

INDEX

Week & Date	Idx	Program	Sign
3	3.1	ATM Cash Withdrawal System	
	3.2	Restaurant Food Ordering System	
	3.3	Employee Payroll Processing System	
	3.4	Smart Highway Toll Plaza Management System	
	3.5	LED Display Board Pattern Generator.	

Problem -1

Problem: ATM Cash Withdrawal System

Scenario Description

A bank has installed a **Smart ATM Machine** that allows customers to withdraw money. For security reasons, a customer is given **a maximum of three attempts** to enter the correct ATM PIN.

The ATM should repeatedly ask the customer to enter the PIN until one of the following occurs:

- The correct PIN is entered.
- The customer exhausts all three attempts.

If the PIN is correct, the customer can proceed to withdraw money. The ATM must then verify whether:

- The withdrawal amount is less than or equal to the available account balance.
- The withdrawal amount is a multiple of ₹100.

If all conditions are satisfied, the withdrawal is successful; otherwise, an appropriate message is displayed.

This scenario naturally requires a **while loop**, as the number of PIN entry attempts is controlled by a condition.

Input Data

Variable	Value
Customer Name	Rahul Verma
Account Number	1023456789
Account Balance	₹45,000
Actual PIN	4321
Maximum Attempts	3
User PINs Entered	1234, 5678, 4321
Withdrawal Amount	₹8,000

Problem Statement

Write a Java program to perform the following tasks.

Step 1 – PIN Verification Using `while`

Allow the customer to enter the PIN until:

- The correct PIN is entered, **or**
- Three attempts have been completed.

Use a `while` loop.

Step 2 – Invalid PIN

If the entered PIN is incorrect,

Display

`Invalid PIN. Try Again.`

Increase the attempt count.

Step 3 – Card Blocking

If three incorrect attempts are made,

Display

`ATM Card Blocked.`
`Please Contact Your Bank.`

Terminate the program.

Step 4 – Withdrawal

If the PIN is correct,

Check whether:

- $\text{Withdrawal amount} \leq \text{Account Balance}$

- Withdrawal amount is divisible by 100

If both conditions are true,

Display

Transaction Successful

Otherwise display the appropriate error message.

Java Solution

```
import java.util.Scanner;

public class ATMSystem {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        String customerName = "Rahul Verma";
        long accountNumber = 1023456789L;

        int actualPIN = 4321;
        int enteredPIN;

        int attempts = 0;
        int maxAttempts = 3;

        double balance = 45000;

        boolean authenticated = false;

        // PIN Verification
        while (attempts < maxAttempts) {

            System.out.print("Enter ATM PIN: ");
            enteredPIN = sc.nextInt();

            if (enteredPIN == actualPIN) {
                authenticated = true;
                break;
            }

            System.out.println("Invalid PIN. Try Again.");

            attempts++;
        }

        if (!authenticated) {

            System.out.println("\nATM Card Blocked.");
            System.out.println("Please Contact Your Bank.");
            return;
        }
    }
}
```

```

    }

    System.out.print("\nEnter Withdrawal Amount: ₹");
    double amount = sc.nextDouble();

    if (amount > balance) {

        System.out.println("Insufficient Balance.");

    } else if (amount % 100 != 0) {

        System.out.println("Amount should be in multiples of ₹100.");

    } else {

        balance -= amount;

        System.out.println("\n===== TRANSACTION SUCCESSFUL =====");

        System.out.println("Customer Name : " + customerName);
        System.out.println("Account No      : " + accountNumber);
        System.out.println("Amount          : ₹" + amount);
        System.out.println("Remaining Bal   : ₹" + balance);

    }

    sc.close();
}
}

```

Sample Output 1

```

Enter ATM PIN: 1234
Invalid PIN. Try Again.

Enter ATM PIN: 5678
Invalid PIN. Try Again.

Enter ATM PIN: 4321

Enter Withdrawal Amount: ₹8000

===== TRANSACTION SUCCESSFUL =====

Customer Name : Rahul Verma
Account No    : 1023456789
Amount       : ₹8000.0
Remaining Bal : ₹37000.0

```

Sample Output 2

```

Enter ATM PIN: 1111
Invalid PIN. Try Again.

Enter ATM PIN: 2222

```

Invalid PIN. Try Again.

