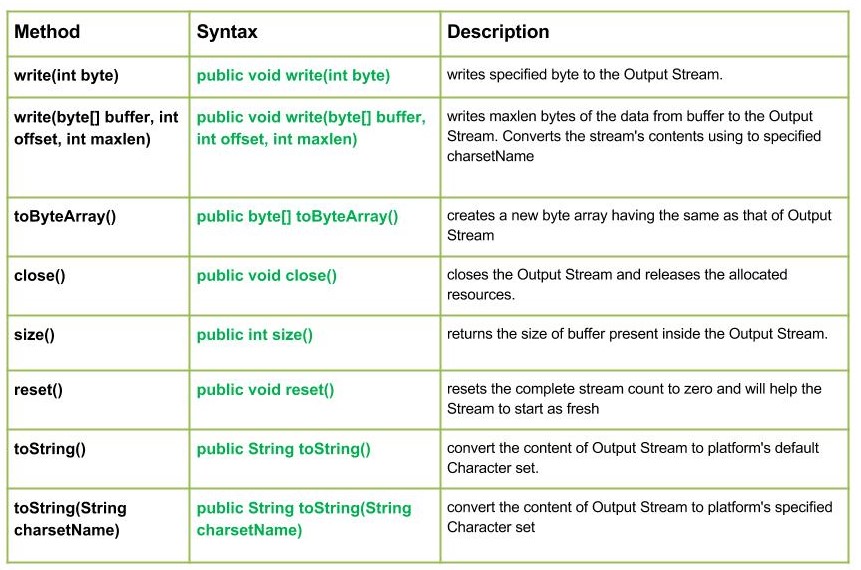
ByteStream classes are used to read bytes from the input stream and write bytes to the output stream. In other words, we can say that ByteStream classes read/write the data of 8-bits. We can store video, audio, characters, etc., by using ByteStream classes. These classes are part of the java.io package.

The ByteArrayInputStream class allows a buffer in the memory to be used as an InputStream. The input source is a byte array.The ByteArrayOutputStream class stream creates a buffer in memory and all the data sent to the stream is stored in the buffer.

**SYNTAX:**

**ByteArrayInputStream:**

 **ByteArrayOutputStream:**



**PROGRAM:**

import java.io.ByteArrayInputStream;

import java.io.ByteArrayOutputStream;

import java.io.\*;

public class Main{

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

String s="welcome to java programming..";

ByteArrayOutputStream baos=new ByteArrayOutputStream();

baos.write(s.getBytes());

System.out.println("Buffer size:"+baos.size());

byte b[]=new byte[baos.size()];

b=(baos.toString()).getBytes();

ByteArrayInputStream in=new ByteArrayInputStream(b);

int c;

while((c=in.read())!=-1)

System.out.print(Character.toUpperCase((char)c));

}}  
**OUTPUT:**

****

**ADDITIONAL PROGRAM 5**

**AIM**: Write a program to demonstrate character stream classes in java

**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.

Java FileWriter class is used to write data to the file.

Java FileReader class is used to read data from the file.  **SYNTAX:**

## **Methods of FileReader class**

|  |  |
| --- | --- |
| **Method** | **Description** |
| int read() | It is used to return a character in ASCII form. It returns -1 at the end of file. |
| void close() | It is used to close the FileReader class. |

## **Methods of FileWriter class**

|  |  |
| --- | --- |
| **Method** | **Description** |
| void write(String text) | It is used to write the string into FileWriter. |
| void write(char c) | It is used to write the char into FileWriter. |
| void write(char[] c) | It is used to write char array into FileWriter. |
| void flush() | It is used to flushes the data of FileWriter. |
| void close() | It is used to close the FileWriter. |

**PROGRAM:**

**import java.io.\*;**

class Main{

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

String str = "File Handling in Java using FileWriter and FileReader";

FileWriter fw=new FileWriter("output.txt");

for (int i = 0; i < str.length(); i++)

fw.write(str.charAt(i));

System.out.println("Writing successful");

fw.close();

int ch;

