Object programing in Java

Assignment 1: Introduction into the Object-oriented Programming

Environment:

- IDE Eclipse Neon (https://eclipse.org/downloads/) **OR**
- Netbeans IDE 8.2 (https://netbeans.org/downloads/)
- Java 8 JDK

Instructions:

a) Within the first assignment you should write two classes: class **Person** and class **Ship**. The class Person is consisted of attributes **first name**, **family name**, and **day of birth**. The class **Ship** should be consisted of the following attributes **captain** (data type of Person!), **name**, **length**, **fuel consumption per hour**, and the **size of the tank**. For each class attribute choose, according to the values it can carry, the right data type.

b) Ensure that the written classes are properly encapsulated. For each attribute within the classes add associated getter and setter method.

c) Each of the written classes should have default constructor. Additionally, add a constructor with parameters first name and family name in the class Person and constructor with parameter name in the class Ship. Parameters, passed in a constructors, **SHOULD BE** named the same as the class attributes.

d) For the completion of the assignment write an additional class named **Launcher**. Within this class you should test proper functioning of the previously written classes **Person** and **Ship**. The class Launcher should allow users of the application to enter value for all class attributes and write theirs values on the standard output (console application). For reading the user input the class **java.util.Scanner** should be used. For more details about java.util.Scanner class look into the **Java 8 documentation**.

Number of points: 3

Objectives of the assignment: data types, class encapsulation, constructors, this reference, Java 8 documentation

Assignment 2: Object-oriented Programming and JDK Tools

Instructions:

a) Check if all of your classes are written in accordance with the rules of class encapsulation. All implemented classes equip with **toString** method, which should return values of all attributes in a class.

b) Add the third constructor to the classes **Person** and **Ship**. The third constructor should accept values of all attributes as parameters. You should avoid code duplication inside a body of constructors of your classes. To achieve this, use these constructor calls inside a class (use this keyword).

c) To the **class Ship** add three methods that calculate the amount of fuel consumed during a trip. The amount of consumed fuel should be calculated according to the following parameters:

- given **duration** of a trip (in hours);
- given **distance** and **average speed;**
- given time **difference between a departure and an arrival** of a ship.

Ship's fuel consumption calculation should be implemented using three methods that have the same name (parametric polymorphism). Use liters per hour as a unit of a ship's fuel consumption.

d) Introduce a new package named **si.um.opj.<your_surname>.logic**. All classes written previously should be placed in this package.

e) Source code in your classes equip with comments and annotations intended for automatic documentation generation. Further, use the JDK tool **javadoc** and manually generate documentation for all your classes. Additionally, demonstrate the usage of JDK tools **java** and **javac**.

f) Upgrade the class Launcher (written within the first assignment) with newly added concepts and functionalities in order to demonstrate their functioning.

Number of points: 4

Goals of the assignment: reference "this", class encapsulation, parametric polymorphism, toString method, constants, tools of JDK.

Assignment 3: Inheritance and Enum Types

Instructions:

a) Add abstract method **getOccupancy** to the class Ship. The method should return a percentage of capacity occupancy of a ship. Abstract method should be implemented in a subclasses of the class Ship.

b) From the class **Person**, written within the first assignment, derive the following subclasses:

- **Captain**, with additional attributes title and number of license;
- **Passenger**, with additional attributes email, preferred meal (you can choose between following options: **MEAT, FISH, VEGETARIAN, DIET**). Attribute preferred meal should be implemented as an **ENUM type**.

Equip newly added classes with constructors (including super class calls), get/set methods and properly overridden toString methods - the toString method of the super class should be called inside the toString method of the child class. From now on, the class Person should be abstract class.

c) From the class Ship, written within the first assignment, derive following subclasses:

- **Cruiser**, with additional attributes **number of cabins**, **number of passengers** on board and **maximum number of passengers** that can be boarded on a cruiser.
- **Cargo Ship**, with additional attributes maximum payload and payload loaded on ship;
- **Sailboat**, with additional attributes **surface of the sails**, **number of beds** onboard and **number of occupied beds** onboard.

Equip newly added classes with the constructors (including super class calls), get/set methods and properly overridden toString methods - toString method of the super class should be called inside the toString method of the child class

d) For each written subclass of the class Ship implement prescribed abstract method getOccupancy:

- **Cruiser** - percentage of occupancy of cabins onboard;
- **Cargo ship** - percentage of occupancy of payload that can be taken onboard;
- **Sailboat** - percentage of occupancy of beds on boat.

By your choice, classes can be supplemented with additional attributes.

e) Meaningfully place so far implemented classes into the following packages:

- si.um.opj.<your_surname>.logic
- si.um.opj.<your_surname>.logic.persons
- si.um.opj.<your_surname>.logic.ships

f) Upgrade the class Launcher, written within the second assignment, with newly added concepts and functionalities in order to demonstrate their functioning

Number of points: 4

Goals of the assignment: abstract classes, abstract methods, inheritance, enum types, packages, reference "super".

