**“A”**

A "/" symbol before the attribute name is wrong in domain model. **F**

A domain model can show UI elements and database in a business project. **F**

Arrow line indicates the relation of a user use use cases in UML Use Case Diagram. **F**

Any UML diagram in OOA must describe the System as Black Box.  **F**

An object in domain layer can send messages to an UI object.  **F**

A timer may be a special actor of the system.  **T**

A domain model is a visual representation of conceptual classes or real-situation objects in a domain. **T**

A use case named 'Manage <X>', such as manage user, means that the user goal is to collapse(集中) CRUD on object X . **T**

An end user can find the operation process in UML Use Case Diagram. **F**

**“C-D”**

Composition in the UML loosely suggests whole-part relationships.  **F**

"condition: actions ..." structure can be used to describe a Use case Main Success Scenario. **F**

"+doSomething(b:B)" of class A means that the class A dependent on the class B. **T**

During OOA, there is an emphasis on finding and describing the objects or conception in the problem domain. **T**

**“I”**

In Design Class Diagram(DCD), a association line between classes may has an association name. **F**

In DCD, use a navigability arrow(导航箭头) to indicate that a Register object has an attribute of one Sale object. **T**

In GoF Composite pattern, the whole object and its part objects have the same interface. **T**

Informally, a use case is a specific sequence of actions and interactions between actors and the system. **F**

In software design, Aggregation over Composition. **F**

It is more than "a few" weeks long for most projects in Inception. **F**

In Agile modeling, a suitable iteration length is two to six weeks time\_box. **T**

In UP, a suitable iteration length is two to six weeks timebox. **T**

In Agile Principles, our highest priority is early and continuously delivers valuable software to customer. **T**

In elaboration, we do not need deliver a runnable product to user for testing. **F**

In the UP, requirements are categorized according to the FURPS+ model. **T**

In Design Class Diagram(DCD), attributes are usually assumed private if no visibility is given. **T**

In practical OOD, we draw interactive diagram firstly, and then draw Design Class Diagram.  **T**

Interface is a UML classifier. **T**

In Agile modeling, the purpose of modeling (sketching UML, …) is primarily to understand, not to document.  **T**

In domian model, only show some meaningful and interesting associations among classes.**T**

Incremental development for the same use case across iterations.  **T**

Inception in one sentence: Envision the product scope, vision, and business case.  **T**

In UP, A Kickoff meeting should be hold to clarify goals in days at the beginning of iteration.  **T**

**“F”**

Finding conceptual classes with noun phrase identification is a useful technique. **T**

"Find Product Help" with underline in a use case text means emphasis. **F**

Finding stakeholders of the system is the most important thing for catching requirements.  **T**

**“T”**

The precondition of buy drink for a vending machine(售货机) is "OK" displayed. **F**

To implement a singleton class, we prefer eager initialization. **F**

The requirements and object-oriented analysis focused on learning to do the thing right. **F**

The sequence of software development in UP should be: 1) define the requirements; 2) design the architecture; 3) implement. **F**

The System Event handled by model layer in the MVC pattern.  **F**

The message 'd = getProductDescription(id)' in sequence diagram is wrong syntax for missing a type of id. **F**

The candidate use case 'Log In' can pass usefull Test, for we use it every day. **F**

The same UML diagram can be used in multiple perspectives. **T**

**“S”**

Software engineering is the application of a systematic, disciplined, quantifiable approach to develop software. **T**

Spend significant time doing class diagrams than UML interactive diagrams. **F**

SSD should involving other external systems. In general, the external system place at the left of the ":system". **F**

Showing a report object in a domain model is always useful.  **F**

**“U”**

Use Case name must start with a verb. **T**

Use-case-driven means the development team employs the use cases from requirements gathering through code and test.  **T**

**“L-P-X”**

Logical Architecture divides the software classes into packages, subsystems, and layers. **T**

Primary actors always be on the left and supporting actors on the right in UML Use Case Diagram.  **T**

XP "story card" practice is acceptable for catching requirements in UP. **T**