FileReader fr = new FileReader("output.txt");

while ((ch=fr.read())!=-1)

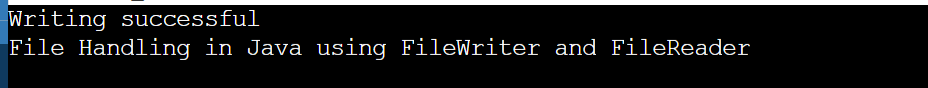
System.out.print((char)ch);

fr.close();

}

}

**OUTPUT:**

****

**ADDITIONAL EXPERIMENT NO: 6**

**AIM:**write a program to show serialization and deserialization.

**DESCRIPTION:**

# Serialization and Deserialization

**Serialization in Java** is a mechanism of *writing the state of an object into a byte-stream*. It is mainly used in Hibernate, RMI, JPA, EJB and JMS technologies.

The reverse operation of serialization is called *deserialization* where byte-stream is converted into an object. The serialization and deserialization process is platform-independent, it means you can serialize an object on one platform and deserialize it on a different platform.

For serializing the object, we call the **writeObject()** method of *ObjectOutputStream*class, and for deserialization we call the **readObject()** method of *ObjectInputStream* class.

We must have to implement the *Serializable* interface for serializing the object.

**SYNTAX:**

**Constructor**

|  |  |
| --- | --- |
| 1) public ObjectOutputStream(OutputStream out) throws IOException {} | It creates an ObjectOutputStream that writes to the specified OutputStream. |
| 1) public ObjectInputStream(InputStream in) throws IOException {} | It creates an ObjectInputStream that reads from the specified InputStream. |

**Important Methods**

|  |  |
| --- | --- |
| **Method** | **Description** |
| 1) public final void writeObject(Object obj) throws IOException {} | It writes the specified object to the ObjectOutputStream. |
| 2) public void flush() throws IOException {} | It flushes the current output stream. |
| 3) public void close() throws IOException {} | It closes the current output stream. |
| 4) public final Object readObject() throws IOException, ClassNotFoundException{} | It reads an object from the input stream |

**PROGRAM:**

import java.io.\*;

import java.io.ObjectOutputStream;

class SD implements Serializable{

int i;

double d;

boolean b;

char c;

SD(int i,double d,boolean b,char c){

this.i=i;

this.d=d;

this.b=b;

this.c=c;

}

void display() {

System.out.println("int value:"+i);

System.out.println("double value:"+d);

System.out.println("boolean value:"+b);

System.out.println("char value:"+c);

}

}

public class Main {

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

ObjectOutputStream os=new ObjectOutputStream(new FileOutputStream("file1.txt"));

SD o1=new SD(1,2.3d,true,'h');

SD o2=new SD(8,3.3d,false,'s');

os.writeObject(o1);

os.writeObject(o2);

os.close();

try{

ObjectInputStream oi=new ObjectInputStream(new FileInputStream("file1.txt"));

SD s1=(SD)oi.readObject();

s1.display();

oi.close();

}

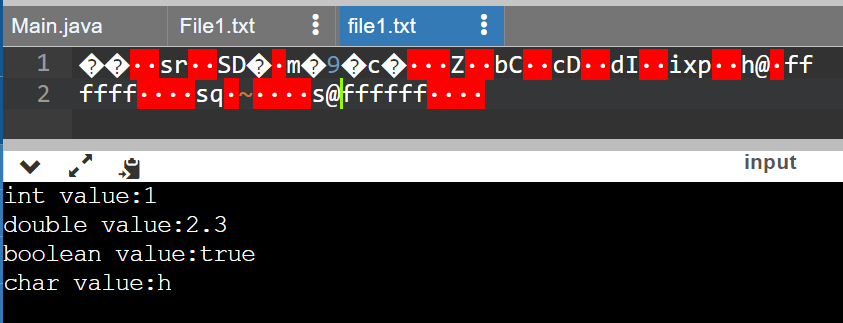
catch(Exception e) {

System.out.println(e);

}

}}

OUTPUT:

****