MAIN.JAVA

Constants:

A. **HEADS:** Integer constant representing heads (value: 0)
B. **TAILS:** Integer constant representing tails (value: 1)
C. **YES:** String constant representing "Y"
D. **NO:** String constant representing "N"
E. **percent:** DecimalFormat object used to format percentages ("##0%")

Libraries:

A. **Java.text.*:**
   DecimalFormat: Formats the percentage difference.
B. **Java.util.*:**
   Random: Generates random numbers for the target head streak.
   Scanner: Reads user input from the console.

1. Initialization:

A. **Simulation coin = new Simulation():** Creates a new Simulation object named coin.
B. **boolean playAgain=true:** Sets a boolean variable playAgain to true to initiate the main loop.

2. Main Loop:

A. **while(playAgain):** The main loop continues as long as playAgain is true.
B. **int flipCount=0:** Initializes a variable flipCount to keep track of the number of flips for each iteration.
C. **double sum=0:** Initializes a variable sum to accumulate the total number of flips across all iterations.

3. Target Streak:

A. **Random t = new Random():** Creates a new Random object to generate random numbers.

B. **int targetHeadStreak=(t.nextInt(7))+6:** Generates a random integer between 6 and 12 (inclusive) to represent the target head streak.
C. **System.out.println**("Your target run is "+ targetHeadStreak +" heads"): Prints the target head streak to the console.

4. User Guess:

A. **Scanner in = new Scanner(System.in):** Creates a new Scanner object to read user input.
B. **System.out.println**("What is your guess of how many coin tosses are needed on average to reach that exact target run?"): Prompts the user for their guess on the average number of flips.

5. Validating User Input:

A. **while(!in.hasNextInt()):** This loop keeps iterating until the user enters a valid integer.
B. **System.out.println("you did not type an int"):** Prints an error message if the user enters anything other than an integer.
C. **String garbage= in.next():** Reads and stores the invalid input to avoid affecting the next iteration.
D. **double yourGuess = in.nextInt():** Stores the user's valid integer guess in the yourGuess variable.

6. Simulation Loop:

A. **for(int i=1; i<=1000;i++):** This loop runs 1000 times to simulate the coin toss experiment.
B. **int headStreak=0:** Initializes a variable headStreak to keep track of the current head streak for each simulation.

7. Inner Loop:

A. **while(headStreak<targetHeadStreak):** This loop continues until the current head streak reaches the target streak.
B. **int randomValue = coin.flip():** Calls the flip() method of the coin object to simulate a coin toss.
C. **flipCount++:** Increments the flipCount for each toss.
D. **if(randomValue==HEADS):** If the toss is heads, increment the headStreak.
E. **else:** If the toss is tails, reset the headStreak to zero.

8. Calculating Average:

A. **sum+=flipCount:** Accumulates the total number of flips for each simulation in the sum variable.
B. **flipCount=0:** Resets the flipCount to zero for the next simulation.

9. Calculating and Reporting Accuracy:

A. **double average= sum/1000:** Calculates the average number of flips across all simulations.
B. **if(yourGuess<average):** If the user guessed less than the average, calculate the absolute difference and format it as a percentage using percent.format().
C. **else if(yourGuess>average):** If the user guessed more than the average, calculate the relative difference and format it as a percentage using percent.format().
D. **System.out.println(...):** Prints a message to the console informing the user of the average number of flips needed and their accuracy relative to the average.

10. Play Again:

A. **System.out.println**("Do you want to play again (Y=yes, N=no)?"): Prompts the user to play the game again.

11. Validating Play Again Input:

A. **while (!valid):** This loop ensures the user enters a valid input (Y or N) to continue or quit.
B. **System.out.println**("Please enter Y or N"): Prompts the user to enter a valid input.
C. **String input = in.next():** Reads the user's input.
if (!(input.equals(NO) || input.equals(YES))): Checks if the input is valid (Y or N). If invalid, the loop continues.