Enter ATM PIN: 3333
Invalid PIN. Try Again.

ATM Card Blocked.
Please Contact Your Bank.

Actual Executed Output:

```
Enter Your Name :  
Hemalatha  
Enter Your A/c Number :  
12345678  
Enter ATM PIN: 4321  
  
Enter Withdrawal Amount: ₹15000  
  
===== TRANSACTION SUCCESSFUL =====  
Customer Name : Hemalatha  
Account No      : 12345678  
Amount          : ₹15000.0  
Remaining Bal   : ₹30000.0
```

```
Enter Your Name :  
Dhanunjay  
Enter Your A/c Number :  
56781234  
Enter ATM PIN: 4567  
Invalid PIN. Try Again.  
Enter ATM PIN: 2345  
Invalid PIN. Try Again.  
Enter ATM PIN: 8765  
Invalid PIN. Try Again.  
  
ATM Card Blocked.  
Please Contact Your Bank.
```

Concepts Covered

This problem helps students practice:

- while loop
- Loop control using a condition
- Counter variable (attempts)
- break statement
- Boolean flag (authenticated)
- Nested if-else statements
- User input using Scanner

- Real-world authentication and transaction processing
-

Learning Outcomes

After solving this problem, students will be able to:

- Use a **while loop** for repeated user input until a condition is met.
- Control loop execution using counters and Boolean flags.
- Apply decision-making with `if-else` inside a loop.
- Use `break` to terminate a loop when a desired condition is satisfied.
- Design a realistic ATM authentication system similar to those used in banking applications.

Problem – 2

Problem: Restaurant Food Ordering System

Scenario Description

A popular restaurant has introduced a **Self-Service Food Ordering Kiosk** where customers can place multiple food orders without the assistance of a waiter.

After selecting an item, the customer is asked whether they want to order another item.

Since **every customer should be able to place at least one order**, the system must display the menu **at least once**, regardless of the customer's decision to continue.

Restaurant Menu

Item No.	Food Item	Price (₹)
1	Veg Burger	120
2	Pizza	250
3	Sandwich	150
4	Pasta	200
5	Coffee	80

Problem Statement

Write a Java program that performs the following tasks.

Step 1 – Display the Menu

Display the food menu.

Step 2 – Accept Customer Order

Allow the customer to:

- Enter the item number.
- Enter the quantity.

Calculate the cost of the selected item and add it to the total bill.

Step 3 – Invalid Item

If the customer enters an invalid item number,

Display

Invalid Item Number.

Do not add anything to the bill.

Step 4 – Continue Ordering

Ask the customer:

Do you want to order another item? (Y/N)

If the customer enters **Y** or **y**, display the menu again.

Otherwise, exit the loop.

Step 5 – Display Final Bill

Display:

- Total number of items ordered
 - Total bill amount
 - Thank-you message
-

Java Solution

```
import java.util.Scanner;

public class RestaurantOrdering {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        int choice, quantity;
        double totalBill = 0;
        int totalItems = 0;
        char continueOrder;

        do {

            System.out.println("\n===== RESTAURANT MENU =====");
            System.out.println("1. Veg Burger    - ₹120");
            System.out.println("2. Pizza      - ₹250");
            System.out.println("3. Sandwich   - ₹150");
            System.out.println("4. Pasta      - ₹200");
            System.out.println("5. Coffee     - ₹80");

            System.out.print("\nEnter Item Number: ");
            choice = sc.nextInt();

            System.out.print("Enter Quantity: ");
            quantity = sc.nextInt();

            double itemCost = 0;

            switch (choice) {
```

```

        case 1:
            itemCost = 120 * quantity;
            break;

        case 2:
            itemCost = 250 * quantity;
            break;

        case 3:
            itemCost = 150 * quantity;
            break;

        case 4:
            itemCost = 200 * quantity;
            break;

        case 5:
            itemCost = 80 * quantity;
            break;

        default:
            System.out.println("Invalid Item Number.");
            itemCost = 0;
    }

    totalBill += itemCost;

    if (choice >= 1 && choice <= 5)
        totalItems += quantity;

    System.out.print("\nDo you want to order another item? (Y/N): ");
    continueOrder = sc.next().charAt(0);

    } while (continueOrder == 'Y' || continueOrder == 'y');

    System.out.println("\n===== FINAL BILL =====");
    System.out.println("Total Items Ordered : " + totalItems);
    System.out.println("Total Bill          : ₹" + totalBill);
    System.out.println("Thank you! Visit Again.");

    sc.close();
}
}

```

Sample Output

===== RESTAURANT MENU =====

```

1. Veg Burger    - ₹120
2. Pizza         - ₹250
3. Sandwich     - ₹150
4. Pasta        - ₹200
5. Coffee       - ₹80

```

Enter Item Number: 2

Enter Quantity: 2

Do you want to order another item? (Y/N): Y

```
===== RESTAURANT MENU =====
1. Veg Burger    - ₹120
2. Pizza         - ₹250
3. Sandwich      - ₹150
4. Pasta         - ₹200
5. Coffee        - ₹80

Enter Item Number: 5
Enter Quantity: 3

Do you want to order another item? (Y/N): N

===== FINAL BILL =====
Total Items Ordered : 5
Total Bill          : ₹740.0
Thank you! Visit Again.
```

Actual Executed Output:

```
D:\AY2026_27\JAVA\LAB\Week3>java RestaurantOrdering
```

```
===== RESTAURANT MENU =====
```

- 1. Veg Burger - Rs.120
- 2. Pizza - Rs.250
- 3. Sandwich - Rs.150
- 4. Pasta - Rs.200
- 5. Coffee - Rs.80

Enter Item Number: 6

Enter Quantity: 1

Invalid Item Number.

Do you want to order another item? (Y/N): Y

```
===== RESTAURANT MENU =====
```

- 1. Veg Burger - Rs.120
- 2. Pizza - Rs.250
- 3. Sandwich - Rs.150
- 4. Pasta - Rs.200
- 5. Coffee - Rs.80

Enter Item Number: 4

Enter Quantity: 1

Do you want to order another item? (Y/N): Y

```
===== RESTAURANT MENU =====
```

- 1. Veg Burger - Rs.120
- 2. Pizza - Rs.250
- 3. Sandwich - Rs.150
- 4. Pasta - Rs.200
- 5. Coffee - Rs.80

Enter Item Number: 5

Enter Quantity: 1

Do you want to order another item? (Y/N): N

```
===== FINAL BILL =====
```

Total Items Ordered : 2

Total Bill : Rs.280.0

Thank you! Visit Again.

Concepts Covered

- `do-while` loop
 - `switch` statement
 - User input using `Scanner`
 - Arithmetic calculations
 - Conditional statements (`if`)
 - Accumulating totals
 - Real-world menu-driven programming
-

Learning Outcomes

After completing this exercise, students will be able to:

- Understand when a **`do-while` loop** is more appropriate than a `while` loop.
- Build a **menu-driven application** that repeatedly performs tasks based on user choice.
- Combine **loops, switch, and conditional statements** in a practical business scenario.
- Develop interactive console applications similar to restaurant ordering systems, shopping carts, and ATM menus.

Problem 3

Problem: Employee Payroll Processing System

Scenario Description

A software company has **50 employees**. At the end of every month, the HR department must process the salaries of all employees.

