	b)	A file contains 5 integers. Write a program to fetch the numbers from the file, and then calculate their sum, and append its sum to the file.	L3	CO3	5 M
		OR			
9	a)	Write a program in Java to demonstrate the use of try, catch, and finally blocks.	L2	CO3	5 M
	b)	What are checked and unchecked exceptions? Explain with examples.	L2	CO3	5 M
		UNIT-V			
10	a)	Write a Java program to create multiple threads by extending the Thread class.	L3	CO4	5 M
	b)	Explain thread life cycle with a neat sketch.	L2	CO4	5 M
		OR			
11	a)	Explain how to insert and delete elements from a list collection with an example.	L2	CO4	5 M
	b)	Explain about hierarchy of collection interfaces.	L2	CO4	5 M

PVP 23

Code: 23CS3302, 23IT3302, 23AM3302, 23DS3302

II B.Tech - I Semester – Regular / Supplementary Examinations
NOVEMBER 2025

OBJECT ORIENTED PROGRAMMING THROUGH JAVA

(Common for CSE, IT, AIML, DS)

Duration: 3 hours Max. Marks: 70

Note: 1. This question paper contains two Parts A and B.
2. Part-A contains 10 short answer questions. Each Question carries 2 Marks.
3. Part-B contains 5 essay questions with an internal choice from each unit. Each Question carries 10 marks.
4. All parts of Question paper must be answered in one place.

BL – Blooms Level CO – Course Outcome

PART – A

		BL	CO
1.a)	List any four Java buzzwords and explain their importance.	L2	CO1
1.b)	Differentiate between while and do-while loops.	L2	CO1
1.c)	Define the terms polymorphism and Encapsulation.	L2	CO1
1.d)	Explain any two methods to extract the characters from string.	L2	CO1
1.e)	What is an abstract class? How is it different from an interface?	L2	CO1
1.f)	How to declare and initialize a two-dimensional array?	L2	CO1
1.g)	Differentiate between throw and throws in exception handling.	L2	CO1

1.h)	Explain the use of the finally block in exception handling.	L2	CO1
1.i)	What is thread priority in Java? How is it set?	L2	CO1
1.j)	Distinguish between a thread and a process.	L2	CO1

PART – B

			BL	CO	Max. Marks
		UNIT-I			
2	a)	What are literals and symbolic constants in Java? Illustrate with examples.	L2	CO1	5 M
	b)	Write a Java program using nested if-else statements to determine the largest of three numbers.	L2	CO1	5 M
		OR			
3	a)	What are escape sequences in Java? Write a program that uses escape sequences to format output.	L2	CO1	3 M
	b)	Develop a program that calculates grades based on the marks for the following ranges: Range — Grade 90-100 — S 80-89 — A 70-79 — B 60-69 — C 50-59 — D 40-49 — E Less than 40 — Fail	L3	CO1	7 M

		UNIT-II			
4	a)	Define classes and objects in Java. Write a program to create a class Student with attributes and methods to display details.	L3	CO2	5 M
	b)	Write a program to count the number of vowels, consonants, spaces, digits and special characters in a string.	L3	CO2	5 M
		OR			
5	a)	What is a final method? How does it differ from a normal method?	L2	CO2	5 M
	b)	Develop a program to demonstrate how to access private members of a class.	L2	CO2	5 M
		UNIT-III			
6	a)	Explain the use of the super keyword in inheritance.	L2	CO2	5 M
	b)	Differentiate between method overloading and method overriding with suitable examples.	L2	CO2	5 M
		OR			
7	a)	Explain the concept of extending an interface with a suitable example.	L2	CO2	5 M
	b)	Write a Java program to accept a matrix of order NxN and interchange the diagonals.	L3	CO2	5 M
		UNIT-IV			
8	a)	Explain the behavior of different access specifiers in packages with examples.	L2	CO3	5 M

PRASAD V POTLURI SIDDHARTHA INSTITUTE OF TECHNOLOGY
(AUTONOMOUS), KANURU, VIJAYAWADA
II B.Tech – I Semester – Regular/supplementary Examinations, NOV 2025
OBJECT ORIENTED PROGRAMMING THROUGH JAVA – 23CS3302(PVP23)
SCHEME OF VALUATION

PART-A

1 a) List any four buzzwords and explain their importance (2 M)
for each buzzword 0.5 mark

