1) EXCEPTIONAL HANDLING

51. Write a Java program to handle arithmetic exception and number format exception

```
import java.util.Scanner;

public class Program51
{
   public static int division(int a, int b)
   {
      return a / b;
   }
   public static void main(String[] args)
   {
      Scanner sc = new Scanner(System.in);
      try
      {
         System.out.print("Enter a number: ");
         int n1 = Integer.parseInt(sc.nextLine());

         System.out.print("Enter another number: ");
         int n2 = Integer.parseInt(sc.nextLine());

         int res = division(n1, n2);

         System.out.println("Result: " + res);
      }
      catch (NumberFormatException e)
      {
         System.out.println("message :"+e.getMessage());
      }
```

```
        catch (ArithmeticException e)
        {
            System.out.println("message :"+e.getMessage());
        }
    }
}
```

OUTPUT:

```
Enter a number: 2.5
message :For input string: "2.5"
(OR)
Enter a number: 5
Enter another number: 0
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

```
import java.util.Scanner;

public class Program52
{
    public static void main(String[] args)
    {
        Scanner sc = new Scanner(System.in);
        try
        {
            int[] n = { 1, 2, 3, 4, 5 };

            System.out.print("Enter an index: ");
            int index = Integer.parseInt(sc.nextLine());

            int element = n[index];
            System.out.println("Element at index " + index + ": " +
element);
        }
        catch (NumberFormatException e)
        {
              System.out.println("Message: "+e.getMessage());
        }
        catch (ArrayIndexOutOfBoundsException e)
        {
            System.out.println("Message: "+e.getMessage());
        }
    }
}
```

OUTPUT:

```
Enter an index: 3
Element at index 3: 4
(OR)
Enter an index: 7
Index 7 out of bounds for length 5
(OR)
Enter an index: 2.3
For input string: "2.3"
```

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

```
class InsufficientFundsException extends Exception
{
    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 Program53
{
    public static void main(String[] args)
    {
        BankAccount acc = 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
```

2) MULTI THREADING

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

```
public class Program54 extends Thread
{
        Program54()
        {
                super();
        }
        Program54(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());
		Program54 m=new Program54("MyThread-0");
		m.start();
		System.out.println("Hello World! from Main Thread");
		Program54 m1=new Program54("MyThread-1");
		Program54 m2=new Program54("MyThread-2");
		m1.start();
		m2.start();
	}
}
```

OUTPUT:

```
main : 5
Hello World! from Main Thread
MyClass Run method: MyThread-0
MyClass Run method: MyThread-2
MyClass Run method: MyThread-1
```

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

```
class MyThread implements Runnable
{
	String s=null;
	MyThread(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 Program55
{
    public static void main(String args[])
    {
        MyThread m=new MyThread("Thread started....");
        Thread t=new Thread(m);
        t.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

```
class Even extends Thread
{
        public void run()
        {
                for(int i=2;i<=6;i+=2)
                {
                        System.out.println("Even "+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("Odd "+j);
                }
                System.out.println("End of thread Odd");
        }
}
public class Program56
{
        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
Even 2
Even 4
Even 6
End of thread Even
Odd 1
Odd 3
Odd 5
End of thread Odd
```

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

```
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 MyBank extends Thread
{
        BankAccount bankAcc;
        MyBank()
        {
                super();
        }
        MyBank(String s, BankAccount ba)
        {
                super(s);
                bankAcc=ba;
        }
        public void run()
        {
                System.out.println(this.getName());
                bankAcc.withdraw(75);
        }
}
class Program57
{
        public static void main(String[] args)
        {
                BankAccount bankAcc=new BankAccount(100);
                System.out.println("Initial Bank Balance is:
"+bankAcc.getBalance());
```

```
        MyBank wBank=new MyBank("Wife",bankAcc);
        wBank.start();
        MyBank hBank=new MyBank("Husband",bankAcc);
        hBank.start();
    }
}
```

