POLITECHNIKA GDAŃSKA

Internet Services Architectures

Laboratory 1

Spring Framework & Spring Boot

During laboratories students will develop an application which subject 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 subject for the application developed during laboratories. Each subjects should define two business classes connected with 1:N relationship. Each class should have at least two fields of different type. Selected model classes should be different than those used as a lecture examples. The best approach is to assume that 1 relationship side is a category and N relationship side is some element belonging to category.

During first laboratory students should get familiar with basic components of Spring Framework (component, repository, service) as well as mechanism of dependency injection and inversion of control. Project should be realized as Maven Spring Boot project. Basic project can be generated using Spring initializer https://start.spring.io/.

There is no need for using persistence storage during first laboratory. All business classes objects can be stored in memory. Persistence storage will be introduced later.

The following tasks must be completed:

1. Implementation of single in-memory storage for business classes as Spring @Component. At this point it is enough to implement storage as component containing two collections exposing those collections instances or data streams to be used inside repositories. (1 point).
2. Implementation of repositories for each business class as Spring @Repository. Repositories should utilize in-memory storage instance provided by the container. Each repository should implement basic at least following operations: save new object, find object by primary key, find all objects, delete existing object. (2 point)
3. Implementation of service for each business class as Spring @Service. Each service should utilize repository instance provided by the container. At this moment each service should delegate repository methods. At this point this can look as services does not introduce any added value but this decomposition can be crucial in developing much more complex applications. (2 point)
4. Implementation of example data initializer launched automatically after start as Spring @Component. The initializer should utilize services instances provided by the container. (1 point)
5. Implementation of command line runner as Spring @Component. The runner should communicate with the user using standard input and output streams and allow for listing available commands, listing all categories, listing all elements,

adding new element (with category selection), delete existing element, stopping the application. The runner should utilize services instances provided by the container. (1 point)

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

```
class Profession {          class Character {

    String name;                String name;

    double moveSpeed;           Profession profession;

    int baseArmor;              int level;

}                           }
```