|  |
| --- |
| **Lab05: Methods and Recursion** |

Designing and implementing Java programs that deal with:

|  |
| --- |
| 1. Static Methods 2. Recursion |

|  |
| --- |
| **Exercises** |

Exercise 1 (Numbers Calculation)

You're tasked with developing a fitness app featuring various workout routines. One of the app's features involves tracking calorie burn rates based on prime numbers, odd numbers, and even numbers. You're required to implement methods to calculate the sum of calories burned in these categories up to a user-provided input. When a user opens the app, they're prompted to input a number representing their workout duration. The app then calculates and displays the sum of calories burned for prime, odd, and even numbers less than the input number. This process continues until the user decides to exit by entering a negative number. Additionally, you should implement the following methods:

1. `calculatePrimeCalories(int input)`: Calculates the sum of calories burned for prime numbers less than the input number.

2. `calculateOddCalories(int input)`: Calculates the sum of calories burned for odd numbers less than the input number.

3. `calculateEvenCalories(int input)`: Calculates the sum of calories burned for even numbers less than the input number.

Exercises 29/86 These methods should be called within a loop in the main method to continue prompting the user for input until a negative number is entered, signaling the end of the session

**Source Code:**

package numberscalculation;

import java.util.Scanner;

public class NumbersCalculation {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

int workoutDuration;

System.out.println("Welcome to the Fitness App!");

System.out.println("Enter a workout duration:");

while (true) {

workoutDuration = scanner.nextInt();

if (workoutDuration < 0) {

System.out.println("Exiting the app. Thank you for using the Fitness App!");

break;

}

int primeCaloriesBurned = calculatePrimeCalories(workoutDuration);

int oddCaloriesBurned = calculateOddCalories(workoutDuration - 1);

int evenCaloriesBurned = calculateEvenCalories(workoutDuration - 1);

System.out.println("Sum of calories burned for prime numbers less than " + workoutDuration + ": " + primeCaloriesBurned);

System.out.println("Sum of calories burned for odd numbers less than " + workoutDuration + ": " + oddCaloriesBurned);

System.out.println("Sum of calories burned for even numbers less than " + workoutDuration + ": " + evenCaloriesBurned);

System.out.println("Enter another workout duration:");

}

scanner.close();

}

public static int calculatePrimeCalories(int workoutDuration) {

int caloriesBurned = 0;

for (int i = 2; i < workoutDuration; i++) {

if (isPrime(i)) {

caloriesBurned += i;

}

}

return caloriesBurned;

}

private static boolean isPrime(int number) {

if (number <= 1) return false;

for (int i = 2; i <= Math.sqrt(number); i++) {

if (number % i == 0) return false;

}

return true;

}

public static int calculateOddCalories(int n) {

if (n <= 0) return 0;

if (n % 2 != 0) {

return n + calculateOddCalories(n - 2);

}

return calculateOddCalories(n - 1);

}

public static int calculateEvenCalories(int n) {

if (n <= 0) return 0;

if (n % 2 == 0) {

return n + calculateEvenCalories(n - 2);

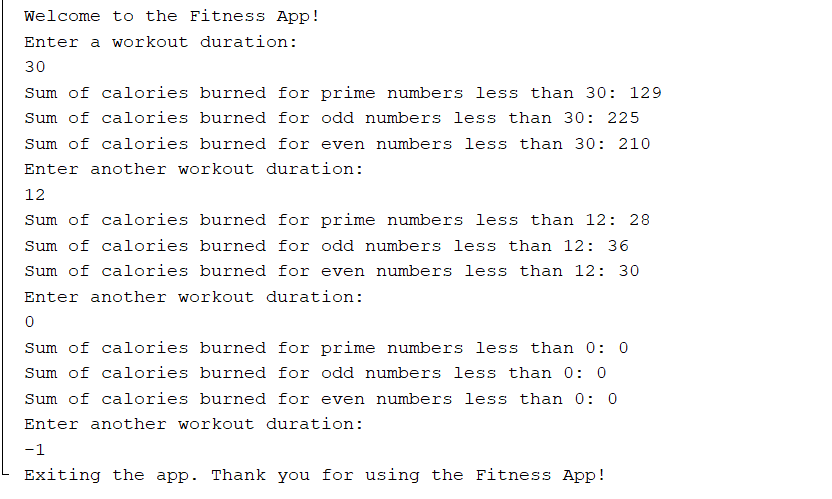
}

return calculateEvenCalories(n - 1);

}

}

Output:



Exercise 2 (Text Processing)

Imagine you're part of a team developing a text processing tool for language learners. One feature you're implementing involves helping users understand word reversals. So, You're tasked with developing a function that reverses words recursively to aid language learners in understanding word structures.