OUTPUT:

```
Initial Bank Balance is: 100.0
Wife
Husband
Amount withdrawn is 75.0 and the remaining bal is: 25.0
Amount withdrawn is 75.0 and the remaining bal is: -50.0
```

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

```
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..");
		}
		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(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 Program58
{
        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 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

```
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(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 Program59
{
        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
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..
```

3) INTER THREAD-COMMUNICATION

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

```
class item
{
   int n;
   boolean valueSet = false;

   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();
    }
    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;
    }
}
class Producer implements Runnable
{
    item q;
    Producer(item q)
    {
        this.q = q;
        new Thread(this, "Producer").start();
    }

    public void run()
    {
        int i = 0;
        while (i<=5)
        {
            q.put(i++);
        }
    }
}
class Consumer implements Runnable
{
    item q;

    Consumer(item q)
    {
        this.q = q;
        new Thread(this, "Consumer").start();
    }

    public void run()
    {
        int i=0;
        while (i<=5)
        {
            q.get();
            i++;
        }
    }
}
class Program60
{
    public static void main(String[] args)
    {
        item q = new item();
        new Producer(q);
        new Consumer(q);
    }
}
```

OUTPUT:

```
Produced: 0
Consumed:0
Produced: 1
Consumed:1
Produced: 2
Consumed:2
Produced: 3
Consumed:3
Produced: 4
Consumed:4
Produced: 5
Consumed:5
```

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

```
class Printno
{
    int maxNumber;
    int currentNumber = 1;
```

```
boolean ThreadTurn = true;

Printno(int maxNumber)
{
    this.maxNumber = maxNumber;
}

public synchronized void printEven()
{
    while (currentNumber < maxNumber)
    {
        while (ThreadTurn)
        {
            try
            {
                wait();
            }
            catch (InterruptedException e)
            {
                e.printStackTrace();
            }
        }
        System.out.println("Even Thread:" + currentNumber);
        currentNumber++;

        ThreadTurn = true;
        notify();
    }
}

public synchronized void printOdd()
{
        while (currentNumber < maxNumber)
        {
            while (!ThreadTurn)
            {
                try
                {
                    wait();
                }
                catch (InterruptedException e)
                {
                    e.printStackTrace();
                }
            }
            System.out.println("Odd Thread:" + currentNumber);
            currentNumber++;

            ThreadTurn = false;
            notify();
        }
    }
}
public class Program61
{
    public static void main(String[] args)
    {
        int maxNumber = 10;
        Printno p = new Printno(maxNumber);

        Thread evenThread = new Thread(() -> {
            p.printEven();
        });
        Thread oddThread = new Thread(() -> {
            p.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
```

4) STRINGS IN JAVA

62. Write a Java program to perform various String operations

```
public class Program62
{
    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());

        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);
    }
}
```

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
int i as toString: 12345
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
```

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

```
class Program63
{
   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);

    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, 31);
    System.out.println(s);

    //deleteCharAt(index)
    s.deleteCharAt(7);
    System.out.println(s);

    //replace(start_index,end_index,string)
    s.replace(0,18,"Mahatma Gandhi ");
    System.out.println(s);
}
```

```
}
```

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 Institute Technology
Chaitana Bharathi Institute Technology
Mahatma Gandhi Institute Technology
```

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

```
public class Program63
{
   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
```

5) ENUMERATION

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

```
enum Apple
{
    PineApple(10),
    GreenApple(9),
    RedApple(15);

    private int price;

    Apple(int p)
    {
        this.price = p;
    }

    int getPrice()
    {
        return price;
    }
}
public class Program64
{
   public static void main(String[] args)
   {
      System.out.println("All Values from Apple Enum are:");
      for (Apple a : Apple.values())
      {
        System.out.println("Cost of " + a + " is: " + a.getPrice());
      }
   }
}
```

OUTPUT:

