Model Answer	Exam	SUM 2025
Subject: Java Programming		K-Scheme
	SUB CODE	314317

Important Instructions to STUDENTS

1) *The model answer given here are prepared from the answers from the previously uploaded model answers by Board.*
2) *These model answers are not uploaded by the MSBTE official site but MSBTE study resources website prepared it for students. This model answer has question paper also inbuilt in it, no need to download it separate.*
3) *Please remembter that answers are not checked word to word but based on keywords which must be present in your answer*
4) *The model answer and the answer written by candidate may vary but the examiner may tryto assess the understanding level of the candidate*
5) *While assessing figures, examiner may give credit for principal components indicated in thefigure. The figures drawn bycandidate and model answer may vary. The examiner may give credit for anyequivalent figure drawn*
6) *For programming language papers, credit may be given to any other program based on equivalentconcept*
7) *Students are advised to prepare all the syllabus from recommended book and use these model answers for the purpose of tests.*

Q		ANSWER	Marking Scheme
1	a)	**Question: List any four features of Java.**	
		Answer:	
		Java has several important features that make it popular among programmers.	
		Platform Independent – Java programs can run on any operating system using the Java Virtual Machine (JVM).	
		Object-Oriented – Java is based on the concept of objects and classes.	
		Simple and Easy to Learn – Java has a clean and easy-to-understand syntax.	
		Secure – Java provides built-in security features like bytecode verification and runtime security checks.	

1	b)	**Question: Define static import. Give its syntax.**	
		Answer:	
		Static import is a feature in Java that allows members (fields and methods) defined in a class as static to be used in another class without specifying the class name.	
		It helps in writing cleaner and more readable code by removing the class name prefix.	
		Syntax:	
		import static package_name.ClassName.static_member;	
		You can also import all static members using:	
		import static package_name.ClassName.;*	

1	c)	**Question: Define exceptions. List its types.**	
		Answer:	
		Exceptions in Java are unwanted or unexpected events that occur during the execution of a program and disrupt its normal flow.	
		They are used to handle errors gracefully so that the program does not crash.	
		Java provides a robust exception handling mechanism using try, catch, throw, throws, and finally blocks.	
		Types of exceptions in Java:	
		Checked Exceptions – These are checked at compile time (e.g., IOException, SQLException).	
		Unchecked Exceptions – These occur at runtime (e.g., NullPointerException, ArithmeticException).	
		Errors – These are serious problems that applications should not try to handle (e.g., OutOfMemoryError).	

1 d) **Question: Define type casting. List its types.**

Answer:

Type casting in Java is the process of converting a variable from one data type to another.
It is used when you need to assign a value of one type to a variable of another type.

Types of type casting:

Implicit Casting (Widening Conversion) – Automatic conversion from a smaller data type to a larger data type (e.g., int to long).

Explicit Casting (Narrowing Conversion) – Manual conversion from a larger data type to a smaller data type using parentheses (e.g., double to int).

1 e) **Question: Write any two differences between AWT and Swing.**

Answer:

Feature	AWT	Swing
Component Type	Heavyweight (uses native OS components)	Lightweight (written entirely in Java)
Appearance	Platform-dependent look and feel	Consistent look and feel across platforms

Question: Define Proxy server and Reverse socket.

1 f)

Answer:

Proxy Server:

A proxy server acts as an intermediary between a client and the internet or another server. It forwards client requests to the target server and then returns the server's response to the client. Proxy servers help improve security, control access, and cache data to speed up requests.

Reverse Socket:

A reverse socket (often called a reverse shell) is a connection where a remote device connects back to a local machine to provide access or control. It is commonly used for remote administration or troubleshooting when the local machine is behind a firewall or NAT, allowing the remote system to initiate the connection.

1	g)	**Question: Write any four methods of Statement.**	
		Answer:	
		The Statement interface in Java provides methods to execute SQL queries on a database. Four common methods of Statement are:	
		executeQuery(String sql) – Executes a SQL query that returns a ResultSet, typically used for SELECT statements.	
		executeUpdate(String sql) – Executes SQL statements like INSERT, UPDATE, or DELETE and returns the number of affected rows.	
		execute(String sql) – Executes any SQL statement and returns a boolean indicating if the result is a ResultSet or an update count.	
		close() – Closes the Statement object and releases database resources associated with it.	

