

# Java Lab Programs

1. Develop a Java program that prints all real solutions to the quadratic equation  $ax^2+bx+c = 0$ . Read in a, b, c and use the quadratic formula. If the discriminate  $b^2 - 4ac$  is negative, display a message stating that there are no real solutions.

Program:

```
import java.util.Scanner;

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

        double a, b, c;
        double discriminant, root1, root2;

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter value of a: ");
        a = sc.nextDouble();

        System.out.print("Enter value of b: ");
        b = sc.nextDouble();

        System.out.print("Enter value of c: ");
        c = sc.nextDouble();

        discriminant = b * b - 4 * a * c;

        if (discriminant < 0) {
            System.out.println("There are no real solutions.");
        }
        else if (discriminant == 0) {
```

```

        root1 = -b / (2 * a);
        System.out.println("One real solution:");
        System.out.println("x = " + root1);
    }
    else {
        root1 = (-b + Math.sqrt(discriminant)) / (2 * a);
        root2 = (-b - Math.sqrt(discriminant)) / (2 * a);

        System.out.println("Two real solutions:");
        System.out.println("x1 = " + root1);
        System.out.println("x2 = " + root2);
    }

    sc.close();
}
}

```

The screenshot shows a Java code editor interface with the following details:

- File:** Quadratic.java
- Code Content:**

```

1 import java.util.Scanner;
2
3 class Quadratic {
4     public static void main(String args[]) {
5         double a, b, c;
6         double discriminant, root1, root2;
7
8         Scanner sc = new Scanner(System.in);
9
10        System.out.print(s: "Enter value of a: ");
11        a = sc.nextDouble();
12
13        System.out.print(s: "Enter value of b: ");
14        b = sc.nextDouble();
15
16        System.out.print(s: "Enter value of c: ");
17        c = sc.nextDouble();
18
19        discriminant = b * b - 4 * a * c;
20
21        if (discriminant < 0) {
22            System.out.println(x: "There are no real solutions.");
23        }
24        else if (discriminant == 0) {
25            root1 = -b / (2 * a);
26            System.out.println(x: "One real solution:");
27
28        }
29
30        root1 = -b / (2 * a);
31        System.out.println("x = " + root1);
32
33        root2 = (-b - Math.sqrt(discriminant)) / (2 * a);
34        System.out.println("x2 = " + root2);
35
36    }
37
38    sc.close();
39}

```
- IDE UI Elements:**
  - Top bar: Help, File, Temp, Chat.
  - Left sidebar: Problems, Output, Debug Console, Terminal, Ports.
  - Bottom status bar: Ln 1, Col 1, Spaces: 4, UTF-8, CRLF, Java, Go Live.
  - Right sidebar: Suggested Actions (Build Workspace, Show Config), Build with Agent (AI responses may be inaccurate, Generate Agent Instructions), and a terminal window showing command-line output.

2. Develop a Java program to create a class Student with members usn, name, an array credits and an array marks. Include methods to accept and display details and a method to calculate SGPA of a student.

```
import java.util.Scanner;

class Student {
    String usn;
    String name;
    int[] credits;
    int[] marks;
    int n;

    void accept() {
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter USN: ");
        usn = sc.nextLine();

        System.out.print("Enter Name: ");
        name = sc.nextLine();

        System.out.print("Enter number of subjects: ");
        n = sc.nextInt();

        credits = new int[n];
        marks = new int[n];

        for (int i = 0; i < n; i++) {
            System.out.print("Credits for subject " + (i + 1) + ": ");
            credits[i] = sc.nextInt();

            System.out.print("Marks for subject " + (i + 1) + ": ");
            marks[i] = sc.nextInt();
        }
    }
}
```

```
}

int getGradePoint(int m) {
    if (m >= 90) return 10;
    else if (m >= 80) return 9;
    else if (m >= 70) return 8;
    else if (m >= 60) return 7;
    else if (m >= 50) return 6;
    else return 0;
}

double calculateSGPA() {
    int totalCredits = 0;
    int sum = 0;

    for (int i = 0; i < n; i++) {
        sum += credits[i] * getGradePoint(marks[i]);
        totalCredits += credits[i];
    }

    return (double) sum / totalCredits;
}

void display() {
    System.out.println("USN: " + usn);
    System.out.println("Name: " + name);

    for (int i = 0; i < n; i++) {
        System.out.println("Subject " + (i + 1) + " Credits: " + credits[i] + " Marks: " + marks[i]);
    }

    System.out.println("SGPA: " + calculateSGPA());
}
}
```

```

public class StudentDemo {
    public static void main(String[] args) {
        Student s = new Student();
        s.accept();
        s.display();
    }
}