```
All Values from Apple Enum are:
Cost of PineApple is: 10
Cost of GreenApple is: 9
Cost of RedApple is: 15
```

6) ANNOTATION

66. WJP to create a custom annotation or user defined annotation

```
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 Program65
 {
   public static void main(String[] args)
 {
       MyAnnotation custom =
Program65.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
```

7) FILE HANDLING

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

```
import java.io.File;
import java.io.IOException;

public class Program67
{
        public static void main(String[] args)
        {
        File myfile = new File("trial.txt");
                try
                {
                        myfile.createNewFile();
                }
        catch(IOException a)
        {
            System.out.println("Unable to create the file");
            a.printStackTrace();
        }
        if (myfile.exists())
        {
            System.out.println("File Name: " + myfile.getName());
            System.out.println("Absolute Path: " +
myfile.getAbsolutePath());
            System.out.println("File Size: " + myfile.length() + "
bytes");
            System.out.println("Is Readable: " +
myfile.canRead());
            System.out.println("Is Writable: " +
myfile.canWrite());
            System.out.println("Is Executable: " +
myfile.canExecute());
            System.out.println("Is a Directory: " +
myfile.isDirectory());
            System.out.println("Is a File: " + myfile.isFile());
            System.out.println("Last Modified: " +
myfile.lastModified());
    }
        else
        {
            System.out.println("File does not exist.");
        }
    }
}
```
OUTPUT:

```
File Name: trial.txt
```

```
Absolute Path: C:\Users\Sushma\Desktop\Java Lab Practice\trial.txt
File Size: 26 bytes
Is Readable: true
Is Writable: true
Is Executable: true
Is a Directory: false
Is a File: true
Last Modified: 1690300679304
```

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

```
import java.io.File;
import java.io.FilenameFilter;

public class Program68
{
        public static void main(String[] args)
        {
                String path = "E:/CBIT/MCA SEM-2/OOPJ/Practical/Java
Programs";
        String extension = ".java";
                File f = new File(path);
        FilenameFilter fnf = new FilenameFilter()
                {
                        @Override
                        public boolean accept(File dir, String name)
                        {
                        return name.endsWith(extension);
                        }
        };
                File[] lst = f.listFiles(fnf);
        if (lst != null)
```

```
                {
                        System.out.println("List of files with extension " +
extension + ":");
                        for (File x : lst)
                        {
                        System.out.println(x.getName());
                        }
        }
                else
                        System.out.println("No files with extension '" +
extension + "' found in the directory.");
        }
}
```

OUTPUT:

```
List of files with extension .java:
AbstractDemo.java
Anonymous.java
AnonymousInnerClass.java
Armstrong.java
BankAccountThread.java
BinarySearch.java
```

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

```
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;

class Program69
{
   public static void main(String args[])
   {
        try
        {
           FileInputStream fis = new FileInputStream("input.txt");
           byte[] b = new byte[1024];
           int byteRead = fis.read(b);
           fis.close();

           FileOutputStream fos = new FileOutputStream("output.txt");
           fos.write(b,0,byteRead);
           fos.close();

            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.

```
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;

class Program70
{
    public static void main(String args[])
```

```
   {
      try
      {
         FileReader fis = new FileReader("give.txt");
         char[] b  new char[1024];
         int charRead = fis.read(b);
         fis.close();

         FileWriter fos = new FileWriter("take.txt");
         fos.write(b, 0, charRead);
         fos.close();

         System.out.println("file copied successfully");
      }
      catch(IOException e)
      {
         e.printStackTrace();
      }
   }
}
```
OUTPUT:
```
file copied successfully
```

8) BUFFER RELATED FILE HANDLING

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

```
import java.io.FileInputStream;
import java.io.BufferedInputStream;
import java.io.IOException;
```

```
class Program71
{
   public static void main(String args[])
   {
      try
      {
         BufferedInputStream b1 = new BufferedInputStream(new
FileInputStream("sample.txt"));
         int ch;
         while((ch=b1.read())!=-1)
            System.out.print((char)ch);
      }
      catch(IOException e)
      {
         e.printStackTrace();
      }
   }
}
```