1 b) Differentiate between while and do-while loops.(2M)

for each difference 1 mark

1 c) Define the terms polymorphism and Encapsulation.(2M)

for definition of encapsulation – 1 mark

for definition of polymorphism – 1 mark

1 d) Explain any two methods to extract the characters from string. (2M)

for each method 1 mark

1 e) What is an abstract class? How is it different from an interface? (2M)

for definition of abstract class – 1 mark

for any one difference – 1 mark

1 f) How to declare and initialize a two-dimensional array? (2M)

correct declaration and initialization – 2 marks

partial correct – 1 mark

1 g) Differentiate between throw and throws in exception handling.(2M)

for any difference --> 2 marks

1 h) Explain the use of finally block in exception handling. (2M)

for correct usage --> 2 marks
for partial correct -> 1 mark

1 i) What is thread priority in java? How is it set? (2M)
purpose of thread priority -> 1 mark
process for setting priority -> 1 mark

1 j) Distinguish between a thread and a process. (2M)
for each difference 1 mark

PART-B

2 a) What are literals and Symbolic constants in Java? Illustrate with examples. (5 M)
for literal and symbolic constants --> 3 marks
for Examples -> 2 marks

b) Write a java program using nested if-else statements to determine the largest of three numbers. (5 M)
for correct program --> 5 marks
if it is partial correct --> 3 marks

3 a) What are escape sequences in java? Write a program that uses escape sequences to format output. (3 M)
Escape sequences --> 2 marks
Example --> 1 mark

b) Develop a Program that calculate grades based on the marks for the following range:(7 M)

Range	Grade
90-100	S
80-89	A
70-79	B
60-69	C
50-59	D
40-49	E
Less than 40	F

For correct program --> 7 marks
For partial correct like incorrect syntax/ logic error ---> 3 marks

4 a) Define class and Object in Java. Write a program to create a class student with attributes and methods to display details. (5M)
for definition of class and object --> 2 marks
for example ---> 3 marks

b) Write a program to count the number of vowels, consonants, spaces, digits, and special character in a string. (5M)
for correct program --> 5 marks
for partial correct --> 3 marks

5 a) What is a final method? How does it differ from a normal method? (5M)
for definition --> 1 mark
for difference with example --> 4 marks

b) Develop a program to demonstrate how to access private members of a class. (5 M)
for correct program ---> 5 marks
for partial correct --> 3 marks

6 a) Explain the use of super keyword in inheritance. (5M)
for demonstration with example --> 5 marks
for demonstration without example --> 2 marks

b) Differentiate between method overloading and method overriding with suitable example. (5M)
for differences --> 2 marks
for example --> 3 marks

7 a) Explain the concept of extending an interface with a suitable example (5M)
for demonstration with example --> 5 marks
for demonstration without example --> 2 marks

b) Write a Java Program to accept a matrix of order NxN and interchange the diagonals. (5M)
for correct program --> 5 marks
for partial correct program --> 3 marks

8 a) Explain the behavior of different access specifiers in packages with examples. (5 M)
for correct program --> 5 marks
for partial correct program --> 3 marks

b) A file contains 5 integers . Write a program to fetch the numbers from the file, and then calculate their sum, and append its sum to the file. (5M)
for correct program --> 5 marks
for partial correct program --> 3 marks

9 a) Write a program in java to demonstrate the use of try, catch and finally blocks. (5 M)
for correct program --> 5 marks
for partial correct program --> 3 marks

b) What are checked and unchecked exceptions? Explain with examples. (5 M)
for definitions --> 2 marks
for demonstration with examples --> 3 marks

10 a) Write a Java program to create multiple threads by extending the thread class. (5 M)
for correct program --> 5 marks
for partial correct program --> 3 marks
b) Explain Thread life cycle with a neat sketch. (5 M)
for diagram ---> 4 marks
for explanation of states --> 1 mark

11 a) Explain how to insert and delete elements from a list collection with an example. (5 M)

for correct program --> 5 marks
for partial correct program --> 3 marks

b) Explain about hierarchy of collection interfaces.(5 M)

for hierarchy of classes and interfaces --> 5 marks

PRASAD V POTLURI SIDDHARTHA INSTITUTE OF TECHNOLOGY
(AUTONOMOUS), KANURU, VIJAYAWADA
II B.Tech – I Semester – Regular/supplementary Examinations, NOV 2025
OBJECT ORIENTED PROGRAMMING THROUGH JAVA – 23CS3302(PVP23)
DETAILED SOLUTIONS

PART-A

1 a) List any four buzzwords and explain their importance (2 M)

Simple:
Java's design aims for ease of learning and use, simplifying complex features found in other languages.
Object-Oriented:
Java is built upon object-oriented principles, utilizing classes, objects, inheritance, polymorphism, and encapsulation.
Platform Independent (or Architecture Neutral):
Java code is compiled into bytecode, which can run on any system with a Java Virtual Machine (JVM), regardless of the underlying hardware or operating system.
Secure:
Java incorporates built-in security features to protect against malicious code and ensure safe execution, especially in networked environments.

