**JAVA LAB PRACTICAL-2**

## **51. Write a Java program to handle arithmetic exceptions and number format exceptions.**

## **CODE:**

**import java.util.Scanner;**

**public class ExceptionHandlingExample**

**{**

**public static int divideNumbers(int dividend, int divisor)**

**{**

**return dividend / divisor;**

**}**

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

**{**

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

**try**

**{**

**System.out.print("Enter a number: ");**

**int number1 = Integer.parseInt(scanner.nextLine());**

**System.out.print("Enter another number: ");**

**int number2 = Integer.parseInt(scanner.nextLine());**

**int result = divideNumbers(number1, number2);**

**System.out.println("Result: " + result);**

**}**

**catch (NumberFormatException e)**

**{**

**System.out.println("Invalid number format entered!");**

**System.out.println("message :"+e.getMessage());**

**}**

**catch (ArithmeticException e)**

**{**

**System.out.println("Cannot divide by zero!");**

**System.out.println("message :"+e.getMessage());**

**}**

**}**

**}**

## **OUTPUT:**

Enter a number: 2.5

Invalid number format entered!

message :For input string: "2.5"

(OR)

Enter a number: 5

Enter another number: 0

Cannot divide by zero!

message :/ by zero

(OR)

Enter a number: 10

Enter another number: 6

Result: 1

## **52. Write a Java program to handle array index out of bounds exception**

## **CODE:**

**import java.util.Scanner;**

**public class ArrayOutOfBound**

**{**

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

**{**

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

**try**

**{**

**int[] numbers = { 1, 2, 3, 4, 5 };**

**System.out.print("Enter an index: ");**

**int index = Integer.parseInt(scanner.nextLine());**

**int element = numbers[index];**

**System.out.println("Element at index " + index + ": " + element);**

**}**

**catch (NumberFormatException e)**

**{**

**System.out.println("Invalid index format entered!");**

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

**}**

**catch (ArrayIndexOutOfBoundsException e)**

**{**

**System.out.println("Index is out of bounds!"+e.getMessage());**

**}**

**}**

**}**

## **OUTPUT:**

Enter an index: 3

Element at index 3: 4

(OR)

Enter an index: 7

Index is out of bounds!Index 7 out of bounds for length 5

(OR)

Enter an index: 2.3

Invalid index format entered!

For input string: "2.3"

## **53. Write a Java program to create and handle user defined exception**

## **CODE:**

**class InsufficientFundsException extends Exception**

**{**

**public InsufficientFundsException(String message)**

**{**

**super(message);**

**}**

**}**

**class BankAccount**

**{**

**private String accountNumber;**

**private double balance;**

**public BankAccount(String accountNumber, double balance)**

**{**

**this.accountNumber = accountNumber;**

**this.balance = balance;**

**}**

**public void withdraw(double amount) throws InsufficientFundsException**

**{**

**if (amount > balance)**

**{**

**throw new InsufficientFundsException("Insufficient funds in the account!");**

**}**

**balance -= amount;**

**System.out.println("Withdrawal of $" + amount + " successful. New balance: $" + balance);**

**}**

**}**

**public class UserExceptions**

**{**

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

**{**

**BankAccount account = new BankAccount("123456789", 500.0);**

**try**

**{**

**account.withdraw(800.0); //Change to 400 so that exception will not be thrown**

**}**

**catch (InsufficientFundsException e)**

**{**

**System.out.println("Exception occurred: " + e.getMessage());**

**}**

**}**

**}**

## **OUTPUT:**

Exception occurred: Insufficient funds in the account!

(OR)

Withdrawal of $400.0 successful. New balance: $100.0

## **54. Write a Java program which extends Thread class to implement multithreading in java**

## **CODE:**

**public class ThreadClassImplementation extends Thread**

**{**

**ThreadClassImplementation()**

**{**

**super();**

**}**

**ThreadClassImplementation(String s)**

**{**

**super(s);**

**}**

**public void run()**

**{**

**System.out.println("MyClass Run method: "+Thread.currentThread().getName());**

**}**

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

**{**

**System.out.println(Thread.currentThread().getName() + " : "+Thread.currentThread().getPriority());**

**ThreadClassImplementation m=new ThreadClassImplementation("MyThread-0");**

**m.start();**

**System.out.println("Hello World! from Main Thread");**

**ThreadClassImplementation m1=new ThreadClassImplementation("MyThread-1");**

**ThreadClassImplementation m2=new ThreadClassImplementation("MyThread-2");**

**m1.start();**

**m2.start();**

**}**

**}**

## **OUTPUT:**

main : 5

Hello World! from Main Thread

MyClass Run method: MyThread-1

MyClass Run method: MyThread-2

MyClass Run method: MyThread-0

## **55. Write a Java program to implement multithreading in java using Runnable interface**

## **CODE:**

**class MyThread1 implements Runnable**

**{**

**String s=null;**

**MyThread1(String s1)**

**{**

**s=s1;**

**}**

**public void run()**

**{**

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

**for(int i=1;i<=4;i++)**

**{**

**System.out.println("It is from thread a :i="+i);**

**}**

**System.out.println("End of thread a");**

**}**

**}**

**public class RunableInterface**

**{**

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

**{**

**MyThread1 m1=new MyThread1("Thread started....");**

**Thread t1=new Thread(m1);**

**t1.start();**

**System.out.println("Main Thread");**

**}**

**}**

## **OUTPUT:**

Main Thread

Thread started....

