Programming Exercises - PRO1 - Session 12

Exercise 12.01

Update the equals method you did in Exercise 11.01, so it's done properly with a parameter of type `Object`, rather than `MyDate`.

Exercise 12.02

Implement a class `Car` holding information of a car. The class should have:

a) 5 instance variables: `make`, `model`, `color`, `licenseNumber` (all of type `String`) and `year` (of type `int`).
b) A 5-argument constructor setting all instance variables to values passed as arguments.
c) A 4-argument constructor with `make`, `model`, `color`, and `year` as arguments. The instance variable `licenseNumber` should be set to some value indicating that the car has no licence number.
d) Get-methods for all instance variables.
e) Set methods for `color` and `licenseNumber`.
f) A method called `copy` that returns a reference to a `Car`-object with the same values for the 5 instance variables.
g) A `toString` method returning all information in a string.
h) An `equals` method returning true if the object passed as argument is a car with the same values of all 5 instance variables.

For those wondering why it doesn't mention making a test class for the `Car` class, then don't worry, you will get to use the `Car` class plenty at our next lesson.

Exercise 12.03

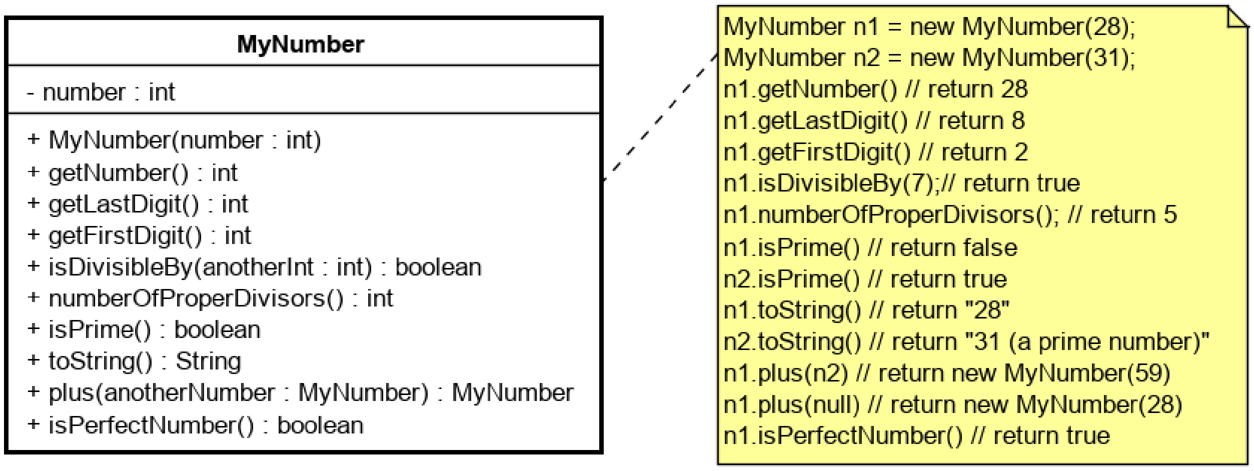
[Gaddis] Find the Error 1, 2, 3, p. 460

[Gaddis] Algorithm Workbench 1, p. 461-462 (start by finding and fixing the errors in the code)

[Gaddis] Programming Challenges 11, p. 467-468

[Gaddis] Programming Challenges 13, p. 469

Exercise 12.04



```
MyNumber n1 = new MyNumber(28);
MyNumber n2 = new MyNumber(31);
n1.getNumber() // return 28
n1.getLastDigit() // return 8
n1.getFirstDigit() // return 2
n1.isDivisibleBy(7);// return true
n1.numberOfProperDivisors(); // return 5
n1.isPrime() // return false
n2.isPrime() // return true
n1.toString() // return "28"
n2.toString() // return "31 (a prime number)"
n1.plus(n2) // return new MyNumber(59)
n1.plus(null) // return new MyNumber(28)
n1.isPerfectNumber() // return true
```

Implement the `MyNumber` class shown above, with the following requirements:

- An instance variable of type `int`, a constructor, and a getter for the instance variable.
- The class must be immutable, i.e. no methods are changing the instance variable.
- A method `getLastDigit()` that return the last digit. *Note:* modulus 10 of a positive number gives the last digit, and modulus 10 of a negative number gives a negative value of the last digit.
  In this method the last digit of 1234 and of -1234 should be returned as 4 in both cases (never -4).
- A method `getFirstDigit()` that return the first digit. *Hint*: Dividing a number by 10 gives all digits except the last one. Doing this in a loop until you get a one-digit number, gives the first digit.
  Example: The first digit of 1234 is 1 because: (1234/10 = 123 → 123/10 = 12 → 12/10 = 1).
- A method `isDivisibleBy(int anotherInt)` that return `true` if the number is divisible by the parameter value, otherwise the method should return `false`.
- A method `numberOfProperDivisors()` that return how many values from 1 to the number (not including the number) that the number is divisible by. *Example*: 28 has 5 proper divisors because 28 is divisible by 1, 2, 4, 7 and 14. *Note*: Only positive values have proper divisors.
  Hint: Use a loop counting how many times the `isDivisibleBy(...)` method return `true`
- A method `isPrime()` that return `true` if the number is a prime number, otherwise `false`.
  Note: A prime number has exactly 1 proper divisor.
- A method `toString()` that return the number as a string, and if it's a prime number then also that info.
  Example 1: if number is 28 then `toString()` return `"28"`
  Example 2: if number is 31 then `toString()` return `"31 (a prime number)"`.
- A method `plus(MyNumber anotherNumber)` takes another `MyNumber` object as argument (not an `int`) and return a new `MyNumber` object with the sum of the two integers. *Note*: If the parameter variable is `null`, then change it to a new `MyNumber` object with the value 0.
  Example 1: if a `MyNumber n1` has the integer 28 and another `MyNumber n2` has the integer 31, then `n1.plus(n2)` return a new `MyNumber` object with the value 59 (i.e. 28 + 31).
  Example 2: if a `MyNumber n1` has the integer 28, then `n1.plus(null)` return a new `MyNumber` object with the value 28 (i.e. 28 + 0).
- Method `isPerfectNumber()` return `true` if it's a perfect number, i.e. if the sum of all proper divisors is equal to the number itself.
  Example: 28 is a perfect number because the sum of its proper divisors equals 28 (1+2+4+7+14=28).