For each employee, the payroll system receives:

- Employee ID
- Employee Name
- Basic Salary
- Overtime Hours

The company follows these payroll rules:

- Overtime Pay = ₹500 per hour
- Income Tax = 10% of Gross Salary (if Gross Salary > ₹50,000)
- Otherwise, no tax is deducted.

Input Data

Assume the HR manager enters details for **5 employees**.

Employee	Basic Salary	Overtime Hours
1	₹40,000	5
2	₹55,000	8
3	₹48,000	2
4	₹62,000	6
5	₹35,000	0

Problem Statement

Write a Java program to perform the following tasks.

Step 1 – Process Employee Details

Use a **for loop** to process the salary details of all employees.

For each employee:

- Read Employee ID
 - Read Employee Name
 - Read Basic Salary
 - Read Overtime Hours
-

Step 2 – Calculate Salary

Calculate:

Overtime Pay = Overtime Hours × ₹500

Gross Salary = Basic Salary + Overtime Pay

Step 3 – Calculate Tax

If

Gross Salary > ₹50,000

Tax

10% of Gross Salary

Else

Tax = ₹0

Step 4 – Calculate Net Salary

Net Salary = Gross Salary - Tax

Step 5 – Display Employee Salary Slip

Display:

- Employee ID
 - Employee Name
 - Gross Salary
 - Tax
 - Net Salary
-

Step 6 – Display Payroll Summary

After processing all employees, display:

- Total Employees Processed
 - Total Payroll Amount
 - Highest Net Salary
 - Average Net Salary
-

Java Solution

```
import java.util.Scanner;

public class PayrollSystem {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        int numberOfEmployees = 5;
```

```

double totalPayroll = 0;
double highestSalary = 0;

System.out.println("===== EMPLOYEE PAYROLL SYSTEM =====");

for (int i = 1; i <= numberOfEmployees; i++) {

    System.out.println("\nEmployee " + i);

    System.out.print("Employee ID : ");
    int empId = sc.nextInt();

    sc.nextLine();

    System.out.print("Employee Name : ");
    String empName = sc.nextLine();

    System.out.print("Basic Salary : ");
    double basicSalary = sc.nextDouble();

    System.out.print("Overtime Hours : ");
    int overtimeHours = sc.nextInt();

    double overtimePay = overtimeHours * 500;

    double grossSalary = basicSalary + overtimePay;

    double tax = 0;

    if (grossSalary > 50000)
        tax = grossSalary * 0.10;

    double netSalary = grossSalary - tax;

    totalPayroll += netSalary;

    if (netSalary > highestSalary)
        highestSalary = netSalary;

    System.out.println("\n----- Salary Slip -----");

    System.out.println("Employee ID : " + empId);
    System.out.println("Employee Name : " + empName);
    System.out.println("Gross Salary : ₹" + grossSalary);
    System.out.println("Tax : ₹" + tax);
    System.out.println("Net Salary : ₹" + netSalary);
}

System.out.println("\n===== PAYROLL SUMMARY =====");

System.out.println("Employees Processed : " + numberOfEmployees);
System.out.println("Total Payroll : ₹" + totalPayroll);
System.out.println("Highest Net Salary : ₹" + highestSalary);
System.out.println("Average Net Salary : ₹"
    + (totalPayroll / numberOfEmployees));

sc.close();
}
}

```

Sample Output

===== EMPLOYEE PAYROLL SYSTEM =====

Employee 1

Employee ID : 101
Employee Name : Rahul
Basic Salary : 40000
Overtime Hours : 5

----- Salary Slip -----

Employee ID : 101
Employee Name : Rahul
Gross Salary : ₹42500.0
Tax : ₹0.0
Net Salary : ₹42500.0

...