It is from thread a :i=1

It is from thread a :i=2

It is from thread a :i=3

It is from thread a :i=4

End of thread a

## **56. Write a Java program to create even and odd threads by extending Thread class**

## **CODE:**

**class Even extends Thread**

**{**

**public void run()**

**{**

**for(int i=2;i<=6;i+=2)**

**{**

**System.out.println("It is from thread Even :i="+i);**

**}**

**System.out.println("End of thread Even");**

**}**

**}**

**class Odd extends Thread**

**{**

**public void run()**

**{**

**for(int j=1;j<=6;j+=2)**

**{**

**System.out.println("It is from thread Odd:j="+j);**

**}**

**System.out.println("End of thread Odd");**

**}**

**}**

**public class EvenOddThread**

**{**

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

**{**

**Even a=new Even();**

**a.start();**

**System.out.println("Main method");**

**Odd b=new Odd();**

**b.start();**

**System.out.println("End of Main");**

**}**

**}**

## **OUTPUT:**

Main method

End of Main

It is from thread Even :i=2

It is from thread Odd:j=1

It is from thread Even :i=4

It is from thread Odd:j=3

It is from thread Odd:j=5

End of thread Odd

It is from thread Even :i=6

End of thread Even

## **57. Write a Java program for Non Synchronized withdraw operation from the shared bank account**

## **CODE:**

**class BankAccount**

**{**

**private double bal;**

**BankAccount()**

**{**

**this.bal=0.0d;**

**}**

**BankAccount(double bal)**

**{**

**this.bal=bal;**

**}**

**public double getBalance()**

**{**

**return this.bal;**

**}**

**public void withdraw(double amt)**

**{**

**this.bal = this.bal-amt;**

**System.out.println("Amount withdrawn is "+ amt +" and the remaining bal is: " + getBalance());**

**}**

**}**

**class MyBankThread extends Thread**

**{**

**BankAccount bankAcc;**

**MyBankThread()**

**{**

**super();**

**}**

**MyBankThread(String s, BankAccount ba)**

**{**

**super(s);**

**bankAcc=ba;**

**}**

**public void run()**

**{**

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

**bankAcc.withdraw(75);**

**}**

**}**

**class BankAccountThread**

**{**

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

**{**

**BankAccount bankAcc=new BankAccount(100);**

**System.out.println("Initial Bank Balance is: "+bankAcc.getBalance());**

**MyBankThread wBank=new MyBankThread("Wife",bankAcc);**

**wBank.start();**

**MyBankThread hBank=new MyBankThread("Husband",bankAcc);**

**hBank.start();**

**}**

**}**

## **OUTPUT:**

Initial Bank Balance is: 100.0

Wife

Husband

Amount withdrawn is 75.0 and the remaining bal is: -50.0

Amount withdrawn is 75.0 and the remaining bal is: 25.0

## **58. Write a Java program for Synchronized withdraw operation from the shared bank account**

## **CODE:**

**class BankAccount**

**{**

**private double bal;**

**BankAccount()**

**{**

**this.bal=0.0d;**

**}**

**BankAccount(double bal)**

**{**

**this.bal=bal;**

**}**

**public double getBalance()**

**{**

**return this.bal;**

**}**

**public synchronized void withdraw(double amt, Thread t)**

**{**

**if (getBalance()<amt)**

**{**

**System.out.println(t.getName()+" No Sufficient Balance in your account..");**

**return;**

**}**

**else**

**{**

**this.bal=this.bal-amt;**

**System.out.println(t.getName()+" withdrawn amount is "+amt +" and the remaining bal is: " + getBalance());**

**}**

**}**

**}**

**class MyBankThread extends Thread**

**{**

**BankAccount bankAcc;**

**MyBankThread()**

**{**

**super();**

**}**

**MyBankThread(BankAccount ba)**

**{**

**bankAcc=ba;**

**}**

**MyBankThread(String s, BankAccount ba)**

**{**

**super(s);**

**bankAcc=ba;**

**}**

**public void run()**

**{**

**System.out.println(this.getName() + " Invoked withdraw Operation");**

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

**{**

**bankAcc.withdraw(25,this);**

**}**

**}**

**}**

**class SynchronizedBankAccountThread**

**{**

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

**{**

**BankAccount bankAcc=new BankAccount(100);**

**System.out.println("Initial Bank Balance is: "+bankAcc.getBalance());**

**MyBankThread wBank=new MyBankThread(bankAcc);**

**wBank.setName("Wife");**

**wBank.start();**

**MyBankThread hBank=new MyBankThread(bankAcc);**

**hBank.setName("Husband");**

**hBank.start();**

**}**

**}**

## **OUTPUT:**

Initial Bank Balance is: 100.0

Wife Invoked withdraw Operation

Husband Invoked withdraw Operation

Wife withdrawn amount is 25.0 and the remaining bal is: 75.0

Wife withdrawn amount is 25.0 and the remaining bal is: 50.0

Wife withdrawn amount is 25.0 and the remaining bal is: 25.0

Husband withdrawn amount is 25.0 and the remaining bal is: 0.0

Husband No Sufficient Balance in your account..

Husband No Sufficient Balance in your account..

## **59. Write a Java program for Synchronized block withdraw operation from the shared bank account**

## **CODE:**

**class BankAccount**

**{**

**private double bal;**

**BankAccount()**

**{**

**this.bal=0.0d;**

**}**

**BankAccount(double bal)**

**{**

**this.bal=bal;**

**}**

**public double getBalance()**

**{**

**return this.bal;**

**}**

**public void withdraw(double amt, Thread t)**

**{**

**if (getBalance()<amt)**

**{**

**System.out.println(t.getName()+" No Sufficient Balance in your account..");**

**return;**

**}**

**else**

**{**

**synchronized(this)**

**{**

**this.bal=this.bal-amt;**

**System.out.println(t.getName()+" withdrawn amount is "+amt +" and the remaining bal is: " + getBalance());**

**}**

**}**

**}**

**}**

**class MyBankThread extends Thread**

**{**

**BankAccount bankAcc;**

**MyBankThread()**

**{**

**super();**

**}**

**MyBankThread(BankAccount ba)**

**{**

**bankAcc=ba;**

**}**

**MyBankThread(String s, BankAccount ba)**

**{**

**super(s);**

**bankAcc=ba;**

**}**

**public void run()**

**{**

**System.out.println(this.getName() + " Invoked withdraw Operation");**

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

**{**

**bankAcc.withdraw(25,this);**

**}**

**}**

**}**

**class SynchronizedBlockBankAccountThread**

**{**

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

**{**

**BankAccount bankAcc=new BankAccount(100);**

**System.out.println("Initial Bank Balance is: "+bankAcc.getBalance());**

**MyBankThread wBank=new MyBankThread(bankAcc);**

**wBank.setName("Wife");**

**wBank.start();**

**MyBankThread hBank=new MyBankThread(bankAcc);**

**hBank.setName("Husband");**

**hBank.start();**

**}**

**}**

## **OUTPUT:**

Initial Bank Balance is: 100.0

Husband Invoked withdraw Operation

Wife Invoked withdraw Operation

Husband withdrawn amount is 25.0 and the remaining bal is: 75.0

Husband withdrawn amount is 25.0 and the remaining bal is: 50.0

Husband withdrawn amount is 25.0 and the remaining bal is: 25.0

Wife withdrawn amount is 25.0 and the remaining bal is: 0.0

Wife No Sufficient Balance in your account..

Wife No Sufficient Balance in your account..

