5. Basics of OOP

TASK 1. Create an object of the **Text File** class using classes of File and Directory. Use the following methods: create, rename, display content to the console, add, delete.

TASK 2. Create a **Payment** class with an inner class that can be used to create a purchase of several products.

TASK 3. Create a **Calendar** class with an inner class that can be used to store information about weekends and holidays.

TASK 4. Create a console application "**The Dragon's Treasure**". A program allows processing information about 100 treasures in a dragons cave. Implement the function of treasure viewing, selection of the most expensive treasure and selection of treasures for a given amount. The application should meet the following requirements:

• The application must be object-oriented, not procedural-oriented.
• Each class should have a meaningful name and informative composition.
• Inheritance should only be used when it makes sense.
• Java code convention must be used.
• Classes should be correctly decomposed into packages.
• The console menu should be minimalistic.
• Files can be used to store data.

TASK 5.

Variant A. Create an application that composes a **flower bouquet** (an object that represents a flower arrangement). **Flowers and a package** are the components of the bouquet.

Variant B. Create an application that composes **gifts,** namely **sweets and packaging.**

In your application you should meet the following requirements:

• Correctly design and implement the problem domain.
• Consider using generative design patterns to create objects from a class hierarchy.
• Implement user input validation, but not client-side.
• Regular expressions can be used to check the correctness of the transmitted data.
• The menu for selecting an action by the user can be omitted, use a stub.
• Special Condition: Overwrite the toString (), equals (), and hashCode () methods where necessary.