1 b) Differentiate between while and do-while loops.(2M)

while loop	**do-while loop**

• Entry-Controlled loop
• The while loop checks its condition before executing the loop's body.
• If the condition is initially false, the loop's body will not execute even once.
• The syntax for a while loop is:
While(condition) {
// body of the loop
}
	• Exit-Controlled loop
	• The do-while loop executes its body at least once before checking the condition.
	• The condition is checked after the first execution of the loop's body.
	• If the condition is initially false, the loop's body will still execute once.
	• The syntax for a do-while loop is:
	do{
	// body of the loop
	}while(condition);

1 c) Define the terms polymorphism and Encapsulation.(2M)

The dictionary definition of *polymorphism* refers to a principle in biology in which an organism or species can have many different forms or stages. This principle can also be applied to object-oriented programming and languages like the Java language. Subclasses of a class can define their own unique behaviors and yet share some of the same functionality of the parent class.

Encapsulation is the process of bundling data (variables) and the methods that operate on that data into a single unit, known as a class. It primarily focuses on restricting direct access to the internal state of an object and providing controlled access through well-defined public method.

1 d) Explain any two methods to extract the characters from string. (2M)

charAt(i) --> extracts single character at index i

substring(start,end)--> extracts a group of characters from index start to index end-1.

1 e) What is an abstract class? How is it different from an interface? (2M)

An abstract class is a class that can contain both abstract methods (without implementation) and concrete methods (with implementation).

Abstract class	Interface
It can have instance variables (fields) of any access modifier.	It can only have public static final fields (constants).
It can have constructors.	It can't have constructors.
A class can only extend one abstract class (single inheritance).	A class can implement multiple interfaces (multiple inheritance of type)

1 f) How to declare and initialize a two-dimensional array? (2M)

Declaration :

```
datatype array_name[][]=new datatype[size][size];
```

Initialization:

```
int a[][]={1,2,3,4,5,6,7,8,9};
```

(or)

```
int a[][]=new int[][]{{1,2,3},{4,5,6},{7,8,9}};
```

1 g) Differentiate between throw and throws in exception handling.(2M)

throw is used to create a new instance of exception object .

```
try {
        throw new ArithmeticException();
}
catch(ArithmeticException e) {
        e.printStackTrace();
}
```

throws is used to specify the exceptions that method can rise in the body at the signature of the method.

```
class Demo{
    public static void main(String[] args) throws IOException {
        // body of the method
    }
}
```

1 h) Explain the use of finally block in exception handling. (2M)

The finally block in Java is an integral part of exception handling, used in conjunction with try and catch blocks. Its primary purpose is to ensure that a specific block of code is executed, regardless of whether an exception occurs in the try block or is caught by a catch block.

Guaranteed Execution:

The finally block is always executed, even if an exception is thrown and caught, or if no exception occurs. It also executes even if a return, break, or continue statement is encountered within the try or catch blocks.

Resource Cleanup:

The finally block is commonly used for resource cleanup operations. This includes closing file streams, database connections, network sockets, or releasing locks, ensuring that these resources are properly managed and preventing resource leaks.

1 i) What is thread priority in java? How is it set? (2M)

In multithreading, if multiple threads are competing for CPU then thread priority is used to specify to the JVM to determine the order in which threads are scheduled for execution.

setPriority(int newPriority) --> used to set the priority

1 j) Distinguish between a thread and a process. (2M)

Thread	Process
The smallest unit of execution within a process.	An executing program with its own memory space.
Shares memory space with other threads in the same process.	Has its own separate memory space.
Resource consumption is Low (lightweight) because it shares resources.	Resource consumption is High (heavyweight) due to separate memory.

PART-B

2 a) What are literals and Symbolic constants in Java? Illustrate with examples. (5 M)

Literals are explicit values inserted into the code that are used to specify a default value for interface attributes or to declare the value for a constant.
Literals can be Boolean (true or false), numeric (integer or floating point), or character-based (a single character or a string).