## 

## **60. Write a Java program for Inter Thread-Communication using Producer Consumer Problem**

## **CODE:**

**class Q {**

**int n;**

**boolean valueSet = false;**

**synchronized int get(){**

**if(!valueSet)**

**try**

**{**

**wait();**

**}**

**catch(InterruptedException e){**

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

**}**

**System.out.println("Consumed:"+n);**

**valueSet=false;**

**notify();**

**return n;**

**}**

**synchronized void put(int n)**

**{**

**if(valueSet)**

**try**

**{**

**wait();**

**}**

**catch(InterruptedException e)**

**{**

**System.out.println("Interrupted Exception caught"}**

**this.n=n;**

**valueSet=true;**

**System.out.println("Produced:"+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();**

**}**

**}**

**}**

**class ProdCons {**

**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:**

Press Control-c to stop

Produced: 0

Consumed: 0

Produced: 1

Consumed: 1

Produced: 2

Consumed: 2

Produced: 3

Consumed: 3

## **61. Write a Java program for Inter Thread-Communication to print Natural Numbers using Even and Odd Threads**

## **CODE:**

**class NaturalNumberPrinter**

**{**

**int maxNumber;**

**int currentNumber = 1;**

**boolean isEvenThreadTurn = true;**

**public NaturalNumberPrinter(int maxNumber)**

**{**

**this.maxNumber = maxNumber;**

**}**

**public synchronized void printEven()**

**{**

**while (currentNumber < maxNumber)**

**{**

**while (isEvenThreadTurn)**

**{**

**try**

**{**

**wait();**

**}**

**catch (InterruptedException e)**

**{**

**e.printStackTrace();**

**}**

**}**

**System.out.println("Even Thread:" + currentNumber);**

**currentNumber++;**

**isEvenThreadTurn = true;**

**notify();**

**}**

**}**

**public synchronized void printOdd()**

**{**

**while (currentNumber < maxNumber)**

**{**

**while (!isEvenThreadTurn)**

**{**

**try**

**{**

**wait();**

**}**

**catch (InterruptedException e)**

**{**

**e.printStackTrace();**

**}**

**}**

**System.out.println("Odd Thread:" + currentNumber);**

**currentNumber++;**

**isEvenThreadTurn = false;**

**notify();**

**}**

**}**

**}**

**public class NosUsingInterThread**

**{**

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

**{**

**int maxNumber = 10;**

**NaturalNumberPrinter printer = new NaturalNumberPrinter(maxNumber);**

**Thread evenThread = new Thread(() -> {**

**printer.printEven();**

**});**

**Thread oddThread = new Thread(() -> {**

**printer.printOdd();**

**});**

**evenThread.start();**

**oddThread.start();**

**}**

**}**

## **OUTPUT:**

Odd Thread:1

Even Thread:2

Odd Thread:3

Even Thread:4

Odd Thread:5

Even Thread:6

Odd Thread:7

Even Thread:8

Odd Thread:9

Even Thread:10

## **62. Write a Java program to perform various String operations**

## **CODE:**

**public class StringDemo {**

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

**char[] ch = {'r', 'a', 'm', 's'};**

**String charStr = new String(ch);**

**System.out.println("Char array based: " + charStr);**

**String charStr2 = new String(ch, 1, 2);**

**System.out.println("Char array based range: " + charStr2);**

**String str2 = new String(charStr);**

**System.out.println("String as input: " + str2);**

**byte[] b = {65, 66, 67, 68, 69};**

**String byteStr = new String(b);**

**System.out.println("Byte array based: " + byteStr);**

**String byteStr2 = new String(b, 2, 2);**

**System.out.println("Byte array based: " + byteStr2);**

**String name = "RamsIT"; // String Literal**

**System.out.println("The length of the string literal is: " + name.length());**

**String primType = String.valueOf(123);**

**System.out.println("Primitive type string: " + primType);**

**int i = 12345;**

**String istr = String.valueOf(i);**

**System.out.println("int i as toString: " + istr);**

**String concatStr = name + i;**

**System.out.println("Concatenated Str: " + concatStr);**

**String name1 = "Rams";**

**String name2 = "Rams";**

**System.out.println("String Pooling");**

**System.out.println("String name1 hashCode: " + name1.hashCode());**

**System.out.println("String name2 hashCode: " + name2.hashCode());**

**String s3 = new String("Rams");**

**System.out.println("name2: " + name1.hashCode());**

**System.out.println("s3: " + s3.hashCode());**

**System.out.println("String name1 identityHashCode: " + System.identityHashCode(name1));**

**System.out.println("String s3 identityHashCode: " + System.identityHashCode(s3));**

**System.out.println("Operator == compares two string addresses");**

**System.out.println("Name1 Vs Name2: " + (name1 == name2));**

**System.out.println("Name1 Vs s3: " + (name1 == s3));**

**System.out.println("equals() compares two strings based on content");**

**System.out.println("Name1 Vs Name2: " + name1.equals(name2));**

**System.out.println("Name1 Vs s3: " + name1.equals(s3));**

**System.out.println("\"Rams\".charAt(2): " + s3.charAt(2));**

**char[] ns = s3.toCharArray();**

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

