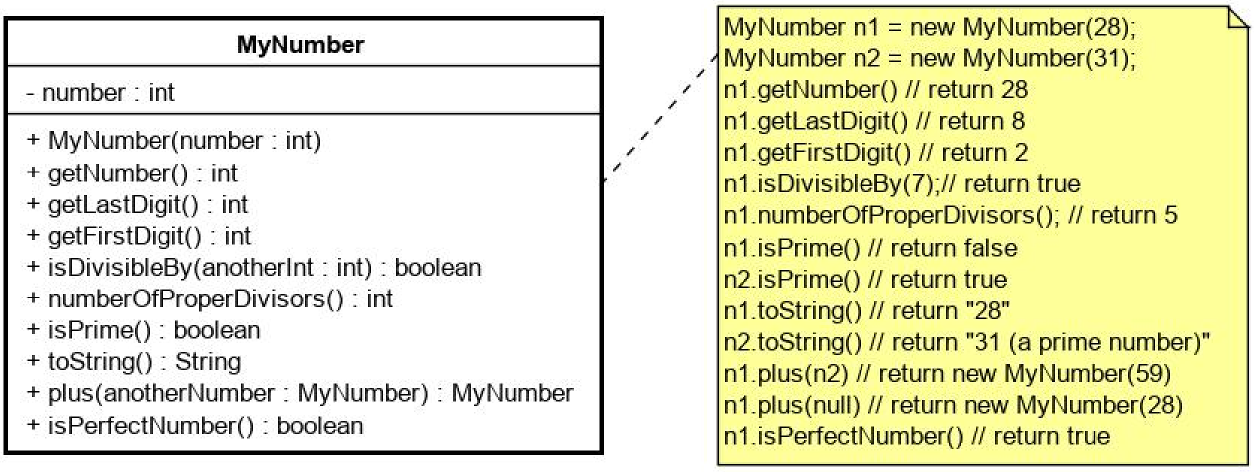
Exercise, SDJ1

Exercise: MyNumber



```
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
```

Requirements to implementation:

- An instance variable of type `int`, a constructor and a getter for the instance variable.
- The class is immutable, i.e. no methods are changing the instance variable
- Method `getLastDigit()` 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.
  Example: the last digit of 1234 and of -1234 is 4 in both cases (and not -4).
- Method `getFirstDigit()` return the first digit.
  Hint: Dividing a number by 10 gives all the digits except the last one. If you do that in a loop until you end up with one digit, then you have found the first digit.
  Example: the first digit of 1234 is 1 because: (1234/10 = 123 → 123/10 = 12 → 12/10 = 1)
- Method `isDivisibleBy(int anotherInt)` return `true` if the number is divisible by the parameter value, otherwise return `false`.
- Method `numberOfProperDivisors()` 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 `isDivisibleBy(...)` return `true`
- Method `isPrime()` return `true` if the number is a prime number, otherwise `false`.
  Note: a prime number has exactly 1 proper divisor.
- Method `toString()` return the number as a string and if it is a prime number, then also this.
  Example 1: if number is 28 then `toString()` return `"28"`
  Example 2: if number is 31 then `toString()` return `"31 (a prime number)"`.
- Method `plus(MyNumber anotherNumber)` is taking 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 is 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)