CPSC 1060: Introduction to Programming in Java

Project 3: Bagels (An Object-Oriented Guessing Game)

Introduction

"Pico, Fermi, Bagel" (also known as "Pico, Fermi, Zilch") is a guessing game similar to Mastermind. The opponent (in this case, the computer) starts by picking an *n*-digit secret number (where *n* is generally 2 or 3, but it can be greater than 3). Then, we make guesses by typing in *n*-digit numbers until we've guessed the secret number correctly. The fewer guesses required, the better. The opponent (computer) provides some feedback after each move:

- "Fermi" is printed for each digit guessed correctly, in the correct place.
- "Pico" is printed for each digit guessed correctly, in the incorrect place.
- "Bagel" is printed if no digits are correct.

Game Details

To clear up some ambiguities about how the game works, read this section and the rest of this document carefully, but don't worry: we've implemented the validation system (a method to compare the guess and the secret number) for you. You're welcome!

All "Pico" results are printed out before any "Fermi" results, and so the user does not know specifically which number is correct. For example, if the secret number is 250 and we guess 245, the feedback would be "Pico Fermi."

The winning guess should receive "Fermi" *n* times. For example, if the secret number is 921 and we guess 921, the feedback would be "Fermi Fermi Fermi".

Note that "Fermi" and "Pico" may show up multiple times (e.g. a secret of 123 and a guess of 231 produce "Pico Pico Pico") but "Bagel" is only printed once if necessary. For example, a secret of 789 and a guess of 123 produce "Bagel".

Though you are welcome to do so, you are ***not*** required to account for invalid user input, including digit values that are too big and guesses that do not have the correct number of digits. You can also assume that the user will not input leading zeros.

Implementation Details

The design of this project has already been done for you, so all you need to do is implement it. You will implement three classes and make use of a fourth, which is provided, to solve this problem in a modular fashion (using classes and methods). Use the UML diagram and documentation provided in the following sections to understand how the project is designed and how you should implement each method. You must adhere to this design exactly! This includes variable names, argument types, and return types.

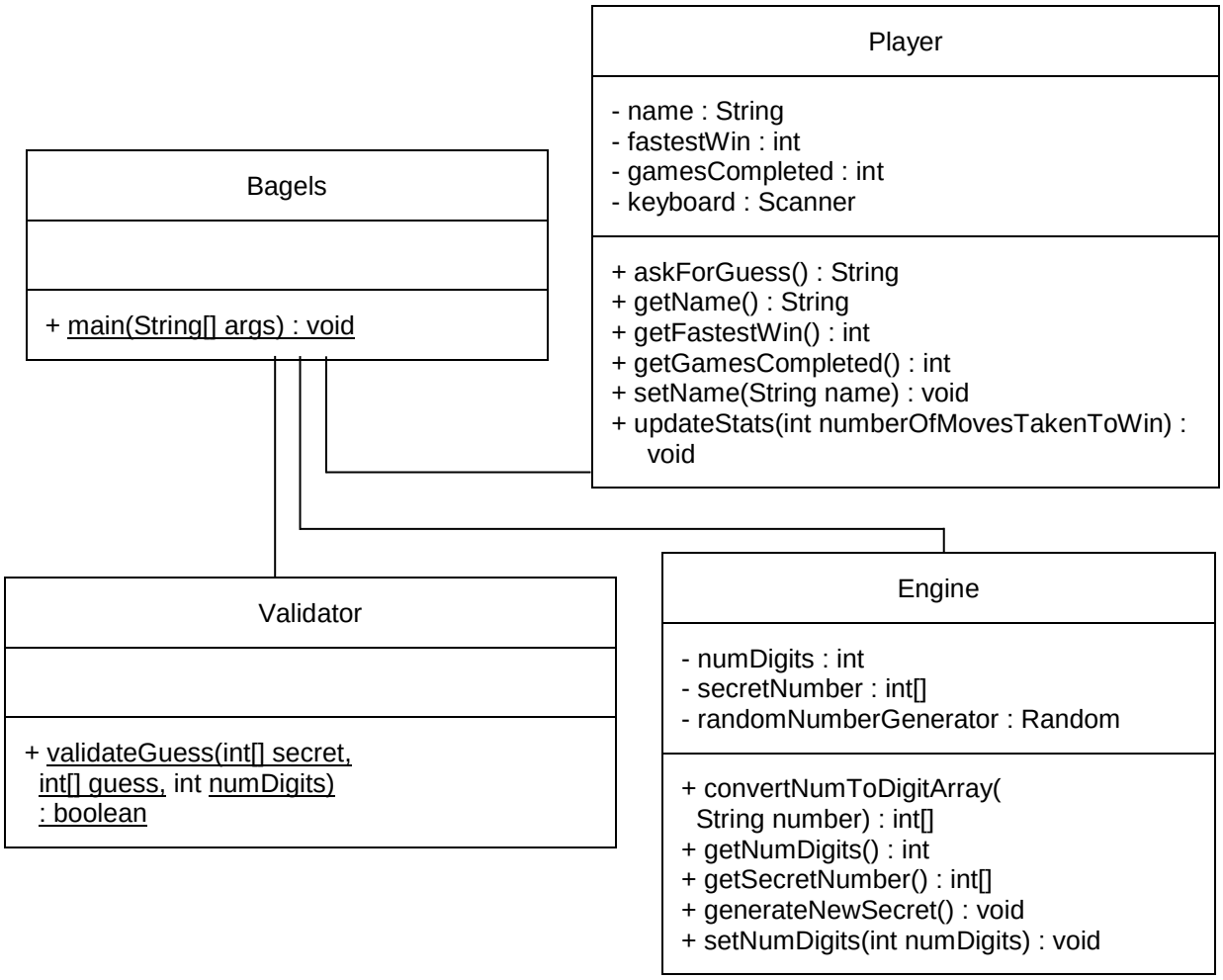
Hopefully, you will find this way of programming easier to work with, since you can write and test your code in smaller, organized blocks, and better reuse code. You may even think of each method as a smaller programming project!