```

The screenshot shows a Java development environment with the following details:

- Terminal:** Shows the execution of the code. The user runs `javac StudentDemo.java` and then `java StudentDemo`. They are prompted for USN and Name, and then enter "1uf25cs090" and "Rahul". They then enter the number of subjects (3), credits for each subject (4, 5, 3), and marks for each subject (99, 83, 83). The output shows the USN, Name, SGPA (6.0), and a summary of subjects and marks.
- Code Editor:** Displays the `StudentDemo.java` file. The code defines a `Student` class with a `display()` method that prints USN, Name, and a table of subjects with their credits and marks. It also defines a `main` method that creates a `Student` object, calls `accept()`, and then `display()`.
- CHAT Panel:** A sidebar on the right shows AI-related features like "Build with Agent" and "SUGGESTED ACTIONS".
- Bottom Status Bar:** Shows the current line (Ln 66), column (Col 1), spaces (Spaces: 4), encoding (UTF-8), and file type (Java).

3.Create a class Book which contains four members: name, author, price, num\_pages. Include a constructor to set the values for the members. Include methods to set and get the details of the objects. Include a `toString()` method that could display the complete details of the book. Develop a Java program to create n book objects.

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

```
class Book {  
    String name;  
    String author;  
    double price;  
    int numPages;
```

```
Book(String name, String author, double price, int numPages) {  
    this.name = name;  
    this.author = author;  
    this.price = price;  
    this.numPages = numPages;  
}
```

```
void setDetails() {  
    Scanner scanner = new Scanner(System.in);  
    System.out.print("Enter book name: ");  
    this.name = scanner.nextLine();  
    System.out.print("Enter author name: ");  
    this.author = scanner.nextLine();  
    System.out.print("Enter price: ");  
    this.price = scanner.nextDouble();  
    System.out.print("Enter number of pages: ");  
    this.numPages = scanner.nextInt();  
    scanner.nextLine(); // consume leftover newline  
} //  ← Missing this closing brace!
```

```
void getDetails() {
```

```

        System.out.println("Book Name: " + name);
        System.out.println("Author: " + author);
        System.out.println("Price: Rs " + price);
        System.out.println("Number of Pages: " + numPages);
    }

    public String toString() {
        return "Book Details:\n" +
            "Name: " + name + "\n" +
            "Author: " + author + "\n" +
            "Price: Rs " + price + "\n" +
            "Number of Pages: " + numPages;
    }
}

class Books {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the number of books: ");
        int n = scanner.nextInt();
        scanner.nextLine(); // consume newline

        Book[] books = new Book[n];

        for (int i = 0; i < n; i++) {
            System.out.println("\nEnter details for Book " + (i + 1) + ":");

            books[i] = new Book("", "", 0.0, 0);
            books[i].setDetails();
        }

        System.out.println("\nDetails of all books:");
        for (int i = 0; i < n; i++) {
            System.out.println("\nBook " + (i + 1) + ":");

            books[i].getDetails();
        }
    }
}

```

```
        scanner.close();
    }
}
```

The screenshot shows a Java code editor interface with the following details:

- File Path:** C:\Users\student\Desktop\Booksjava
- Code Content:**

```
1 import java.util.Scanner;
2
3 class Book {
4     String name;
5     String author;
6     double price;
7     int numPages;
8
9     Book(String name, String author, double price, int numPages) {
10         this.name = name;
11         this.author = author;
12         this.price = price;
13         this.numPages = numPages;
14     }
15
16     void setDetails() {
17         Scanner scanner = new Scanner(System.in);
18         System.out.print("Enter book name: ");
19         this.name = scanner.nextLine();
20         System.out.print("Enter author name: ");
21         this.author = scanner.nextLine();
22         System.out.print("Enter price: ");
23         this.price = scanner.nextDouble();
24         System.out.print("Enter number of pages: ");
25         this.numPages = scanner.nextInt();
26     }
27
28 }
```
- Output Window:** Shows the execution of the code, displaying two books:

```
Book 1:
Book Details:
Name: Kill a Mockingbird
Author: Lee Harper
Price: Rs 100.0
Number of Pages: 300

Complete details of all books:

Book 2:
Book Details:
Name: Pride and Prejudice
Author: Jane Austen
Price: Rs 550.0
Number of Pages: 340
```
- Terminal:** Shows the command used to run the program: powershell
- Status Bar:** Includes file statistics (Ln 54, Col 14), encoding (UTF-8), and system information (ENG US, 11:46:49, 05-11-2025).

4. Develop a Java program to create an abstract class named Shape that contains two integers and an empty method named printArea(). Provide three classes named Rectangle, Triangle and Circle such that each one of the classes extends the class Shape. Each one of the classes contain only the method printArea() that prints the area of the given shape

```
abstract class Shape {  
    int x;  
    int y;  
  
    Shape(int x, int y) {  
        this.x = x;  
        this.y = y;  
    }  
  
    abstract void printArea();  
}  
  
class Rectangle extends Shape {  
    Rectangle(int l, int b) {  
        super(l, b);  
    }  
  
    void printArea() {  
        System.out.println("Area of Rectangle = " + (x * y));  
    }  
}  
  
class Triangle extends Shape {  
    Triangle(int b, int h) {  
        super(b, h);  
    }  
  
    void printArea() {  
        System.out.println("Area of Triangle = " + (0.5 * x * y));  
    }  
}
```

```
        }
    }

class Circle extends Shape {
    Circle(int r) {
        super(r, 0);
    }

    void printArea() {
        System.out.println("Area of Circle = " + (3.14 * r * r));
    }
}

public class ShapeDemo {
    public static void main(String[] args) {
        Shape r = new Rectangle(10, 5);
        Shape t = new Triangle(6, 4);
        Shape c = new Circle(7);

        r.printArea();
        t.printArea();
        c.printArea();
    }
}
```

A screenshot of a Java development environment, likely IntelliJ IDEA, showing a code editor, terminal, and AI integration features.

**Code Editor:** The main window displays `ShapeDemo.java` with the following code:

```
1 abstract class Shape {  
2     int x;  
3     int y;  
4  
5     Shape(int x, int y) {  
6         this.x = x;  
7         this.y = y;  
8     }  
9  
10    abstract void printArea();  
11 }  
12  
13 class Rectangle extends Shape {  
14     Rectangle(int l, int b) {  
15         super(l, b);  
16     }  
17  
18     void printArea() {  
19         System.out.println("Area of Rectangle = " + (x * y));  
20     }  
21 }  
22  
23 class Triangle extends Shape {  
24     Triangle(int b, int h) {  
25         super(b, h);  
26     }  
27  
28     void printArea() {  
29         System.out.println("Area of Triangle = " + (0.5 * x * y));  
30     }  
31 }
```

**Terminal:** The terminal shows the output of running the Java application:

```
PS C:\Users\Buste\UrbanTech-Offices\OneDrive\Desktop\Temp> java ShapeDemo  
>>  
Area of Rectangle = 50  
Area of Triangle = 12.0  
Area of Circle = 153.86  
PS C:\Users\Buste\UrbanTech-Offices\OneDrive\Desktop\Temp> []
```

**AI Integration:** A sidebar on the right titled "Build with Agent" includes a message about AI responses being inaccurate and a button to "Generate Agent Instructions".