**Source Code:**

package textprocessing;

public class TextProcessing {

public static void main(String[] args) {

String word = "aymankhan";

String reversedWord = reverseWord(word);

System.out.println("Original word: " + word);

System.out.println("Reversed word: " + reversedWord);

}

public static String reverseWord(String word) {

if (word.isEmpty()) {

return word;

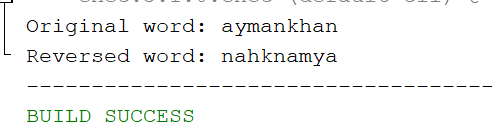
}

return word.charAt(word.length() - 1) + reverseWord(word.substring(0, word.length() - 1));

}

}

**Output:**



Exercise 3 (Fibonacci Calculator)

You're working on a Fibonacci calculator for a mathematics tutoring app. Users can input a number, and the app calculates the Fibonacci of that number using a recursive function.

**Source Code:**

package fibonaccicalculator;

import java.util.Scanner;

public class FibonacciCalculator {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.println("Enter a number to calculate its Fibonacci value:");

int number = scanner.nextInt();

int result = calculateFibonacci(number);

System.out.println("Fibonacci of " + number + " is: " + result);

scanner.close();

}

public static int calculateFibonacci(int n) {

if (n <= 1) {

return n;

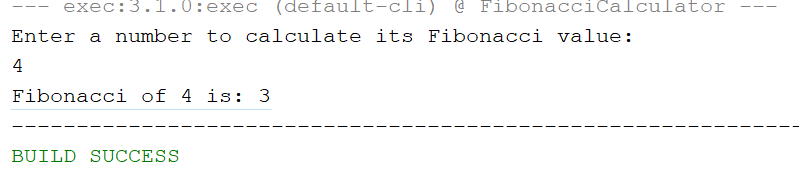
}

return calculateFibonacci(n - 1) + calculateFibonacci(n - 2);

}

}

**Output:**

****

Exercise 4 (Time Tracking Tasks Calculation)

You're developing a time-tracking tool where users can input their start and end times for different tasks. You want to implement a feature that calculates the total time spent on a task by summing up all the individual time intervals between the start and end times. Also calculate the total time spend by the user in the entire week. If user has spend less than 40 hours then system must display a warning message.

**Source Code:**

package timetracking;

import java.time.Duration;

import java.time.LocalTime;

import java.util.ArrayList;

import java.util.Scanner;

public class TimeTracking {

public static void main(String[] args) {

Scanner inputScanner = new Scanner(System.in);

ArrayList<Duration> dailyTaskDurations = new ArrayList<>();

int weeklyTotalMinutes = 0;

System.out.println("Welcome to the Time Tracking Tool!");

System.out.println("Please enter your start and end times for each task in the format HH:MM.");

for (int dayIndex = 1; dayIndex <= 7; dayIndex++) {

int minutesPerDay = 0;

System.out.println("\nDay " + dayIndex + ":");

while (true) {

System.out.print("Enter start time : ");

String startTimeInput = inputScanner.next();

if (startTimeInput.equalsIgnoreCase("done")) {

break;

}

System.out.print("Enter end time: ");

String endTimeInput = inputScanner.next();

LocalTime taskStartTime = LocalTime.parse(startTimeInput);

LocalTime taskEndTime = LocalTime.parse(endTimeInput);

Duration taskDuration = Duration.between(taskStartTime, taskEndTime);

minutesPerDay += taskDuration.toMinutes();

System.out.println("Time spent on this task: " + formatTime(taskDuration));

}

Duration dailyDuration = Duration.ofMinutes(minutesPerDay);

dailyTaskDurations.add(dailyDuration);

weeklyTotalMinutes += minutesPerDay;

System.out.println("Total time spent on Day " + dayIndex + ": " + formatTime(dailyDuration));

}

Duration weeklyDuration = Duration.ofMinutes(weeklyTotalMinutes);

System.out.println("\nTotal time spent in the week: " + formatTime(weeklyDuration));

if (weeklyDuration.toHours() < 20) {

System.out.println("Warning: You have spent less than 20 hours this week.");

}

inputScanner.close();

}

private static String formatTime(Duration duration) {

long hours = duration.toHours();

long minutes = duration.toMinutes() % 60;

return String.format("%02d:%02d", hours, minutes);

}

}

**Output:**