Data type	Literal specification	Example

Integer	Base 2: prefix 0b or 0B
	Base 10: no prefix default
	Base 8: prefix 0
	Base 16: prefix 0x or 0X
	Long : suffix l or L
		int x = 0b1101;
		int x = 253;
		int x = 024;
		int x = 0x17A;
		long x = 2867893L;

Floating-point	Float: suffix f or F Double : no suffix default	float y=2.57f; double d= 4.56783;
Character	Character : ‘ ’ String : “ ”	char ch=’a’; String str = “java world”;
Boolean	true or false	boolean flag= true;

In Java, a symbolic constant is a named constant value that is defined once and whose value cannot be changed after initialization. They are used to improve code readability, maintainability, and prevent unintended modifications of important values within a program.

syntax:

public static final data_type CONSTANT_NAME = value;

Example:

```
public class ConstantDemo {

   public static final double PI = 3.14159;
   public static final int MAX_USERS = 100;
   public static final String APP_VERSION = "1.0.0";

   public static void main(String[] args) {
      System.out.println("Value of PI: " + MyConstants.PI);
      System.out.println("Maximum users allowed: " + MyConstants.MAX_USERS);
      System.out.println("Application version: " + MyConstants.APP_VERSION);
   }
}
```

b) Write a java program using nested if-else statements to determine the largest of three numbers. (5 M)

```
import java.util.Scanner;

public class Largest {
        public static void main(String[] args) {
                Scanner sc=new Scanner(System.in);
                System.out.print("Enter x value : ");
                int x=sc.nextInt();
                System.out.print("Enter y value : ");
                int y=sc.nextInt();
                System.out.print("Enter z value : ");
                int z=sc.nextInt();
                if(x>y && x>z)
                        System.out.println( x + " is greatest ");
                else if(y>z)
                        System.out.println(y+ " is greatetst ");
                else
                        System.out.println(z+ " is greatetst ");
                sc.close();
        }
}
```

3 a) What are escape sequences in java? Write a program that uses escape sequences to format output. (3 M)

In Java, an escape sequence is a series of characters beginning with a backslash (`\`) that represents a special character or action within a string literal. These sequences are interpreted by the Java compiler and allow you to include characters that would otherwise be difficult or impossible to represent directly in a string, such as newline characters, tabs, or quotation marks.

- \n: Newline - Inserts a line break, moving the cursor to the beginning of the next line.
- \t: Tab - Inserts a horizontal tab space.
- \r: Carriage Return - Moves the cursor to the beginning of the current line, potentially overwriting previous text.
- \": Double Quote - Inserts a double quotation mark within a string literal.
- \': Single Quote - Inserts a single quotation mark within a string literal.
- \\: Backslash - Inserts a literal backslash character.
- \b: Backspace - Moves the cursor back one position, potentially deleting the preceding character.

Example:

```
public class EscapeSequenceDemo {

    public static void main(String[] args) {
            System.out.println("Hello\nJava!"); // Newline
     System.out.println("Item1\tItem2"); // Tab
     System.out.println("Hai, \"How are you?\""); // Double quote
     System.out.println("Path: C:\\Program Files\\Java"); // Backslash
    }

}
```

b) Develop a Program that calculate grades based on the marks for the following range:(7 M)

Range	Grade
90-100	S
80-89	A
70-79	B
60-69	C
50-59	D
40-49	E
Less than 40	F

```
import java.util.Scanner;

public class CalcualteGrade {

        public static void main(String[] args) {
                Scanner sc=new Scanner(System.in);
                System.out.println("Enter Marks: ");
                int marks=sc.nextInt();
                if(marks>=90 && marks <=100)
                        System.out.println("S");
                else if(marks>=80 && marks<=89)
                        System.out.println("A");
                else if(marks>=70 && marks<=79)
                        System.out.println("B");
                else if(marks>=60 && marks<=69)
                        System.out.println("C");
                else if(marks>=50 && marks<=59)
                        System.out.println("D");
                else if(marks>=40 && marks<=49)
                        System.out.println("E");
                else
                        System.out.println("Fail");

        }

}
```

4 a) Define class and Object in Java. Write a program to create a class student with attributes and methods to display details. (5M)

Class is a specification of an object. (or) It is collection of instance variables and methods.
Object is an instance of class (or) It is a real-world entity.

