50002870006 - Java Enterprise Application Development

Academic Year 2025-2026 Semester 1

Programming Examination

Time Slot: 1:30-4:15pm, 22 October 2025

Venue: Lab 430, Jishi Building, Tongji University

Notice:

1. Compress all source code folders and files into one ZIP file, and submit it via the *Canvas* system. No IDE-related file should be submitted. No binary file (such as .class/.jar file) or IDE-related file should be submitted.

2. When it reaches 4:15pm, you must immediately stop your programming work and initiate the submission process. You will be given 15 minutes for uploading your code and solving technical issues encountered. The *Canvas* submission portal will be automatically closed at 4:30pm.

3. When solving the problems, please also pay attention to the design, readability and maintainability of your source code.

4. Please write your matriculation number and full name (as source code comments) in the beginning of each source code file submitted.

5. You may read technical documents and search for information over the Internet during the examination. However, it is STRICTLY NOT ALLOWED to use any communication service (such as sending/receiving emails, asking for help in a forum, using WeChat/QQ), and STRICTLY NOT ALLOWED to use any AIGC service (such as ChatGPT) regardless of the form.

6. During grading, JDK 21 will be used for testing your code.

GarfieldTask: A Command-Line Tool for Task Management

(100 Marks)

Garfield, a lazy but clever cat, finds it hard to stay organized and productive when managing his daily routines—especially when balancing his meals, naps, and occasional study sessions. To help him overcome procrastination, he needs a simple but practical command-line productivity tool called **GarfieldTask**, which can manage his to-do items and assist him with focus sessions through the Pomodoro technique.

Your mission is to design and implement this system in **Java**, following **object-oriented programming principles and techniques** (such as encapsulation, inheritance and polymorphism). The final program should be a **command-line application built with *Picocli***.

Detailed Tasks

1. Object-Oriented Design and Implementation (40 Marks)

Requirements:

- **Task Definition**
  - Create an abstract superclass `Task` that contains: `id`, `title`, `createdAt`, `status` (enum: `OPEN`/`DONE`), `priority` (enum: `LOW`/`MEDIUM`/`HIGH`) and any attribute that might be used.
  - Define an abstract method `describe()` that each subclass must override to present its details.
  - Implement at least two subclasses:
    - `SimpleTask`: a basic to-do item.
    - `DeadlineTask`: a task with a `deadline` (using `java.time.LocalDateTime`), which supports the display of the time remaining or "*Overdue*" if passed.
- **Repository and Data Management**
  - Design a generic interface `GarfieldTaskRepository`. It should contain standard CRUD methods like `save`, `findById`, `findAll`, and `deleteById`, as well as other methods necessary for supporting the application's features.
  - At a minimum, the interface should contain:
    - `void save(Task task);`
    - `Task findById(String id) throws TaskException;`
    - `List<Task> findAll();`
    - `void deleteById(String id) throws TaskException;`
    - `List<Task> findByStatus(Task.Status status);`
    - `void update(Task task) throws TaskException;`
  - Provide a concrete implementation `InMemoryGarfieldTaskRepository` by utilizing `Map<String, Task>`.
- **Exception Handling**
  - Handle invalid situations (such as empty title, duplicate id, missing record) with a custom exception named `TaskException`.

2. Pomodoro Timer for Time Management (20 Marks)

Requirements:

- Implement a class named `PomodoroTimer` that encapsulates the timer logic.
- This class contains a method `startTimer(int workSeconds, int breakSeconds, int cycles)` that utilizes `java.util.concurrent.ScheduledExecutorService` or `java.util.TimerTask` to run the timer. (Note: we use **seconds** rather than **minutes** to facilitate easy testing in this exam).
- To implement the logic for a single work or break session, you will need to create one or more objects that implement the `java.lang.Runnable` interface. These `Runnable` tasks are what you will submit to the scheduler (e.g., using `executor.schedule(...)`) or encapsulate within a `TimerTask`.
- The above method should run on a separated thread so that it does not block the main application loop.
- The timer should display clear start/end messages for work and break sessions, and optionally, a countdown progress.
- **Important Note on Asynchronous Output:** The timer's output will be printed to the console asynchronously. It is **acceptable and expected** that the timer's messages (e.g. "Break time") may appear while the user is typing a new command in the prompt. You are not required to build a complex UI to synchronize the prompt and the timer output.

3. Data Export (20 Marks)

Requirements:

- Implement a class named `TaskExporter` that handles serializing and saving tasks.
- This class should have a method `exportTasks(List<Task> tasks, String filename)` that takes the current list of tasks and a filename.
- Inside this method, use a library like **Jackson** or **Gson** to serialize the list into a well-structured JSON document.
- Write the resulting JSON string to the specified file. Handle any potential `IOException`.

4. Command-Line User Interface with Picocli (20 Marks)

Requirements:

- Utilize **Picocli** to implement the `main` method to launch an **interactive shell** (a REPL - Read-Evaluate-Print Loop) that allows the end-user to run commands continuously.
- A **simple demo project (as a ZIP file)** has been provided for use. It contains the necessary `pom.xml` and the main application REPL structure. Please build your application upon this demo project (i.e., simply start your coding by modifying the source code of the demo project provided).
- The shell should instantiate one instance of your `InMemoryGarfieldTaskRepository` and other services (such as timer and exporter) and keep them alive for the entire session.
- The shell must read user inputs (i.e. utilizing `Scanner`) in a loop.

- Use **Picocli** to parse the user's input string inside the loop to execute the desired command.
- The shell must integrate all previously implemented components by injecting services into the **command classes**.
- The main command (`GarfieldTask`) must support `help` and `version`.

Required Commands (to be typed into the shell):

1. `add`: Add a new task. The command should create a `DeadlineTask` if the `--deadline` option is provided; otherwise, it creates a `SimpleTask`.

   ```
   garfield> add --title "Review design patterns" --priority HIGH
   garfield> add --title "Submit lab report" --deadline "2025-10-14 23:00"
   ```

2. `list`: Display the current tasks. It should support filtering by status or priority.

   ```
   garfield> list --status OPEN
   ```

3. `done`: Mark a task as completed.

   ```
   garfield> done --id T-1001
   ```

4. `remove`: Delete a task.

   ```
   garfield> remove --id T-1001
   ```

5. `timer`: Start the Pomodoro timer following the logic from Section 2 above.

   ```
   garfield> timer --work 25 --break 5 --cycles 2
   ```

6. `export`: Save the current session's tasks to a JSON document following the logic from Section 3.

   ```
   garfield> export --file my-tasks.json
   ```

7. `exit`: Exits the application.

   ```
   garfield> exit
   Goodbye!
   ```

Important Rules

- **Libraries Allowed for Use in this Exam:** Picocli, Jackson or Gson (optional for JSON), and all standard JDK libraries.

Submission Requirements

Each candidate must submit:

1. **Source code** (including `pom.xml` or `build.gradle`).
2. **README.md** with necessary information, including your matriculation number and full name as well as build commands and running examples.

References

- **Picocli:** `https://picocli.info`
- **Apache Maven Guide:** `https://maven.apache.org/guides/index.html`
- **Java Concurrency Tutorial:** `https://docs.oracle.com/javase/tutorial/essential/concurrency`
- **java.time API:** `https://docs.oracle.com/javase/tutorial/datetime/index.html`
- **Jackson (JSON Library):** `https://github.com/FasterXML/jackson`
- **Gson (JSON Library):** `https://github.com/google/gson` (See User Guide: `https://google.github.io/gson/UserGuide.html`)

The End