

## QAP 4 – Advanced Java: Persisting Data with a Database and Writing to a Text File (SOLO ONLY)

### Software Developer Course Assessment

Quantitative Assessment Practice - 4

**Course Name:** Advanced Programming (Java)

**Current Week:** 20<sup>th</sup> November 2025

**Submission Deadline:** 28<sup>th</sup> November 2025

#### Introduction:

This assessment is designed to gauge your understanding of the review material covered in recent weeks. Its primary goal is to improve our teaching strategies by identifying areas that may require additional attention.

Participation in this assessment is **mandatory**. You must complete **at least 80% of the assigned QAPs** for each course. Failure to do so will result in your course being marked as incomplete, regardless of your other grades.

If you **submit a solution**, it will be marked as **complete**, and you will receive full participation marks—**regardless of correctness**. We encourage you to submit your best attempt, even if incomplete. **Do not copy** others' work; the purpose is to identify learning gaps honestly.

❖ Final Submission Date: **25<sup>th</sup> July 2025 (End of Day)**

❖ Ensure all questions in the **Instructions** section are answered in your submission.

#### Grading Criteria:

**QAPs are marked on a 1 to 5 scale as follows:**

Grade	Description
1	Incomplete: Severe lack of understanding; the solution is non-functional or unrelated.
2	Partially Complete: Shows basic understanding; partial or buggy implementation.
3	Mostly Complete: Demonstrates good grasp of core ideas; mostly functional with minor bugs.
4	Complete (Pass): Functional, all requirements met; minor, non-critical bugs only.
5	Complete with Distinction (Pass Outstanding): Exceeds expectations; handles edge cases, shows mastery.

#### Instructions (To be included in your submission):

Please answer the following:

1. How many hours did it take you to complete this assessment?  
*(Please estimate per problem if possible)*

2. What online resources did you use?  
*(e.g., Lectures, YouTube, Stack Overflow, etc.)*
3. Did you get help from any classmates?  
*(If yes, provide names; they must be enrolled in your class)*
4. Did you ask for help from an instructor?  
*(Mention the number of questions/help sessions)*
5. Rate the difficulty of each problem and your confidence in solving similar problems in the future.

## Purpose

The purpose of this project is to help you demonstrate your ability to **persist and read data** using two widely used Java techniques:

1. Writing Java objects to a **file** using file I/O
2. Writing Java objects to a **PostgreSQL database** using JDBC

By completing this project, you will gain experience with both **file-based** and **database-backed** persistence, an essential skill for any software developer

## What You Will Build

You will create a simple Java program **from scratch** that:

- Defines 2 **custom entity** classes
- Has the functionality to persist data in a database and in a text file
- Has the functionality to read data from both a database and a text file
- Provides a **menu-based interface** for the user which allows them to select options to:
  - Save data to a **text file**
  - Save data to a **PostgreSQL database**
  - Read data from the file (show all drugs)
  - Read data from the database (show all patients)

### ► Entity Classes

- Drug (drugId, drugName, drugCost, dosage)
- Patient (patientId, patientFirstName, patientLastName, patientDOB)

### ► File I/O

Choose one of the entities and write the data to a text file

Use Java's file handling classes to:

- Write/ save data to a file
- Read the data from the file and be able to print the data using the menu option, just a print showing all the data for the one text file is perfect

## ► Database I/O (PostgreSQL)

- ★ Use the other entity and be able to write the data to a database
- ★ Add the PG driver to your project so you can connect to a PG database or use something like Maven to handle Java dependencies
- ★ The easiest way to get this to work is to create the PGSQl table in PGADMIN outside of your IDE. Then JDBC will just make the connection and read from the DB

Use **JDBC** to:

- Connect to a **PostgreSQL** database
- Insert data into the database
- Retrieve and display data from the database

## Scanner Menu

Use a scanner to create a menu that

- Provides the user with 4 options
  - Save data to a file
  - Read the data from the file
  - Save Data to the database
  - Read data from the database

## Submission Checklist

Make sure your submission includes **all** the following:

- A single **GitHub repository** link (submitted in Teams)
- A **video or screenshots** showing:
  - Data being saved to the **file**
  - Data being saved to the **database**
  - Data being read from both the file and the database
- A **.sql** file containing your table creation script
- A **README.md** file in your repository with:
  - A short project description, a short summary of how it went for you
  - Screenshots can be put in the readme