**Suggested Actions:** A panel on the right lists "Build Workspace" and "Show Config".

5. Develop a Java program to create a class Bank that maintains two kinds of account for its customers, one called savings account and the other current account. The savings account provides compound interest and withdrawal facilities but no cheque book facility. The current account provides cheque book facility but no interest. Current account holders should also maintain a minimum balance and if the balance falls below this level, a service charge is imposed. Create a class Account that stores customer name, account number and type of account. From this derive the classes Cur-acct and Sav-acct to make them more specific to their requirements. Include the necessary methods in order to achieve the following tasks: a) Accept deposit from customer and update the balance. b) Display the balance. c) Compute and deposit interest d) Permit withdrawal and update the balance Check for the minimum balance, impose penalty if necessary and update the balance.

```
import java.util.Scanner;
class Account {
    protected String customerName;
    protected String accountNumber;
    protected String accountType;
    protected double balance;

    public Account(String customerName, String accountNumber, String
accountType, double initialBalance) {
        this.customerName = customerName;
        this.accountNumber = accountNumber;
        this.accountType = accountType;
        this.balance = initialBalance;
    }

    public void acceptDeposit(double amount) {
        if (amount > 0) {
            balance += amount;
            System.out.println("Deposited: " + amount);
        } else {
    }
```

```
        System.out.println("Invalid deposit amount.");
    }
}
public void showBalance() {
    System.out.println("Account Balance: " + balance);
}

public void allowWithdrawal(double amount) {
    System.out.println("Withdrawal not implemented for base account.");
}
}

class SavAcct extends Account {
    private double interestRate;

    public SavAcct(String customerName, String accountNumber, double initialBalance, double interestRate) {
        super(customerName, accountNumber, "Savings", initialBalance);
        this.interestRate = interestRate;
    }

    public void applyInterest() {
        double interest = balance * interestRate / 100;
        balance += interest;
        System.out.println("Interest of " + interest + " has been added to your
savings account.");
    }

    public void allowWithdrawal(double amount) {
        if (amount > 0 && balance >= amount) {
            balance -= amount;
            System.out.println("Withdrawn: " + amount);
        } else {
```

```
        System.out.println("Insufficient balance or invalid withdrawal
amount.");
    }
}
}

class CurAcct extends Account {
    private double minimumBalance;
    private double serviceCharge;

    public CurAcct(String customerName, String accountNumber, double
initialBalance, double minimumBalance, double serviceCharge) {
        super(customerName, accountNumber, "Current", initialBalance);
        this.minimumBalance = minimumBalance;
        this.serviceCharge = serviceCharge;
    }

    public void allowWithdrawal(double amount) {
        if (amount <= 0) {
            System.out.println("Invalid withdrawal amount.");
            return;
        }

        if (balance - amount >= minimumBalance) {
            balance -= amount;
            System.out.println("Withdrawn: " + amount);
        } else {
            balance -= serviceCharge;
            System.out.println("Balance fell below minimum. Service charge of "
+ serviceCharge + " applied.");
        }
    }

    public void showBalance() {
```

```
super.showBalance();
if (balance < minimumBalance) {
    System.out.println("Warning: Your balance is below the minimum
required. Service charges may apply.");
}
}

public void showChequeBookFacility() {
    System.out.println("Cheque book facility is available for current
account holders.");
}
}
```