You will need to use everything you've learned so far to complete this project, including arrays and classes/objects/methods. You will also likely want to use the following classes provided by Java at some point (though, not required):

- `int nextInt(int n)`, an instance method in the Random class
  - Returns a pseudorandom number between 0 (inclusive) and *n* (exclusive). Hint: you can use simple mathematical operations to specify a different lower bound and upper bound.
  - Also, you may use Math.random() to generate a pseudorandom number.
- `double pow(double a, double b)`, a static method in the Math class
  - Returns the value of a^b.
- `int MAX_VALUE`, a static field in the Integer class
  - The largest value an integer can have (2^{31}-1). Hint: this may serve as a good initial value.

- `int parseInt`, a static method in the `Integer` class that converts a String input to an integer.

UML Diagram



Documentation

Class: Bagels – The "driver" class that users will execute to play the game.

Methods

Type	Method Signature	Description
`static void`	`main(String[])`	Called automatically when the program is run. It should run according to the flowchart below and the examples at the end of this project.

Class: Player – Models the data and actions associated with the person playing the game. ***Do not add a constructor to this class!*** We will use the default constructor (which is built into every class that does not contain a constructor) to test this class.

Fields

Type	Name	Description
`String`	`name`	The first name of the player.
`int`	`fastestWin`	Like a high score. How quickly was the player able to guess the number? Best score possible is 1.
`int`	`gamesCompleted`	The number of games that have been completed (and thus won).
`Scanner`	`keyboard`	Used for user input in `askForGuess()`. This should be initialized once and reused for each call to `askForGuess()`.

Methods

Type	Method Signature	Description
`String`	`askForGuess()`	Using the keyboard field, the guess is read in from the keyboard.
`String`	`getName()`	Returns the player's name.
`int`	`getFastestWin()`	Returns the fastest win.
`int`	`getGamesCompleted()`	Returns the number of games completed.
`void`	`setName(String)`	Sets the player's name.
`void`	`updateStats(int)`	Should be called once after finishing each game. It updates the `gamesCompleted` field and possibly the `fastestWin` field.

Class: Engine – Encapsulates data and actions related to the actual game in progress. ***Do not add a constructor to this class!*** We will use the default constructor (which is built into every class that does not contain a constructor) to test this class.

