Herman Mann

Professor Alexander

CMSC 203

Assignment 2 Final Design Document (Pseudocode)

**Start**

1. Importing the import **java.util.Scanner;** class for scanning input from keyboard.
2. Naming the driver method(main class) **public RandomNumberGuesser**.
3. Declaring and initializing the attributes **randNum, nextGuess, highGuess to 100** and **lowGuess** to **0**, and assigning three other variables as a Boolean **isFirstGuess = true;** (used for the first guess, when user enters a first guess, and then turns into false when first guess flag isn’t used anymore), the Boolean **isGuessMatched = false;** (it is assigned as false until the randNum equals the user’s next guess and then changed to true later on). The last one, the Boolean **isExit = false;**(used for if the user wants to go again and enter another guess for another random number iteration, then assigns it to true if the user wants to stop running the program and to exit from guessing another time for the random number).
4. Writing the **scanner method** to take input of user from keyboard and using it with **(System.in).**
5. Declaring and initializing final String variables that are needed throughout this project/program run through. **final String MSG\_HEADER, MSG\_FOOTER\_THANK\_YOU, PROGRAMMER\_NAME, MSG\_FIRST\_GUESS, MSG\_ENTER\_NUMBER, MSG\_AND, MSG\_NUMBER\_OF\_GUESS, MSG\_GUESS\_TOO\_HIGH, MSG\_GUESS\_TOO\_LOW, MSG\_CONGRATULATION, MSG\_TRY\_AGAIN, MSG\_YES, MSG\_NO**, these variables assigned accordingly to description as Message String Constants.
6. Introducing a modularizing method of a **private void initialize()** method that doesn’t take any parameters and sets the variables previously described **isGuessMatched = false; isFirstGuess = true; isExit = false; lowGuess = 0; highGuess = 100;** (basically initializing these variables to their default values)**.**
7. Introducing another modularizing method of a **private void doNextGuess()** method which won’t take any methods, and will initialize and assign a boolean isInvalidGuess = false; at the beginning to emphasize if the user had entered any user input out of the bounds of the previous low or high guesses and during program run/compilation of the previous entered low or high guesses according to the random number not said or defined visually. Also, this method uses a do-while loop and takes a user’s next guess and scans it while also including the previous line spaces if any, and turning any string entered as a number by the user to an integer number as scanned and defined: nextGuess = Integer.parseInt(reader.nextLine()); Then this method will call the RNG.inputValidation(nextGuess, lowGuess, highGuess) using an if statement conditional statement: if(RNG.inputValidation(nextGuess, lowGuess, highGuess)) and will act accordingly depending on if the nextGuess the user enters is higher or lower guess than the actual randNum(Random number) or if

it is equal to the randNum itself.

**private void doNextGuess()** {

boolean isInValidGuess = false;

do {

if (RNG.inputValidation(nextGuess, lowGuess, highGuess)) {

System.out.println(MSG\_NUMBER\_OF\_GUESS+RNG.getCount());

if (nextGuess > randNum) {

System.out.println(MSG\_GUESS\_TOO\_HIGH);

highGuess = nextGuess;

} else if (nextGuess < randNum) {

System.out.println(MSG\_GUESS\_TOO\_LOW);

lowGuess = nextGuess;

} else if (nextGuess == randNum) {

isGuessMatched = true;

}

isInValidGuess = false;

} else {

isInValidGuess = true;

} while (isInValidGuess);

}

1. Introducing another modularizing method of **private void doFirstGuess()**, which displays and prints out the first guess message so the user can start the program compilation and enter his/her first guess. Calls the initialize() method of the variables that were declared and initialized, is necessary for beginning of Program execution successfully. Then, initializing the randNum = RNG.rand(); the static method from the RNG class. Calling the other method inside the doFirstGuess method for the user to enter the next guess and then to see if that user’s guess was appropriate for what if conditional statement. Finally then setting the isFirstGuess = false; (The first guess previously entered by the user was done so now we initialized the isFirstGuess variable to false).

private void doFirstGuess() {

System.out.println(MSG\_FIRST\_GUESS);

initialize();

randNum = RNG.rand();

doNextGuess();

isFirstGuess = false;

}

1. Introducing another Modularizing method the **private void doGuessMatch()**, which will repeat the guessing number the user enters for the random number until that specific number the user enters matches up with the random number exactly, ==. It will display and print a message to the user depending on if the user enters a number that was within a lower number guess than the random number or if the user enters a guess much higher than that of the random number itself. The message will be printed and displayed “ Enter your next guess between the previous low and high number”. Like this:

**private void doGuessMatch()** {

do {

System.out.println(MSG\_ENTER\_NUMBER + lowGuess + MSG\_AND + highGuess);

doNextGuess();

} while (!isGuessMatched);

}

1. Introducing another Modularizing method of the **private void playAgain()**, which for this specific method there is going to be a do-while loop for the user to enter a choice of a yes or no ignoring the case sensitivity, and after the first iteration is done, the user will enter yes if he/she wants to do another iteration with a different random number. This method will display and print “Try again? (yes or no)” and then the next line the user would have to enter a string choice of yes or no like said before. If the user enters yes, the random number game will reset and then **RNG.resetCount()** will reset the number of counts/guesses and then will change the variable **isFirstGuess = true;** and then a **break** within those lines of code after reset of the previous guesses/iteration(s). Else if the choice is a String of choice of no, the program will exit program execution completely and assign the variable **isExit = true;** and **break;** and then should later on print and display message “Thanks for playing…” and then display and print “PROGRAMMER: Herman Mann”

**private void playAgain()** {

do {

System.out.println(MSG\_TRY\_AGAIN);

String choice = reader.nextLine();

if (choice.equalsIgnoreCase(MSG\_YES)) {

RNG.resetCount();

isFirstGuess = true;

break;

} else if (choice.equalsIgnoreCase(MSG\_NO)) {

isExit = true;

break;

}

} while (true);

}

1. Introducing another modularizing method of public void startRandomGuesser(), which is the biggest method that displays and prints the program Header of this Program, and then introduces a do-while loop that is designed to check for the first guess and then eventually do the first guess and if the whole process of guessing a number that is equal to random number is true of first guess and doing it, else we do the guess matching method until the user’s next guess or whatever matches to the random number defined indirectly. If it is the user’s first guess, call the doFirstGuess() method, else call the doGuessMatch() method. If the user’s next guess or first guess matches the random number defined indirectly and implicitly print and display the message “Congratulations, you guessed correctly”, and then calls the playAgain() method, if the user wants to enter another iteration, else print and display message “Thanks for playing…”, “PROGRAMMER: Herman Mann”.

**public void startRandomGuesser()** {

System.out.println(MSG\_HEADER);

System.out.println("");

do {

if (isFirstGuess) {

doFirstGuess();

} else {

doGuessMatch();

}

if (isGuessMatched) {

System.out.println(MSG\_CONGRATULATION);

playAgain();

}

} while (!isExit);

System.out.println(MSG\_FOOTER\_THANK\_YOU);

System.out.println();

System.out.println(PROGRAMMER\_NAME);

}

1. After all of the previous things/steps are done we introduce the main method, which creates the instance of the RandomNumberGuesser class and creates a new instance of this class all in all, RandomNunberGuesser randomNumberGuesser = new RandomNumberGuesser(); Then we get the new instance of the RandomNumberGuesser class and call the startRandomGuesser() class and then start playing the random number guessing game, randomNumberGuesser.startRandomGuesser();

**public static void main(String[] args)** {

RandomNumberGuesser randomNumberGuesser = new RandomNumberGuesser();

randomNumberGuesser.startRandomGuesser();

}

}

1. Display and print Header, and let user guess first time and scan the value the user enters into nextGuess = Integer.parseInt(reader.nextLine());
2. Displaying and printing the message to allow the user another guess to enter a number between the lowest guess generated and the first guess the user inputted if it is the lower or higher guess accordingly next time.
3. Displaying and printing the total number of guesses to the screen to the output.
4. If the user enters the correct guess of the random number, the program will ask the user if he/she wants to try again using **System.out.println(“Try again? (yes or no)”);**
5. Displaying and printing the Programmer’s name to the screen’s output, following the user’s guess being correct of the random number.
6. Displaying and printing the message **System.out.println(“Thanks for playing…”);**
7. Using the **inputValidation(int nextGuess, int lowGuess, int highGuess)** to check to see if the next guess the user inputs from keyboard is between the lowest and highest guess, and returns a true if it satisfies that condition, and false if it doesn’t. If the user enters a wrong guess after the range goes to the user’s first guess value of the random number than the message will be displayed and printed to the screen as **System.out.println(“>>> Guess must be between the first guess and second guess of the user’s inputted choice before).**
8. Using the **getCount() method** to get the total count of the user-inputted guesses and then returning the count as a integer value variable towards the end of the program execution.
9. Using the **resetCount() method** to reset the number of guesses the user will about to enter through the use of the keyboard to 0.

**End**