# Flowchart

## reset()

Chart, diagram, box and whisker chart

Description automatically generated

Figure . Flowchart of reset()

## run()

Diagram

Description automatically generated

Figure . Flowchart of run()

## main()

Diagram

Description automatically generated

Figure . Flowchart of main()

# Test plans for algorithms

## For loop in method reset()

|  |  |  |  |
| --- | --- | --- | --- |
| value | value <= MAX ? | sum | Notes |
| 1 | Yes | 1 | Loop runs, sum is increased by 1 |
| 2 | Yes | 3 | Loop runs, sum is increased by 2 |
| 3 | Yes | 6 | Loop runs, sum is increased by 3 |
| 4 | Yes | 10 | Loop runs, sum is increased by 4 |
| 5 | Yes | 15 | Loop runs, sum is increased by 5 |
| 6 | Yes | 21 | Loop runs, sum is increased by 6 |
| 7 | Yes | 28 | Loop runs, sum is increased by 7 |
| 8 | Yes | 36 | Loop runs, sum is increased by 8 |
| 9 | Yes | 45 | Loop runs, sum is increased by 9 |
| 10 | Yes | 55 | Loop runs, sum is increased by 10 |

## While loop in method run() trace table for running out of fuel user does not guess answer

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| fuelAvailable | isWon | fuelAvailable > 0 &&  isWon == false ? | User input | Notes |
| 27 | false | Yes | 10 | Loop runs, user enters 10, fuelAvailable becomes 17, isWon remains false |
| 17 | false | Yes | 10 | Loop runs, user enters 10, fuelAvailable becomes 7, isWon remains false |
| 7 | false | Yes | 10 | Loop runs, user enters 10, fuelAvailable becomes -7, isWon remains false |
| -7 | false | No | User can’t input now | Loop runs into if() loop, user can’t enter now ,fuelAvailable becomes 0, isWon remains false |
| 0 | false | No | User can’t input now | Loop ends. |

## While loop in method run() trace table for user does guess answer

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| fuelAvailable | isWon | fuelAvailable > 0 &&  isWon == false ? | User input | Notes |
| 27 | false | Yes | 5 | Loop runs, user enters 5, fuelAvailable becomes 22, isWon remains false |
| 22 | false | Yes | 4 | Loop runs, user enters 4, fuelAvailable becomes 18, isWon becomes true |
| 18 | true | No | User can’t input now | Loop ends. |

## While loop in method run() trace table for invalid input

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| fuelAvailable | isWon | fuelAvailable > 0 &&  isWon == false ? | User input | Notes |
| 27 | false | Yes | 0 | Loop runs, user enters 0, fuelAvailable doesn’t change, isWon remains false |
| 27 | false | Yes | -1 | Loop runs, user enters 4, fuelAvailable becomes 28, isWon remains false |
| 28 | false | Yes | -10 | Loop runs, user enters 4, fuelAvailable becomes 38, isWon remains false |

Invalid input will not change isWon (false) and (fuelAvailable > 0 && isWon ) is always false. Therefore, the loop will run forever with invalid input.

## Do-While loop in method main() testing that program continues and exits

|  |  |  |
| --- | --- | --- |
| User input | shouldContinue = CONTINUE\_GAME ? | Notes |
| y | Yes | User entered “y” to continue the game, loop continues |
| n | No | User entered ”n” to end the game, loop ends |