2 a) **Question: Write a program to accept age from user and throw an exception if age is less than 18.**

Answer:

Here is a simple Java program that accepts the user's age and throws an exception if the age is less than 18:

1

```
import java.util.Scanner;

class AgeNotValidException extends Exception {
    public AgeNotValidException(String message) {
        super(message);
    }
}

public class AgeCheck {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter your age: ");
        int age = scanner.nextInt();

        try {
            if (age < 18) {
                throw new AgeNotValidException("Age must be at least 18.");
            } else {
                System.out.println("Age is valid.");
            }
        } catch (AgeNotValidException e) {
            System.out.println("Exception caught: " + e.getMessage());
        }

        scanner.close();
    }
}
```

1

1

1

2	b)	**Question: Explain garbage collection in Java.**	
		Answer:	
		Garbage collection in Java is the process of automatically deleting unused objects from memory to free up space. It is done by the Java Virtual Machine (JVM), not the programmer. This helps in managing memory efficiently and makes the program run better without memory problems.	
		When a Java program creates objects, they are stored in a part of memory called the heap. Some of these objects may no longer be used as the program runs. The garbage collector finds such unused objects and removes them to make space for new ones. This process happens in the background while the program is running.	
		Benefits of garbage collection:	
		1.Frees memory automatically by removing unused objects.	
		2.Prevents memory leaks, which can slow down or crash the program.	
		3.Makes memory management easier for the programmer.	
		4.Improves program performance and reliability.	
		5.Reduces the chances of errors caused by manual memory handling.	
		Garbage collection is one of the key features that makes Java a simple and powerful language for building applications.	

2 c) **Question: Write a program to use Label, TextField, and Button to create a login form. When the user clicks the button, a login successful/unsuccessful message should be displayed.**

Answer:

Here is a simple Java Swing program that creates a login form using JLabel, JTextField, JPasswordField, and JButton. When the user clicks the button, it checks the username and password and displays a message:

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

public class LoginForm extends JFrame implements ActionListener {
    JLabel userLabel, passLabel, messageLabel;
    JTextField userText;
    JPasswordField passText;
    JButton loginButton;

    public LoginForm() {
        // Create labels
        userLabel = new JLabel("Username:");
        userLabel.setBounds(50, 50, 100, 30);

        passLabel = new JLabel("Password:");
        passLabel.setBounds(50, 100, 100, 30);

        // Create text fields
        userText = new JTextField();
        userText.setBounds(150, 50, 150, 30);

        passText = new JPasswordField();
        passText.setBounds(150, 100, 150, 30);
```

```
        // Create login button
        loginButton = new JButton("Login");
        loginButton.setBounds(150, 150, 100, 30);
        loginButton.addActionListener(this);

        // Message label to display result
        messageLabel = new JLabel();
        messageLabel.setBounds(50, 200, 300, 30);

        // Add components to the frame
        add(userLabel);
        add(passLabel);
        add(userText);
        add(passText);
        add(loginButton);
        add(messageLabel);

        // Frame settings
        setSize(400, 300);
        setLayout(null);
        setVisible(true);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setTitle("Login Form");
    }
```

```
    @Override
    public void actionPerformed(ActionEvent e) {
        String username = userText.getText();
        String password = new String(passText.getPassword());

        // Simple validation: username=admin, password=1234
        if (username.equals("admin") && password.equals("1234")) {
            messageLabel.setText("Login Successful!");
        } else {
            messageLabel.setText("Login Unsuccessful. Try again.");
        }
    }