Example:

```
class Student{
        int rno;
        String name;
        double marks;
        Student (int rno,String name,double marks){
                this.rno=rno;
                this.name=name;
                this.marks=marks;
```

```
    }
    public void display() {
        System.out.println("student roll number:"+rno);
        System.out.println("Student name: "+name);
        System.out.println("Marks: "+marks);
    }
}
public class StudentDemo {
    public static void main(String[] args) {
        Student s1=new Student(101,"william",25.5);
        s1.display();
        Student s2=new Student(102,"Henry",28.2);
        s2.display();
    }

}
```

Note: Student can write any example that contains attributes and methods to display the data.

b) Write a program to count the number of vowels, consonants, spaces, digits, and special character in a string. (5M)

```
public class StringDemo {
    public static void main(String[] args){
        String s="hello123 xyz.uvz@gmail.com";
        int vowels = 0, consonant = 0, specialChar = 0, digit = 0, spaces = 0;
      for (int i = 0; i < s.length(); i++) {
        char ch = s.charAt(i);
       if ( (ch >= 'a' && ch <= 'z') || (ch >= 'A' && ch <= 'Z') ) {
        ch = Character.toLowerCase(ch);
        if (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u')
            vowels++;
        else
            consonant++;
        }
       else if (ch >= '0' && ch <= '9')
            digit++;
       else if (ch==' ')
             spaces++;
       else
            specialChar++;
      }
      System.out.println("Vowels: " + vowels);
      System.out.println("Consonant: " + consonant);
      System.out.println("Spaces: "+spaces);
      System.out.println("Digit: " + digit);
      System.out.println("Special Character: " + specialChar);

    }
}
```

5 a) What is a final method? How does it differ from a normal method? (5M)

A method specified with the final access modifier is known as final method. that cannot be be overridden by subclasses, ensuring its behavior remains constant throughout the inheritance hierarchy

```
public final return_type method_name(argument_list){

        // method body;
}
```

Final Method	Normal Method
A method with the final keyword that cannot be overridden.	A regular method that can be overridden by subclasses.
Can be inherited but not modified by subclasses.	Can be inherited and its behavior can be changed (overridden) by subclasses.
To lock down the implementation of a method to prevent changes through inheritance, ensuring critical logic is enforced.	To provide a general implementation that can be specialized by subclasses.

```
class A {
        // can be overridden in subclass
        public void method1(){
          System.out.println("non final method");
        }
        // cannot be overriden
        public final void method2(){
          System.out.println("final method in class A");
}
class B extends A {

        public void method1(){
           System.out.prntln("overridden in sub class");
        }
}
```

b) Develop a program to demonstrate how to access private members of a class. (5 M)

```
class Student{
        private int rno;
        private String name;
        private double marks;
        Student (int rno,String name,double marks){
                this.rno=rno;
                this.name=name;
                this.marks=marks;
        }
        public int getRno() {
                return rno;
```

```
    }
    public void setRno(int rno) {
        this.rno = rno;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public double getMarks() {
        return marks;
    }
    public void setMarks(double marks) {
        this.marks = marks;
    }
    public void display() {
        System.out.println("student roll number:"+rno);
        System.out.println("Student name: "+name);
        System.out.println("Marks: "+marks);
    }
}
public class StudentDemo {
    public static void main(String[] args) {
        Student s1=new Student(101,"william",25.5);
        s1.display();
        s1.setMarks(26.2);
        s1.setName("William Henry");
        s1.display();
    }
}
```

6 a) Explain the use of super keyword in inheritance. (5M)

super keyword can be used for two purposes

1. to access parent class constructor explicitly.
2. to access parent class members if child and parent class have same name.

```
class A{
    int x,y;
    A(int x,int y){
        this.x=x;
        this.y=y;
    }
}
class B extends A{
    int x,y,z;
    B(int x,int y,int z){
            // usage 1
        super(x,y);
        this.z=z;
    }
    void modify(){
            // usage 2
        z=z+super.x+super.y;
```

```
        }
        int getZ() {
                return z;
        }
    }
    public class SuperDemo {
        public static void main(String[] args) {
                B obj=new B(10,20,30);
                obj.modify();
                System.out.println(obj.getZ());
        }
    }
```

b) Differentiate between method overloading and method overriding with suitable example. (5M)

Method overloading is a way of defining two or more methods with same method name but differ in argument type and number of arguments in a class.