OUTPUT:

```
tomorrow should be holiday
```

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

```
import java.io.FileOutputStream;
import java.io.BufferedOutputStream;
import java.io.IOException;

class Program72
{
   public static void main(String args[])
   {
        try
        {
           BufferedOutputStream b1 = new BufferedOutputStream(new
FileOutputStream("sample.txt"));
           String s = "tomorrow should be holiday";
           b1.write( s.getBytes());
           b1.close();
           System.out.println("content written to file successfully");
        }
        catch(IOException e)
        {
           e.printStackTrace();
        }
    }
}
```

OUTPUT:

```
content written to file successfully
```

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

```
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;

public class Program73
{
    public static void main(String[] args)
    {
           try
```

```
        {
            BufferedReader r = new BufferedReader(new
FileReader("input.txt"));
            BufferedWriter w = new BufferedWriter(new
FileWriter("output.txt"));

            String line;
            while ((line = r.readLine()) != null)
            {
                w.write(line);
                w.newLine();
            }
            r.close();
            w.close();
            System.out.println("File read and write successfully!");
        }
        catch (IOException e)
        {
            e.printStackTrace();
        }
    }
}
```

OUTPUT:

```
File read and write successfully!
```

9) SERIALIZATION AND DESERIALIZATION

74. Write a Java program to demonstrate serialization and deserialization

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

```
class Student implements Serializable
{
   String name;
   int age;
   String course;

   Student(String name, int age, String course)
   {
      this.name = name;
      this.age = age;
      this.course = course;
   }
   public void display()
   {
      System.out.println("Name: " + name);
      System.out.println("Age: " + age);
      System.out.println("Course: " + course);
   }
}

public class Program74
{
   public static void main(String[] args)
   {
      // Serialization
      try
      {
         FileOutputStream fileOut = new
FileOutputStream("student.ser");
         ObjectOutputStream objectOut = new
ObjectOutputStream(fileOut);
```

```
          Student obj = new Student("John Doe", 25, "Computer
Science");
          objectOut.writeObject(obj);
          System.out.println("Object serialized and stored in
student.ser");

          objectOut.close();
          fileOut.close();
        }
        catch (IOException e)
        {
          e.printStackTrace();
        }

        // Deserialization
        try
        {
          FileInputStream fileIn = new FileInputStream("student.ser");
          ObjectInputStream objectIn = new
ObjectInputStream(fileIn);

          Student obj = (Student) objectIn.readObject();
          System.out.println("Object deserialized:");
          obj.display();

          objectIn.close();
          fileIn.close();
        }
        catch (IOException | ClassNotFoundException e)
        {
          e.printStackTrace();
        }
```

```
    }
}
```

OUTPUT:

```
Object serialized and stored in student.ser
Object deserialized:
Name: John Doe
Age: 25
Course: Computer Science
```

10) ARRAYLIST

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

```
import java.util.ArrayList;

class Emp
{
    String name;
    double sal;

    Emp(String x, double y)
    {
        name = x;
        sal = y;
    }
    public void disp()
    {
        System.out.println("Name: "+name+"\tSalary: "+sal);
    }
}
public class Program75
```

```
{
   public static void main (String[] args)
   {
      ArrayList<Emp> obj = new ArrayList<>();

      obj.add(new Emp("Sushma",500));
      obj.add(new Emp("Maneesh",800));
      obj.add(new Emp("Laxmi",650));

      for(Emp x: obj)
         x.disp();
   }
}
```

OUTPUT:

```
Name: Sushma     Salary: 500.0
Name: Maneesh    Salary: 800.0
Name: Laxmi      Salary: 650.0
```