    public static void main(String[] args) {
        new LoginForm();
    }
}
```

Question : Differentiate between TCP and UDP.

2 d)

Question: Differentiate between TCP and UDP.

Answer:

The differences between TCP (Transmission Control Protocol) and UDP (User Datagram Protocol) are given below in tabular form:

TCP (Transmission Control Protocol)	UDP (User Datagram Protocol)
TCP is **connection-oriented.**	UDP is **connectionless.**
It is **reliable** as it ensures data delivery.	It is **unreliable**, delivery is not guaranteed.
Data is sent in **sequence.**	Data may arrive **out of order.**
Slower due to error checking and correction.	**Faster** as there is no error checking.
Used for applications like **web browsing, email.**	Used for **video streaming, gaming.**
Header size is **larger** (20–60 bytes).	Header size is **smaller** (8 bytes).

This table shows how TCP focuses on reliable communication, while UDP focuses on speed and simplicity.

3 a) **Question: Explain constructor with its types. Give example.**

Answer:
A constructor in Java is a special method that is used to initialize objects. It has the same name as the class and does not have a return type, not even void. When an object is created, the constructor is called automatically to set initial values for the object.

Types of constructors:

Default Constructor:
This constructor does not take any parameters. If no constructor is defined in a class, Java provides a default constructor automatically, which initializes objects with default values.

Parameterized Constructor:
This constructor takes parameters to initialize an object with given values. It allows creating objects with different initial states.

```java
class Student {
    String name;
    int age;

    // Default constructor
    Student() {
        name = "Unknown";
        age = 0;
    }

    // Parameterized constructor
    Student(String n, int a) {
        name = n;
        age = a;
    }

    void display() {
        System.out.println("Name: " + name);
        System.out.println("Age: " + age);
    }
}

public class Main {
    public static void main(String[] args) {
        Student s1 = new Student();              // Calls default constructor
        Student s2 = new Student("Alice", 20);   // Calls parameterized constructor

        s1.display();
        s2.display();
    }
}
```

3 b) **Question: Write a program to create two threads. Thread A should print even numbers from 1 to 50 and thread B should print odd numbers from 1 to 50. Each thread should relinquish control to the other frequently.**

Answer:

Here is a simple Java program that creates two threads: one for printing even numbers and the other for odd numbers from 1 to 50. Each thread calls Thread.yield() to give control to the other thread frequently.

```
class EvenThread extends Thread {
    public void run() {
        for (int i = 2; i <= 50; i += 2) {
            System.out.println("Even: " + i);
            Thread.yield();  // Relinquish control
        }
    }
}

class OddThread extends Thread {
    public void run() {
        for (int i = 1; i <= 50; i += 2) {
            System.out.println("Odd: " + i);
            Thread.yield();  // Relinquish control
        }
    }
}

public class TwoThreads {
    public static void main(String[] args) {
        EvenThread t1 = new EvenThread();
        OddThread t2 = new OddThread();

        t1.start();
        t2.start();
    }
}
```

3 c) **Question: Explain final with respect to inheritance. Describe how final method is different from abstract method.**

Answer:

In Java, the **'final'** keyword is used to restrict modification or inheritance. When a class is declared as final, it cannot be inherited. When a method is declared as final, it cannot be overridden by any subclass.

This is useful when you want to prevent further changes in the class or method behavior.

Difference between Final Method and Abstract Method:

Final Method	Abstract Method
Has a body (already implemented).	Has no body (no implementation).
Cannot be overridden in a subclass.	Must be overridden in a subclass.
Used to stop changes in method logic.	Used to force subclasses to define logic.
Can be present only in a concrete class.	Can be present only in an abstract class.

3 d) **Question: Write a program to list five states using JComboBox. Display selected item in JTextField.**

Answer:

Here is a simple Java Swing program that uses a JComboBox to list five states. When the user selects a state, the selected item is shown in a JTextField.

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

public class StateSelector {
    public static void main(String[] args) {
        JFrame frame = new JFrame("State Selector");

        // Create ComboBox with state names
        String[] states = {"Maharashtra", "Gujarat", "Punjab", "Kerala", "Tamil Nadu"};
        JComboBox<String> comboBox = new JComboBox<>(states);
        comboBox.setBounds(50, 50, 150, 30);

        // Create TextField to display selected state
        JTextField textField = new JTextField();
        textField.setBounds(50, 100, 200, 30);

        // Add ActionListener to ComboBox
        comboBox.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                String selectedState = (String) comboBox.getSelectedItem();
                textField.setText(selectedState);
            }
        });

        // Add components to frame
        frame.add(comboBox);
        frame.add(textField);

        frame.setSize(300, 200);
        frame.setLayout(null);
        frame.setVisible(true);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}
```

4 a) **Question: Write a program to use JTextField, JLabel and JButton to add two numbers and display the output. Use GridLayout.**

Answer:

Here is a simple Java Swing program using GridLayout to add two numbers entered by the user:

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

public class AddNumbersGrid {
    public static void main(String[] args) {
        JFrame frame = new JFrame("Add Two Numbers");
        frame.setLayout(new GridLayout(4, 2)); // 4 rows, 2 columns

        JLabel label1 = new JLabel("Enter First Number:");
        JTextField text1 = new JTextField();

        JLabel label2 = new JLabel("Enter Second Number:");
        JTextField text2 = new JTextField();

        JButton addButton = new JButton("Add");

        JLabel resultLabel = new JLabel("Result:");
        JTextField resultField = new JTextField();
        resultField.setEditable(false);
```

```
        // Add components to frame
        frame.add(label1);
        frame.add(text1);
        frame.add(label2);
        frame.add(text2);
        frame.add(new JLabel()); // Empty cell
        frame.add(addButton);
        frame.add(resultLabel);
        frame.add(resultField);
```

```
        // Add button action
        addButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                try {
                    int num1 = Integer.parseInt(text1.getText());
                    int num2 = Integer.parseInt(text2.getText());
                    int sum = num1 + num2;
                    resultField.setText(String.valueOf(sum));
                } catch (NumberFormatException ex) {
                    resultField.setText("Invalid input");
                }
            }
        });

        frame.setSize(300, 200);
        frame.setVisible(true);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}
```

4	b)	**Question: Define package. Explain how to create and access user-defined package in Java.**	
		Answer:	
		A package in Java is a way to group related classes, interfaces, and sub-packages. It helps organize code properly and avoids naming conflicts between classes. Java provides built-in packages (like java.util, java.io) and allows users to create their own packages, called user-defined packages.	
		Creating a User-Defined Package:	
		To create a package, use the package keyword at the beginning of the class file.	
		The class file should be saved in a folder with the same name as the package.	
		Example:	
		If we write package mypack; at the top of the file, the file should be saved inside a folder named mypack.	
		After writing the class, compile it using the -d option. This tells Java where to place the class file in the correct folder structure.	
		Accessing a User-Defined Package:	
		Use the import keyword to include the class from the package.	
		Then create an object of that class and use it as usual.	

4 c) **Question: Write a program to read and display content from the URL "http://www.google.com".**

Answer :

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

public class ReadURL {
    public static void main(String[] args) {
        try {
            URL url = new URL("http://www.google.com");
            BufferedReader reader = new BufferedReader(
                new InputStreamReader(url.openStream())
            );

            String line;
            while ((line = reader.readLine()) != null) {
                System.out.println(line);
            }

            reader.close();
        } catch (Exception e) {
            System.out.println("Error: " + e);
        }
    }
}
```

Explanation:

The URL class is used to create a connection to the given website.

openStream() opens an input stream to read data from the website.

BufferedReader reads the content line by line.

The program prints each line until the end of the page.

If there is any error (like no internet), the catch block will handle it and print an error message.

This program displays the HTML code of the web page at "http://www.google.com".

4 d) **Question: Explain the different types of JDBC drivers.**

Answer:

JDBC (Java Database Connectivity) drivers are used to connect Java applications with databases. There are four types of JDBC drivers, each working differently to establish the connection between a Java program and a database.

1. Type 1: JDBC-ODBC Bridge Driver

Uses the ODBC (Open Database Connectivity) driver to connect to the database.

Translates JDBC method calls into ODBC calls.

Advantage: Easy to use for testing.

Disadvantage: Requires ODBC driver to be installed. Not suitable for production.

2. Type 2: Native-API Driver

Converts JDBC calls into native database API calls using database-specific libraries.

Advantage: Better performance than Type 1.

Disadvantage: Platform dependent. Requires native DB libraries on the client machine.

3. Type 3: Network Protocol Driver

Sends JDBC calls to a middleware server which then connects to the database.

The middleware server handles the database communication.

Advantage: No client-side database library needed. Good for internet-based applications.

Disadvantage: Requires middleware server setup.

4. Type 4: Thin Driver (Pure Java Driver)

Directly converts JDBC calls to the database-specific protocol.

Written entirely in Java.

Advantage: Platform independent, best performance, widely used.

Disadvantage: One driver per database (e.g., MySQL, Oracle, etc.).

