CTS 02.04.2024

Number 2

Name:
Group Number:
Grade: ☐ /10p

A software application is being developed for a company that hires contractors for various companies.

4 points: The implemented application must allow the creation of various employment contracts, depending on the requirements of the companies. In addition to the elements that cannot be missing from such a contract: start date (String), end date (if undetermined, it will be marked as "undetermined"), salary; some contracts may also contain many specifications such as annual bonus grant, stock option grant, confidentiality clause, non-compete clause, etc. Once a contract is created, it is final and cannot be modified. The implementation will be carried out starting from the **Icontract** interface.

1 point: In main(), a relevant code sequence must be implemented to highlight the implemented design pattern.

4 points: For each created contract, the occupational safety document will be generated, which is the same document for all employees.

The occupational safety document is a complex one, its generation takes quite some time because of the information that needs to be extracted from an external service, therefore the implementation must ensure the optimization of the poster creation process. The posters will, of course, contain the details of the promoted event. The implementation of this poster generation component will be done through the implementation of the **IOccupationalSafety** interface.

```
public interface IContract{
   public void printDetails ();
}
```

```
public interface IOccupationalSafety {
   public void print();
}
```

1 point: In main(), a relevant code sequence must be implemented to highlight the implemented design pattern.

OBS:

1. The implemented patterns must respect the definition and structure discussed in the courses and laboratories. Variations or incomplete implementations are not accepted.
2. The pattern must be implemented correctly and completely (in its entirety) to be considered.
3. The solution does not contain compilation errors.
4. Patterns can be treated separately or can be implemented on the same set of classes.
5. Implementations that are not functionally related to the requirements in the subject will not be considered (copying an example from other sources will not be scored).
6. Modifying the classes received is *not* allowed.
7. Solutions will be cross-checked using MOSS. Sharing code between students is not allowed. Solutions with a similarity degree higher than 30% will be canceled.

Mandatory Clean Code Requirements (the solution is deducted by 2 points for each requirement not respected) - up to 8 points may be lost.

1. For naming classes, functions, attributes, and variables, the Java Mix CamelCase naming convention is followed as discussed.
2. Patterns and the class containing the main() method are defined in distinct packages with the form
   a. cts.firstname.lastname.groupNumber.pattern_name,
   b. cts.firstname.lastname.groupNumber.main (students in the additional year replace groupNumber with "as").
3. Classes and methods are implemented respecting the principles of KISS, DRY, and SOLID (pay attention to DIP).
4. Class names, method names, variables, as well as messages displayed in the console, must be related to the subject received (generic names are not accepted). Functionally, methods will display messages in the console to simulate the required action or will implement simple processing.