Chapter 9 Reflection Logs

- The code is placed under the Mastery package, which is intended for grouping related classes.

Main Method:

- A Scanner object is created to read input from the user.
- The user is prompted to input a word or phrase, and the input is stored in the input variable.
- The isPalindrome method is called with the user input to check whether it's a palindrome.

- The string is converted to lowercase using toLowerCase() to ensure case-insensitive comparison.
- Two-pointer technique:
  - Two variables (left and right) are used to point to the beginning and end of the string, respectively.
  - A while loop is used to iterate through the string and compare the characters at left and right. If at any point the characters don't match, the method returns false (indicating the string is not a palindrome).
  - The loop moves the pointers towards the center, ensuring that all characters are compared until they meet in the middle.
- If no mismatches are found, the method returns true, indicating that the string is a palindrome.

```
/*
Program: Palindrome     Date: November 15, 2024
Purpose: This program will tell if the word or phrase is a palindrome
School: CHHS
Course: Computer Science 20
*/
package Mastery;
import java.util.Scanner;
public class Palindrome {
  public static void main(String[] args) {

      Scanner scanner = new Scanner(System.in);
      // Prompt the user to enter a word or phrase
      System.out.print("Enter a word or phrase, and this program will identify if it's a palindrome: ");
      String input = scanner.nextLine();
      // Check if the string is a palindrome
      if (isPalindrome(input)) {
          System.out.println("The string is a palindrome.");
      } else {
          System.out.println("The string is not a palindrome.");
      }

  }
  // Method to check if a string is a palindrome
  public static boolean isPalindrome(String string) {

      string = string.replaceAll("[^a-zA-Z0-9]", "").toLowerCase();

      int left = 0;
      int right = string.length() - 1;
      while (left < right) {

          if (string.charAt(left) != string.charAt(right)) {
              return false;  // Not a palindrome if characters don't match
          }
          left++;
          right--;
      }
      return true;  // The string is a palindrome
  }
}
/* screen dump
Test case1
Enter a word or phrase, and this program will identify if it's a palindrome: billy
The string is not a palindrome.
Test case2
Enter a word or phrase, and this program will identify if it's a palindrome: hannah
The string is a palindrome.
*/
```

Pailondrom

—---

- Random Number Generation: The program uses the Random class to generate random numbers between 0 and 99. This is done inside a loop that runs 25 times (for (int i = 0; i < 25; i++)).

- Use of Strings for Storage: The program uses String variables (evens and odds) to accumulate the even and odd numbers. Each time a number is classified, it's appended to the respective string.
    - Pros: The code is simple and easy to understand, especially for beginners. The String concatenation approach is straightforward for handling small datasets like this.

    - Cons: Appending strings repeatedly in a loop can be inefficient, especially with larger datasets, because each concatenation creates a new string object. A better alternative would be to use a StringBuilder, which is more efficient for string manipulation in loops.

- Random Number Generation: The Random class is used to generate random integers within the specified range (0 to 99), the program could benefit from ensuring the random number generation is uniformly distributed, though for this simple task, Random suffices.

```
/*
Program: EvensAndOdds      Date: November 14, 2024
Purpose: This will generate 25 random numbers between 0-90 and display all even and all odd
School: CHHS
Course: Computer Science 20
*/
package Mastery;
import java.util.Random;
public class EvensAndOdds {
  public static void main(String[] args) {
    // Create a Random object to generate random numbers
    Random random = new Random();

    String evens = "";
    String odds = "";
    // Generate 25 random integers between 0 and 99
    for (int i = 0; i < 25; i++) {
      int number = random.nextInt(100);  // Generate random number between 0 and 99
      // Check if the number is even or odd
      if (number % 2 == 0) {
        evens += number + " ";
      } else {
        odds += number + " ";
      }
    }
    // Display the even numbers
    System.out.println("Even numbers: " + evens);
    // Display the odd numbers
    System.out.println("Odd numbers: " + odds);
  }
}
/* screen dump
Even numbers: 94 52 8 82 72 14 30 18 94 18 96 14 34 12
Odd numbers: 33 93 3 79 23 3 81 79 75 71 9
*/
```

EvensAndOdds

The code demonstrates the core functionality of a grade-book system that:

- Accepts grades for students and tests.
- Computes averages for students and tests.
- Displays all the grades and calculated averages.

