Exercises module 14 – Getters & Setters

14.1 – Writing getters and setters

1. Create a new class named **Customer** like:

```
public class Customer
{
    private int id;
    private String name;
    private String address;
    private String zipcode;
    private String city;
    private String email;
}
```

2. Create a constructor that sets all the values when created.

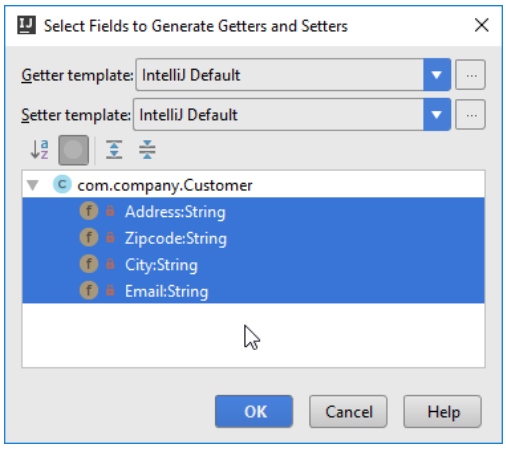
3. Create **Getters** and **Setters** so that the properties can be accessed from the outside the class.

   As a start, only write the Getters and Setters methods for the id and Name field.

> ***Important***
> The convention is that **Getters** don't take any parameters and **Setters** always returns void.

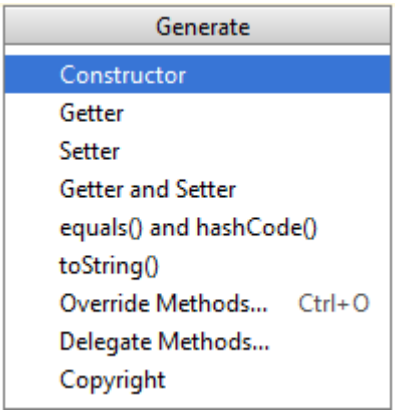
4. Instead of writing them manually, we can let IntelliJ create them for us.
   a. Place the cursor inside the Customer class
   b. Press **alt+insert** and you should see this menu appear:

c. Select **Getter and Setter** and then select the remaining fields to generate Getter and Setter for:



5. Now every field should have its own **Getter** and **Setter** method.

6. Explore the other available generators that are available when you press **alt+insert** inside a class, like creating constructors!



14.2 – Input validation

1. Setter methods are useful to validate that the data provided is valid and within accepted range/rules. Add the following validation rules to the class created in the previous step.

    a. **ID**, if negative, set it to Zero.
    b. **Name**, if the length of the name is zero, then set it to "Unknown".
    c. **Address**, if the address is longer than 20 characters, then cut it after 20 characters and ignore the rest.
    d. **Zipcode** should always be 5 characters long and all numbers. Use the **Character.isDigit** method to check the characters. If not valid set it to "00000".
    e. **Email** must have a length> 5 characters and contain a '@' and a '.' character. The last found '.' character must be located after the '@' character. If not valid make it empty "".

2. Write simple tests to verify the functionality in the validation rules by using the corresponding Getters.

3. The main reason for using getters/setters is that it makes it possible to change the internal implementation without changing the public interface. This makes it easier to refactor and maintain your application in the long run.

    Let's **refactor** and make the internal representation for the **id** a **string** instead of an integer without changing the public contract/interface of the class.

```
private String id;
```

    Make sure it compiles and the validation rules still works!

4. Now the values that we set via the constructor is never checked and we need to make sure that the data passed to the object is always checked and valid. Complying with the rules in the setters.

    To make sure that the object always is valid we need to modify the constructor so that it calls the corresponding setter methods and don't set the fields directly.

Questions and concepts to study further on your own:

- Getters/Setters. Evil. Period.
  http://www.yegor256.com/2014/09/16/getters-and-setters-are-evil.html
- Refactoring
  https://en.wikipedia.org/wiki/Code_refactoring
- Data verifications in Getter/Setter or elsewhere?
  http://stackoverflow.com/questions/2750

- (no) Properties in Java?
  http://stackoverflow.com/questions/70471/no-properties-in-java