Programing Technologies
Lesson 2

1. Carefully follow the presentation about assertion in Java (available online at the page of the class).

Create a project in which you manage information about students registering to a lab.

2. The name of the project is `Lesson2`, the main class is called `Main`, the package name is derived from the name of the project.
3. Add a new class that can store information about a student (name, age).
4. Add `get`ters, `set`ters, `toString`, and a parameterized `constructor` to the class.
5. Add another class to the project named `Lab`. In a `Lab` you have a private `int` holding the headcount and a private `ArrayList` holding the list of students registered to the lab.
6. Update the Lab class. Add getters to the class and add the `addStudent(Student s)`, and the `removeStudent(Student s)`. The constructor may initialize the headcount and the list.

```
public void addStudent(Student s) {
    headcount++;
    getStudents().add(s);
}//addStudent
public void removeStudent(Student s) {
    headcount--;
    getStudents().remove(s);
}//removeStudent
```

7. Add another method to the class that prints out the list of students alphabetically ordered.
    - Find out how you can sort the elements of a `ListArray`.
    - Modify the `Student` class as it is needed (implement either the `Comparator` or the `Comparable` interface).
8. In the main method create a Lab named `lab01` and add three students to it.

Update this project and use assertion during the test and debug process.

9. With an assert make sure that no matter what you make wrong in the management of the list, the size of it and the headcount remain the same.
10. Try to remove a student from the list that is not present. Print out the headcount and the size of the list. (Here should have been an assert. Why we do not see it?)
11. Enable asserts in your project and run the code again. Solve the problem...
12. Add a new property to the `Student` so that at the instantiation it can be told if the student is male or female.
13. Modify the `toString()` method so that if the gender is '`m`' you add (male) and if it is '`f`' you add female to the end of the returned String. There may not be other possibilities so in the rest of the cases use an assert to stop the program. Run the program. What happens?

Create a project in which you represent players of a blackjack table.

14. The name of the project is `Lesson2`, the main class is called `Main`, the package name is derived from the name of the project.
15. Add a new class that can store information about a card. For this you will have to create two `enum` types. One to store the possible colors, and one to store the possible values.
    - Write the `CardColor enum` type
    - Write the `CardValue enum` type
    - Add the Card class to the project. This class has two properties the value and the color. Use the above defined two types for these attributes.
16. Add `get`ters, `set`ters, `toString`, and a parameterized `constructor` to the `Card` class.
17. Add another class to the project named `Hand`. In a `Hand` you can store `Cards`. Add methods to add and to remove `Cards` to and from the `Hand`. Add also a `getCards` method that returns the `List` of `Cards` in the hand. It is important to pay attention not to let more `Cards` of the same value and color to be in a `Hand`. To solve this introduce `SameCardsInHandException` to the project and use the equals method. Finally override the `toString` method as well.
18. In the main class of the project instantiate a new Hand, give some cards to it.
    - Read some more cards from an input file. In the file the following format is used: `CardColor CardValue` (e.g. `HEART FIVE`)
    - Be careful if from the file an already existing card is read you must not add the card, but you have to read the remaining ones.