Type	Name	Platform Independent	Uses Native Code	Performance	Usage
1	JDBC-ODBC Bridge	No	Yes	Low	Testing only
2	Native-API Driver	No	Yes	Medium	Limited use
3	Network Protocol Driver	Yes	No	Medium	Web apps
4	Thin Driver	Yes	No	High	Most common

4 e) **Question: Write a program to use KeyListener to display characters entered by the user.**

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

public class KeyListenerExample extends Frame implements KeyListener {
    Label label;
    TextField textField;

    public KeyListenerExample() {
        setLayout(new FlowLayout());

        label = new Label("Entered Characters:");
        add(label);

        textField = new TextField(20);
        textField.addKeyListener(this);
        add(textField);

        setSize(300, 150);
        setTitle("KeyListener Example");
        setVisible(true);

        // Close window on close button click
        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent we) {
                System.exit(0);
            }
        });
    }
```

```
    public void keyTyped(KeyEvent e) {
        char ch = e.getKeyChar();
        label.setText("Entered Characters: " + textField.getText() + ch);
    }

    public void keyPressed(KeyEvent e) {
        // Not used
    }

    public void keyReleased(KeyEvent e) {
        // Not used
    }

    public static void main(String[] args) {
        new KeyListenerExample();
    }
}
```

5 a) **Question: Write a program to display the list of employees of deptno 10 from the table employee(eno, ename, salary, deptno). Use PreparedStatement.**

Answer :

```
import java.sql.*;

public class EmployeeList {
    public static void main(String[] args) {
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            Connection con = DriverManager.getConnection(
                "jdbc:mysql://localhost:3306/yourDB", "root", "password");

            String sql = "SELECT eno, ename, salary, deptno FROM employee WHERE deptno = ?";
            PreparedStatement ps = con.prepareStatement(sql);
            ps.setInt(1, 10);

            ResultSet rs = ps.executeQuery();

            System.out.println("ENO\tENAME\tSALARY\tDEPTNO");
            while (rs.next()) {
                System.out.println(rs.getInt("eno") + "\t" + rs.getString("ename") + "\t" +
                                   rs.getDouble("salary") + "\t" + rs.getInt("deptno"));
            }

            rs.close();
            ps.close();
            con.close();
        } catch (Exception e) {
            System.out.println("Error: " + e);
        }
    }
}
```

5 b) **Question: Write a program to implement the following:**

i) Create a Vector and add 2 Integer, 2 Double, and 1 Float object.
ii) Add a String object at the 3rd position.
iii) Remove an element specified by the user.
iv) Display all elements in the Vector.
v) Remove all elements from the Vector.
vi) Display the capacity of the Vector.

```
import java.util.*;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        // 1) Create Vector and add 2 Integer, 2 Double, 1 Float
        Vector<Object> v = new Vector<>();
        v.add(10);            // Integer
        v.add(20);            // Integer
        v.add(15.5);          // Double
        v.add(25.75);         // Double
        v.add(5.5f);          // Float

        // 2) Add String at 3rd position (index 2)
        v.add(2, "Hello");

        // 3) Remove element specified by user
        System.out.print("Enter element to remove: ");
        String input = sc.nextLine();
```

```
    // Try to remove input as Integer, Double, Float, or String
    boolean removed = false;
    try {
        // Try Integer
        int intVal = Integer.parseInt(input);
        removed = v.remove((Integer) intVal);
    } catch (NumberFormatException e1) {
        try {
            // Try Double
            double dblVal = Double.parseDouble(input);
            removed = v.remove((Double) dblVal);
        } catch (NumberFormatException e2) {
            // Remove as String
            removed = v.remove(input);
        }
    }
```

```
        if (removed)
            System.out.println("Element removed.");
        else
            System.out.println("Element not found.");

        // 4) Display all elements
        System.out.println("Elements in Vector:");
        for (Object obj : v) {
            System.out.println(obj);
        }

        // 5) Remove all elements
        v.clear();
        System.out.println("All elements removed.");

