RUPEE COIN CHANGER

mini project submitted by ‘ ASHISH MEWARA’

(24MCA20482)

Introduction

In day-to-day life, especially in retail and banking environments, making accurate and efficient change using the **least number of coins** is a common task. The **Coin Changer** project addresses this problem through a simple yet effective Java application that takes a given amount in rupees and calculates the **minimum number of coins** needed to make up that amount using standard Indian coin denominations (₹20, ₹10, ₹5, ₹2, and ₹1).

This project is designed as a **console-based utility tool** developed in Java. It applies the **greedy algorithm** to achieve optimal results by always choosing the highest denomination available at each step. The program is interactive and user-friendly, offering continuous usage in a loop until the user decides to exit.

Through this project, core programming concepts such as **arrays**, **loops**, **methods**, **conditional logic**, and **user input validation** are put into practical use. It is a great demonstration of how simple algorithms can be used to solve real-world problems in an efficient way.

The Coin Changer project not only enhances programming skills but also provides insights into algorithmic thinking and problem-solving in a structured manner.

Abstracts

The **Coin Changer** is a Java-based console application designed to compute the **minimum number of coins** required to make up a given amount in rupees. It uses a **greedy algorithm** that selects the highest possible denomination at each step to efficiently break down the amount using standard Indian coins (₹20, ₹10, ₹5, ₹2, ₹1).

This project aims to simulate a real-world scenario commonly encountered in financial transactions, where returning change with the least number of coins is both practical and time-saving. The program is interactive, performs input validation, and provides accurate results in a user-friendly manner. It demonstrates essential programming concepts such as loops, conditionals, arrays, and method usage, making it a valuable learning project for beginners.

Overall, the Coin Changer offers a simple yet effective solution for coin change problems, showcasing the practical application of algorithms in daily life situations.

Problem Statement

In real-life scenarios such as retail transactions, banking, and vending systems, it is often necessary to return change to customers using **the least number of coins** possible. Manually determining the optimal combination of coins can be inefficient and error-prone, particularly when speed and accuracy are critical

.

There is a need for an application that can **automatically calculate the minimum number of coins required** to make up a specific amount in rupees using standard Indian coin denominations (₹20, ₹10, ₹5, ₹2, and ₹1). The solution should be simple, efficient, and user-friendly, providing accurate results for any valid positive amount entered by the user.

This project aims to develop a **Java-based console application** that:

- Accepts a positive integer input representing the amount in rupees.
- Validates user input to handle invalid or negative values.
- Applies a **greedy algorithm** to compute the minimum number of coins needed.
- Displays the breakdown of each coin denomination used.
- Allows the user to perform multiple conversions in a single session.

Challenges to be addressed

While developing the **Coin Changer** application, several challenges and considerations had to be addressed to ensure the solution is efficient, user-friendly, and reliable:

1. Input Validation

- Ensuring that the user enters only **positive integers**.
- Handling invalid inputs such as letters, symbols, negative numbers, or empty values without crashing the program.

2. Optimal Coin Calculation

- Implementing a correct and efficient **greedy algorithm** to ensure the minimum number of coins is used.
- Making sure that the algorithm works accurately for all possible values, regardless of the input size.

3. User Interaction and Looping

- Providing a smooth and clear user experience through the console.
- Allowing the user to **repeat the process** without restarting the program.
- Handling user responses like “yes”, “no”, or their variants correctly.

4. Fixed Denomination Limitation

- The program currently works with predefined Indian coin denominations.
- Adapting the algorithm in the future to accept **custom denominations** would require structural changes.

5. Scalability and Extensibility

- Designing the logic in a way that allows future upgrades like:
    - Adding support for decimal values (paise).
    - Introducing GUI or file handling capabilities.
    - Supporting limited coin supply or real-world constraints.

JAVA CODE

```
import java.util.Scanner;

public class CoinChanger {


    static int[] coins = {20, 10, 5, 2, 1};

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

        System.out.println("==================================");
        System.out.println("        🏛 COIN CHANGER        ");
        System.out.println("==================================");

        boolean runAgain = true;

        while (runAgain) {
            System.out.print("\nEnter amount in rupees (positive integer): ");

            if (!scanner.hasNextInt()) {
                System.out.println("❌ Invalid input. Please enter a number.");
                scanner.next(); // Clear invalid input
                continue;
            }

            int amount = scanner.nextInt();

            if (amount <= 0) {
                System.out.println("❌ Amount must be greater than 0.");
                continue;
            }

            System.out.println("\nCalculating minimum coins needed for ₹" + amount
+ "...");
            getMinimumCoins(amount);

            // Ask user if they want to continue
            System.out.print("\nWould you like to convert another amount? (yes/no):
");
```

```
            scanner.nextLine(); // consume leftover newline
            String choice = scanner.nextLine().trim().toLowerCase();

            if (!choice.equals("yes") && !choice.equals("y")) {
                runAgain = false;
                System.out.println("\nThank you for using Coin Changer! 😊");
            }
        }

        scanner.close();
    }

    // Method to calculate and display minimum coins
    public static void getMinimumCoins(int amount) {
        int totalCoins = 0;

        for (int coin : coins) {
            int count = amount / coin;

            if (count > 0) {
                System.out.println("→ " + coin + " rs coin x " + count);
                amount %= coin;
                totalCoins += count;
            }
        }

        System.out.println("☑ Total coins used: " + totalCoins);
    }
}
```

