Write an application that will be used to manage the logistics of a seaport in terms of a transhipment terminal for container ships. The application will be used to unload and load containers from/to the ship. In case of unloading, the containers can be transferred to the warehouse or directly to the railway transport carriage.

Each ship has a different capacity and deadweight defined by:

- the maximum number of containers with toxic or explosive cargo that can be loaded within the ship
- the maximum number of heavy containers
- the maximum number of containers requiring connection to the electricity network
- the maximum number of all containers
- the maximum weight load

Each ship stores basic information about itself (unique identification number assigned automatically when creating the object, name, home port, transport origin and destination).

In the case of load capacity of containers for toxic and explosive cargo - this value is common for both types of containers. E.g. if the ship has three places for containers with explosive or toxic cargo, these places can be loaded respectively by:

- 3 containers with explosive cargo
- 2 containers with explosive cargo and 1 container with toxic cargo
- 1 containers with explosive cargo and 2 container with toxic cargo
- 3 container with toxic cargo

Each type of container has a different set of attributes (e.g. information about the sender, tare, information about the security of the container, net weight, gross weight, information about certificates, etc.). Also, each container has its own unique identification number assigned automatically when creating the object. For each type of container, you must additionally devise at least one unique attribute throughout all container types.

We have at our disposal, among others:

- standard container,
- heavy container, which is a type of standard container,
- a refrigerated container, which is a type of heavy container, requiring connection to the ship's electrical network,
- a container for liquids cargo, which is a type of standard container,
- a container for explosives cargo, which is a type of standard container,
- a container for toxic cargo, which is a type of heavy container, which is available in two versions: a container for toxic powdery cargo and a container for toxic liquid cargo, which is not only a kind of heavy container, but also has the features of a container for liquids cargo.

It should be remembered that when it comes to a ship, we are limited not only by the maximum number of containers, but also by the maximum weight capacity of the ship, therefore, before loading the next container, we must be sure that the next load will not exceed the permissible safe load capacity of the ship.

There is no need to create a specialized loading algorithm (e.g. will the ship start to heel due to uneven loading).

We also must have, among others the possibility of creating a new ship and containers of any type from the menu, and the possibility of loading the container onto the ship and unloading the container onto the railway transport carriage or to the warehouse.

When we unload a container onto a railway transport carriage, we assume that one train can accommodate 10 containers. After filling it up, wait 30 seconds for the departure of the current squad and the arrival of the next one (implement this based on threads, not *Timer* class).

Transhipment warehouse has a limited number of containers that it can hold, defined when creating an object of type ***Warehouse***. Any type of container may be stored within the warehouse, however, containers with explosive cargo can only be stored for 5 days, containers with toxic liquid cargo for 10 days, and containers with toxic powdery cargo for a maximum of 14 days.

Therefore, you should implement a mechanism for passing time using the thread mechanism (do not use the *Timer* class). The thread should move the date 1 day forward every 5 seconds, simulating the passage of time. At the same time, storage matters should be carried out and, in the event that the stored container exceeds the maximum storage time, it should be disposed of by taking it from the port area (and then forgotten from the point of view of the application).

Store all dates used in the program using ***LocalDate*** or ***LocalDateTime*** class.

When a container is disposed of, the sender receives a warning in the form of an exception object of the type ***IrresponsibleSenderWithDangerousGoods***. This warning contains information about the container (the date of arrival at the warehouse, the date of disposal and the unique identification number). When the sender receives two warnings, in the case of any other containers requiring storage, such containers will not be accepted and will be left on the container ship with an order to be returned to the sender.

The sender has more than the previously mentioned data, such as: name, surname, PESEL number, address, date of birth (date of birth is not stored specifically within the class field - we implement it in the form of a method that will display the date of birth as a *LocalDate* object on on the basis of the PESEL number).

For each ship, we must be able to unload and load the container, as well as departure of the ship from the port. Similarly, in the case of a warehouse, we must be able to at least store the container, take the container from the warehouse and load it on the indicated ship, check the current stock, display the contents of the warehouse and dispose of the indicated container.

Application durability should be ensured, i.e. it should be possible to save the current state of the entire seaport to a text file (including all information from the application, such as data of senders, ships, warehouse, containers and the complete set of information about them). Saving takes place after selecting the appropriate functionality from the menu. Information stored in the file should be written clearly and legibly for a human, in accordance with the following rules:

- Ship containers should be sorted in ascending order by weight.
- Container ships should be sorted in descending order of ship name
- Containers stored in the warehouse should be sorted in descending order according to the storage start date (from the longest stored to the shortest), and if the storage time is the same, the next sorting criterion is the sender's name.
- ***Do not use any type of serialization!***

The data written to the file must be read during the next application start to be able to start working from the last saved state. Multiple files can be used for this purpose.

You should handle any exceptions that may arise and properly notify the user of the following situation without interrupting the program.

Principle of program operation:

- In the ***main*** method, create a seaport with at least five different types of container ships loaded with different containers. Additionally, several containers of different types should be placed in the warehouse.
- After starting the application, the user should be able to call all of functionalities via the command console and the implemented menu.

The project is based on PPJ and GUI material.

Attention:

- ***In the case of receiving a project with significant deficiencies in implementation or a non-compiling solution, the result for such a project will be 0 points.***
- ***Lack of knowledge of any line of code or plagiarism will result in obtaining 0 points for this project with the possibility of failing the entire subject.***
- ***Not only the practical and substantive correctness of the solution will be assessed, but also the optimality, quality and readability of the code written by you.***
- ***An important part of the project is the use of: inheritance, collections, interfaces or abstract classes, lambda expressions, Java Generics, additional functionalities or structures and other characteristic elements presented in the classes and lecture.***