PROJECT Looping Control with Java - Game Play: Seize the Station

Objective To construct a program that demonstrates repetition program control.

PROJECT DESCRIPTION

Looping program control occurs when code statements are executed in a repetitive fashion.

Type, compile and run a computer program that uses repetitive control structures to simulate this game play scenario.

Your opponents have captured the railway transportation station and you have been given the task to seize and secure it.

There is only 300 feet between your current position and the station. You will move towards the target in step increments, determined at random, until you reach your goal. But there will be a series of obstacles along your path.

You must be successful!

Design your program such that each of these requirements are satisfied. Some starter code statements are available in **Figure 1** to assist you in programming this application. The starter code has a loop that allows for a fixed number of chances to re - take the station and thus to achieve your objective.

The core of the starter code in **Figure 1** is a `for()` loop.

Each time the body of the `for()` loop is executed, the player enters a signal that he / she is ready for the next step in reaching the target, which is to re - capture the railway station. A random encounter with an obstacle will then ensue and a health amount is added or subtracted from the player's current health level depending on the player's ability to overcome it. If the player successfully overcomes the obstacle, then his / hers distance to the target will lessen depending on a random number subtracted from their current distance to the target. If the player's health is greater than zero, gameplay continues until the objective is accomplished.

Then compile and run your program, observe the output and then modify the program.

Information About This Project

Game and simulation applications comprise the use of typical gameplay logic.

For this application, the core gameplay steps will be as follows:

- The player will start at a certain distance from the target / objective. In this program, we assume that we are 300 feet away from our objective. The player begins with 100 % health.
- The player enters a signal that they are ready to move forward toward the objective.
- When the player moves forward he / she may encounter an obstacle that can reduce the player’s health. At this point the type of obstacle will determine the distance that the player will move closer to the target. We also track the total distance remaining to reach the objective.
- The player must reach the target with more than 0 % health otherwise the mission is a failure.

PROJECT Looping Control with Java - Game Play: Seize the Station

Steps to Complete This Project

STEP 1 **Open an Integrated Development Environment (IDE)**

Open Eclipse, Net Beans, MS Visual Studio or similar programming text editor / compiler.

STEP 2 **Write the Program Code**

Write the program code that will satisfy the requirements of this project.

Review the starter code that is given in **Figure 1** . The starter code has some suggested objects, variables and code blocks for which you can modify the code to conform to your own particular gameplay preferences.

Into your **Code** window, copy in the program code shown in **Figure 1** .

Place your own name in the source code, as the program writer.

You can run the starter code to become acquainted with the gameplay format. Here is a sample program output of the initial starter code.

[Sample Program Output]

```
-----------------------------------

Are you ready to proceed? ( Y or N)
Y
circumvent the next obstacle

-----------------------------------

Are you ready to proceed? ( Y or N)
Y
you will move forward by 9 feet

-----------------------------------

Are you ready to proceed? ( Y or N)
```

Complete the starter code by modifying / augmenting the code statements to perform the tasks, summarized below, which were listed on the prior page

- Enter a loop and indicate whether you wish to proceed.
- When you proceed a randomly defined obstacle could prevent your next advance to your goal.
- If a randomly generated letter is within the included range of letter a through letter m , then the player moves forward by a random distance of 1 to 20 feet, inclusive. Otherwise, the player is detained by the obstacle and loses some health.
- The player then decides to continue moving to the objective or quit the mission.
- As the player continues towards the objective, the player's health could diminish as obstacles are encountered. Also, the total distance remaining to the target is computed.
- When the player has ample health to reach the goal then the game ends and the mission is successful.

PROJECT Looping Control with Java - Game Play: Seize the Station

Figure 1 Program Code for the Gameplay Application

```java
import java.util.Random;
import java.util.Scanner;
public class SeizeTheStation
{
  public static void main(String args[])
  {
    // declare a Scanner class object and Random class object
    Scanner scan = new Scanner(System.in);
    Random randomGen = new Random();
    // variable to decrease distance to target
    int randAddDist = 0;
    // variable to track remaining distance to target
    int randDistToMove = 0;
    // variable to use to supplement player / game interaction
    int randInteract = 0;
    // variable to define player obstacle
    char interact = '\0';
    // variable to allow player to proceed to target
    char again = '\0';
    // variable to set initial distance to goal
    int goal = 300;
    // variable to set initial player health
    int health = 100;
    // define a loop for at most twenty actions
    for (int count = 1; count <= 20; count++)
    {
      // signal the intention of the player
      System.out.println("\n-----------------------------------");
      System.out.println("\nAre you ready to proceed? ( Y or N )");
      again = scan.next().charAt(0);
      if (again != 'Y') break;
      // define an obstacle
      interact = (char)(randomGen.nextInt(26) + 'a');
      if (interact >= 'a' && interact <= 'm')
      {
        // random number sets distance to move toward the objective
        randDistToMove = 1 + randomGen.nextInt(20);
        System.out.println("move forward " + randDistToMove + " ft");
      }
      else
      {
        System.out.println("circumvent the next obstacle");
        health -= 10;
      }
    }
  }
}
```

PROJECT Looping Control with Java - Game Play: Seize the Station

STEP 3 **Build, Compile and Run the Program**

Compile and run your program code statements. Correct any syntax or compile errors.

STEP 4 **Test the Program**

Once you have successfully compiled your program, run your program and enjoy the gameplay!

STEP 5 **Verify Your Output**

Verify that your program is functioning correctly. Take screen snapshots of the **Console** window showing a successful reach of the objective and an unsuccessful attempt.

You may have to adjust the given random value ranges to ensure that a successful or unsuccessful gameplay is obtained.

STEP 6 **Modify Your Program**

With your program running successfully, alter your program code to perform these additional tasks.

- Track the number of attempt (loops) that were necessary for goal to be achieved.
- Display a summary of the player's actions and health by showing the number of loops that were necessary to reach the goal and the health value achieved at the end of the game.
- After five loops of the program, use a randomized value that will determine whether an additional companion will team with the player to assist in reaching the goal in a faster pace. You can adjust the health and distance to target variable values to allow for the entrance of the companion player.

STEP 7 **Submit Your Project**

Once you have determined that your modified program is satisfying this project's requirements, complete the submission process as follows:

Open MS Word and type a heading for a new document that includes your full name, course number, lab number and date.

Within the document paste a snapshot of your program code. Label your snapshot with a reasonable description.

After the snapshot, paste the output that appears in your **Console** screen.

Note - you can use the **Windows Snipping Tool**, which is part of the Windows Accessories Group, to easily capture your **Console** window**.**

Your score for this project will generally be based upon the following factors: documentation, output correctness, content, organization, style and creativity. Submit your Word document to the appropriate course Submittal Box.

STEP 8 **Questions and Answers Concerning this Computer Project**

Answer the following questions in your own words.

Open MS Word and, within your lab submittal document, place your responses to each of these questions. Submit your completed MS Word document for credit.

PROJECT Looping Control with Java - Game Play: Seize the Station

(1) What is meant by repetition program control? Can sequential program control be used in conjunction with repetition program control?

(2) How is repetition program control used in this application?

(3) The original starter code for this application has a `for()` loop with twenty iterations? Were you able to reach the objective of seizing the station within this number of iterations? Explain your answer.

(4) How were random numbers used in your program?

(5) For implementation of repetition control in this application, could a `while()` loop be used here instead of a `for()` loop?