**Object-Oriented Analysis and Design**

**성명:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

|  |
| --- |
| **※ 총 4 페이지, 제한 시간 60분.**  **※ Open book. Closed neighbor.**  **※ 문제지에 직접 답안을 작성한 후 제출하세요.** |

**Part I.** *(32점)* **객관식 문제**

**1.** *(4점)* 다음과 같이 클래스 **B**가 클래스 **A**의 서브클래스로 선언된 경우, 그 의미와 **관계 없는** 것은?  **(2)**

**class B extends A { ... }**

① Every object of type **B** is also an object of type **A**, but *not vice-versa*.

② Anything that is true of an object of type **B** is also true of an object of type **A**, but *not vice-versa*.

③ **A** represents a more general concept than **B**, and **B** represents a more specialized concept than **A**.

④ Anywhere an object of type **A** can be used, an object of type **B** can be used as well, but *not vice-versa*.

**2.** *(3점)* 다음중 객체와 속성(attribute)의 관점에서 볼때, 열거된 다른 것과 가장 관계가 먼 것은? **(4)**

1. Course ② Department ③ Student ④ Age

**3.** *(3점)* 다음중 개발프로세스(development process)가 정의하고 있는 것과 가장 거리가 먼 것은? **(3)**

1. Who ② When ③ Where ④ What ⑤ How

**4.** *(3점)* Unified Process (UP)의 각 phase를 순서대로 올바르게 나열한 것은? **(4)**

① Elaboration-Inception-Construction-Transition

② Transition-Inception-Elaboration-Construction

③ Construction-Elaboration-Transition-Inception

④ Inception-Elaboration-Construction-Transition

**5.** *(3점)* 다음 중 UP의 best practices와 관계 **없는** 것은? **(1,5)**

① Tackle low-risk issues in early iterations.

② Build a cohesive, core architecture in early iterations.

③ Adopt iterative and incremental development approach.

④ Apply use cases.

⑤ Adopt linear, document-driven approaches.

**6.** *(3점)* 다음 중 사실과 거리가 먼 것은? **(4)**

① Analysis emphasizes an investigation, understanding, and discovery of the problem domain and requirements.

② Analysis does not concern how a logical solution is defined.

③ OO design (OOD) is primarily a process of invention and adaptation of conceptual solution.

④ All the vocabularies used in the analysis ~~and design~~ must come from the problem domain.

**7.** *(3점)* 다음 중 소프트웨어 개발 프로세스에서 분석(analysis)의 목적을 **가장 잘** 설명하고 있는 것은? **(3)**

① Analysis allows programmers to check that their data structures and algorithms are correct before their code is integrated into the system.

② Analysis translates system requirements into UML classes and relationships. This is important because the UML classes are then easily implemented in object-oriented languages such as Java or C++.

③ Analysis results in artifacts that capture the requirements of the system and document the domain model of the problem domain that is shared by all parties (developers and clients) who have an interest in the system that is being developed.

④ Analysis helps clients understand how computers and software applications are developed by translating computer science concepts and terminology into the equivalent concepts in the problem domain.

**8.** *(3점)* 다음 중 설계(Design)의 목적을 가장 잘 설명하고 있는 것은? **(2)**

① The system design must be documented so that the clients will understand how to interpret the program code. This will help them to test the system once it is delivered.

② The development of a logical model of the system that is consistent with the domain model and satisfies the system requirements.

③ It prevents the reinvention of code that already exists in other applications and could be easily reused in the current application.

④ The assignment of responsibility of program modules to programmers. Each programmer should be given responsibility for a module that best matches his or her skills and experience.

**9.** *(3점)* 다음 중 Use Case와 관계 **없는** 것은? **(3)**

① Informally, use cases are stories of using a system to meet goals.

② A use case is a set of use-case instances, where each instance is a sequence of actions that yields an observable result of value to a particular actor.

③ Use cases document the behavior (i.e., functional requirements) of the system from the developer's point of view.

④ A source for most of the quality attributes.

**10.** *(4점)* 다음은 UP Phase의 목적을 나타내는 목록이다.

① Feasibility Study

② Beta Test, Tuning

③ Baseline architecture

④ Implementation of Lower risks

위의 목록 가운데 아래 각 Phase의 목적과 가장 밀접한 관련이 있는 것을 고르시오.