        // 6) Display capacity
        System.out.println("Vector capacity: " + v.capacity());

        sc.close();
    }
}
```

5 c) **Question: Explain the life cycle of thread.**

Answer:

In Java, a thread passes through several stages in its life cycle. The main stages of a thread's life cycle are:

1.New (Created) State:

When a thread object is created using the Thread class but not yet started using start(), it is in the new state.

2.Runnable State:

After calling the start() method, the thread enters the runnable state. It is ready to run and waiting for CPU time.

3.Running State:

When the thread scheduler picks the thread from the runnable pool, it enters the running state. The run() method is executed in this state.

4.Blocked or Waiting State:

A thread enters the waiting or blocked state when it has to wait for a resource or condition. For example, when a thread is waiting for input or sleeping for a while.

5.Timed Waiting State:

A thread that is sleeping or waiting for a specified period enters the timed waiting state. Methods like sleep (time) or join(time) cause this.

6.Dead State:

When a thread finishes its execution or is forcefully terminated, it enters the terminated (dead) state. It cannot be restarted once it is dead.

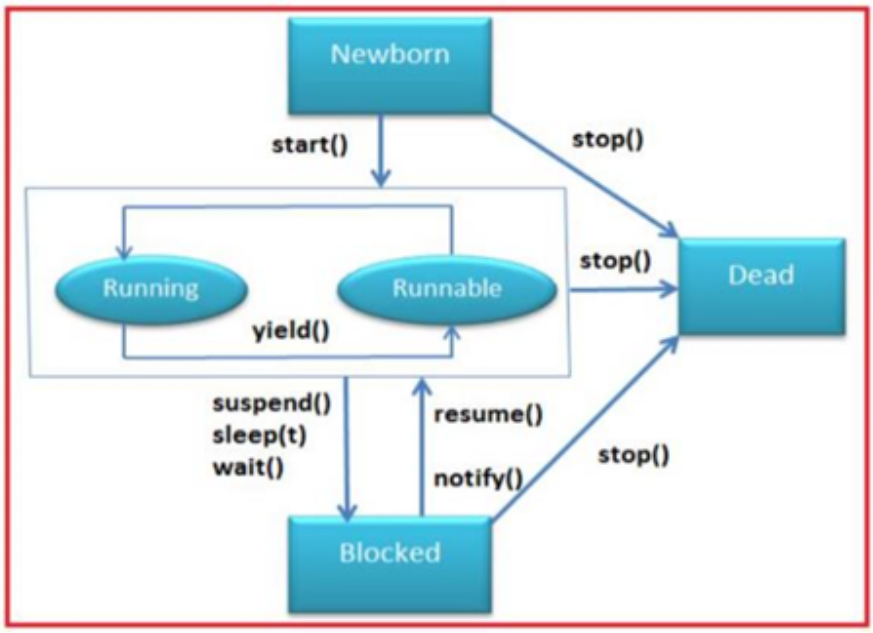
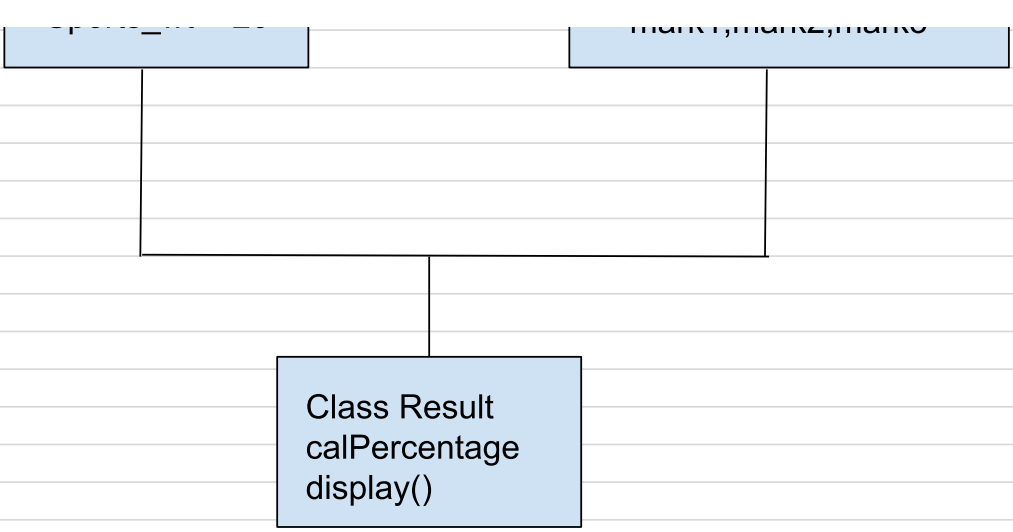


Fig: Life cycle of Thread

6 a **Write a Program to Implement the following fig. No. 1**

Interface Exam
Sports_wt = 20

Class Student
rollNo, name
mark1,mark2,mark3


mark1,mark2,mark3
Class Result
calPercentage
display()

Solution :