**String nam = "This is Ramesh Ponnala handling java";**

**String newname = nam.replace("a", "n");**

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

**String newname2 = nam.replaceFirst("a", "n");**

**System.out.println(newname2);**

**String newname3 = nam.replaceAll("a", "n");**

**System.out.println(newname3);**

**}**

**}**

## **OUTPUT:**

Char array based: rams

Char array based range: am

String as input : rams

Byte array based: ABCDE

Byte array based: CD

The length of the string literal is :6

primitive type string :123

int i as toString12345

concatenated Str :RamsIT12345

String Pooling

String name1 hashCode :2539573

String name2 hashCode :2539573

name2 :2539573

s3 :2539573

String name1 identityhashCode:401424608

String s3 identityhashCode:1348949648

operator == compares two string addresses

Name1 Vs Name2: true

Name1 Vs s3: false

equals() compares two strings based on content

Name1 Vs Name2: true

Name1 Vs s3: true

"Rams".charAt(2) :m

Rams

This is Rnmesh Ponnnln hnndling jnvn

This is Rnmesh Ponnala handling java

This is Rnmesh Ponnnln hnndling jnvn

## **63. Write a Java program to work with StringBuffer and its operations**

## **CODE:**

**class Main{**

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

**{**

**StringBuffer s = new StringBuffer("Chaitanya Bharathi");**

**int p = s.length();**

**System.out.println("Length of string ="+ p);**

**int q = s.capacity();**

**System.out.println("Capacity of string =" + q);**

**//append(string)**

**s.append(" Institute Technology");**

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

**//insert(index,string)**

**s.insert(28, " of");**

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

**s.reverse();**

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

**s.reverse();**

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

**//delete(start\_index,end\_index)**

**s.delete(28, 32);**

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

**//deleteCharAt(index)**

**s.deleteCharAt(7);**

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

**StringBuffer str=new StringBuffer("Ramesh");**

**//replace(start\_index,end\_index,string)**

**str.replace(3,6,"Ponnala");**

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

**}**

**}**

## **OUTPUT:**

Length of string =18

Capacity of string =34

Chaitanya Bharathi Institute Technology

Chaitanya Bharathi Institute of Technology

ygolonhceT fo etutitsnI ihtarahB aynatiahC

Chaitanya Bharathi Institute of Technology

Chaitanya Bharathi InstituteTechnology

Chaitana Bharathi InstituteTechnology

RamPonnala

## **64. Write a Java program to work with StringBuilder and its operations**

## **CODE:**

**public class StringBuilder**

**{**

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

**{**

**StringBuilder str= new StringBuilder("RamsIT");**

**System.out.println("String = "+ str);**

**//reverse()**

**str.reverse();**

**System.out.println("Reverse String = "+ str);**

**//appendCodePoint(integer\_value)--this method will append the char 'a' to the string**

**str.appendCodePoint(97);**

**System.out.println("Modified String = "+ str);**

**int capacity = str.capacity();**

**System.out.println("Capacity of StringBuilder = "+ capacity);**

**}**

**}**

## **OUTPUT:**

String = RamsIT

Reverse String = TIsmaR

Modified String = TIsmaRa

Capacity of StringBuilder = 22

Strings are immutable and are of fixed size. To overcome this problem we use StringBuffer and StringBuilder.Both string buffer and string builder serve the same purpose i.e., to modify existing strings but the key difference between them is synchronization.StringBuffer is synchronized i.e., thread safe whereas StringBuilder is not synchronized, it is not thread safe.

## **65. Write a Java program to create an Enumeration and assign values using constructor**

## **CODE:**

**enum Apple**

**{**

**Jonathan(10),**

**GoldenDel(9),**

**RedDel(15),**

**Cortland(8);**

**private int price;**

**Apple(int p)**

**{**

**this.price = p;**

**}**

**int getPrice()**

**{**

**return price;**

**}**

**}**

**public class Main**

**{**

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

**{**

**Apple ap = Apple.GoldenDel;**

**System.out.println("GoldenDel costs: " + ap.getPrice());**

**System.out.println("All Values from Apple Enum are:");**

**for (Apple a : Apple.values())**

**{**

**System.out.println("Cost of " + a + " is: " + a.getPrice());**

**}**

**}**

**}**

## **OUTPUT:**

GoldenDel costs: 9

All Values from Apple Enum are:

