ITSC 315 – Security for Developers Assignment 1

Java Secure Coding Guidelines

Introduction

You are being given a pre-written application called **`ManageComputers`**. This application allows the user to manage a list of computers of two types: Desktop and Laptop. When run the application prompts the user to enter a menu selection for what operation s/he wants to perform in the system. The user can Load, Save, List, Add, Delete, and Edit computer data via the menu, and exit from the program.

The **`main()`** method for the application is contained in a file called **`ManageComputers.java`**, and the remaining classes are contained in a package called **`domain`**. The **`domain`** classes are:

- **`BaseComputer`**:
  - Holds as attributes: the type of the computer (desktop or laptop); CPU type; RAM size; and disk size.
- **`LaptopComputer`**:
  - Inherits from **`BaseComputer`**, and adds a screen size attribute.
- **`DesktopComputer`**:
  - Inherits from **`BaseComputer`**, and adds a GPU (video card) attribute.

The domain objects for the desktop and laptop computers are held in a single **`ArrayList`** declared and managed in the **`ManageComputers.java`** file. The objects in the **`ArrayList`** can be saved (serialized) into files on a drive. The serialized files can also be loaded (deserialized) back into the **`ArrayList`** by the program.

Saving computer data is performed by deleting any and all existing serialized files on the drive first, and then looping through the **`ArrayList`** of computers in memory and writing out new files (one per computer in the **`ArrayList`**).

Loading computer data is performed by first clearing the **`ArrayList`** in memory, and then deserializing any serialized files on the drive back into the **`ArrayList`**.

Assignment Requirements

The application as written does not attempt to follow any of the ***Java Secure Coding Guidelines*** (as covered in class over the last few weeks). You must ***refactor***[1] the code for the application so that it does the following:

- Uses ***composition*** instead of inheritance to reuse existing functionality between the **`BaseComputer`** and the **`LaptopComputer`** and **`DesktopComputer`** classes.
- Disallows any attempt at ***subclassing*** of any of the domain classes by other classes.
- ***Instantiation*** of domain objects must only be possible by calling a **`static getInstance()`** method in the appropriate class. It must ***not*** be possible to instantiate domain objects using the ***new*** keyword.
- Uses only ***immutable*** domain objects.
- Uses ***input validation*** to ensure that only valid input data is accepted into the application.
- All String data being stored in ***serialized data files*** must be ***encrypted*** and ***decrypted*** using a ***Caesar cipher*** that encodes string data by moving all of the letters in a string three places to the right in the alphabet when encoding data, and left by three letters to decode it, as required. String data must not be easily readable in the serialized data files.
- Only allows the application to run when under the supervision of a Java **`SecurityManager`** and ***security policy*** file (called **`security.policy`**). The policy file must contain permissions allowing the application to create, read, write and delete files in the application's data directory (but nowhere else). Trying to run the application without a **`SecurityManager`** should cause the application to output the message "***No `SecurityManager` available, application stopping!!***" and it should quit immediately.
- The domain classes must be defined and loaded by the application from ***a single sealed .JAR file called Assign1DDomain.jar*** when the application is run.

Additional requirements:

- **Submissions must *NOT* consist of IDE-based projects**. Only a version of the application that can be run from the command line will be accepted (see the "***Log Operations***" section below for details).
- When changing the application you ***must refactor the provided application code***, you ***cannot rewrite large pieces of the existing application, or the entire application from scratch***.

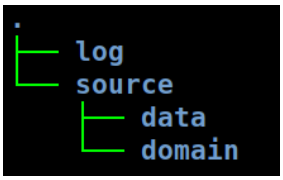
Submission requirements:

You must submit the following in a file called **`assignment1D.zip`** (***not .rar***, etc., only **.zip** files please!):

1 "***refactor***" means *changing existing code*, without changing its functionality. Not writing a lot of new code.

- The refactored source code files, including:
  - **`ManagerComputers.java`**, **`BaseComputer.java`**, **`DesktopComputer.java`** *and* **`LaptopComputer.java`**.
  - A file called **`Assign1DDomain.jar`** that contains a prebuilt **sealed** version of the domain package classes.
  - **`MANIFEST.MF`** file used to create the sealed domain package file.
  - ***Security policy*** file used by a **`SecurityManager`** when program is run.
- An application run log file called **`logfle.txt`** containing terminal output created by the refactored version of the application. See the section below called "***Log operations***" for information on what this file must contain.

The directory structure for the above files **must** be as follows:



- ***log*** directory: copy of your **`logfle.txt`**.
- ***source*** directory: **`ManageComputers.java`** and **`.class`** files, and **`MANIFEST.MF`** file. Also, a copy of your sealed **`Assign1DDomain.jar`** file containing the **`domain`** package. Your **`security.policy`** file should be included here as well,
- ***source/domain*** directory: **`BaseComputer.java`**, **`LaptopComputer.java`** and **`DesktopComputer.java`** files, and associated **`.class`** files.
- ***source/data*** directory: copies of a few serialized computer data files created by your refactored application.

Log Operations

After you have finished refactoring the application you must run it in a Terminal in the course VM using the following command line (run this in the directory where you have located your application **`ManageComputer.class`** file and **`Assign1DDomain.jar`** file):

```
java -cp .:./Assign1Domain.jar -Djava.security.manager -Djava.security.policy=./security.policy ManageComputers
```

Then perform the following actions:

1. List the computers, to show that initially the program contains no data in **`ArrayList`**.
2. Load computer data from the drive into the program.
3. List the computers to show the newly-loaded data on screen.
4. Add a new Laptop computer using invalid values, e.g. an i6 CPU, 32GB of RAM, 120GB drive size and a 24 inch screen size. The application should show a message rejecting the invalid values, and the invalid computer should not be added to the application.
5. List the computers again, to show that the invalid computer has not been added.
6. Add a valid desktop computer using the values i7 CPU, 16GB of RAM, 250GB hard drive and an AMD video card/GPU.
7. List the computers to show the new computer has been added to the application.
8. Save the computer data to the drive.
9. Exit from the program.
10. Try running the program without the Security Manager using the command line "**`java -cp .:./Assign1Domain.jar ManageComputers`**" and show the program refusing to run without it.
11. Use the following command in the Terminal to display the contents of the first serialized data file: ***`cat ./data/1D.txt`***.

Copy the text in the terminal that was output for the operations outlined above and paste it into your **`logfle.txt`**. That is the file you will submit as part of your Assignment 1 deliverables.

Suggested Approach for this Assignment

1. Start by compiling and running the existing application and understand its operations.
2. *Carefully* read the application code and make sure you understand it.
3. Only then continue on to refactor the code according to the requirements outlined above.

When refactoring existing code it is very important to understand what the existing code is doing before making changes, hence the advice above.