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

        // Savings Account input
        System.out.println("Enter details for Savings Account:");
        System.out.print("Customer Name: ");
        String savName = sc.nextLine();
        System.out.print("Account Number: ");
        String savAccNo = sc.nextLine();
        System.out.print("Initial Balance: ");
        double savBalance = sc.nextDouble();
        System.out.print("Interest Rate (%): ");
        double interestRate = sc.nextDouble();
        sc.nextLine();

        SavAcct savingsAccount = new SavAcct(savName, savAccNo,
        savBalance, interestRate);
        System.out.println("Savings Account created for " + savName + ".");
        savingsAccount.showBalance();
```

```
System.out.print("Enter amount to deposit in savings account: ");
double savDeposit = sc.nextDouble();
savingsAccount.acceptDeposit(savDeposit);

savingsAccount.applyInterest();
savingsAccount.showBalance();
```

```
System.out.print("Enter amount to withdraw from savings account: ");
double savWithdraw = sc.nextDouble();
savingsAccount.allowWithdrawal(savWithdraw);
savingsAccount.showBalance();
```

```
sc.nextLine();
System.out.println("\nEnter details for Current Account:");
System.out.print("Customer Name: ");
String curName = sc.nextLine();
System.out.print("Account Number: ");
String curAccNo = sc.nextLine();
System.out.print("Initial Balance: ");
double curBalance = sc.nextDouble();
System.out.print("Overdraft Limit: ");
double overdraftLimit = sc.nextDouble();
System.out.print("Penalty Fee: ");
double penaltyFee = sc.nextDouble();
sc.nextLine();
```

```
CurAcct currentAccount = new CurAcct(curName, curAccNo,
curBalance, overdraftLimit, penaltyFee);
System.out.println("Current Account created for " + curName + ".");
currentAccount.showBalance();
```

```
System.out.print("Enter amount to deposit in current account: ");
double curDeposit = sc.nextDouble();
currentAccount.acceptDeposit(curDeposit);
```

```

System.out.print("Enter amount to withdraw from current account: ");
double curWithdraw = sc.nextDouble();
currentAccount.allowWithdrawal(curWithdraw);
currentAccount.showBalance();

System.out.print("Enter amount to withdraw from current account: ");
double curWithdraw2 = sc.nextDouble();
currentAccount.allowWithdrawal(curWithdraw2);
currentAccount.showBalance();

sc.close();
}

}

```

The screenshot shows an IDE window with the following details:

- File Menu:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Toolbar:** RUN AND DEBUG, Welcome, J bank.java 7.
- Code Editor:** Displays Java code for `J bank.java`. The code defines three classes: `Account`, `SavAcct`, and `CurAcct`. The `Account` class has a static variable `count` and a method `allowWithdrawal`. The `SavAcct` class extends `Account` and overrides `allowWithdrawal` to print withdrawal details. The `CurAcct` class extends `Account` and overrides `allowWithdrawal` to handle minimum balance and service charges. The code also includes `showBalance` and `showChequeBookFacility` methods.
- Left Sidebar:**
  - VARIABLES:** Shows variables for the current scope.
  - WATCH:** Shows watched variables.
  - CALL STACK:** Shows the call stack with threads: Finalizer, Signal Dispatcher, Attach Listener, Reference Handler, Notification Thread, Common-Cleaner, all marked as **RUNNING**.
  - BREAKPOINTS:** Shows uncaught exceptions and caught exceptions.
- Bottom Status Bar:** Shows the line number (Ln 39 Col 1), spaces (Spaces 4), encoding (UTF-8), file type (CRLF), Java, and a timestamp (11:54:20 15-10-2025).
- System Tray:** Shows weather (23°C, Mostly cloudy), system icons, and network status.

6.Create a package CIE which has two classes - Personal and Internals. The class Personal has members like usn, name, sem. The class Internals has an array that stores the internal marks scored in five courses of the current semester of the student. Create another package SEE which has the class External which is a derived class of Personal. This class has an array that stores the SEE marks scored in five courses of the current semester of the student. Import the two packages in a file that declares the final marks of n students in all five courses.