Cost of Jonathan is: 10

Cost of GoldenDel is: 9

Cost of RedDel is: 15

Cost of Cortland is: 8

## **66. Write a java program to create a custom annotation or user defined annotation**

## **CODE:**

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

**@Target(ElementType.TYPE)**

**@Inherited**

**@Retention(RetentionPolicy.RUNTIME)**

**@interface MyAnnotation**

**{**

**String author() default "Ramesh Ponnala";**

**String course() default "OOP Java";**

**}**

**@MyAnnotation**

**class CustomAnnotation**

**{**

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

**{**

**MyAnnotation custom = CustomAnnotation.class.getAnnotation(MyAnnotation.class);**

**System.out.println("Annotated Author is: " + custom.author());**

**System.out.println("Annotated course is: " + custom.course());**

**}**

**}**

## **OUTPUT:**

Annotated Author is: Ramesh Ponnala

Annotated course is: OOP Java

## **67. Write a Java program to create file object and get its properties using file object**

## **CODE:**

**import java.io.File;**

**public class FilePropertiesDemo {**

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

**// Replace "path/to/your/file.txt" with the actual file path**

**String filePath = "./input.txt";**

**// Create a File object using the file path**

**File file = new File(filePath);**

**// Check if the file exists**

**if (file.exists()) {**

**System.out.println("File Name: " + file.getName());**

**System.out.println("Absolute Path: " + file.getAbsolutePath());**

**System.out.println("File Size: " + file.length() + " bytes");**

**System.out.println("Is Readable: " + file.canRead());**

**System.out.println("Is Writable: " + file.canWrite());**

**System.out.println("Is Executable: " + file.canExecute());**

**System.out.println("Is a Directory: " + file.isDirectory());**

**System.out.println("Is a File: " + file.isFile());**

**System.out.println("Last Modified: " + file.lastModified());**

**} else {**

**System.out.println("File does not exist.");**

**}**

**}**

**}**

## **OUTPUT:**

File Name: input.txt

Absolute Path: /config/workspace/./input.txt

File Size: 77 bytes

Is Readable: true

Is Writable: true

Is Executable: false

Is a Directory: false

Is a File: true

Last Modified: 1690094992000

## **68. Write a Java Program to display list of file names with specified extension using FilenameFilter**

## **CODE:**

**import java.io.File;**

**import java.io.FilenameFilter;**

**public class FileFilterDemo {**

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

**// Replace "path/to/your/directory" with the actual directory path**

**String directoryPath = "./";**

**// Replace "extension" with the desired file extension, for example, ".txt" or ".java"**

**String extension = ".java";**

**File directory = new File(directoryPath);**

**// Create a FilenameFilter to filter files by the specified extension**

**FilenameFilter filter = new FilenameFilter() {**

**@Override**

**public boolean accept(File dir, String name) {**

**return name.endsWith(extension);**

**}**

**};**

**// Get the list of files in the directory that match the extension**

**File[] fileList = directory.listFiles(filter);**

**if (fileList != null) {**

**System.out.println("List of files with extension '" + extension + "':");**

**for (File file : fileList) {**

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

**}**

**} else {**

**System.out.println("No files with extension '" + extension + "' found in the directory.");**

**}**

**}**

**}**

## **OUTPUT:**

List of files with extension '.java':

BufferedInputStreamDemo.java

EmpHash.java

BuffRead.java

ByteStreamDemo.java

……etc

## **69. Write a Java Program illustrating the Byte Stream to copy contents of one file to another file.**

## **CODE:**

**import java.io.FileInputStream;**

**import java.io.FileOutputStream;**

**import java.io.IOException;**

**public class ByteStreamDemo {**

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

**String inputFile = "input.txt";**

**String outputFile = "output.txt";**

**try (FileInputStream fis = new FileInputStream(inputFile);**

**FileOutputStream fos = new FileOutputStream(outputFile)) {**

**byte[] buffer = new byte[1024];**

**int bytesRead;**

**while ((bytesRead = fis.read(buffer)) != -1) {**

**fos.write(buffer, 0, bytesRead);**

**}**

**System.out.println("File copied successfully!");**

**} catch (IOException e) {**

**e.printStackTrace();**

**}**

**}**

**}**

## **OUTPUT:**

File copied successfully!

## **70. Write a Java Program illustrating the Character Stream to copy contents of one file to another file.**

## **CODE:**

**import java.io.FileReader;**

**import java.io.FileWriter;**

**import java.io.IOException;**

**public class CharStreamDemo {**

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

**String inputFile = "input.txt";**

**String outputFile = "output.txt";**

**try (FileReader reader = new FileReader(inputFile);**

**FileWriter writer = new FileWriter(outputFile)) {**

**char[] buffer = new char[1024];**

**int charsRead;**

**while ((charsRead = reader.read(buffer)) != -1) {**

**writer.write(buffer, 0, charsRead);**

**}**

**System.out.println("File copied successfully!");**

**}**

**catch (IOException e) {**

**e.printStackTrace();**

**}**

**}**

**}**

## **OUTPUT:**

File read and write successfully!

## **71. Write a Java program to read the file content and display using BufferedInputStream**

## **CODE:**

**import java.io.BufferedInputStream;**

**import java.io.FileInputStream;**

**import java.io.IOException;**

**public class BufferedInputStreamDemo {**

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

**String inputFile = "input.txt";**

**try (BufferedInputStream bis = new BufferedInputStream(new FileInputStream(inputFile))) {**

**int data;**

**while ((data = bis.read()) != -1) {**

**System.out.print((char) data);**

**}**

**} catch (IOException e) {**

**e.printStackTrace();**

**}**

**}**

**}**

## **OUTPUT:**

Sample text from the input file

## **72. Write a Java program to write the content into file using BufferedOutputStream**

## **CODE:**

**import java.io.BufferedOutputStream;**

**import java.io.FileOutputStream;**

**import java.io.IOException;**

**public class BufferedOutputStreamDemo {**

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

**String outputFile = "output.txt";**

**try (BufferedOutputStream bos = new BufferedOutputStream(new FileOutputStream(outputFile))) {**

**String content = "Hello, this content will be written to the file!";**

**bos.write(content.getBytes());**

**System.out.println("Content written to the file successfully!");**

**} catch (IOException e) {**

**e.printStackTrace();**

**}**

**}**

**}**

## **OUTPUT:**

Content written to the file successfully!