Breakdown of the Code

Let me walk through the key components of the simple Coin Change Dispenser implementation:

Import and Class Declaration

java

Copy

```
import java.util.Scanner;

public class CoinChanger {
```

- Imports the Scanner class to get user input
- Defines a class named CoinChanger

Static Variable

java

Copy

```
static int[] coins = {20, 10, 5, 2, 1};
```

- Declares a static array of integers called coins
- Contains the coin denominations in descending order (20, 10, 5, 2, 1 rupees)
- Being static means it's shared across all instances of the class

Main Method

java

Copy

```
public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    // Rest of the code...
}
```

- Entry point of the program
- Creates a Scanner object to read user input

Welcome Message

java

Copy

```
System.out.println("==================================");
System.out.println("          🪙 COIN CHANGER         ");
System.out.println("==================================");
```

- Displays a formatted title with decorative borders

Main Program Loop

java

Copy

```
boolean runAgain = true;
while (runAgain) {
    // Program logic here
}
```

- Creates a boolean flag runAgain initialized to true
- The while loop continues executing as long as runAgain is true

User Input and Validation

java

Copy

```
System.out.print("\nEnter amount in rupees (positive integer): ");
if (!scanner.hasNextInt()) {
    System.out.println("❌ Invalid input. Please enter a number.");
```

```
            scanner.next(); // Clear invalid input
            continue;
        }
        int amount = scanner.nextInt();
        if (amount <= 0) {
            System.out.println("❌ Amount must be greater than 0.");
            continue;
        }
```

- Prompts user for an amount
- Checks if input is an integer using hasNextInt()
- If not an integer, displays error message, clears invalid input, and restarts the loop with continue
- Reads the integer value with nextInt()
- Validates that amount is positive, otherwise shows error and restarts the loop

Processing and Output

```
        System.out.println("\nCalculating minimum coins needed for ₹" + amount + "...");
        getMinimumCoins(amount);
```

- Displays a message showing the amount being calculated
- Calls the getMinimumCoins() method to do the actual calculation

Continue or Exit

```
        System.out.print("\nWould you like to convert another amount? (yes/no): ");
        scanner.nextLine(); // consume leftover newline
        String choice = scanner.nextLine().trim().toLowerCase();
        if (!choice.equals("yes") && !choice.equals("y")) {
```

```
        runAgain = false;
        System.out.println("\nThank you for using Coin Changer! 😊");
    }
```

- Asks if user wants to continue
- The first nextLine() consumes the leftover newline from previous nextInt() call
- Reads user choice and converts to lowercase
- If user enters anything other than "yes" or "y", sets runAgain to false to exit the loop
- Displays a goodbye message when exiting

Cleanup

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

- Properly closes the Scanner object to avoid resource leaks

Coin Calculation Method

```
public static void getMinimumCoins(int amount) {
    int totalCoins = 0;
    for (int coin : coins) {
        int count = amount / coin;
        if (count > 0) {
            System.out.println("→ " + coin + " rs coin x " + count);
            amount %= coin;
            totalCoins += count;
        }
    }
    System.out.println("☑ Total coins used: " + totalCoins);
}
```

- Takes an integer amount as input
- Initializes a counter for total coins used
- Uses an enhanced for loop to iterate through each coin denomination
- For each denomination:
    - Calculates how many coins of that value can be used (count = amount / coin)
    - If at least one coin can be used, prints the number of coins
    - Updates the remaining amount using modulo (amount %= coin)
    - Adds the number of coins used to the total count
- Finally displays the total number of coins used

    This code implements a greedy algorithm which works correctly for this specific set of coin denominations.

OUTPUT

```
================================
         🪙 COIN CHANGER
================================

Enter amount in rupees (positive integer): 37

Calculating minimum coins needed for ₹37...
→ 20 rs coin x 1
→ 10 rs coin x 1
→ 5 rs coin x 1
→ 2 rs coin x 1
✅ Total coins used: 4

Would you like to convert another amount? (yes/no): no

Thank you for using Coin Changer! 😊
```