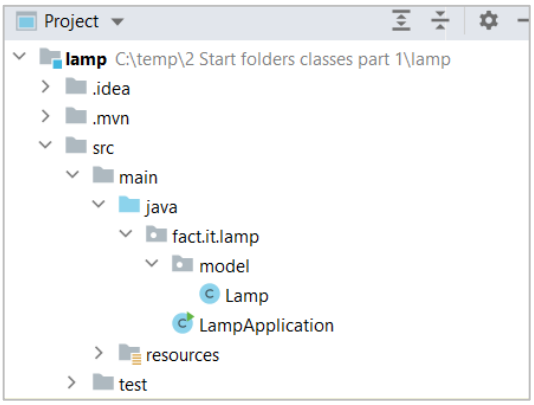
Exercises Classes Part 1

Use the start folders on Canvas.
The classes are already present and need to be completed.
The test file is always coloured red in the start folders. This is normal. As long as you have not completed the necessary code, it will remain red...

Exercise 1



Complete the class **Lamp** with 2 attributes:

- *brand*: String
- *power*: int

In the Lamp class, generate a *no-arg constructor* (= constructor without parameters) and *getters* and *setters* for each attribute.

Create *a* method *showPowerInTermsOfStars()* which returns a String showing as many "*" as the number of power of the lamp.
Example with power = 15:

```
***************
```

Test your code with the given unit tests.

Create in the application an object of this class (brand = Osram, power = 8) and use its methods in order to show the following output:

```
You created a Lamp-object with the following values:
Osram - 8 Watt(********)
```

Exercise 2

Complete the **Runner** class with 3 attributes:

- *name*: String
- *age*: int
- *bestPerformance*: double (this is the runner's best performance of the 100m sprint in terms of seconds)

In the Runner class, generate a *no-arg constructor*, *getters* and *setters* for each attribute.
In the setter, make sure that the age can never be set negatively. If this does happen you have to set the age = 0.

Create a method *getSpeed* which returns the best performance of the 100m in terms of km/h. You can calculate this in the following way:

360 / (best performance achieved in seconds)

Test your code with the given unit tests.

Create in the application an object of this class and use its methods in order to show the following output:

```
You created a Runner-object with the following values:
Tyson Gay (37 years), best performance on the 100m = 9.58
this runner achieved 37.578288100208766 km/h on the 100m sprint
```

Exercise 3

Complete the class **Spell** with 2 attributes:
- *text*: String
- *numberOperations*: int

In the Spell class, generate a *no-arg constructor.* Then generate a *getter* and *setter* for the attribute *text*. For *numberOperations* you should only generate a *getter.*

Create the following methods:
- *echo* doubles the *text* and increases the *number of operations* by 1 (so the text "pass" becomes "passpass"). This method has no parameters and no return value.

- *completeWith(String word):* the parameter is pasted at the end of the existing text and *numberOperations* is increased by 1. This method has no return value.

- *addAt(String word, char location):* the word is pasted to the existing text and *numberOperations* is increased by 1. The location is 'b', 'f' or 's' which means that the text is pasted at the back, front, or both sides (use a switch to work out this method). This method has no return value.

Test your code with the given unit tests.

Create in the application an object of this class, set the *text* to "pass" and call the method *echo()* twice. After that, show the values of this object as follows:

```
You created a Spell-object with the following values:
The text = passpasspasspass and 2 operations were performed on it
```

Exercise 4

Complete the **Student** class with 3 attributes:
- *name*: String
- *level*: char
- *grade*: int

In the Student class, generate a *no-arg constructor* and *getters* and *setters* for each attribute.
- Make sure that the grade can only have values from 1 to 6 (inclusive). If this is different from one of these values, the grade will be set to 1. Do this in the setter of the *grade* attribute.

- Make sure that the *level* can only be set to 'S', 'M' or 'E'. If this is different from one of these values, the level will be set to 'S'. Do this in the setter of the attribute *level*.

Create the following methods:

- *increaseGrade* (no parameter or return value) increases the *grade* by 1 .
  Attention: if the grade = 6, you may not raise it any more. In this case, the grade must therefore remain 6.
- *getLevelExplained* gives the meaning (return type = String) of the *level*.
  The list below gives the corresponding text for each value:
  'S' "Starting level"
  'M' "Middle level"
  'E' "End level"

Test your code with the given unit tests.
Create in the application an object of this class and use its methods in order to show the following output:

```
You created a Student-object with the following values:
Jan (grade = 2) currently has level E. This means:
End level
```