Java SE 8 Programming Language

Training Assignments

Document Code	25e-BM/HR/HDCV/FSOFT
Version	1.1
Effective Date	20/11/2012

Hanoi, 06/2019

RECORD OF CHANGES

No	Effective Date	Change Description	Reason	Reviewer	Approver
1	01/Oct/2018	Create a new Assignment	Create new	DieuNT1	VinhNV
2	19/Jun/2019	Update template	Fsoft template	DieuNT1	VinhNV

Contents

CODE:	JPL.L.L201
TYPE:	LONG
LOC:	N/A
DURATION:	180 MINUTES

Long Assignment 2 – Option 1: Java I/O Fudamentals

Objectives:

- Understand and practise with Advanced OOP in java.
- Understand and apply with Collection classes in java: List, Set, Map.
- Understand and practise with Exception handing, JavaIO.

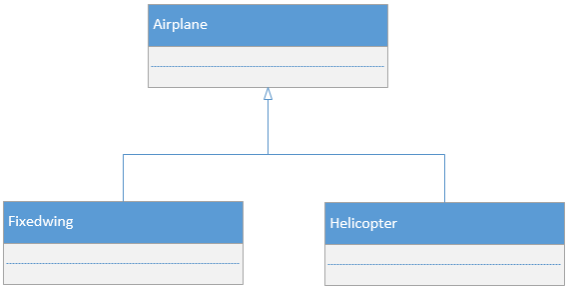
Working requirements:

- JDK 1.8
- Eclipse

Assignment Specifications:

Create a Java Console application bases on Java Classes/Objects, OOP, Exception Handing, IO, Java Collection to manage an Airport (Airport Management System) with **fixed wing airplane** and **helicopter.**

For the class hierarchy is as follows, the trainee let's create the java classes install this class diagram to be able to relationship between them.



- The fixedwings class contains the information about fixed wing airplanes. Each fixed wing airplane has its **ID**, **model**, **plane type**, **cruise speed**, **empty weight**, **max takeoff weight**, **min needed runway si**ze and **fly method** (“fixed wing”)
- The helicopters class contains the information about helicopters. Each helicopter has its, **ID model**, **cruise speed**, **empty weight**, **max takeoff weight**, **range** and **fly method** (“rotated wing”)
- The airports class contains the information about 4 airports. Each airport has an **ID**, a **name**, **runway size**, **max fixed wing parking place**, **list of fixed wing airplane ID**, **max rotated wing parking place**, **list of helicopter ID**.

Business Rules:

- ID is a string of 7 characters, started by “FW” for fixed wing airplane, “RW” for helicopter and “AP” for airport, followed by 5 digits. ID is unique.
- The model size is maximum 40 characters.
- Three fixed wing airplane type are CAG (Cargo), LGR (Long range) and PRV (Private).
- If a fixed wing airplane is parked in an airport, its min runway size does not excess the airport runway size.
- The max takeoff weight of helicopter does not excess 1.5 times of its empty weight.

Functional Requirements:

a. The program has the function to create a new airport. The name, runway size and capacity of the new airport are selected when a new airport is creating.

b. The program has the function to add to an airport one or more fixed wing airplane(s) which currently does not participate to an airport and which has the min needed runway size shorter than the airport runway size.

c. The program has the function to remove one or more helicopter(s) from an airport.

d. The program has the function to add to an airport one or more helicopter(s) which currently does not participate to an airport.

e. The program has the function to remove one or more helicopter(s) from an airport having.

f. The program has the function to change plane type and min needed runway size of fixed wing airplane.

Storage Data:

» The user inputs data from the keyboard.

» Data is stored in the a text file.

» Output data is displayed on the console.

User Interface Requirements:

Create a new class named **AirplaneManagement** that contains a main() method to display user interface.

The main screen allows selecting the functions for:

» Input data from keyborad

» Airport management

- Display list of all airport information, sorted by airport ID
- Display the status of one airport, selected by airport ID

» Fixed wing airplane management

- Display list of all fixed wing airplanes with its parking airport ID and name

» Helicopter management group

- Display list of all helicopters with its parking airport ID and name

» Close program

Each unsuccessful action will be informed to user by an error message.

Guildelines:

- Create a project named **JPL.L.A201**, create package ***fa.training.entities*** that contains classes/interfaces: **Airplane**, **Fixedwing**, **Helicopter**.
- Create other package ***fa.training.services*** that contains classes to implement Functional Requirements. The package ***fa.training.utils*** to implement Validation Rules requirements.

  Note, the functional requirements related to any entity, you have to create the service class corresponding to that entity. Ex: FixedwingService, …
- Create a package named ***fa.training.main*** contains **EmployeeManagement** class.

-- THE END --