J – Looping Structures

Student Exercises

# Exercises

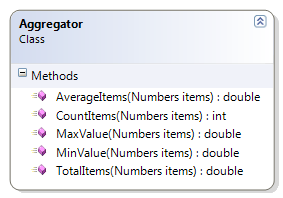
1. **Aggregator** – countItems(), minValue()
2. **Math** – factorial(), isPrime(), power(), and greatestCommonDenominator()
3. **GuessingGame** – This example expands on the sample GuessingGame by 1) re-working the guess() method to make guessing more efficient by using the SecretNumber’s guessWithHint() method, and 2) overloading the guess() method to continue guessing until the correct answer is found (and returning the number of attempts it took to find the correct answer).

# Aggregator

## Problem Statement

Expand on the sample Aggregator class to perform the following aggregations.

* **CountItems()** – Return the count of the number of values in the Numbers object.
* **MinValue()** – Return the smallest value in the set of values of the supplied Numbers object. If no values exist in the Numbers object, return the largest whole number supported by the programming language.

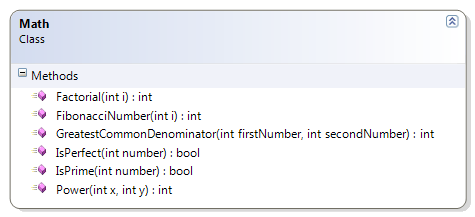


# Math

## Problem Statement

Expand on the sample Math class by providing methods to perform the following.

* **Factorial()** – Calculate the factorial of a given number. If the given number is negative, throw an exception. If the calculated value for the factorial will exceed the maximum integer value (int.MaxValue), then return zero (0) as a result.
* **IsPrime()** – Identify if a number is or is not a prime number. A prime number is a number with only two divisors: 1 and the number itself. By definition for this problem, numbers less than one are not considered prime numbers.
* **Power()** – Calculate the value of x to the power of y (as in **xy**). Use looping logic (addition) to get the result.
* **GreatestCommonDenominator()** – Find the greatest common denominator between two whole numbers.



# GuessingGame

This example expands on the sample GuessingGame by adding more guess methods to make guessing more efficient by using the SecretNumber’s guessWithHint() method.

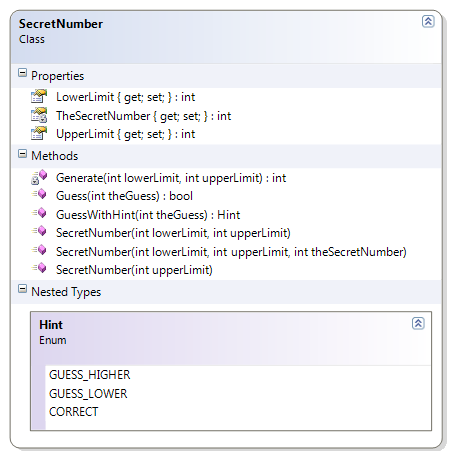
## Problem Statement

In a regular number guessing game between two people, one person asks another person to guess a whole number between a pair of values. In this example, two different classes take on the roles of the two people in the game: The SecretNumber class takes on the role of the person who has picked a value between some lower and upper limit, while the GuessingGame class takes on the role of the person who has to guess what that hidden number is.

### SecretNumber

Objects of this class will store a hidden value between some upper and lower limit (inclusive). The SecretNumber class supports these public methods:

* **GetLowerLimit()** – Returns a number representing the lower end (inclusive) of the range of possible values for the hidden value.
* **GetUpperLimit()** – Returns a number representing the upper end (inclusive) of the range of possible values for the hidden value.
* **GuessWithHint()** – Returns CORRECT if the supplied value matches the hidden value, otherwise it returns GUESS\_HIGHER if the attempted guess was too low or GUESS\_LOWER if the attempted guess was too high.



**🡸 This supporting class is already coded for you.**

### GuessingGame

This class simulates the action of guessing what hidden value is stored inside of a SecretNumber object. The SecretNumber is supplied to the constructor of the GuessingGame, and the following methods attempt to find out what that number is.

* **GuessUsingHints()** – This first method simply tries to guess the hidden value of the SecretNumber. It has “unlimited” guesses, and it will return the number of attempts it took to find out what that hidden number is. **Use the GuessWithHint method of the SecretNumber object.**
* **GuessUsingHints(MaxAttempts : Integer)** – This method will also try to guess the SecretNumber’s hidden value, but it is limited to a maximum number of guesses. This method will return the actual number of attempts it took to find the hidden value, or it will return the GUESS\_FAILED constant if it was unable to guess the secret number. **Use the GuessWithHint method of the SecretNumber object.**

As an additional exercise, create a simple driver that instantiates a SecretNumber object and supplies it to a GuessingGame object. Show how many attempts it took to guess the hidden number when using hints.

