LAB 211 Assignment

Type:	**Long Assignment**
Code:	**J1.L.P0011**
LOC:	**500**
Slot(s):	**N/A**

Title

Covid-19 Vaccine Management

Background

You are required to develop a program to manage the vaccination process of FPT university students. The vaccination management program must allow administrators to manage student vaccination information. In addition, the program also provides a function for students to view information about the vaccine that they have been vaccinated. To simplify the application building, we give the following business roles:

1. Each student can only inject a **maximum of 2 injections of the vaccine**.
2. The second dose of vaccine **must be given 4 to 12 weeks after the first injection**
3. Two vaccines **must be of the same type**.

Injection information is stored in the **injection.dat file**

The **student.dat file** has stored student information **including studentID and name fields.**

The **vaccine.dat file** has stored vaccine information **including vaccineID and name fields.**

Program Specifications

Build a management program for manager (doctor). With the following basic functions

0. Build your data structure
1. Show information all students have been injected
2. Add student's vaccine injection information
3. Updating information of students' vaccine injection
4. Delete student vaccine injection information
5. Search for injection information by studentID
6. Others- Quit

Each menu choice should invoke an appropriate function to perform the selected menu item. Your program must display the menu after each task and wait for the user to select another option until the user chooses to quit the program.

Each injection information has the properties such that injection id, 1st injection place, 2nd injection place, 1st injection date, 2nd injection date, student ID and vaccine ID. **The lecturer provides an student.dat and vaccine.dat file that contains information about 10 records for each.**

Features:

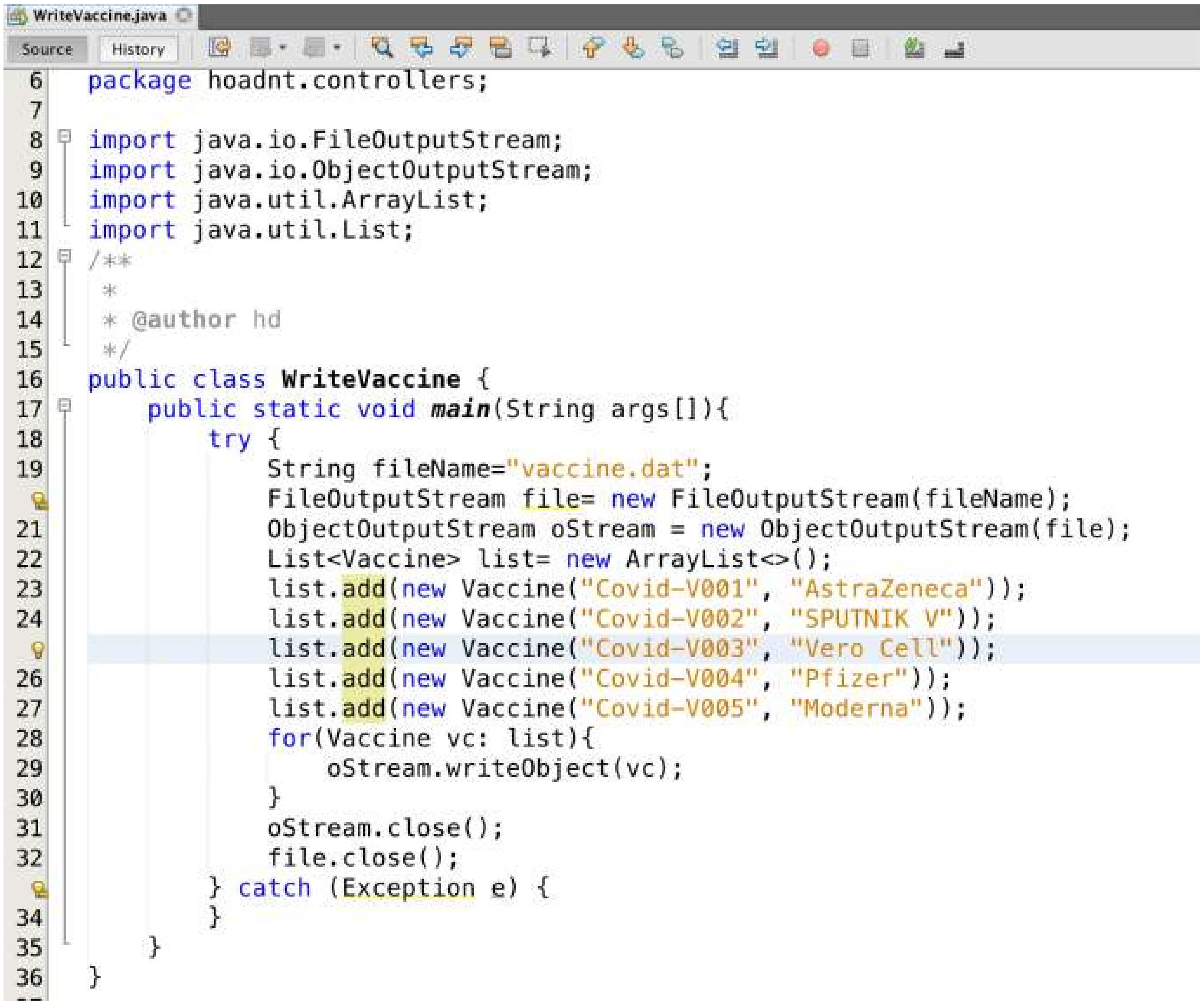
This system contains the following functions:

Display a menu and ask users to select an option.

- **Function 0: Build the data structure – 100 LOC**
  - Classes, abstract classes, Interfaces.
  - **The injectionID, studentID, and vaccineID fields cannot be changed after created**.
  - Must implement the polymorphism properties of object-oriented programming.
- **Function 1: Show information the injection information – 50 LOC**
  - Show all data in the **injection.dat** file into the screen.
- **Function 2: Add new injection – 100 LOC**
  - Create a submenu that allows the doctor add new injection information.

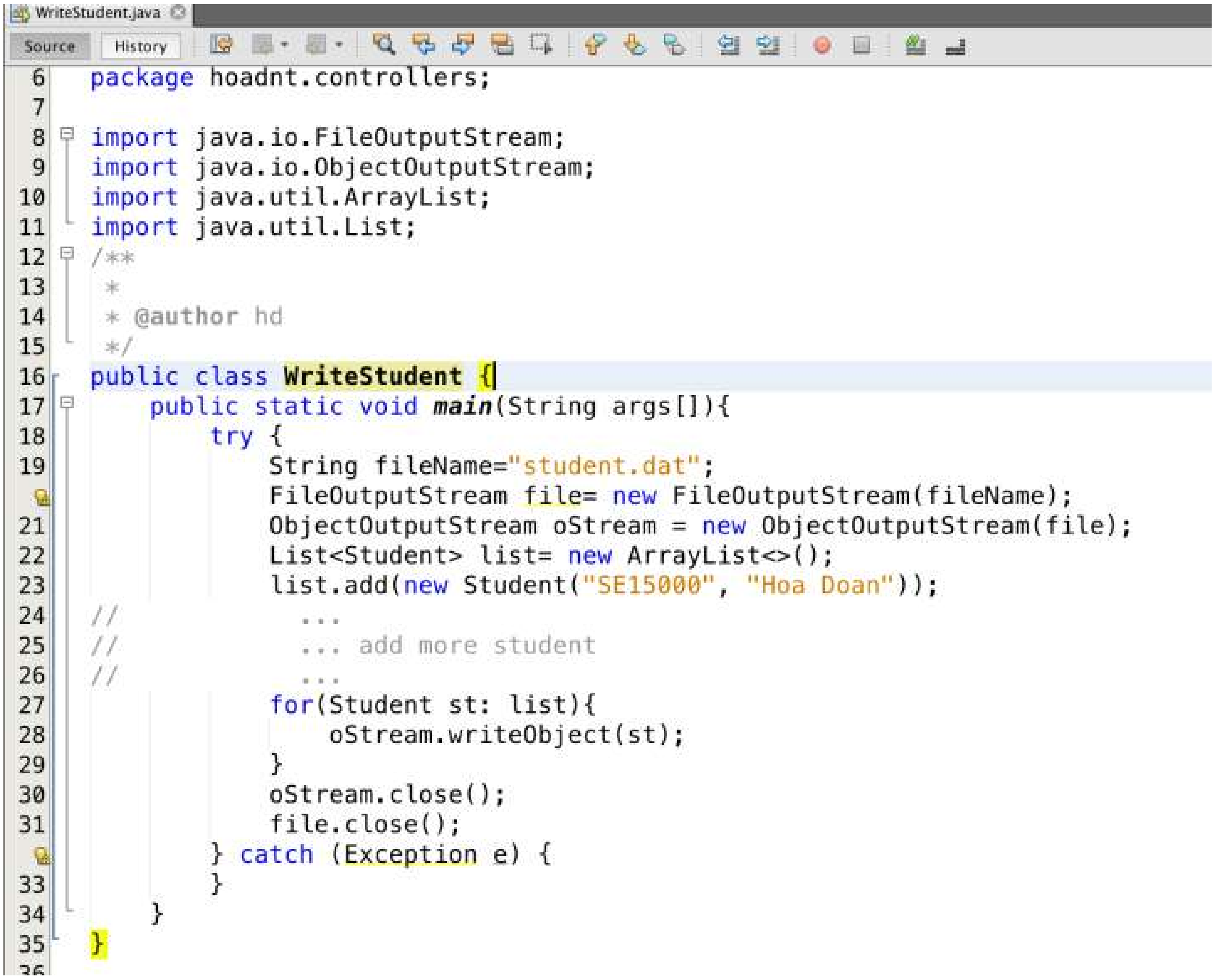
  - **Remember that the constraints must be checked: the injectionID's value cannot duplicate, student and vaccine must be existed and all business role above.**
  - Add the new injection to collection. If any information is left blank, it means that the information does not need to be input at the present time (2nd information).
  - Ask to continuous create new injection or go back to the main menu.
- **Function 3: Update Injection information – 100 LOC**
  - User is required enter the injection id
  - If injection does not exist, the notification "Injection does not exist". Otherwise, doctor can start update 2nd information only (place, place).
  - **Remember that new information must be validated.**
  - Then, system must print out the result of the updating. If the student has fully injected two injections, the system will display the message "Student has completed 2 injections!"
  - After updating, the program returns to the main screen.
- **Function 4: Delete Injection – 50 LOC**
  - Doctor can delete the Injection in the list.
  - Before the delete system **must** show confirm message.
  - Show the result of the delete: success or fail.
  - After delete, the program returns to the main screen
- **Function 5: Search injection by studentID – 50 LOC**
  - The doctor enters search text: studentID
  - The system searches the list, and returns the injection information of the input student.
  - Show result list: injection information.
- **Function 6: Store data to file – 50 LOC**
  - Store data in collection to injection.dat file.