가. Construction ........................................( **Implementation of Low risks** )

나. Inception    ........................................( **Feasibility Study** )

다. Transition   .........................................( **Beta Test, Tuning** )

라. Elaboration  ........................................( **Baseline Architecture** )

**Part II.** *(14점)* **다음의 각 물음에 간단히 답하시오.**

**1.** *(5점)* Analysis는 크게 requirements analysis와 object analysis (또는 domain analysis)로 구분된다. 각각의 목적을 기술하고 각 결과물이 표현되는 모델이 무엇인지 기술하시오.

**Requirement Analysis : Functional Requirement 와 Non-functional Requirement를 조사하고, Use Case Model 로 Capture 된다.**

**Object Analysis : Problem domain에서 object를 찾고 기술하는데 중점을 두고 있으며, Domain Model로 Capture 된다.**

**2.** *(4점)* Use Case에서 일반적으로 actor라 함은 시스템과 직접적인 interaction을 하는 외부의 사람, 소프트웨어 시스템 또는 장치(device)로 정의된다. 이 정의에 의하면 offstage actor는 이 범위에서 벗어나지만, 그럼에도 불구하고 그(것)들을 actor로 반드시 고려해야 하는 이유는 무엇인가?

**Stakeholder의 모든 필요한 Interests 가 Capture 되고 Satisfied 되었는지 확인하기 위해서**

**To ensure that the all the necessary interests are captured and satisfied**

3. (*5점*) Use Case에서 일반적으로 actor라 함은 시스 GRASP 패턴을 적용하여 collaboration diagram을 작성하는 도중, Expert 패턴과 Creator 패턴이 모두 적용될 수 있는 상황에 부딪혀, 서로 다른 두 가지의 해결책이 존재한다면 어떻게 이 문제를 해결하도록 하는 것이 바람직한지 기술하시오.

**High Cohesion Loose Coupling**

**Part III.** *(16점)* 다음에 주어진 설명을 잘 읽고,  언급된 conceptual class들 사이의 각종 관계(association, aggregation, composition aggregation, generalization/specialization, multiplicity, etc.)를 가능한 한 상세히 보여주는 UML class diagram을 그리시오.

(1) Draw a class diagram to describe programs.  A program consists of statements. A statement can be either a compound statement or a simple statement. In other words, compound statements and simple statements are special types of statements. A compound statement can contain either simple statements or compound statements or both.

[2] Draw a class diagram to describe directed graphs. A directed graph consists of a set of vertices and a set of edges. Edges connect pairs of vertices (source vertex and destination vertex). Your diagram should capture only the structure of the graphs (i.e. connectivity), and need not be concerned with geometrical details such as lengths of edges. A typical graph is shown below.

**Part IV.** *(8점)* 다음아래 그림에서 보는 바와 같이, Clock 객체는 자신의 notify() method가 호출되면 자신과 연결된 Counter 객체의 next()를 호출하여 Counter 객체가 자신의 속성인 value를 증가시킬 수 있도록 설계되었다.

그러나, 이러한 설계는 Clock을 항상 Counter와 함께 사용해야 한다는 제약을 가지고 있다. Clock의 notify()가 호출되었을 때, 자신과 연결된 임의의 객체의 next()를 호출할 수 있도록 함으로써 Clock 객체를 특정 클래스(예, Counter)와 독립적으로 사용할 수 있도록 위의 설계를 변경하시오.

**Part VI.** *(10점)*

다음에 주어진 Use Case를 읽고 Main Success Scenario로부터 도출되는 SSD(System Sequence Diagram)을 보이시오.

|  |
| --- |
| **Use Case: Checkout books**  **Primary Actor:** Librarian  **Main Success Scenario:**  1. The Librarian enters the identity of a customer who wishes to check out books.  2. The System confirms that the customer is allowed to checkout books, and remembers the customer's identity.  3. The Librarian enters the identity of a book that the customer is checking out.  4. The System confirms that the book can circulate, calculates the due date based on whether the customer is a faculty member or a student, and records that the customer has checked out this book, which is due on the calculated due date.  5. The System tells the Librarian the due date.  The Librarian repeats steps 3-5 until indicates done. |

**Part V.** *(20점)*

[1] 아래에 주어진 sequence diagram으로부터 도출된 class diagram을 그리시오. (단, 객체간의 visibility는 모두 attribute visibility로 가정하시오.

(2) 위의 sequence diagram을 동일한 communication diagram으로 보이시오. 단, 메시지의 번호는 dewey

decimal 표기법을 따르시오.