Method overriding is a way of overriding the parent class method in the child or sub class.

```
class A {

        public int sum(int a,int b){
                return a+b;
        }
        public double sum(double a,double b,double c){
                return a+b+c;
        }
        public void display(){
                System.out.println("In parent class");
        }
}
class B extends A {
        public void display(){
                System.out.println("In child class");
        }
}

public class OverloadDemo {

        public static void main(String[] args) {
                A obj=new A();
                int result=obj.sum(10, 20);
                System.out.println(result);
                double d=obj.sum(2.3, 4.1, 5.6);
                System.out.println(d);
                obj.display();
                obj=new B();
                obj.display();
        }
}
```

7 a) Explain the concept of extending an interface with a suitable example (5M)

```
package demo;
interface Shape{
        public double area();
}
interface Polygon extends Shape{
        public double perimeter();
}
class Rectangle implements Polygon{
        double length,breadth;
        Rectangle(double length,double breadth){
                this.length=length;
                this.breadth=breadth;
        }
        public double area() {
                return length*breadth;
        }
        public double perimeter() {
                return 2*(length+breadth);
        }

}
class Triangle implements Polygon{
        double base,height;
        Triangle(double base,double height){
                this.base=base;
                this.height=height;
        }
        public double area() {
                return 0.5*base*height;
        }
        public double perimeter() {
                return base+height+Math.sqrt(base*base+height*height);
        }
}
public class InterfaceDemo {
        public static void main(String[] args) {
                Rectangle s=new Rectangle(10.0,20.0);
                System.out.println("Area of a rectangle "+s.area());
                System.out.println("Perimeter of a rectangle "+s.perimeter());
                Triangle t=new Triangle(10.0,5.0);
                System.out.println("Area of a Triangle "+t.area());
                System.out.println("Perimeter of a Triangle "+t.perimeter());
        }
}
```

b) Write a Java Program to accept a matrix of order NxN and interchange the diagonals. (5M)

```
    public class InterchangeDiagonals {
        public static void main(String[] args) {
                int a[][]=new int[][]{{1,2,3,4},{5,6,7,8},{9,10,11,12},{13,14,15,16}};
                int l=a.length-1;
```

```
                System.out.println("Before interchange");
                for(int[] row:a) {
                        for(int ele:row)
                                System.out.print(ele+" ");
                        System.out.println();
                }

                for(int i=0;i<a.length;i++) {
                        for(int j=0;j<a[0].length;j++) {
                                if(i==j) {
                                        int t=a[i][j];
                                        a[i][j]=a[i][l-i];
                                        a[i][l-i]=t;
                                }
                        }
                }
                System.out.println("Before interchange");
                for(int[] row:a) {
                        for(int ele:row)
                                System.out.print(ele+"  ");
                        System.out.println();
                }
                }

        }
```

8 a) Explain the behavior of different access specifiers in packages with examples. (5 M)

Access specifier	Within a same package
Public	Accessible
Proctected	Accessible
No modifier	Accessible
Private	Not Accessible

```
package demo;
class Access{
        private int pri_a=10;
        int a=20;
        protected int pro_a=30;
        public int pub_a=40;
}
public class AccessSpecifierDemo {
        public static void main(String[] args) {
                Access obj=new Access();

        //      System.out.println("private member of Access:"+ obj.pri_a); // not accessible
                System.out.println("no modifeir of Access:"+ obj.a);
                System.out.println("protected member of Access:"+ obj.pro_a);
                System.out.println("public member of Access:"+ obj.pub_a);
        }
}
```

b) A file contains 5 integers . Write a program to fetch the numbers from the file, and then calculate their sum, and append its sum to the file. (5M)

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

public class FileRead {

	public static void main(String[] args) throws IOException{
		BufferedReader br=new BufferedReader(new FileReader("numbers.txt"));
		String line;
		int sum=0;
		while((line=br.readLine())!=null) {
			sum=sum+Integer.parseInt(line);
		}
		br.close();
		BufferedWriter bw=new BufferedWriter(new FileWriter("numbers.txt",true));
		bw.write('\n');
		bw.write(Integer.toString(sum));
		bw.close();
	}

}
```

9 a) Write a program in java to demonstrate the use of try, catch and finally blocks. (5 M)

