OOSE Practice Quiz [Spring 2020]

- This is a closed book exam. You may **not** use any written/printed/online notes.
- This quiz has 5 problems (including this) and total of **50 points**.
- You have 50 minutes to complete the quiz.
- You may **not** use any communication devices (phone, etc.) while examination is in progress.
- Don't spend too much time on any problem.
- Good luck!

True/False

Each statement is worth **1 point**.

Statements

1. In Object-Oriented programming (OOP), the primary purpose of "polymorphism" is to reuse your code.
2. Object-Oriented Analysis and Design (OOAD) is a software engineering approach that models a system as a group of interacting objects.
3. In the Class, Responsibility, Collaborator (CRC) model, "responsibility" is something that a class **knows** or **does**.
4. Coupling is a measure of how strongly one element in code (such as a class or a method) is *focused* and comprise of responsibilities which belong together.
5. Gradle is a build automation tool that can also be used to manage your (software) project dependencies.
6. Adapter is a structural design pattern that allows objects with *incompatible* interfaces to collaborate.
7. Open-closed principle states when extending a class, consider that you should be able to pass objects of the subclass in place of objects of the parent class without breaking the client code.

```
// Answers
1. false (Inheritace)
2. true
3. true
4. false (Cohesion)
5. true
6. true
7. false (Liskov Substituition Principle)
```

Multiple-Choices

Each question is worth **2 points**.

Questions

1. In a User Story, …… represent the simplest candidates to be classes. Moreover, …… are candidates for classes' behavior.
   - [x] a. nouns, verbs
   - [ ] b. subjects, objects
   - [ ] c. verbs, nouns
   - [ ] d. objects, subjects
2. Git technology is **primarily** used for …
   - [ ] a. Incremental development
   - [x] b. Source code version control
   - [ ] c. Continues integration
   - [ ] d. Collaborative programming
3. The **primary** use of "Pull Request" on GitHub is …
   - [ ] a. to allow a project to move in multiple different directions simultaneously.
   - [ ] b. for combining different versions of code.
   - [x] c. to merge a branch of a repository with another branch of the same repository.
   - [ ] d. to make a copy of a repository so that any changes to the copy would not affect the original.
4. What is an advantage to using the MVC (Model-View-Controller) design pattern?
   - [ ] a. Model information can only be accessed and manipulated by the view.
   - [ ] b. The application as a whole adheres to client-server architecture.
   - [ ] c. Dependency inversion is automatically guaranteed to happen between model objects.
   - [x] d. Each section (model, view, controller) adheres to the single responsibility principle.

Scenario

Suppose you are designing a software application that will allow the users to perform task management. The user can add tasks to the system and can group tasks together into projects. Projects can be added as sub-projects of other projects, nested arbitrarily deep. Each tasks has an estimated time for completion that is specified when the task is constructed. You want to be able to treat individual tasks and projects in the same way. In particular, you want to be able to get the time needed to complete a task or a project. The time taken to complete a project is the total time needed to complete all the tasks in the project or in sub-projects of that project.

Part 1

Based on the software description, write two "must have" (functional) requirement in form of User Stories. [**3 pts**]

```
// Sample answer
- As a user, I want to group tasks into a project so that I can better manage my tasks.
- As a user, I want to assign estimated completion time to each task so that the software give me the time needed to complete a project.
```

Part 2

This application conforms to the Client-Server software architecture. To show your understanding of this architecture, describe one use-case (a scenario involving a user using the proposed software) and indicate the interaction between different entities (user, client, server, database, ...) involved in the use-case. [**4 pts**]

```
// Sample answer
User clicks on "add task" button on the client application.

The client application collects the information provided for the task and sends a request to the server to create and store the task.

The server recieves the requests. It creates a task with the provided information. Stores the task in the database. Upon sucessfull completion of this process, the server sends a response to the client application.

The client application, upon reciving the server's response, displays the newly added task in the list of tasks.
```

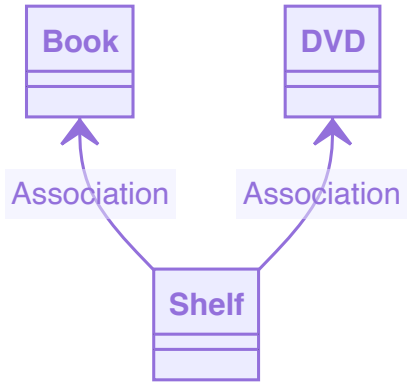
Part 3

Based on the software description, what design pattern(s) [among those we covered in lecture/readings] would apply to the design of this application. **Name** the design pattern and **elaborate** (briefly) on the problem and proposed solution (how it fits here). [**13 pts**]

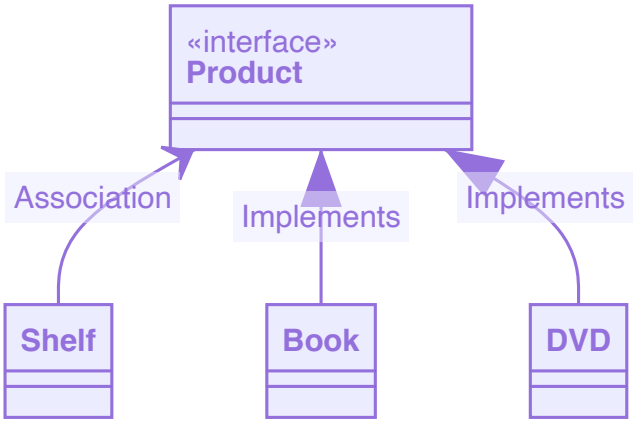
```
// Sample answer
-- Composite Design Pattern --

>>> Problem >>>
We need to manipulate a hierarchical collection of "primitive" and "aggregate"
objects. Moreover, we need to process (treat) aggregate objects the same way as
primitive objects.

The primitive object is a Task.
The aggregate object is a Project.

>>> Solution >>>
Following the pattern, we can create an abstraction like AbstractTask class with
an abstract method getEstimatedCompletionTime().
We implement a Task class that extends AbstractTask.
We implement a Project class that extends AbstractTask but also contains a
collection of AbstractTasks with methods to add/remove AbstractTasks.
```

Design Review

A team of students are building a Bookstore Management Software. The UML design for the *first iteration* includes the following:



The team advisor suggested this alternative design:



Briefly explain what SOLID design principle(s) the advisor's revised design adheres to. [**10 pts**]

```
-- Dependency Inversion Principle --
An abstraction (Product interface) is introduced between the high-level classes
(Shelf) and low-level classes (Book & DVD) that changes the direction of the
dependency and splits the dependency between the high-level and low-level
modules.

-- Open/Closed Principle --
To extend the application, e.g. to use other products, all needs to be done is to
add another concrete implementation of the Product interface. The extention
(adding a new Product) will not require any changes to the classes already
exsiting in the model.
```

Update your Resume!

After taking this class, you've updated your resume and added "Participated in designing and development of software using agile development practices" under EN.6001.421 OOSE. During a phone interview, a recruiter asks you to elaborate on the agile practices you've incorporated for your OOSE course project.

Summarize your answer (which you would give to the recruiter) in a paragraph. We expect you to highlight two agile practices in your answer. [**5 pts**]

```
// Sample answer
We adopted incremental development approach spanned over several short (two-week)
iterations. Each iteration started with planning on what to achieve and ended
with a retrospective. At regular intervals, we reflected as a team on how to
become more effective. We gathered the software requirement in form of a product
backlog (a collection of User Stories) and delivered finished product increments
at the end of each iteration. After recieving feedback about the delivered
features, we adjusted and updated the requirements at each iteration.
```