**Software Architecture and Design**

Laboratory work 6

CREATIONAL DESIGN PATTERNS. PROTOTYPE, SINGLETON AND FACTORY METHOD

Variant - 8

*Prepared by:*

*student of SE-224B,*

*Anna Kuts’*

*Accepted by:*

*Vasilieva M.D.*

**Task**

1. Learn creational design patterns. Know the general characteristics of

creational patterns and the purpose of each of them.

2. Study in detail the creational patterns for software design – Prototype,

Singleton and Factory Method.

For each of them:

• study the pattern, its purpose, motivation, cases when its use is appropriate,

and the results of such use;

• know the peculiarities of the implementation of the pattern, related templates,

known cases of its use in software applications;

• know the structure of the pattern, the assignment of its classes and the

relationships between them;

• be able to recognize a pattern in the UML class diagram and write the code

that implement the pattern.

3. Create the com.labwork6 software package. In the package, develop

interfaces and classes that implement tasks (according to the variant) using one or

more patterns (item 2). The methods that implement business logic should be closed

with stubs that output information about the called method and its arguments to the

console.

Example of business method implementation:

void draw(int х, int у)

{

System.out.println(“Method draw with parameters x=”+x+” y=”+y);

}

4. Complete documentation of the developed classes (also methods and fields)

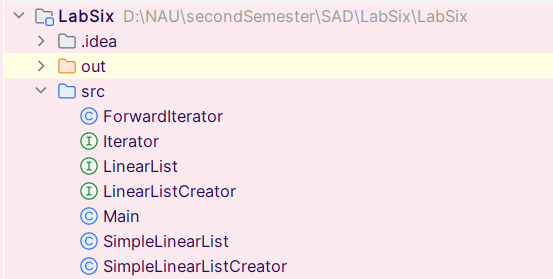
using Javadoc automated tools, while the documentation should sufficiently

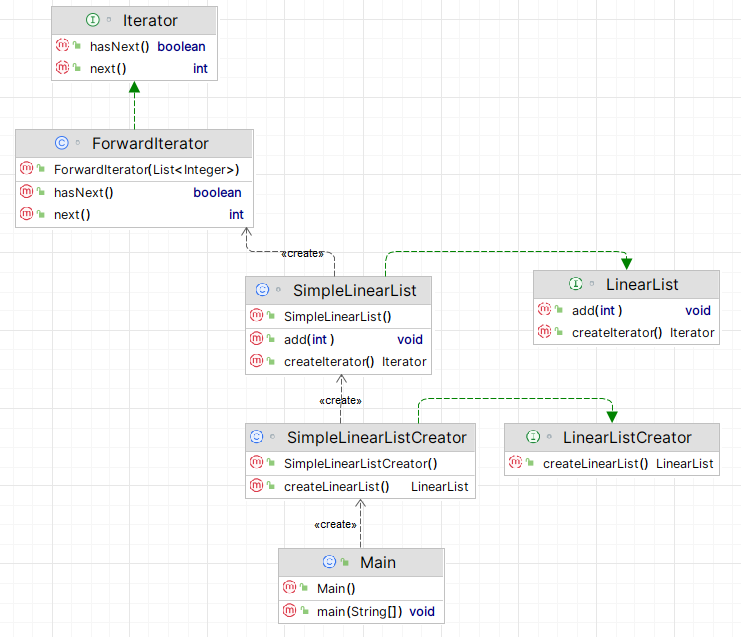
highlight the role of a certain class in the general structure of the pattern and the

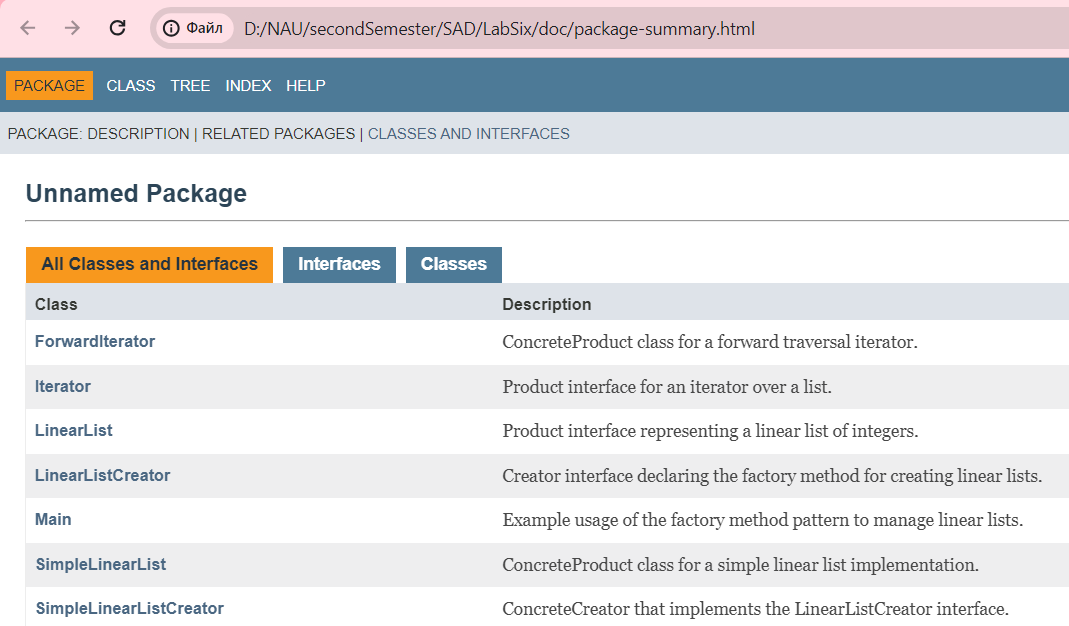
specifics of certain implementation.

|  |  |
| --- | --- |
| **№** | **Task variant** |
| 8 | Define class specifications that encapsulate a linear list of integers and a  forward traversal iterator in an ordered structure. |

**Solution**

****

****

****

<https://github.com/Jasokaa/LabSix>

**Conclusion**

Creational design patterns like Prototype, Singleton, and Factory Method address object creation mechanisms, enhancing flexibility and reuse of existing code. They help manage complexity in system architecture by controlling how objects are created, which is crucial for large-scale software development.