Mohamed Khider University - Biskra
Department of Computer Science
Module: Software Engineering (Génie Logiciel)
Year: 3rd Year License

13-10-2024

Practical Work No. 2

You should respect the Java Naming Conventions in these exercises:

Exercise 1 :

The goal is to develop a library management system using object-oriented programming, interfaces, and abstract classes in Java.

Part 1: Basic Classes

- **Create a Book class:**
  - **Attributes:** title, author, publicationYear, available (boolean), stockQuantity.
  - **Methods:**
    - **`borrow()`** set available to false and decrement stock quantity,
    - **`return()`** set available to true and increment stock quantity),
    - **`toString()`** method to return a string with the book details.
- **Create a CD class:**
  - **Attributes:** title, artist, publicationYear, available (boolean), stockQuantity.
  - **Methods:** **`borrow()`** (set available to false and decrement stock quantity), `return()` (set available to true and increment stock quantity), and **`toString()`** method to return a string with the CD details.

Part 2: Abstract Classes and Interfaces

- **Create an Empruntable interface:** With the operations `borrow()` and `return()`.
- **Modify the Book class** to implement Empruntable.
- **create a class** called `**Person**` with the following private instance variables: `firstName`, `secondName`, `age`, `gender`, and `country`.

- **Create an abstract class AbstractPerson:** With the abstract methods `borrowBook(Empruntable book)` and `returnBook(Empruntable book)`, and with the following private attributes: `firstName`, `secondName`, `age`, `gender`, and `country`
- **Add a new Student class:** This class should also extend **AbstractPerson**.
- Implement the `borrowBook` and `returnBook` methods to reflect the specific behavior of a student (for example, a limit on the number of books borrowed).
- **Add a new Teacher class:** This class should also extend **AbstractPerson**.
- Implement the `borrowBook` and `returnBook` methods to reflect the specific behavior of a teacher (for example, no limit on the number of books borrowed).

Part 3: Library Management

- **Create a Library class:** Attributes: a list of Empruntable items (books and CDs) and a list of persons registered with the library.
- **Methods:** `addEmpruntable(Empruntable item)`, `addPerson(AbstractPerson person)`, `toString()`, `displayAvailableItems()`, `displayRegisteredPeople()`.
- Add the following additional functionalities to the library:
  - Managing overdue returns with a fine.
  - Managing book reservations.
  - Displaying the top 10 most borrowed items.

With the Student and Teacher classes, you can now further diversify the behavior of people who borrow books from the library, allowing students to borrow a limited number of books, while teachers have no borrowing limit.

Exercise 2 : Project: Task Manager (To-Do List)

Part 1 Project Description: The project involves developing a task manager (to-do list) with a graphical user interface using Java and a UI library such as Swing, SWT, or JavaFX. The task manager should allow users to create, view, edit, and delete tasks. It should also enable users to mark tasks as completed or not. For this project, students will need to use abstract classes and interfaces to organize their code.

Project Requirements:

1. **Task Interface:** Create an interface `Task` that defines the necessary methods for managing a task. Methods may include creating, editing, deleting, and marking a task as completed.
2. **Abstract TaskList Class:** Create an abstract class `TaskList` that implements basic methods for managing a list of tasks. This abstract class can include methods for adding, deleting, editing, and displaying tasks.
3. **TaskImpl Class:** Implement a concrete class `TaskImpl` that implements the `Task` interface. This class represents a specific task with attributes such as name, description, due date, etc.
4. **TaskListImpl Class:** Create a concrete class `TaskListImpl` that extends the abstract `TaskList` class. This class should manage a list of tasks using appropriate data structures (such as a list or array).
5. **Application GUI Interface (TaskManagerGUI class):** Use Swing, SWT, or JavaFX to create a user-friendly graphical interface that allows users to create, view, edit, and delete tasks. The task list should be displayed in a dedicated area of the graphical interface.
6. **Additional Options:** For a more options, students should add functionalities such as:

    - Filtering tasks by status (completed/not completed, abandoned),
    - Sorting tasks by due date,
    - Search Functionality: Implement a search feature that allows users to search for tasks based on keywords in their names or descriptions.
    - Task Comments: Allow users to add comments or notes to tasks for additional context or reminders.
    - Exporting/importing the task list to/from a file or database.

This project encourages practice in object-oriented programming, the design of abstract classes and interfaces, and the development of an application with a user interface. It also allows exploration of list management and data manipulation concepts.

Work to do:

1. Create a use case diagram to illustrate the interactions between users and the system, identifying main functionalities of this software.
2. Create a class diagram for this system.
3. Implement this software in Java.
4. Add comments to your source code and use the Javadoc tool to generate a documentation for your code.

Evaluation: Students will be evaluated based on the quality of their code, the effectiveness of their graphical interface, the implementation of abstract classes and interfaces, task management, and project documentation.