Software Requirements Specification

For

How Many Are In The Jar

Version 2.2.0 Approved By: Amir Yousef

Prepared By: Amir Yousef

December 11th, 2017

TeamTreeHouse

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Amir Yousef	12/03/2017	Beta Version Needs Modifications	1.0.0
Amir Yousef	12/10/2017	Final Version After All Modifications	2.2.0

1. Introduction

1.1 Purpose

This is java project required for TeamTreeHouse project1 submission.

1.2 Document Conventions

My standard or reference is the instructions given by the instructor:

Create a jar class that accepts an item name and a maximum number of items that can fit in the jar. Your program should have an administrator who is prompted to identify what type of item should fill the jar, and the maximum number of items the jar can hold, and should include the phrase "What type of item". The type of item should be repeated in the second prompt. For example, if the administrator specifies “gumballs” as the type of item, the next prompt should ask “What is the maximum amount of gumballs?”
As a player of the game, for each new game I should be presented with a new jar filled with a random number of items, so that I can play multiple games and still find the game challenging.
Write a fill method for your Jar class that uses the Random class to fill the jar with a random number of items. That number should be between 1 and the maximum number of items set by the administrator.
Hint: Read the code in the project instructions for help!
As a player of the game, I should be shown the maximum possible number of items in the jar.
When the player begins the game, your program should display the maximum number of items the jar can hold. The prompt should also repeat the type of item in the jar.
For example, if the jar is filled with a maximum number of 1,500 jelly beans, the prompt should read “How many jellybeans are in the jar? Pick a number between 1 and 1500.”
As a player of the game, I should be prompted to guess the number of the items until I get the correct answer.
Use a while loop to prompt the user for a new guess until the correct answer is guessed
As a winner of the game, I should know how many attempts it took me to get to the right answer, so that I am encouraged to beat my score.
Your winning message should be in this format: "You got it in X attempt(s)".
When returning the number of guesses, be careful of off by 1 errors!
Use System.out for output and a reader on System.in for input
There are tests that have been written and included in this project. If you would like to run them, from the ~/workspace directory run the command ./gradlew test.
The first time will take about a minute as it downloads required libraries. Subsequent test runs will be quicker.
If you are attempting to get an exceeds expectation grade, change the appropriate line in src/main/resources/treehouse.properties to true and the tests will accommodate for your changes.
Please ensure you upload all the files provided in the starter workspace to GitHub.
Before you submit your project for review, make sure you can check off all of the items on the Student Project Submission Checklist.
The checklist is designed to help you make sure you’ve met the grading requirements and that your project is complete and ready to be submitted!

1.3 Intended Audience and Reading Suggestions

The document is intended for the project reviewers for grading purpose and for the future employers to show writing skills.

1.4 Product Scope

There will be an administrator who determines which items and the maximum amount of items in each jar. They can then call a fill method to randomly fill the jar.
As long as the player has not guessed the correct answer, they must make another guess. Once they get the correct random number, they should know how many guesses it took them.

1.5 References

Project_1_ UML_Diagram.PNG & Project1_QA_Test_Case.xls documents.

1.6 Product Perspective

A simple diagram shows the major components attached in Appendix B: UML Diagram.

1.7 Product Functions

The major function is to receive integer from the user and validate it against a random number generated till the user win the game.

1.8 User Classes and Characteristics

Used three classes GuessingGame.java, Jar.java and Prompter.java

1.9 Operating Environment

The environment in which the software will operate is any windows operating system such as Windows7 or Windows8, and required to have Java JDK to be able to compile and run.

1.10 Design and Implementation Constraints

The only limitation is to have Java installed.

1.11 User Documentation

No documentation required.

1.12 Assumptions and Dependencies

Not applicable, the software can be shared without a problem.

2. External Interface Requirements

2.1 User Interfaces

The user will run the application and type the numbers using the keyboard.

2.2 Hardware Interfaces

The computer, monitor and keyboard.

2.3 Software Interfaces

Windows, Java Runtime and Java JDK. Attached Appendix D “Compile and Run Instructions”

2.4 Communications Interfaces

Not Required.

3. System Features

The functional requirements for the product:

3.1 System Feature 1

The admin will be prompted to enter the type of items and the maximum number of items the Jar can contain during the setup.

3.2 System Feature 2

The user will be prompted to enter a number between 1 and the maximum number of items till the user enter the matching random number that generated from the admin maximum number.

4. Other Nonfunctional Requirements

Not Applicable.

4.1 Performance Requirements

Not Required.

4.2 Safety Requirements

Not Required.

4.3 Security Requirements

Not Required.

4.4 Software Quality Attributes

Tested and passed QA.

4.5 Business Rules

Not Required.

5. Other Requirements

Not Required.

Appendix A: Glossary

Not Applicable.

Appendix B: Analysis Models