## **73. Write a Java program to read and write from file using BufferedReader and BufferedWriter**

## **CODE:**

**import java.io.BufferedReader;**

**import java.io.BufferedWriter;**

**import java.io.FileReader;**

**import java.io.FileWriter;**

**import java.io.IOException;**

**public class BuffRead {**

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

**// Define the paths of the input and output files**

**String inputFile = "input.txt"; // Replace with the path of the file you want to read from**

**String outputFile = "output.txt"; // Replace with the path of the file you want to write to**

**try {**

**// Create a BufferedReader to read from the input file**

**BufferedReader reader = new BufferedReader(new FileReader(inputFile));**

**// Create a BufferedWriter to write to the output file**

**BufferedWriter writer = new BufferedWriter(new FileWriter(outputFile));**

**// Read each line from the input file and write it to the output file**

**String line;**

**while ((line = reader.readLine()) != null) {**

**writer.write(line); // Write the current line to the output file**

**writer.newLine(); // Move to the next line in the output file**

**}**

**// Close the resources to free up system memory**

**reader.close();**

**writer.close();**

**// Print a success message**

**System.out.println("File read and write successful!");**

**} catch (IOException e) {**

**// If an error occurs during file read/write, print the error trace**

**e.printStackTrace();**

**}**

**}**

**}**

## **OUTPUT:**

File read and write successful!