```
package CIE;
```

```
public class Personal {  
    public String usn;  
    public String name;  
    public int sem;  
  
    public Personal(String usn, String name, int sem) {  
        this.usn = usn;  
        this.name = name;  
        this.sem = sem;  
    }  
}
```

```
package CIE;
```

```
public class Internals {  
    public int[] marks = new int[5];  
  
    public Internals(int[] m) {  
        for (int i = 0; i < 5; i++) {  
            marks[i] = m[i];  
        }  
    }  
}
```

```
package SEE;

import CIE.Personal;

public class External extends Personal {
    public int[] marks = new int[5];

    public External(String usn, String name, int sem, int[] m) {
        super(usn, name, sem);
        for (int i = 0; i < 5; i++) {
            marks[i] = m[i];
        }
    }
}

import java.util.Scanner;
import CIE.*;
import SEE.*;

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

        System.out.print("Enter number of students: ");
        int n = sc.nextInt();

        for (int i = 0; i < n; i++) {
            System.out.print("USN: ");
            String usn = sc.next();

            System.out.print("Name: ");
            String name = sc.next();

            System.out.print("Semester: ");
            int sem = sc.nextInt();
        }
    }
}
```

```
int[] im = new int[5];
int[] em = new int[5];

System.out.println("Enter internal marks:");
for (int j = 0; j < 5; j++)
    im[j] = sc.nextInt();

System.out.println("Enter SEE marks:");
for (int j = 0; j < 5; j++)
    em[j] = sc.nextInt();

Internals in = new Internals(im);
External ex = new External(usn, name, sem, em);

System.out.println("Final Marks:");
for (int j = 0; j < 5; j++) {
    System.out.println("Subject " + (j + 1) + ": " + (in.marks[j] +
ex.marks[j]));
}
sc.close();
}
```

The screenshot shows a Java development environment with the following interface elements:

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Project Explorer:** Shows a tree view of the project structure under the "PROJECT" category, including CIE, SEE, and JAVA PROJECTS sections. "Main.java" is selected.
- Code Editor:** Displays the content of Main.java. The code reads student information from standard input and stores it in arrays. The code editor includes line numbers and syntax highlighting.
- Terminal:** Shows the command "java Main" being run, followed by the output of the program which asks for student details and prints them to the console.
- Suggested Actions:** A panel on the right suggests "Build Workspace" and "Show Config".
- Bottom Status:** Shows file statistics (Ln 12, Col 27), workspace settings (Spaces: 4, UTF-8, CRLF), and Java-related icons.

7. Write a program that demonstrates handling of exceptions in inheritance tree. Create a base class called “Father” and derived class called “Son” which extends the base class. In Father class, implement a constructor which takes the age and throws the exception WrongAge( ) when the input age<0. In Son class, implement a constructor that uses both father and son’s age and throws an exception if son’s age is >=father’s age

```
import java.util.Scanner;
class WrongAge extends Exception {
    public WrongAge(String message) {
        super(message);
    }
}
```

```
class Father {
    int fatherAge;

    public Father(int age) throws WrongAge {
        if (age < 0) {
            throw new WrongAge("Father's age cannot be negative!");
        }
        this.fatherAge = age;
    }
}
```

```
class Son extends Father {
    int sonAge;

    public Son(int fatherAge, int sonAge) throws WrongAge {
        super(fatherAge); // Call Father constructor
    }
}
```