# Program codes

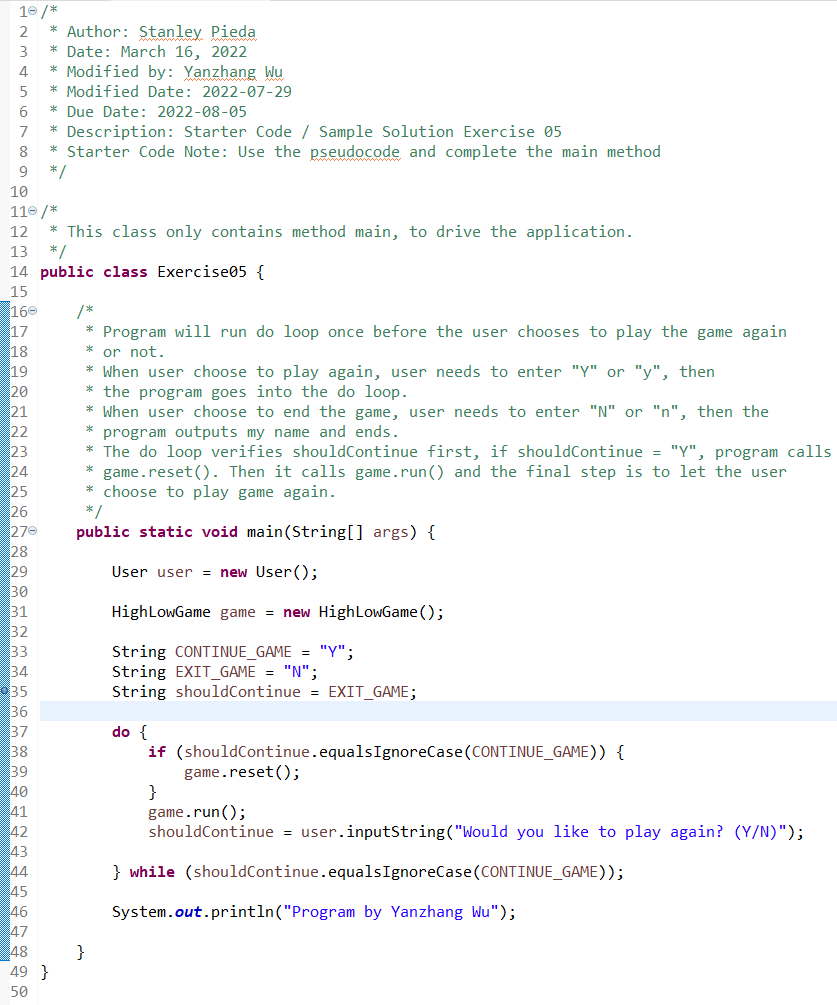
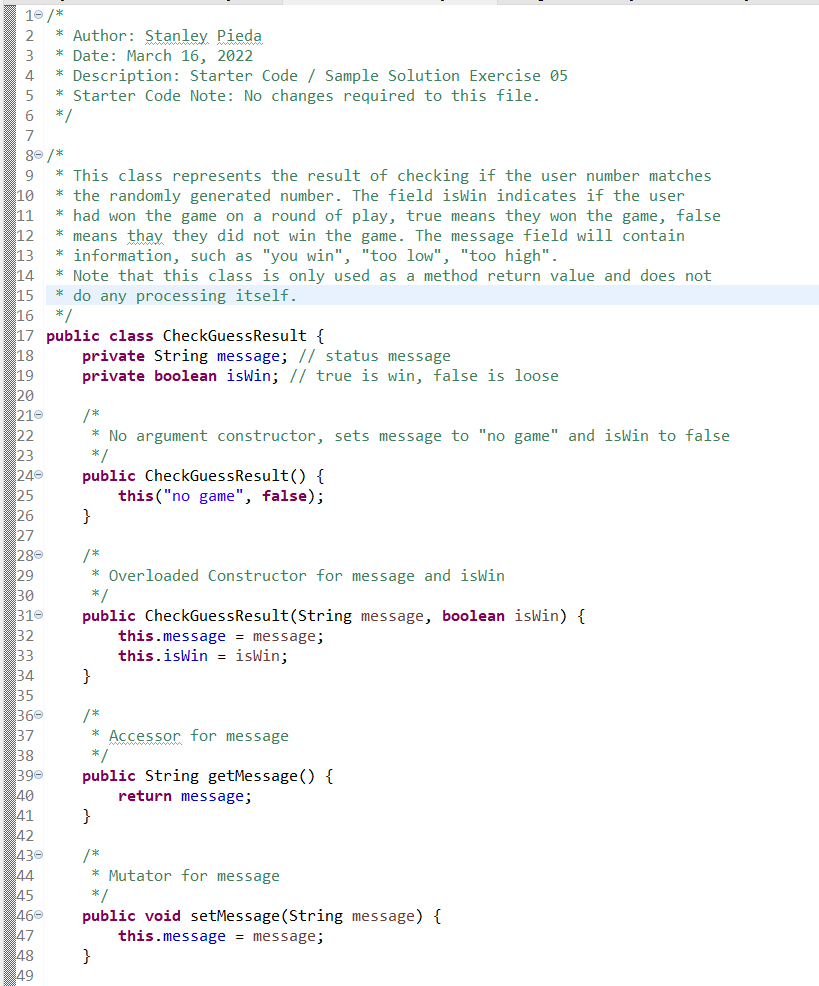


Figure . Class Exercise05

 Graphical user interface, text, application

Description automatically generated

Figure . Class CheckGuessResult

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated Graphical user interface, text, application

