**Assignment: Adventure Math Game in Java**

2024F ICS-114-X01AB-X02AB

Instructor: Olga Gould

[GouldO@camosun.ca](mailto:GouldO@camosun.ca)

See due dates in D2L.

**Adventure Game Programming Assignment**

**Introduction:** In this assignment, you will help a character in an adventure game complete various challenges. Each task will test your knowledge of loops (for, do-while, while), conditionals (if-else), and switch statements. You will implement the functionality to help the player through different stages of the game. The game will also provide practical challenges related to your career as a computer scientist.

**Pair Programming:** You are encouraged to work in pairs or small groups for this assignment. Collaborating with your peers can lead to a better understanding of concepts, enhance problem-solving skills, and foster original ideas. Make sure to divide the tasks and communicate effectively with your partner(s).

**Tasks**

**Task 1: Summing Points to Level Up**

* **Problem:** The player collects points to level up. Your task is to calculate the sum of all points from 1 to the maximum value entered by the player.
* **Java Concepts:** for loop

import java.util.Scanner;

public class AdventureGame {

// Method: Calculate the sum of points from 1 to max

public static void calculatePointsSum() {

// TODO: Ask the player to enter the maximum number of points they can collect

// TODO: Initialize sum to 0

// TODO: Use a 'for' loop to sum all numbers from 1 to max

// TODO: Print out the total points collected

}

}

**Task 2: Defeating the Boss - Largest Weapon Selection**

* **Problem:** To defeat the boss, the player must choose the largest weapon (number) from a set of five values entered by the player.
* **Java Concepts:** for loop, if statement

import java.util.Scanner;

public class AdventureGame {

// Method: Find the largest of 5 numbers (weapon values)

public static void selectLargestWeapon() {

// TODO: Initialize a variable 'largest' to store the largest number

// TODO: Use a 'for' loop to let the player input five weapon values

// TODO: Inside the loop, check if the current value is larger than 'largest'

// TODO: Print the largest weapon value selected

}

}

**Task 3: Mini-Boss Fight - Target the Weakest Mini-Boss**

* **Problem:** The player encounters three mini-bosses. Each mini-boss has a health value. The player must target the weakest (smallest number) first.
* **Java Concepts:** for loop, if statement

import java.util.Scanner;

public class AdventureGame {

// Method: Find the smallest of 3 mini-boss health values

public static void targetWeakestMiniBoss() {

// TODO: Initialize a variable 'smallest' to store the smallest health value

// TODO: Use a 'for' loop to input three health values from the player

// TODO: Inside the loop, check if the current health value is smaller than 'smallest'

// TODO: Print the smallest health value (the weakest mini-boss)

}

}

**Task 4: Factorial Power-Up**

* **Problem:** The player needs to calculate the factorial of a number to unlock a special power. You will use a do-while loop to calculate the factorial.
* **Java Concepts:** do-while loop

import java.util.Scanner;

public class AdventureGame {

// Method: Calculate the factorial of a number for a power-up

public static void calculateFactorialPowerUp() {

// TODO: Ask the player for a number to calculate its factorial

// TODO: Initialize factorial to 1

// TODO: Use a 'do-while' loop to calculate the factorial

// TODO: Print out the power-up points (factorial result)

}

}

**Task 5: Switch-Based Puzzle Challenge**

* **Problem:** The player encounters a puzzle with five different doors. Each door has a number from 1 to 5, and the player must choose which door to open based on the number they input.
* **Java Concepts:** switch statement

import java.util.Scanner;

public class AdventureGame {

// Method: Solve the door puzzle using a switch statement

public static void doorPuzzle() {

// TODO: Ask the player to choose a door number (1-5)

// TODO: Use a switch statement to determine what happens with each door

// Hint: Provide different messages for each case (e.g., treasure, trap, ally, locked door, secret passage)

}

}

**(Bonus) Loop Until Victory - Dragon Battle**

* **Problem:** The player battles a dragon, and the battle continues until one of them runs out of health. Use a while loop to simulate the battle. Each round, the player either deals damage to the dragon or takes damage.
* **Java Concepts:** while loop

import java.util.Scanner;

public class AdventureGame {

// Method: Simulate a battle with a dragon using a while loop

public static void battleDragon() {

// TODO: Initialize player's health and dragon's health

// TODO: Use a while loop to continue the battle until one of the health values drops to 0

// TODO: Deal damage or defend based on the player's action (ask for input in each round)

// TODO: Print the result of the battle (victory or defeat)

}

}

**Final Testing**

After completing each method, test them by calling them in the main method.

public class AdventureGame {

public static void main(String[] args) {

// TODO: Uncomment the methods below after you implement each one

// calculatePointsSum();

// selectLargestWeapon();

// targetWeakestMiniBoss();

// calculateFactorialPowerUp();

// doorPuzzle();

// battleDragon();

}

// Implement all methods here: calculatePointsSum, selectLargestWeapon, targetWeakestMiniBoss, etc.

}

**Reflection**

After completing the assignment, each student (or pair) must submit a brief reflection (approximately 1-2 paragraphs) discussing:

* The coding process: What approach did you take to solve the tasks?
* Challenges faced: What difficulties did you encounter, and how did you overcome them?
* Learning outcomes: What new concepts or skills did you learn during this assignment?

**Unique Challenges**

In addition to the tasks outlined above, each pair must implement a unique feature or variation of one of the tasks. Examples of unique challenges include:

* **Task 1 Variation:** Allow the player to collect points in increments (e.g., 2, 3, or 5 points at a time) and calculate the sum based on the chosen increment.
* **Task 2 Variation:** Implement a feature that allows the player to input weapon values until they decide to stop, then find the largest weapon value.
* **Task 3 Variation:** Introduce additional mini-bosses, allowing the player to select a strategy based on the health values.
* **Task 4 Variation:** Extend the factorial calculation to handle multiple numbers and print all factorial results.
* **Task 5 Variation:** Add more doors or create different outcomes based on player choices, such as unlocking special abilities or rewards.

**Grading Criteria**

Your assignment will be graded based on the following criteria:

* **Functionality:** The program compiles and runs without error (2 marks).
* **Correctness:** Each task meets its requirements and functions as intended (2 marks).
* **Coding Style:** The code is well-organized, readable, and follows best practices (2 marks).
* **Reflection:** A thoughtful reflection is submitted, highlighting the learning process and challenges (2 marks).
* **Unique Challenge:** A creative and functional implementation of the unique feature (2 marks).
* **Collaboration:** Evidence of effective collaboration and communication with your partner(s) (2 marks).

**Total: 12 marks.**