## **74. Write a Java program to demonstrate serialization and deserialization**

## **CODE:**

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

**class Student implements Serializable {**

**private static final long serialVersionUID = 1L;**

**private String name;**

**private int age;**

**private String course;**

**public Student(String name, int age, String course) {**

**this.name = name;**

**this.age = age;**

**this.course = course;**

**}**

**@Override**

**public String toString() {**

**return "Student{" +**

**"name='" + name + '\'' +**

**", age=" + age +**

**", course='" + course + '\'' +**

**'}';**

**}**

**}**

**public class SerialDeSerial{**

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

**// Serialization**

**try (FileOutputStream fileOut = new FileOutputStream("student.ser");**

**ObjectOutputStream objectOut = new ObjectOutputStream(fileOut)) {**

**Student student = new Student("John Doe", 25, "Computer Science");**

**objectOut.writeObject(student);**

**System.out.println("Object serialized and stored in student.ser");**

**} catch (IOException e) {**

**e.printStackTrace();**

**}**

**// Deserialization**

**try (FileInputStream fileIn = new FileInputStream("student.ser");**

**ObjectInputStream objectIn = new ObjectInputStream(fileIn)) {**

**Student student = (Student) objectIn.readObject();**

**System.out.println("Object deserialized: " + student);**

**} catch (IOException | ClassNotFoundException e) {**

**e.printStackTrace();**

**}**

**}**

**}**

## **OUTPUT:**

Object serialized and stored in student.ser

Object deserialized: Student{name='John Doe', age=25, course='Computer Science'}

## **75. Write a Java program to create a list of Employees Information using ArrayList**

## **CODE:**

**import java.util.ArrayList;**

**// import java.util.List;**

**class Emp{**

**String name;**

**double sal;**

**Emp(String n, double s){name =n;sal =s;}**

**public void dis(){**

**System.out.println("name : "+name+"\nSalary : "+sal);**

**}**

**}**

**public class ArrayListDemo {**

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

**ArrayList<Emp> empList = new ArrayList<>();**

**Emp e1 = new Emp("maneesh", 500);**

**Emp e2 = new Emp("vamshi", 660);**

**Emp e3 = new Emp("raki", 688);**

**empList.add(e1);**

**empList.add(e2);**

**empList.add(e3);**

**for (Emp employee : empList) {**

**employee.dis();**

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

**}**

**}**

**}**

## **OUTPUT:**

name : maneesh

Salary : 500.0

name : vamshi

Salary : 660.0

name : raki

Salary : 688.0

## **76. Write a Java program to demonstrate LinkedList**

## **CODE:**

**import java.util.Iterator;**

**import java.util.LinkedList;**

**public class LinkedListDemo {**

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

**// Create a LinkedList to store integers**

**LinkedList<Integer> myLinkedList = new LinkedList<>();**

**// Adding elements to the LinkedList**

**myLinkedList.add(10);**

**myLinkedList.add(20);**

**myLinkedList.add(30);**

**myLinkedList.add(40);**

**// Displaying the LinkedList elements**

**System.out.println("LinkedList elements: " + myLinkedList);**

**// Adding elements at specific positions in the LinkedList**

**myLinkedList.add(1, 15); // Add 15 at index 1**

**myLinkedList.addFirst(5); // Add 5 at the beginning**

**myLinkedList.addLast(50); // Add 50 at the end**

**// Displaying the updated LinkedList elements**

**System.out.println("Updated LinkedList elements: " + myLinkedList);**

**// Accessing elements in the LinkedList**

**System.out.println("Element at index 2: " + myLinkedList.get(2));**

**System.out.println("First element: " + myLinkedList.getFirst());**

**System.out.println("Last element: " + myLinkedList.getLast());**

**// Removing elements from the LinkedList**

**myLinkedList.removeFirst(); // Remove the first element**

**myLinkedList.removeLast(); // Remove the last element**

**myLinkedList.remove(2); // Remove element at index 2**

**// Displaying the final LinkedList elements**

**System.out.println("Final LinkedList elements: " + myLinkedList);**

**// Size of the LinkedList**

**System.out.println("Size of the LinkedList: " + myLinkedList.size());**

**System.out.println("LinkedList elements using iterator:");**

**Iterator<Integer> iterator = myLinkedList.iterator();**

**while (iterator.hasNext()) {**

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