Description automatically generated

Figure . Class HighLowGame

Text

Description automatically generated Text

Description automatically generated

Figure . Class User

# Test plan

|  |  |  |
| --- | --- | --- |
| Input | Actual output | Description |
| 5  y  5  8  6  7  n | Guess the number from 1 to 10  You have 27 guess-fuel remaining.  guess:  5  You guessed the number  You win! It took you 1 guesses  Would you like to play again? (Y/N)  y  Guess the number from 1 to 10  You have 27 guess-fuel remaining.  guess:  5  too low, 22 guess-fuel remaining  guess:  8  too high, 14 guess-fuel remaining  guess:  6  too low, 8 guess-fuel remaining  guess:  7  You guessed the number  You win! It took you 4 guesses  Would you like to play again? (Y/N)  n  Program by Yanzhang Wu | Enter numbers as required, the program runs normally as expected. “y” and “n” are not case sensitive. |
| 0  -5  Hi | Guess the number from 1 to 10  You have 27 guess-fuel remaining.  guess:  0  too low, 27 guess-fuel remaining  guess:  -5  too low, 32 guess-fuel remaining  guess:  hi  Exception in thread "main" java.util.InputMismatchException  at java.base/java.util.Scanner.throwFor(Scanner.java:939)  at java.base/java.util.Scanner.next(Scanner.java:1594)  at java.base/java.util.Scanner.nextInt(Scanner.java:2258)  at java.base/java.util.Scanner.nextInt(Scanner.java:2212)  at User.inputInteger(User.java:20)  at User.inputInteger(User.java:28)  at HighLowGame.run(HighLowGame.java:100)  at Exercise05.main(Exercise05.java:41) | 0 and negative numbers not decrease the guess fuel, and the user needs to input the guess number again and again. For input of string, the program will end with error. |
| 10  10  10  hi | Guess the number from 1 to 10  You have 27 guess-fuel remaining.  guess:  10  too high, 17 guess-fuel remaining  guess:  10  too high, 7 guess-fuel remaining  guess:  10  too high, 0 guess-fuel remaining  You did not win, you used 3 guesses  Would you like to play again? (Y/N)  hi  Program by Yanzhang Wu | Enter 10 three times, the guess-fuel becomes 0 and the program keeps this play. For the input of choosing to play, if the user inputs strings others rather than “y” and “n”, the program ends |
| 10  10  10  0 | Guess the number from 1 to 10  You have 27 guess-fuel remaining.  guess:  10  too high, 17 guess-fuel remaining  guess:  10  too high, 7 guess-fuel remaining  guess:  10  too high, 0 guess-fuel remaining  You did not win, you used 3 guesses  Would you like to play again? (Y/N)  0  Program by Yanzhang Wu | For the input of choosing to play, if the user inputs numbers rather than “y” and “n”, the program ends |
| 5.5 | Guess the number from 1 to 10  You have 27 guess-fuel remaining.  guess:  5.5  Exception in thread "main" java.util.InputMismatchException  at java.base/java.util.Scanner.throwFor(Scanner.java:939)  at java.base/java.util.Scanner.next(Scanner.java:1594)  at java.base/java.util.Scanner.nextInt(Scanner.java:2258)  at java.base/java.util.Scanner.nextInt(Scanner.java:2212)  at User.inputInteger(User.java:20)  at User.inputInteger(User.java:28)  at HighLowGame.run(HighLowGame.java:100)  at Exercise05.main(Exercise05.java:41) | If the input is a double type, the program ends with error. |

# Screen shot

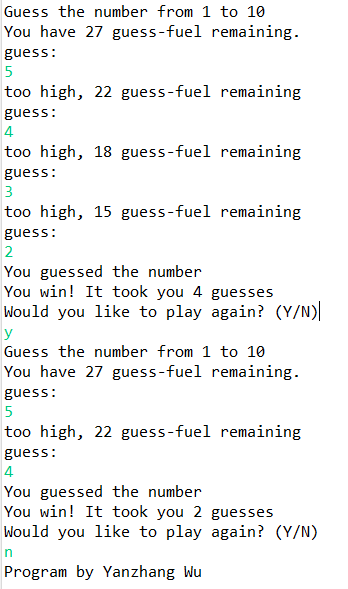


Figure . Program runs normally

Text

Description automatically generated

Figure . Invalid input

Text

Description automatically generated

Figure . Invalid String input for "Y"'or "N"

Text

Description automatically generated

Figure . Invalid number input for "Y"'or "N"

Text, letter

Description automatically generated

Figure . Invalid double input

# Question(s)

1. (Reflective) Examine method reset() and re-write the for loop as a while loop using pseucode and submit this as part of your write up in the MS Word document.
   * Which is your preferred version of the loop, and why? (Compare and contrast from perspective of ease of use and ease of reading between a for-loop, and a while-loop that both solve the same problem).

## Pseucode while loop

value = MIN

while value <= MAX

sum = sum + value

value++

end while

## for loop VS while loop

The same problem could be solved by for loop and while loop. In other words, for loop and while loop do the same thing.

However, if we can estimate the number of loop times, for loop is better since the condition and the change of the loop control variable are listed together in the parenthesis. Therefore, it’s easier to read. But for some problems, we don’t know how many times the loop will run. In this case, while loop is better.

Moreover, loop control variable for for loop could be defined as block-level-scope variable. It will be released after the for loop ends and we can’t access it any more. However, for while loop, the loop control variable needs to be defined before the while loop, for example, variable value above. After the while loop ends, the loop control variable can be accessed again. Therefore, if we don’t need to access the control variable after the loop, it’s better to use for loop since it can save more memory.

1. (Technical) When using a for loop the loop control variable is written in-line within the loop header.
   * What is the scope of this variable (class-scope, local-scope, or block-level-scope)?

block-level-scope

* + Can you access the loop control variable to print the last value it held below the loop body (yes/no)?

no