```
// Interface
interface Exam {
    int Sports_wt = 20; // Sports weight is constant
}

// Student class
class Student {
    int rollNo;
    String name;
    int mark1, mark2, mark3;

    Student(int rollNo, String name, int m1, int m2, int m3) {
        this.rollNo = rollNo;
        this.name = name;
        this.mark1 = m1;
        this.mark2 = m2;
        this.mark3 = m3;
    }
}

// Result class inherits Student and implements Exam
class Result extends Student implements Exam {
    float percentage;

    Result(int rollNo, String name, int m1, int m2, int m3) {
        super(rollNo, name, m1, m2, m3);
    }
```

```
    void calPercentage() {
        int total = mark1 + mark2 + mark3 + Sports_wt;
        percentage = total / 4.0f;
    }

    void display() {
        System.out.println("Roll No: " + rollNo);
        System.out.println("Name: " + name);
        System.out.println("Marks: " + mark1 + ", " + mark2 + ", " + mark3);
        System.out.println("Sports Weight: " + Sports_wt);
        System.out.println("Percentage: " + percentage + "%");
    }
}
```

```java
// Main class
public class Main {
    public static void main(String[] args) {
        Result r = new Result(101, "Amit", 75, 80, 85);
        r.calPercentage();
        r.display();
    }
}
```

6 b

Write a Program To display Folder structure of Computer using JTree

```
import javax.swing.*;
import javax.swing.tree.*;
import java.io.File;

public class FolderStructure extends JFrame {

    public FolderStructure() {
        File rootFile = new File("C:\\");  // You can change the path as needed
        DefaultMutableTreeNode root = new DefaultMutableTreeNode(rootFile.getName());
        createTree(rootFile, root);

        JTree tree = new JTree(root);
        JScrollPane scrollPane = new JScrollPane(tree);

        add(scrollPane);
        setTitle("Folder Structure");
        setSize(400, 500);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setVisible(true);
    }

    void createTree(File fileRoot, DefaultMutableTreeNode node) {
        File[] files = fileRoot.listFiles();
        if (files == null) return;

        for (File file : files) {
            if (file.isDirectory()) {
                DefaultMutableTreeNode child = new DefaultMutableTreeNode(file.getName());
                node.add(child);
                // Recursively add subfolders
                createTree(file, child);
            }
        }
    }

    public static void main(String[] args) {
        new FolderStructure();
    }
}
```

6 c) **Write a program to implement the following using serversocket and socket.**
i) Client should send a number to server
ii) Server should check if the number is even or odd and send response to client
iii) Client displays output.

Answer:

This program consists of two parts — Server program and Client program.

Part 1) Server.java

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

public class Server {
    public static void main(String[] args) throws Exception {
        ServerSocket ss = new ServerSocket(5000);
        System.out.println("Server started. Waiting for client...");

        Socket s = ss.accept();
        DataInputStream dis = new DataInputStream(s.getInputStream());
        DataOutputStream dos = new DataOutputStream(s.getOutputStream());

        int number = dis.readInt();
        String result;

        if (number % 2 == 0) {
            result = "Even";
        } else {
            result = "Odd";
        }

        dos.writeUTF("Number is " + result);

        dis.close();
        dos.close();
        s.close();
        ss.close();
    }
}
```

Part 2) Client.java

```java
import java.io.*;
import java.net.*;
import java.util.Scanner;

public class Client {
    public static void main(String[] args) throws Exception {
        Socket s = new Socket("localhost", 5000);

        DataInputStream dis = new DataInputStream(s.getInputStream());
        DataOutputStream dos = new DataOutputStream(s.getOutputStream());
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter a number: ");
        int num = sc.nextInt();

        dos.writeInt(num);

        String response = dis.readUTF();
        System.out.println("Server Response: " + response);

        dis.close();
        dos.close();
        s.close();
    }
}
```