- The above specifications are only basic information; you must perform a requirements analysis step and build the application according to real requirements.
- The lecturer will explain the requirement only once on the first slot of the assignment.



```java
package hoadnt.controllers;

import java.io.FileOutputStream;
import java.io.ObjectOutputStream;
import java.util.ArrayList;
import java.util.List;
/**
 *
 * @author hd
 */
public class WriteVaccine {
    public static void main(String args[]){
        try {
            String fileName="vaccine.dat";
            FileOutputStream file= new FileOutputStream(fileName);
            ObjectOutputStream oStream = new ObjectOutputStream(file);
            List<Vaccine> list= new ArrayList<>();
            list.add(new Vaccine("Covid-V001", "AstraZeneca"));
            list.add(new Vaccine("Covid-V002", "SPUTNIK V"));
            list.add(new Vaccine("Covid-V003", "Vero Cell"));
            list.add(new Vaccine("Covid-V004", "Pfizer"));
            list.add(new Vaccine("Covid-V005", "Moderna"));
            for(Vaccine vc: list){
                oStream.writeObject(vc);
            }
            oStream.close();
            file.close();
        } catch (Exception e) {
        }
    }
}
```



```java
package hoadnt.controllers;

import java.io.FileOutputStream;
import java.io.ObjectOutputStream;
import java.util.ArrayList;
import java.util.List;
/**
 *
 * @author hd
 */
public class WriteStudent {
    public static void main(String args[]){
        try {
            String fileName="student.dat";
            FileOutputStream file= new FileOutputStream(fileName);
            ObjectOutputStream oStream = new ObjectOutputStream(file);
            List<Student> list= new ArrayList<>();
            list.add(new Student("SE15000", "Hoa Doan"));
//                ...
//                ... add more student
//                ...
            for(Student st: list){
                oStream.writeObject(st);
            }
            oStream.close();
            file.close();
        } catch (Exception e) {
        }
    }
}
```