===== PAYROLL SUMMARY =====

Employees Processed : 5
Total Payroll : ₹223650.0
Highest Net Salary : ₹58500.0
Average Net Salary : ₹44730.0

Actual Executed Output:

```
D:\AY2026_27\JAVA\LAB\Week3>javac PayrollSystem.java
```

```
D:\AY2026_27\JAVA\LAB\Week3>java PayrollSystem
```

```
Enter no. Of Employees :
```

```
2
```

```
===== EMPLOYEE PAYROLL SYSTEM =====
```

```
Employee 1
```

```
Employee ID : 101
```

```
Employee Name : Hema
```

```
Basic Salary : 50000
```

```
Overtime Hours : 5
```

```
----- Salary Slip -----
```

```
Employee ID : 101
```

```
Employee Name : Hema
```

```
Gross Salary : Rs.52500.0
```

```
Tax : Rs.5250.0
```

```
Net Salary : Rs.47250.0
```

```
Employee 2
```

```
Employee ID : 102
```

```
Employee Name : latha
```

```
Basic Salary : 30000
```

```
Overtime Hours : 0
```

```
----- Salary Slip -----
```

```
Employee ID : 102
```

```
Employee Name : latha
```

```
Gross Salary : Rs.30000.0
```

```
Tax : Rs.0.0
```

```
Net Salary : Rs.30000.0
```

```
===== PAYROLL SUMMARY =====
```

```
Employees Processed : 2
```

```
Total Payroll : Rs.77250.0
```

```
Highest Net Salary : Rs.47250.0
```

```
Average Net Salary : Rs.38625.0
```

Java Concepts Covered

- `for` loop
- User input using `Scanner`
- `if` statement
- Arithmetic operations
- Variables and data types
- Accumulator pattern (`totalPayroll`)
- Finding the maximum value (`highestSalary`)
- Computing an average after loop completion

Learning Outcomes

After completing this exercise, students will be able to:

- Use a **`for` loop** when the number of iterations is predetermined.
- Process records sequentially in a real-world business application.
- Maintain running totals, calculate averages, and determine maximum values.
- Combine loops with conditional statements to implement payroll logic.
- Develop practical console applications similar to payroll systems used in organizations.

Extension Activity: Enhance the program by categorizing employees based on their net salary (e.g., Gold, Silver, Bronze payroll bands), or by tracking the number of employees who paid income tax. This encourages students to perform additional calculations within the same `for` loop.

Problem 4

Problem: Smart Highway Toll Plaza Management System

Scenario Description

A national highway authority has developed a **Smart Toll Plaza Management System** to automate toll collection and monitor traffic.

During a day, vehicles pass through the toll plaza from different lanes. The system records the type of each vehicle and calculates the toll amount. At the end of the day, the management wants to analyse the traffic flow.

This scenario is ideal for demonstrating **different variations of the `for` loop** in Java.

Toll Charges

Vehicle Type	Toll Charge
Car	₹100
Bus	₹250
Truck	₹350
Bike	₹50

Input Data

Assume:

- Number of vehicles = **10**
 - Vehicle types are entered by the operator.
-

Problem Statement

Write a Java program to perform the following tasks.

Part 1 – Standard `for` Loop

Use a **standard `for` loop** to process the details of **10 vehicles**.

For each vehicle:

- Read the vehicle type.
- Calculate the toll amount.

- Display the toll for that vehicle.
- Add it to the total collection.

Example:

```
for(int i = 1; i <= 10; i++)
```

Part 2 – Multiple Initialization and Update Expressions

Use a `for` loop with **multiple variables** to display the lane number and remaining vehicles.

Example:

```
for(int lane = 1, remaining = 10;  
    lane <= 10;  
    lane++, remaining--)
```

Display:

```
Lane 1 - Remaining Vehicles: 10  
Lane 2 - Remaining Vehicles: 9  
...  
Lane 10 - Remaining Vehicles: 1
```

Part 3 – Omitted Initialization

Assume the operator has already initialized:

```
int i = 1;
```

Use

```
for(; i <= 5; i++)
```

to display

```
Checking CCTV Camera 1  
Checking CCTV Camera 2  
...  
Checking CCTV Camera 5
```

Part 4 – Omitted Update

Use

```
for(int counter = 1; counter <= 5; )
```

Increment the counter **inside the loop**.