Assignment 4: Java Interfaces

Instructions:

a) Prepare the class **Cruise**. Individual cruise should include attributes such as **cruise title**, **cruise ship** with which the cruise is conducted, the **date of departure** and **duration of travel in days**.

b) The class Ship has to be changed in the following manner: add the list (array) of 4-passenger cabins on the ship. For the implementation use the array.

c) Enable the ticket sales for the chosen cruise. Cruise class should implement the **Salable interface**, which describes the following methods:

- void **sellTicket**(Cabin cabin, Passenger passenger), which reserves the chosen cabin for the selected passenger
- void **cancelTicket**(Cabin cabin, Passenger passenger), which releases the place in the cabin (removes the passenger from the cabin)
- boolean **cabinEmpty(**Cabin cabin), which checks if the cabin is empty (if there is at least one bed vacant - for example if in the cabin, there are 3 passengers, there is still one bed unoccupied)

OR

d) **Alternative 1:** If you will implement the assignment with the use of the **2-dimensional array**, the interface Salable describes the following methods: (The cabin has to be the first index in the 2 dimensional array):

- void **sellTicket**(int cabin, Passenger passenger) which reserves the chosen cabin for the selected passenger
- void c**ancelTicket**(int cabin, Passenger passenger), which releases the place in the cabin (removes the passenger from the cabin)
- boolean **cabinEmpty**(int cabin), which checks if the cabin is empty (if there is at least one bed vacant - for example if in the cabin, there are 3 passengers, there is still one bed unoccupied), where cabin presents the first index in the 2-dimensional field.

e) In the console application, demonstrate the newly added functionalities.

Number of points: 4

Goals of the assignment: You have to master the concepts such as: arrays, interfaces.

Assignment 5: Graphical User Interface

Instructions:

a) Prepare a graphical user interface used to operate a travel agency, which offers the cruises. The user interface has to be implemented with the use of the **javax.swing library**. Implement the following functionalities:

- **editing** (adding, editing, deleting) of the cruises
- **selling** the cruises to customers

Instances of the captains and ships can be statically implemented (they do not have to be specifically edited). To achieve this, use at least **2 different layout managers**. Graphical interface should be uniform (all views should be displayed within the same window).

In the application, you can implement the views (panels) with the help of the tools for visual building of the graphical interfaces. Next, you should manually insert the Implemented panels into the application - here you have to manually add the program code for the missing components (menus, status bars …). You have to use **CardLayout** layout manager.

You will implement the responses for the events in the next assignment.

Newly created classes should be placed into **si.unimb.opj.<your_surname>.ui** package.

Number of points: 4

Goals of the assignment: You have to master the preparation of graphical interfaces.

Assignment 6: Events

Instructions:

a) Using the responses to the events, you have create a **working graphical user interface** for the sales of cruse tickets, which was prepared in the previous assignment.

For the implementation of the user interface listeners, you **HAVE TO** use the following class types:

- outer class,
- inner class,
- anonymous inner class.

When implementing the graphical interface, you can also upgrade the existing classes and add the needed missing methods.

Number of points: 4

Goals of the assignment: After finishing the assignment, you will have to master the preparation of the graphical interfaces, the implementation of the response to the graphical interface events and the comprehension of the different types of classes: anonymous inner class, special class with the responses to the events, responses to the events in the class (which is graphical interface at the same time).

Assignment 7: Exceptions, Collections and Streams

Instructions:

b) Manually prepare the exception **CabinOccupiedException**, which is triggered when adding the passenger into an occupied cabin. Complete the methods for the calculation of the fuel consumption with the exception **java.lang.IllegalArgumentException**, which has to be triggered when the user fills the methods with the invalid data (for example - the negative value of the travel distance).

c) Where applicable, use the **collections from the java.util** (Vector, ArrayList, Set, Map) package. Collections should be of a parametrized type.

d) Enable the permanent storage of the entered data into the binary file with the use of the **serialization of the objects**. Make sure that the data is **compressed.** Decide if any of the attributes can be ignored when serializing. When starting the application, make sure that you refresh the flight data from the **binary file**.

e) Store the important events of the application and caught exceptions into the text file - enter the time of the event, the priority of the event (**CRITICAL, ERROR, WARNING, REMARK**) and the description.

Number of points: 4

Goals of the assignment: You have to master the work with the collections, exceptions, usage of the sorting mechanisms, working with files, serializations and the usage of the generics in the collections.

Assignment 8: Java Concurrency

Instructions:

a) You have to implement the **simulation of the regatta** (boat race). In the regatta, there can be any number of the boat involved. Their goal is to sail the specified distance in the shortest time possible. The progression of the boats in the regatta has to be dependent on the wind speed at the given time. The wind speed can range from **0-40 km/h** and has to be randomly changed every **100 - 2500 ms**. For the purpose of generating random numbers, you can use the class **java.util.Random**.

b) In the assignment, use the Sailboat class, which has been developed in the assignment **no. 3**. For the purpose of this assignment, use the class and change it into the Thread. The boat has to move independently until it does not finish the whole track of the regatta. You can specify the **formula for the movement of the boat on your own.** The assignment can be implemented as a console application.

c) To make sure that the simulation will not progress too quickly, the boat has to **wait every 500 ms**. The winner of the regatta is the boat, which finishes the whole track of the regatta first.

d) Make sure that every thread **will be finished successfully** after the end of the regatta.

Number of points: 3

Goals of the assignment: You have to master Java threads.