**}**

**}**

**}**

## **OUTPUT:**

LinkedList elements: [10, 20, 30, 40]

Updated LinkedList elements: [5, 10, 15, 20, 30, 40, 50]

Element at index 2: 15

First element: 5

Last element: 50

Final LinkedList elements: [10, 15, 30, 40]

Size of the LinkedList: 4

LinkedList elements using iterator:

10

15

30

40

## **77. Write a Java program to demonstrate HashSet**

## **CODE:**

**import java.util.HashSet;**

**import java.util.Iterator;**

**public class HashSetDemo {**

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

**// Create a HashSet to store strings**

**HashSet<String> myHashSet = new HashSet<>();**

**// Adding elements to the HashSet**

**myHashSet.add("Apple");**

**myHashSet.add("Banana");**

**myHashSet.add("Orange");**

**myHashSet.add("Grapes");**

**// Displaying the HashSet elements**

**System.out.println("HashSet elements: " + myHashSet);**

**// Checking if an element exists in the HashSet**

**System.out.println("is empty ? ->"+ myHashSet.isEmpty());**

**System.out.println("contains Apple ?->"+ myHashSet.contains("Apple"));**

**System.out.println("HashSet elements using iterator:");**

**Iterator it = myHashSet.iterator();**

**while (it.hasNext()) {**

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

**}**

**}**

**}**

## **OUTPUT:**

HashSet elements: [Apple, Grapes, Orange, Banana]

is empty ? ->false

contains Apple ?->true

HashSet elements using iterator:

Apple

Grapes

Orange

Banana

## **78. Write a Java program to demonstrate TreeSet**

## **CODE:**

**import java.util.TreeSet;**

**public class TreeSetDemo {**

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

**// Create a TreeSet to store integers in sorted order**

**TreeSet<Integer> numbersSet = new TreeSet<>();**

**// Add elements to the TreeSet**

**numbersSet.add(5);**

**numbersSet.add(2);**

**numbersSet.add(8);**

**numbersSet.add(1);**

**numbersSet.add(7);**

**// Print the elements in sorted order**

**System.out.println("Sorted TreeSet: " + numbersSet);**

**Integer lowerValue = numbersSet.lower(5);**

**System.out.println("Value strictly less than 5: " + lowerValue);**

**Integer higherValue = numbersSet.higher(5);**

**System.out.println("Value strictly greater than 5: " + higherValue);**

**}**

**}**

## **OUTPUT:**

Sorted TreeSet: [1, 2, 5, 7, 8]

Value strictly less than 5: 2

Value strictly greater than 5: 7

## **79. Write a Java program to define a HashMap which maps to employee names to their salary**

## **CODE :**

**import java.util.HashMap;**

**import java.util.Map;**

**import java.util.Scanner;**

**public class EmpHash {**

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

**// Create a HashMap to store employee names and their salaries**

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

**Map<String, Double> employeeSalaries = new HashMap<>();**

**char ch;**

**// Add employee names and salaries to the HashMap**

**while (true) {**

**System.out.println("enter emp name and salary");**

**employeeSalaries.put(sc.next(), sc.nextDouble());**

**System.out.println("wanna add another record?[y/n] :");**

**ch = sc.next().charAt(0); //taking string and considering 1st char**

**if(ch=='y')continue;**

**else if(ch=='n')break;**

**else continue;**

**}**

**System.out.println("\nEmployee Salaries:");**

**//printing keys and values**

**for ( String empName : employeeSalaries.keySet() ) {**

**Double value = employeeSalaries.get(empName);**

**System.out.println(empName + ": " + value);**

**}**

**}**

**}**

## **OUTPUT :**

enter emp name and salary

man 300

wanna add another record?[y/n] :

y

enter emp name and salary

sam 500

wanna add another record?[y/n] :

n

Employee Salaries:

man: 300.0

sam: 500.0

## 