Display

Generating Daily Report...

five times.

Part 5 – Empty Body (`for` Loop Ending with Semicolon)

The toll plaza has collected:

₹100 every hour

Use

```
int total = 0;
for(int hour = 1; hour <= 24; total += 100, hour++);
```

After the loop,

Display

Total Collection for 24 Hours

Part 6 – Infinite `for` Loop

Display

```
for(;;)
```

Explain that it continuously monitors emergency vehicles until the program is manually terminated.

(Do not execute this loop in the program.)

Java Solution

```
import java.util.Scanner;

public class TollPlazaSystem {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        double totalCollection = 0;
```

```

System.out.println("==== SMART TOLL PLAZA =====");

// Part 1 - Standard for Loop
for (int i = 1; i <= 10; i++) {

    System.out.print("\nEnter Vehicle Type (Car/Bus/Truck/Bike): ");
    String vehicle = sc.next();

    double toll = 0;

    switch(vehicle.toLowerCase()) {

        case "car":
            toll = 100;
            break;

        case "bus":
            toll = 250;
            break;

        case "truck":
            toll = 350;
            break;

        case "bike":
            toll = 50;
            break;

        default:
            System.out.println("Invalid Vehicle");
    }

    totalCollection += toll;

    System.out.println("Toll = ₹" + toll);
}

System.out.println("\nTotal Collection = ₹" + totalCollection);

// Part 2
System.out.println("\nLane Status");

for(int lane = 1, remaining = 10;
    lane <= 10;
    lane++, remaining--) {

    System.out.println(
        "Lane " + lane +
        " Remaining Vehicles : " + remaining);
}

// Part 3
System.out.println("\nChecking Cameras");

int i = 1;

for(; i <= 5; i++) {

    System.out.println("Checking CCTV Camera " + i);
}

```

```

// Part 4
System.out.println("\nGenerating Reports");

for(int counter = 1; counter <= 5; ) {

    System.out.println("Generating Daily Report...");

    counter++;

}

// Part 5
int total = 0;

for(int hour = 1; hour <= 24;
    total += 100, hour++);

System.out.println("\n24-Hour Collection = ₹" + total);

sc.close();
}
}

```

Sample Output

```

===== SMART TOLL PLAZA =====

Vehicle 1
Car
Toll = ₹100

Vehicle 2
Truck
Toll = ₹350

...

Total Collection = ₹1950

Lane 1 Remaining Vehicles : 10
Lane 2 Remaining Vehicles : 9
...

Checking CCTV Camera 1
Checking CCTV Camera 2
...

Generating Daily Report...
Generating Daily Report...
...

24-Hour Collection = ₹2400

```

Variations Covered

for Loop Variation	Example in the Scenario
Standard <code>for</code> loop	Processing 10 vehicles
Multiple initialization and update	Lane number and remaining vehicles
Initialization omitted	CCTV camera inspection
Update omitted	Report generation
Empty-body <code>for</code> loop	Calculating 24-hour collection
Infinite <code>for (;;)</code> (<i>conceptual</i>)	Continuous emergency vehicle monitoring

Learning Outcomes

After completing this exercise, students will be able to:

- Use the standard `for` loop for fixed iterations.
- Initialize and update multiple loop variables.
- Omit the initialization or update expressions when appropriate.
- Understand the purpose of an empty-body `for` loop.
- Recognize the use case of an infinite `for (;;)` loop in systems that require continuous monitoring.
- Apply different `for` loop variations in a practical, real-world application.

Problem 5

Problem: LED Display Board Pattern Generator

Scenario Description

A city is hosting an **Annual Cultural Festival**. At the entrance of the venue, a large **LED Display Board** is installed to welcome visitors. The display board can show different decorative patterns such as squares, triangles, pyramids, and diamonds before each event.

