GuessGame Design Documentation

# Introduction

This documentation describes the software design and build of GuessGame, a number-guessing Java application.

Please see the Use Case documentation for more details.

Please see the Test Case documentation for various test cases.

# Assumption

**Positive Integer Number**

All numbers used in the application are positive integer numbers.

**Range of the Number**

The application guesses a positive integer number within a given range. The upper and lower bound values of the range are set in Java system properties or with “-D” Java options.

Note that the upper bound number must be greater than the lower bound number, or the application will not start.

**Case Insensitive**

User’s response is case insensitive. That is, a user can respond with “higher”, “HIGHER”, or “Higher”.

**Start a New Game**

When a user respond a “yes” or “end” to finish the current round of the game, the application will start a new game by prompting “Are you ready for the game?”

# Application Requirements

The application is developed and tested in Java environment with JDK 1.7. To run the application, you must have JDK or JRE 1.7 installed.

You can also install Maven to build and test the application from command line. The application was built with Apache Maven 3.0.5.

# Algorithm to Generate the Guess Number

**Calculate the Mean Value**: Mean To speed up the game, the application will calculate the mean value of a pair of high and low numbers.

However, to get the initial guess number, the application will not use the mean value. Instead, the application will generate a random integer number.

The speed will then be in the order of O(log N), where N is the total numbers between the upper and low bound values.

# Operation

**Validation and Unrecognized Answer**

The application must validate the user’s response. If the user type in a response that is not in the valid list, the application shall display a message to show the user with valid set of responses.

**Detect Wrong Answer**

When the application responds a number, the user will answer either “higher, “lower”, or “yes”. If the user has entered a wrong answer previously (for example wrong higher or lower) thus causing the application will not guess the right number, the application shall detect such situation and prompt the user. If it happens, the application shall quit and start a new game.

# Build the Application

Run “mvn package” command from under “GuessGame” directory. A JARA file named “GuessGame.jar” will be created in “GuessGame\target” sub-folder.

# Test the Application

Run “mvn test” command from under “GuessGame” directory.

# Run the Application

**Run Command**

The application is built in a JAR format by Maven utility. Run the following command from under “GuessGame” directory.

java -cp target\GuessGame-1.0.jar -Dupperbound=100 -Dlowerbound=1 com.cp.app.GuessGame

Where the “upperbound” parameter defines the upper bound value and the “lowerbound” parameter defines the lower bound value.

**Default Values:**

If these two values are not set in the system parameters, the application will use default value 100 as the upper bound value and 1 as the lower bound value.

# Future Extension

1. The upper and lower bound numbers can be defined in a **property file** or use a **Spring bean** to inject (direct injection) their values.
2. Create a batch file or shell script to run the application.
3. All messages prompting to users can be stored in a file and loaded once (in a singleton) when the application starts.
4. The application shows the list of guessed numbers when a user gives a wrong answer.