GDAŃSK UNIVERSITY OF TECHNOLOGY

Internet Services Architectures

Laboratory 1

Java SE

During laboratories students will develop an application which topic should be selected during first meeting. Each student should keep own work after each laboratory in order to be able to expand it during next one.

During first laboratory each student needs to select own topic for the application developed during laboratories. Alternatively topic can be assigned by the teacher. It can be the same for the whole group or individual for each student, decision is up to the teacher. Each topic should define two business classes connected with 1:N relationship. Each class should have at least two fields of different type (not counting relationships fields). Selected entity classes should be different than those used as a lecture examples. The best approach is to assume that the 1 relationship side is a category and the N relationship side represents elements belonging to that category. Example is provided at the end of this instruction.

During first laboratory students should revise already known Java SE topics. Project should be realized as Apache Maven project. If the teacher allows different JVM languages can be used. Moreover if the teacher allows Gradle instead Apache Maven can be used. Using external libraries not used in lecture examples should be consulted with the teacher. Project Lombok is always allowed. Above rules apply to all laboratories.

The following tasks should be completed:

1. Implementation of entity classes with appropriate comparison mechanisms (based on both hash and natural ordering) and text representation. Be careful about circuital dependencies. For element class additional implement DTO class with original fields but instead whole category use only one category field (the one best identifying it). For object creation builder pattern should be used. When some task requires to print object, the text representation should be used. (1 point)
2. At the application start-up, collection of categories filled with elements (remember about two way relationships) should be created. At this moment there is no need for user interaction. Objects should be created in code using appropriate creation methods. Then using nested for each lambda print all categories and elements in original order. (1 point)
3. Using single Stream API pipeline create `Set` collection all elements (from all categories). Then using second pipeline print it. (1 point)
4. Using single Stream API pipeline filter elements collection created earlier (by one selected property), then sort it (by one different property) and print it. (1 point)
5. Using single Stream API pipeline transform elements collection created earlier into steam of DTO objects, then sort them using natural order and collect them into `List` collection. Then using second pipeline print it. (1 point)

6. Using serialization mechanism store collection of categories in the binary file, then read it from it and print call categories with elements. (1 point)
7. Using Stream API parallel pipelines with custom thread pool execute some task on each category. For example task can be printing each collection elements with intervals using `Thread.sleep()` to simulate workload. Observer result with different custom pool sizes. For thread pool use `ForkJoinPool` Remember about closing the thread pool. (2 points)

Example data model – RPG game heroes representation (access modifiers, accessors, constructors and other methods omitted for the sake of simplification):

```
class Profession {
    String name;
    int baseArmor;
    List<Character>characters
}
```

```
class Character {
    String name;
    int level;
    Profession profession;
}

class CharacterDto {
    String name;
    int level;
    String profession;
}
```