The event organizer wants a simple Java program that allows the operator to choose a pattern from a menu and display it on the LED board.

Since each pattern consists of rows and columns, the program should use **nested for loops** and a **switch statement**.

Problem Statement

Write a Java program that repeatedly displays the following menu until the user chooses **Exit**.

```
===== LED DISPLAY MENU =====
```

1. Solid Square
2. Increasing Triangle
3. Decreasing Triangle
4. Right-Aligned Triangle
5. Pyramid
6. Diamond
7. Exit

After the user enters a choice:

- Display the selected pattern.
 - Assume the number of rows is **5**.
 - Continue displaying the menu until the user selects **7 (Exit)**.
-

Sample Output

If the user selects 1

```
*****
*****
*****
*****
*****
```

If the user selects 2

```
*
**
***
****
*****
```

If the user selects 3

```
*****
****
***
**
*
```

If the user selects 4

```
  *
  **
 ***
****
*****
```

If the user selects 5

```
  *
  **
 ***
****
*****
*****
*****
```

If the user selects 6

```
  *
  **
 ***
****
*****
*****
*****
*****
****
 ***
  **
  *
```

Simple Java Solution

```
import java.util.Scanner;

public class PatternGenerator {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        int choice;
        int rows = 5;

        do {

            System.out.println("\n===== LED DISPLAY MENU =====");
            System.out.println("1. Solid Square");
            System.out.println("2. Increasing Triangle");
            System.out.println("3. Decreasing Triangle");
            System.out.println("4. Right-Aligned Triangle");
            System.out.println("5. Pyramid");
            System.out.println("6. Diamond");
            System.out.println("7. Exit");

            System.out.print("Enter your choice: ");
            choice = sc.nextInt();

            switch (choice) {

                case 1:
```

```

    for (int i = 1; i <= rows; i++) {
        for (int j = 1; j <= rows; j++) {
            System.out.print("*");
        }
        System.out.println();
    }
    break;

```

case 2:

```

    for (int i = 1; i <= rows; i++) {
        for (int j = 1; j <= i; j++) {
            System.out.print("*");
        }
        System.out.println();
    }
    break;

```

case 3:

```

    for (int i = rows; i >= 1; i--) {
        for (int j = 1; j <= i; j++) {
            System.out.print("*");
        }
        System.out.println();
    }
    break;

```

case 4:

```

    for (int i = 1; i <= rows; i++) {

        for (int j = 1; j <= rows - i; j++) {
            System.out.print(" ");
        }

        for (int j = 1; j <= i; j++) {
            System.out.print("*");
        }

        System.out.println();
    }
    break;

```

case 5:

```

    for (int i = 1; i <= rows; i++) {

        for (int j = 1; j <= rows - i; j++) {
            System.out.print(" ");
        }

        for (int j = 1; j <= (2 * i - 1); j++) {
            System.out.print("*");
        }

        System.out.println();
    }

    break;

```

```

case 6:

    // Upper Half
    for (int i = 1; i <= rows; i++) {

        for (int j = 1; j <= rows - i; j++) {
            System.out.print(" ");
        }

        for (int j = 1; j <= (2 * i - 1); j++) {
            System.out.print("*");
        }

        System.out.println();
    }

    // Lower Half
    for (int i = rows - 1; i >= 1; i--) {

        for (int j = 1; j <= rows - i; j++) {
            System.out.print(" ");
        }

        for (int j = 1; j <= (2 * i - 1); j++) {
            System.out.print("*");
        }

        System.out.println();
    }

    break;

case 7:

    System.out.println("Thank you! Exiting Pattern
Generator...");
    break;

default:

    System.out.println("Invalid Choice!");

}

} while (choice != 7);

sc.close();
}

```

Java Concepts Covered

This single program demonstrates:

-  for loop
-  Nested for loops
-  do-while loop

- ☒ switch statement
 - ☒ Menu-driven programming
 - ☒ Pattern generation
 - ☒ User input using Scanner
-