Question 1

**What is an object (instance) and what is a class?**

Your Answer:

A class is a defined structure of an object. It contains the properties and functionality. An object in an instantiation of the class. It will access the properties and functionality of the class.

An object is an instance of a class

Classes should group things that have a common structure, exhibit a common behavior, or have the same relationships

Question 2

**What does the term polymorphism mean with reference to software engineering?**

Your Answer:

Polymorphism is the concept of defining the same functionality in different ways. This is generally done through overloading (maybe overriding as well).

Polymorphism allows objects of varying type, but with the same interfaces, to be substituted for each other at runtime. Polymorphism promotes software extensibility; however, it can impact performance and make code difficult to understand. In the example below, the move() method behaves differently in both classes

Question 3

**What is the difference between class and type inheritance?**

Your Answer:

Class inheritance is when one class inherits the attributes and methods of another class. Type inheritance is when an object inherits the structure of another. The type defines what attributes the object must or can have.

Question 4

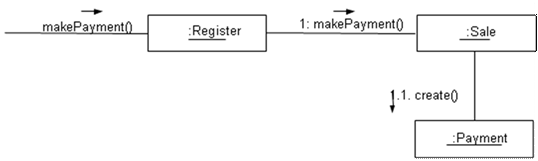
**What do the terms coupling and cohesion mean?**

Your Answer:

**LOW COUPLING:** The intent of this pattern is to assign a responsibility so that coupling remains low.

Coupling is a measure of how strongly one element is connected to, has knowledge of, or relies on other elements. **Low coupling** is an evaluative pattern that dictates how to assign responsibilities for the following benefits:

* lower dependency between the classes,
* change in one class having a lower impact on other classes,
* higher reuse potential.



**HIGH COHESION:** Cohesion is a measure of how strongly related and focused the responsibilities of an element are. An element with highly related responsibilities and which does not do a tremendous amount of work has high cohesion. The intent of this pattern is that cohesion remains high.

Question 5

**What is a pattern? Name two.**

Your Answer:

A Pattern is a design concept and template used to achieve some form of functionality. I don't remember many patterns specifically, but I believe Observer is one. I'm not sure I can recall a second by name.

Question 6

**The UML is a:**

language

standard

technique

Question 7

**What is the major difference between a use case and a user story?**

Your Answer:

A use case describes a specific action to achieve a specific outcome. There are usually many different use cases. A user story is more of general description of the journey a user may interact with the system to reach some outcome. You can describe a user story with many use cases.

Question 8

**Analysis is often referred to as modeling the problem and design as modeling the solution, but in object-oriented systems we base the design model on the problem domain. Why? What advantages are gained in this approach?**

**Your Answer:**

The design model is based on the problem domain because it is beneficial to design the software around the root of the problem that you are trying to solve. This way to can design a solution out of the problem and not have to worry about implementation details. An advantage of this is being able to use design patterns for the general problem. It also removes any distraction that may arrive from specific implementations so you can design your solution around the problem and not the language or data model.

#### ***​*Hierarchy**

The relationships just described provide support for hierarchies. Generalization supports inheritance hierarchies. We have *class hierarchy* which is the implementation of inheritance (kind-of relationship) – then we have a *type hierarchy* which is interface inheritance (type-of relationship). Aggregation/composition support hierarchies of parts (part-of relationship)

Abstractions reduce to only the essential characteristics, however, there may be more different abstractions than we can handle at one time. Therefore, encapsulation (hiding the inside view of the abstraction) and modularity (cluster logically related abstractions) isn’t enough. Inheritance is the most important “is a” hierarchy. It is an essential element of the o-o systems. It defines a relationship among classes. But remember what was said about inheritance earlier, it can cause tight/high coupling (not good).

As you can gather from the content in this lesson, effective software is more about the design coding.