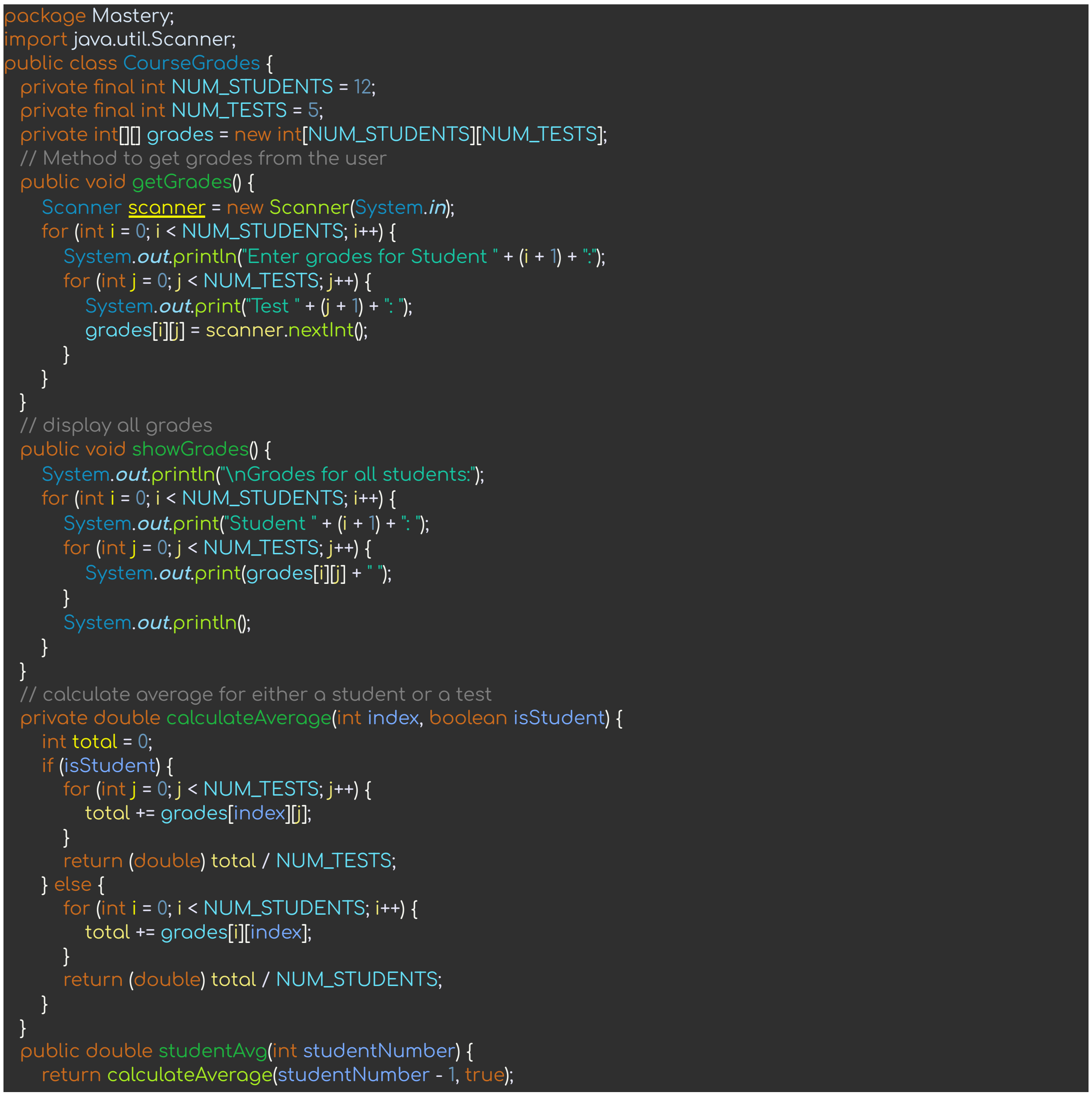
The core logic in the CourseGrades class, the user interaction takes place in the main() method. The code's flow is straightforward, and the naming conventions used for methods and variables are meaningful and easy to understand.

- The 2D array grades are defined with dimensions NUM_STUDENTS x NUM_TESTS, making it a suitable way to store grades for multiple students across multiple tests.

- The getGrades() method uses Scanner to get input from the user. It prompts for each student's grades on each test in a nested loop.

- The code uses scanner.nextInt() to read input. This works fine for integers but doesn't handle invalid input well. This could be enhanced by checking the input to ensure it is valid.

- The showGrades() method prints all the grades for each student. It formats the output so that the grades are grouped by student, which makes it easy to visualize the data.



```java
package Mastery;
import java.util.Scanner;
public class CourseGrades {
  private final int NUM_STUDENTS = 12;
  private final int NUM_TESTS = 5;
  private int[][] grades = new int[NUM_STUDENTS][NUM_TESTS];
  // Method to get grades from the user
  public void getGrades() {
    Scanner scanner = new Scanner(System.in);
    for (int i = 0; i < NUM_STUDENTS; i++) {
      System.out.println("Enter grades for Student " + (i + 1) + ":");
      for (int j = 0; j < NUM_TESTS; j++) {
        System.out.print("Test " + (j + 1) + ": ");
        grades[i][j] = scanner.nextInt();
      }
    }
  }
  // display all grades
  public void showGrades() {
    System.out.println("\nGrades for all students:");
    for (int i = 0; i < NUM_STUDENTS; i++) {
      System.out.print("Student " + (i + 1) + ": ");
      for (int j = 0; j < NUM_TESTS; j++) {
        System.out.print(grades[i][j] + " ");
      }
      System.out.println();
    }
  }
  // calculate average for either a student or a test
  private double calculateAverage(int index, boolean isStudent) {
    int total = 0;
    if (isStudent) {
      for (int j = 0; j < NUM_TESTS; j++) {
        total += grades[index][j];
      }
      return (double) total / NUM_TESTS;
    } else {
      for (int i = 0; i < NUM_STUDENTS; i++) {
        total += grades[i][index];
      }
      return (double) total / NUM_STUDENTS;
    }
  }
  public double studentAvg(int studentNumber) {
    return calculateAverage(studentNumber - 1, true);
```

```java
   }
   public double testAvg(int testNumber) {
       return calculateAverage(testNumber - 1, false);
   }
   public static void main(String[] args) {
       Scanner scanner = new Scanner(System.in);
       CourseGrades gradeBook = new CourseGrades();
       // Get grades and show them
       gradeBook.getGrades();
       gradeBook.showGrades();
       // Display student average
       System.out.print("\nEnter student number (1-12) to see their average: ");
       int studentNumber = scanner.nextInt();
       System.out.println("Average grade for Student " + studentNumber + ": " +
gradeBook.studentAvg(studentNumber));
       // Display test average
       System.out.print("\nEnter test number (1-5) to see the average grade: ");
       int testNumber = scanner.nextInt();
       System.out.println("Average grade for Test " + testNumber + ": " + gradeBook.testAvg(testNumber));
   }
}
```

CourseGrades