```
public class ExceptionDemo {
	public static void main(String[] args) {
		try {
		// Code that might throw an exception
		int[] numbers = {1, 2, 3};
		System.out.println(numbers[10]); // This will cause an
ArrayIndexOutOfBoundsException
		System.out.println("This line will not be executed if an exception occurs.");
	    } catch (ArrayIndexOutOfBoundsException e) {
		// Code to handle the specific exception
		System.out.println("Caught an ArrayIndexOutOfBoundsException: " +
e.getMessage());
	    } catch (Exception e) {
		// Code to handle any other type of exception (more general)
		System.out.println("Caught a general exception: " + e.getMessage());
	    } finally {
		// Code that will always be executed, regardless of whether an exception occurred
or not
		System.out.println("The 'finally' block always executes, for cleanup or final
actions.");
	    }
	    System.out.println("Program continues after the try-catch-finally block.");
	}
}
```

b) What are checked and unchecked exceptions? Explain with examples. (5 M)

In Java, exceptions are broadly categorized into two types: checked exceptions and unchecked exceptions. The key difference lies in when they are detected and whether the compiler mandates handling them.

Checked Exceptions:

- These exceptions are checked at compile time. The Java compiler forces you to handle them.
- You must either handle a checked exception using a try-catch block or declare that your method might throw it using the throws keyword in the method signature. Failure to do so results in a compilation error.
- They typically represent predictable external conditions that a program might encounter and from which it can reasonably recover. Examples include IOException (e.g., file not found), SQLException (e.g., database connection issues), and ClassNotFoundException.
- All Exception subclasses that are not RuntimeException or its subclasses are checked exceptions.

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

public class CheckedExceptionExample {
    public static void main(String[] args) {
        try {
            FileReader file = new FileReader("nonexistent.txt"); // IOException is a checked exception
            // Further operations with the file
        } catch (IOException e) {
            System.err.println("Error reading file: " + e.getMessage());
        }
    }
}
```

Unchecked Exceptions:

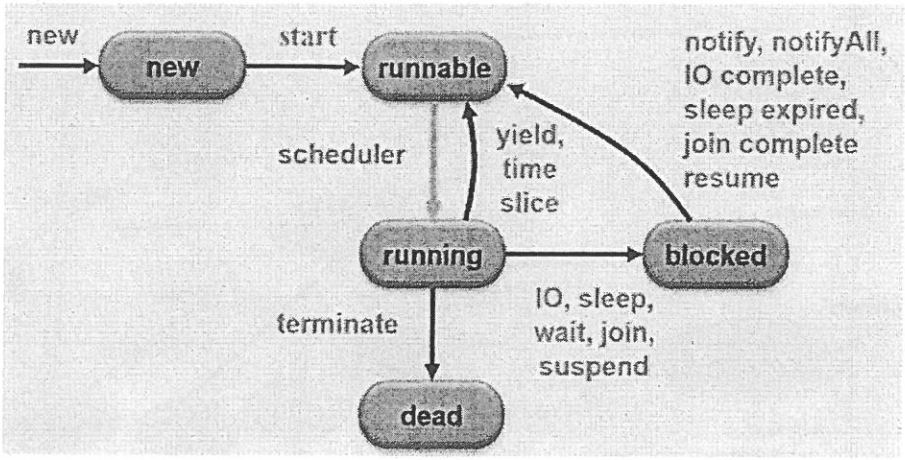
- These exceptions are checked at runtime. The compiler does not mandate handling them.
- You are not required to handle unchecked exceptions, though you can if necessary. They often indicate programming errors that should ideally be fixed rather than caught.
- They usually represent programming errors or logical flaws within the code that are difficult to anticipate and recover from gracefully. Examples include NullPointerException (e.g., accessing a method on a null object), ArrayIndexOutOfBoundsException (e.g., trying to access an array element outside its bounds), and ArithmeticException (e.g., division by zero).
- All RuntimeException and its subclasses are unchecked exceptions. Error and its subclasses are also considered unchecked.

```
public class UncheckedExceptionExample {
    public static void main(String[] args) {
        String text = null;
        // The following line will throw a NullPointerException at runtime
        // The compiler does not force you to handle it
        System.out.println(text.length());
    }
}
```

10 a) Write a Java program to create multiple threads by extending the thread class. (5 M)

```
class Thread1 extends Thread{
        String name;
        Thread1(String name){
                this.name=name;
        }
        public void run() {
                for(int i=1;i<=5;i++)
                        System.out.println(name+" : "+i);
        }
}
public class ThreadDemo {
        public static void main(String[] args) {
                Thread t1=new Thread1("Thread A");
                Thread t2=new Thread1("Thread B");
                Thread t3=new Thread1("Thread C");
                t1.start();
                t2.start();
                t3.start();
        }
}
```

b) Explain Thread life cycle with a neat sketch. (5 M)



New state : A thread is new state means memory is created , but it is not yet running

Runnable : If you invoke a start() method, then the thread changes to new to runnable state, means it running.