```

        if (sonAge < 0) {
            throw new WrongAge("Son's age cannot be negative!");
        }

        if (sonAge >= super.fatherAge) {
            throw new WrongAge("Son's age cannot be greater than or equal to
Father's age!");
        }

        this.sonAge = sonAge;
    }
}

```

```

public class ExceptionInheritanceDemo {
    public static void main(String[] args) {
        int fage, sage;
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter father's age: ");
        fage=sc.nextInt();
        System.out.println("Enter Son's age: ");
        sage=sc.nextInt();
        try {
            Son s=new Son(fage,sage);
        }
        catch(WrongAge e){
            System.out.println("Exception caught "+e.getMessage());
        }
        try {
            Father s=new Father(fage);
        }
        catch(WrongAge a){
            System.out.println("Exception caught "+a.getMessage());
        }
    }
}

```

{}

The screenshot shows a Java IDE interface with a dark theme. The top menu bar includes File, Edit, Selection, View, Go, Run, Terminal, Help, and a search bar. The left sidebar shows project files: External.java, StudentMarks.java, ExceptionInheritanceDemo.java, and Son.java. The main editor area contains the following Java code:

```
1 import java.util.Scanner;
2 class WrongAge extends Exception {
3     public WrongAge(String message) {
4         super(message);
5     }
6 }
7
8
9 class Father {
10     int fatherAge;
11
12     public Father(int age) throws WrongAge {
13         if (age < 0) {
14             throw new WrongAge("Father's age cannot be negative!");
15         }
16         this.fatherAge = age;
17     }
18 }
19
20
21 class Son extends Father {
22     int sonAge; // Variable sonAge is never read
23
24     public Son(int fatherAge, int sonAge) throws WrongAge {
25         super(fatherAge); // Call Father constructor
26
27         if (sonAge < 0) {
28             throw new WrongAge("Son's age cannot be negative!");
29         }
30     }
31 }
```

The code editor highlights a warning: "Variable sonAge is never read". The terminal below shows the execution of the program:

```
Enter father's age:
-20
Enter Son's age:
5
Exception caught Father's age cannot be negative!
Exception caught Father's age cannot be negative!
```

8. Write a program which creates two threads, one thread displaying “BMS College of Engineering” once every ten seconds and another displaying “CSE” once every two seconds.

```
class Clg extends Thread {  
    public void run() {  
        try {  
            while (true) {  
                System.out.println("BMS College of Engineering");  
                Thread.sleep(10000);  
            }  
        } catch (InterruptedException e) {  
            System.out.println("Thread interrupted");  
        }  
    }  
}  
  
class CSE extends Thread {  
    public void run() {  
        try {  
            while (true) {  
                System.out.println("CSE");  
                Thread.sleep(2000);  
            }  
        } catch (InterruptedException e) {  
            System.out.println("Thread interrupted");  
        }  
    }  
}  
  
public class ThreadDemo {  
    public static void main(String[] args) {  
        Clg a = new Clg();  
        CSE b = new CSE();
```

```

    a.start();
    b.start();
}
}

```

The screenshot shows a Java IDE interface with two tabs open: `ThreadDemo.java` and `ThreadDemo.java`. The code in `ThreadDemo.java` defines two classes, `CSE` and `Clg`, which extend `Thread`. The `CSE` class has a `run()` method that prints "CSE" and sleeps for 2000ms. The `Clg` class has a `run()` method that prints "Clg". Both threads are started in the `main()` method.

```

1  class Clg extends Thread {
2     public void run() { Add @Override Annotation
10    }
11  }
13
14 class CSE extends Thread {
15     public void run() { Add @Override Annotation
16         try {
17             while (true) {
18                 System.out.println("CSE");
19                 Thread.sleep(2000); Thread.sleep called in loop
20             }
21         } catch (InterruptedException e) {
22             System.out.println("Thread interrupted");
23         }
24     }
}

```

The output window shows the following text:

```

CSE
CSE
CSE
CSE
BMS College of Engineering
CSE
CSE
CSE
CSE
CSE
CSE
CSE
BMS College of Engineering
CSE
CSE
CSE
CSE
CSE
CSE
BMS College of Engineering
CSE
CSE
CSE

```

The status bar at the bottom right indicates the current time is 11:18:50 on 10-12-2025.