# Training materials

* The Java Tutorials. [Lambda Expressions](https://docs.oracle.com/javase/tutorial/java/javaOO/lambdaexpressions.html).
* The Java Tutorials. [Method References](https://docs.oracle.com/javase/tutorial/java/javaOO/methodreferences.html).
* The Java Tutorials. [The Collection Interface](https://docs.oracle.com/javase/tutorial/collections/interfaces/collection.html). Subsection Aggregate Operations.
* The Java Tutorials. [Lesson: Aggregate Operations](https://docs.oracle.com/javase/tutorial/collections/streams/index.html).
* The Java Tutorials. [Reduction](https://docs.oracle.com/javase/tutorial/collections/streams/reduction.html).
* The Java Tutorials. [The Reflection API](https://docs.oracle.com/javase/tutorial/reflect/index.html).

# Code Exercise

Create the superclass Trial for a trainee trial. Use the only field account to present a trainee. Any trial consists of two tests to be estimated by integer values (mark1 and mark2) from 0 to 100 inclusively. A trial is considered to be passed if the sum of marks is not less than some class constant.

Create subclasses for following kinds of trials:

⎼ LightTrial - simplified (or light) trial. It also includes two tests. But a trial is passed if both marks are not less than some constants for every test.

⎼ StrongTrial - complicated (or strong) trial with two tests too. A trial is passed if the sum of a half mark1 and a whole mark2 is not less than the same constant as in the base trial.

⎼ ExtraTrial - extraordinary (or extra) trial. It contains an additional test. This trial is passed if the base trial is passed and a mark3 for an additional test is not less than some constant.

A totally valid json file contains an array of objects in the format:

[

{

class: value, //"Trial", "LightTrial" or "StrongTrial"

args: {

account: value,

mark1: value,

mark2: value

}

},

…,

{

class: "ExtraTrial”,

args: {

account: value,

mark1: value,

mark2: value,

mark3: value

}

},

...

]

If some object contains extra data, it is considered correct. This data must be registered at the WARN level. Otherwise an object is not correct, and it must be registered at the ERROR level.

Define the Runner class in the default package. An argument of the Runner is the name of a csv file.

The algorithm of the method main( ):

1. Create the List implementation from a json file. Use [Gson library](https://github.com/google/gson) to parse it and to identify trial entities.

2. Print the collection content (one element per line).

3. Print the number of passed trials.

4. Sort the collection by the sum of first and second marks.

5. Print sums of first and second marks from the collection (one sum per line).

6. Create a new collection from unpassed trials, clear all marks and print this collection. Check whether all trials are failed (the result type is boolean).

7. Create a numeric array from sums of first and second marks of sorted collection (see item 4) and print it in the format:

sum[0], sum[1], … , sum[sum.length - 1]

#### Замечания и ограничения

– Для преобразования из строки в класс используйте [Reflection API](https://docs.oracle.com/javase/tutorial/reflect/index.html).

– Для обработки коллекции используйте стримы.

– Этапы выполнения задачи ([см. описание процедуры](https://docs.google.com/document/d/1N-JIvSlmT_VqNA7aensAfMDhacMGLUDKhAIOdmax-p8/view) в задаче start1)

Этап 1. Реализация нулевой версии решения.

Имя архива **start3.zip**.

Этап 2. Методические рекомендации по решению.

Заполните [форму рецензирования](https://docs.google.com/forms/d/e/1FAIpQLSf6jW2oiE7AiPTZAS2cOUbFj76rEPW7z-WS-D_Xaq6_wKnPcg/viewform), указав в поле **Result** значение **yes**.

Этап 3. Контроль.

Имя архива **start3\_2.zip**.