Fields

Type	Name	Description
`int`	`numDigits`	The number of digits the user wishes to play with.
`int[]`	`secretNumber`	The computer's secret number. This is initialized by generateNewSecret().
`Random`	`randomNumberGenerator`	An instance of Java's Random class, used to generate pseudorandom secret numbers.

Methods

Type	Method Signature	Description
`int[]`	`convertNumToDigitArray (String)`	Converts a number (as a String) to an array of ints. The ordering **must** go from most to least significant digits. For example, if the number String is "732" then index 0 of the array should contain 7.
`int`	`getNumDigits()`	Returns the number of digits.

`int[]`	`getSecretNumber()`	Returns the secret number.
`void`	`generateNewSecret()`	Changes the `secretNumber` field to a random number in the range 10^{n-1} and 10^n-1.
`void`	`setNumDigits(int)`	Sets the number of digits to the input parameter.

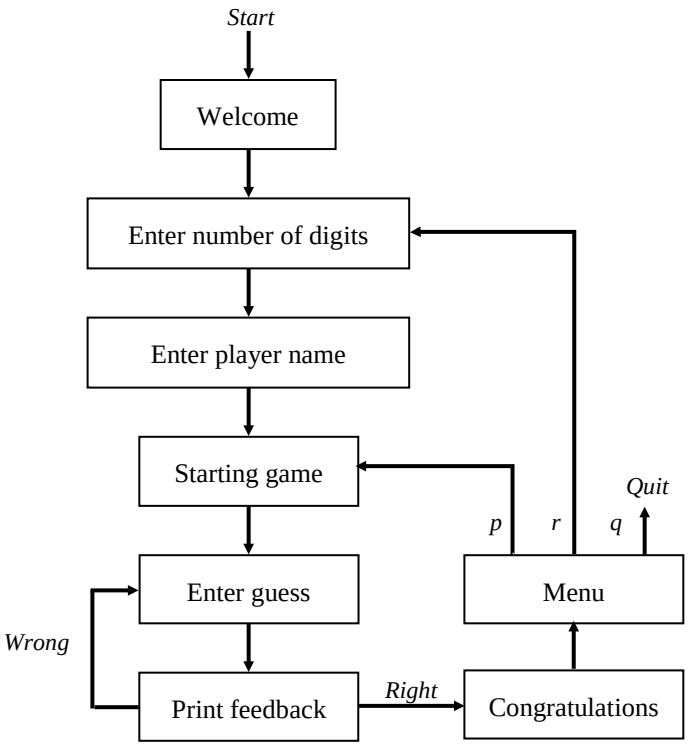
Class: Validator – Contains a single static method to validate guesses. This class is provided and should ***NOT*** be modified.

Methods

Type	**Method Signature**	**Description**
`static boolean`	`validateGuess(int[], int[], int)`	Prints out the feedback from the opponent (e.g. "Fermi Pico") and returns `true` if the guess was correct or `false` if the guess was incorrect. The last argument (`numDigits`) should match the length of both arrays.

Flowchart

Study the flowchart (in addition to the sample runs) below when writing your main method.



Additional Instructions

1. Your program must include a comment between the import statements and the class declaration. Copy and agree to the comments below, and fill in the file name, your name, the submission date, and the program's purpose.

```
/*
* [Class name here].java
* Author:  [Your name here]
* Submission Date:  [Submission date here]
*
* Purpose: A brief paragraph description of the
* program. What does it do?
*
* Statement of Academic Honesty:
*
* The following code represents my own work. I have neither
* received nor given inappropriate assistance. I have not copied
* or modified code from any source other than the course webpage
* or the course textbook. I recognize that any unauthorized
* assistance or plagiarism will be handled in accordance
* with the policies at Clemson University and the
* policies of this course. I recognize that my work is based
* on an assignment created by the School of Computing
* at Clemson University. Any publishing or posting
* of source code for this project is strictly prohibited
* unless you have written consent from the instructor.
*/
```

2. Your program cannot use a **break** statement. The use of a break statement for this project will result in a grade of zero.
3. Your program cannot use a **continue** statement. The use of a continue statement for this project will result in a grade of zero.

All instructions must be followed for full credit to be awarded.