11) LINKEDLIST

76. Write a Java program to demonstrate LinkedList

```
import java.util.LinkedList;


public class Program76
{
   public static void main (String[] args)
   {
      LinkedList<Integer> obj = new LinkedList<>();


      obj.add(10);
```

```
        obj.add(20);
        obj.add(30);
        obj.add(40);
        System.out.println("linkedList Elements: "+obj);

        obj.add(1,15);
        obj.addFirst(5);
        obj.addLast(50);
        System.out.println("linkedList after adding elements: "+obj);

        System.out.println("linkedList element at 2 index: "+obj.get(2));
        System.out.println("linkedList first elements: "+obj.getFirst());
        System.out.println("linkedList last elements: "+obj.getLast());

        obj.remove(2);
        obj.removeFirst();
        obj.removeLast();
        System.out.println("linkedList after removing elements: "+obj);

        System.out.println("linkedList size: "+obj.size());
    }
}
```

OUTPUT:

```
linkedList Elements: [10, 20, 30, 40]
linkedList after adding elements: [5, 10, 15, 20, 30, 40, 50]
linkedList element at 2 index: 15
linkedList first elements: 5
linkedList last elements: 50
linkedList after removing elements: [10, 20, 30, 40]
linkedList size: 4
```

12) HASHSET

77. Write a Java program to demonstrate HashSet

```
import java.util.HashSet;
import java.util.Iterator;

public class Program77
{
   public static void main (String[] args)
   {
      HashSet<String> obj = new HashSet<>();

      obj.add("Apple");
      obj.add("Orange");
      obj.add("Banana");
      obj.add("Grapes");

      System.out.println("Hashset Elements: "+obj);

      System.out.println("is it empty? "+obj.isEmpty());
      System.out.println("contains apple? "+obj.contains("Apple"));

      System.out.println("Hashset elements using Iterator");

      Iterator it = obj.iterator();
      while(it.hasNext())
         System.out.println(it.next());
   }
}
```

OUTPUT:

```
Hashset Elements: [Apple, Grapes, Orange, Banana]
is it empty? false
contains apple? true
Hashset elements using Iterator
Apple
Grapes
Orange
Banana
```

13) TREESET

78. Write a Java program to demonstrate TreeSet

```
import java.util.TreeSet;

public class Program78
{
   public static void main (String[] args)
   {
      TreeSet<Integer> obj = new TreeSet<>();

      obj.add(5);
      obj.add(8);
      obj.add(0);
      obj.add(2);
      obj.add(1);

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

      System.out.println("Value Strictly lower than 5: "+obj.lower(5));
      System.out.println("Value Strictly higher than 5: "+obj.higher(5));
   }
}
```

OUTPUT:

```
Sorted TreeSet: [0, 1, 2, 5, 8]
Value Strictly lower than 5: 2
Value Strictly higher than 5: 8
```

14) HASHMAP

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

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

public class Program79
{
   public static void main (String[] args)
   {
      Scanner sc = new Scanner(System.in);
      HashMap<String,Double> obj = new HashMap<>();
      char ch;
      while(true)
      {
         System.out.print("Enter Name and Salary: ");
         obj.put(sc.next(),sc.nextDouble());

         System.out.print("Wanna add another record? [y/n]: ");
         ch = sc.next().charAt(0);
         if(ch == 'y' || ch == 'Y')
            continue;
         else
```

```
            break;
        }
        System.out.println("Employees Details");
        for(String x: obj.keySet())
        {
           double y = obj.get(x);
           System.out.println(x+":"+y);
        }
    }
}
```

OUTPUT:

```
Enter Name and Salary: Sushma 250000
Wanna add another record? [y/n]: y
Enter Name and Salary: Maneesh  50000
Wanna add another record? [y/n]: y
Enter Name and Salary: Laxmi  180000
Wanna add another record? [y/n]: n
Employees Details
Sushma:250000.0
Laxmi:180000.0
Maneesh:50000.0
```