Running : Currently executed by the processor.

Blocked / Waiting: When a thread is blocked or waiting, it is temporarily inactive. It doesn't execute any code and consumes minimal resources. It is up to the thread scheduler to reactivate it.

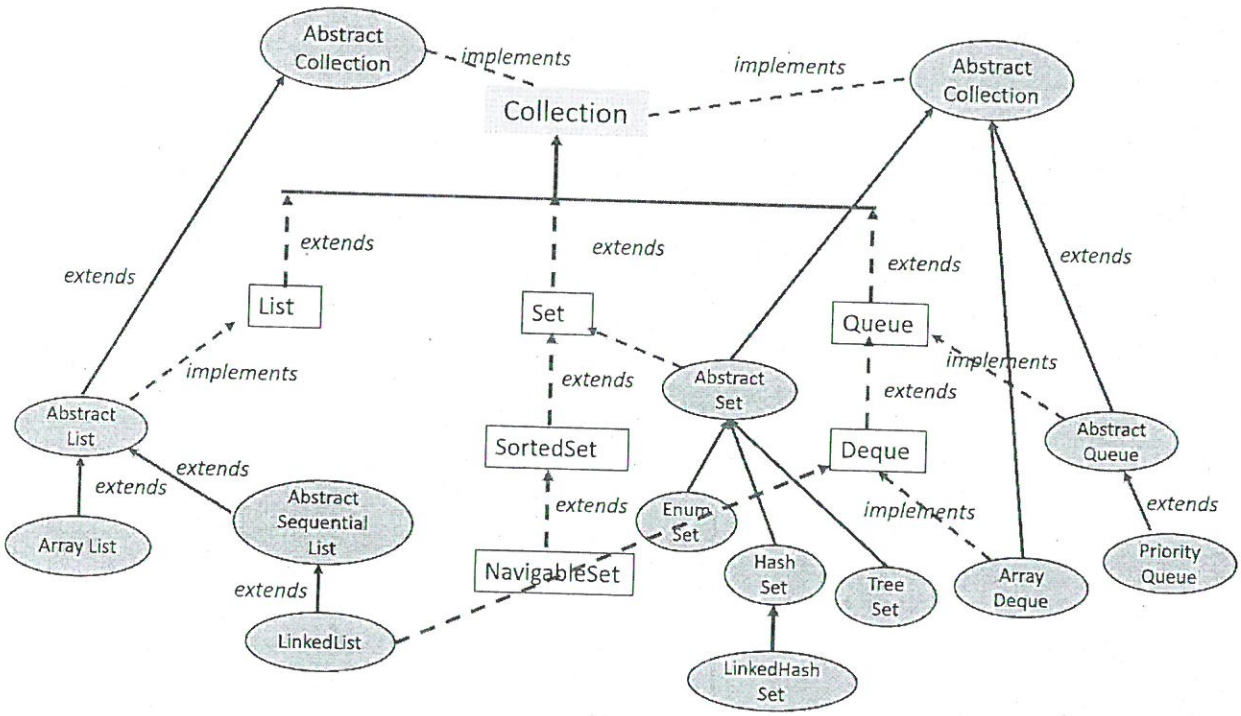
Dead : A thread is dead for one of two reasons:

- It dies a natural death because the run method exits normally.
- It dies abruptly because an uncaught exception terminates the run method.

11 a) Explain how to insert and delete elements from a list collection with an example. (5 M)

```
public class CollectionDemo {
        public static void main(String[] args) {
                List<Integer> l=new ArrayList<>();
                l.add(10);
                l.add(20);
                l.add(30);
                l.add(40);
                l.add(50);
                System.out.println("List elements are: "+l);
                // remove index element
                l.remove(0);
                System.out.println("List elements are: "+l);
                Integer ele=20;
                // remove specified element
                l.remove(ele);
                System.out.println("List elements are: "+l);
        }
}
```

b) Explain about hierarchy of collection interfaces.(5M)



Interface List: Provides sequential access of data. The classes that extends AbstractList and AbstractSequentialList are

- ArrayList
- LinkedList

Set: Provides unique elements. The classes that extends AbstractSet are

- EnumSet

- HashSet
- TreeSet
- LinkedHashSet

Queue & Deque : used to provide constant time insertion and deletion . The classes that extend Abstract Queue are

- Array Deque
- PriorityQueue