Project Submission

After you have thoroughly tested your program with all the examples provided plus your own examples, submit **`Bagels.java`**, **`Player.java`**, and **`Engine.java`** to our course website. You should not include **`Validator.java`** or any **`.class`** files.

Project Grading

All projects are graded out of a possible 100 points. Programs not submitted before this assignment's deadline will receive a grade of zero. Programs that do not compile, use a break statement, or use a continue statement will also receive a grade of zero. You must make absolutely certain your program compiles before submitting, and you must thoroughly test your program and each class and their methods with different inputs to verify that it is working correctly. You must make absolutely certain your project conforms to all UML specifications; otherwise, it may not compile or run correctly with our testing program(s), which may result in a failing grade on this project.

This project will be graded for both correctness and style:

Style [20pts]

- 5 points for including the class comment required for all projects in all of your submitted **.java** files

- 5 points for commenting all methods
- 10 points for adhering to the UML diagram and documentation described above

Correctness [80pts]

- 80 points for correct output on various test cases. These test cases may test individual classes to see if the class is correct based on the specifications and examples found in this project's instructions. Also, these test cases may include testing multiple or all classes together to see if they are correct based on the specifications and examples found in this project's instructions.

Example Runs

Your program should be consistent with the examples below and all specifications in this project. All input/output should be formatted as shown.

```
Welcome!
Enter the number of digits to use: 2
Enter the player's name: Kyle

Starting game #1.
Enter your guess: 12
Bagel
Enter your guess: 34
Bagel
Enter your guess: 56
Fermi
Enter your guess: 57
Fermi
Enter your guess: 58
Fermi
Enter your guess: 59
Fermi Fermi
Congratulations!  You won in 6 moves.

Statistics for Kyle:
Games completed: 1
Number of digits: 2
Fastest win: 6 guesses
p - Play again
r - Reset game
q - Quit

What would you like to do? p


Starting game #2.
Enter your guess: 83
Bagel
Enter your guess: 72
Fermi
Enter your guess: 92
Fermi Fermi
Congratulations!  You won in 3 moves.
```

```
Statistics for Kyle:
Games completed: 2
Number of digits: 2
Fastest win: 3 guesses
p - Play again
r - Reset game
q - Quit

What would you like to do? r

Enter the number of digits to use: 3
Enter the player's name: Gertrude

Starting game #1.
Enter your guess: 123
Pico
Enter your guess: 456
Pico
Enter your guess: 789
Pico
Enter your guess: 147
Pico
Enter your guess: 258
Pico Fermi
Enter your guess: 618
Fermi Fermi
Enter your guess: 628
Fermi
Enter your guess: 418
Fermi Fermi
Enter your guess: 218
Fermi Fermi
Enter your guess: 118
Fermi Fermi
Enter your guess: 418
Fermi Fermi
Enter your guess: 318
Fermi Fermi
Enter your guess: 518
Fermi Fermi Fermi
Congratulations!  You won in 13 moves.

Statistics for Gertrude:
Games completed: 1
Number of digits: 3
Fastest win: 13 guesses
p - Play again
r - Reset game
q - Quit

What would you like to do? q

Goodbye!
```

```
Welcome!
Enter the number of digits to use: 3
Enter the player's name: Fanny

Starting game #1.
Enter your guess: 182
Pico
Enter your guess: 314
Fermi
Enter your guess: 516
Fermi Fermi
Enter your guess: 517
Pico Fermi
Enter your guess: 716
Fermi Fermi Fermi
Congratulations!  You won in 5 moves.

Statistics for Fanny:
Games completed: 1
Number of digits: 3
Fastest win: 5 guesses
p - Play again
r - Reset game
q - Quit

What would you like to do? p


Starting game #2.
Enter your guess: 258
Bagel
Enter your guess: 149
Bagel
Enter your guess: 367
Pico
Enter your guess: 700
Fermi
Enter your guess: 607
Fermi Fermi
Enter your guess: 606
Fermi Fermi Fermi
Congratulations!  You won in 6 moves.

Statistics for Fanny:
Games completed: 2
Number of digits: 3
Fastest win: 5 guesses
```

```
p - Play again
r - Reset game
q - Quit

What would you like to do? q

Goodbye!
```