***Note Q 7, Q8, Q9 not available in this file.**

OOAD 2013 Solved Short Questions

Question No 1:

A UML Operation contract identifies system state changes when an operation happens. Effectively, it will define what each system operation does. An operation is taken from a system sequence diagram. It is a single event from that diagram. A domain model can be used to help generate an operation contract. The domain model can be marked as follows to help with the operation contract:

Question No 2:

A design pattern is a general solution to a recurring problem. Design patterns are more conceptual than tangible and can be modified to fit the exact need. However, abstract classes and interfaces can be reused to implement certain patterns.

Question NO 3:

A **container class** is a data type that is capable of holding a collection of items. In C++, **container classes** can be implemented as a **class**, along with member functions to add, remove, and examine items.

- Create an empty container (via a constructor)
- Insert a new object into the container
- Remove an object from the container
- Report the number of objects currently in the container
- Empty the container of all objects
- Provide access to the stored objects
- Sort the elements (optional)

Question NO 4:

Inception is the smallest phase in the project, and ideally it should be quite short. If the Inception Phase is long then it may be an indication of excessive up-front specification, which is contrary to the spirit of the Unified Process.

Elaboration phase the project team is expected to capture a healthy majority of the system requirements. However, the primary goals of Elaboration are to address known risk factors and to establish and validate the system architecture. Common processes undertaken in this phase include the creation of use case diagrams, conceptual diagrams (class diagrams with only basic notation) and package diagrams (architectural diagrams).

Question NO 5:

Cohesion is the indication of the relationship within **module**.	**Coupling** is the indication of the relationships between modules.
Cohesion shows the module's relative **functional** strength.	Coupling shows the relative**independence** among the modules.
Cohesion is a degree (quality) to which a component / module focuses on the**single** thing.	Coupling is a degree to which a component / module is connected to the**other** modules.
While designing you should strive for **high cohesion** i.e. a cohesive component/ module focus on a single task (i.e., **single-mindedness**) with little interaction with other modules of the system.	While designing you should strive for **low coupling** i.e. **dependency** between modules should be less.
Cohesion is the kind of natural extension of data hiding for example, **class**having all members visible with a package having default visibility.	Making private fields, private methods and non public classes provides loose coupling.
Cohesion is **Intra – Module** Concept.	Coupling is **Inter -Module** Concept.

Question NO 6:

An **object** can be a variable, a data structure, or a function, and as such, is a location in memory having a value and possibly referenced by an identifier.

In the class-based object-oriented programming paradigm, "object" refers to a particular instance of a class where the object can be a combination of variables, functions, and data structures.

- Function object: an object with a single method (in C++, this method would be the function operator, "operator()") that acts much like a function (like a C/C++ pointer to a function).
- Immutable object: an object set up with a fixed state at creation time and which does not change afterward.
- First-class object: an object that can be used without restriction.
- Container object: an object that can contain other objects.
- Factory object: an object whose purpose is to create other objects.
- Metaobject: an object from which other objects can be created (compare with a class, which is not necessarily an object).
- Prototype object: a specialized metaobject from which other objects can be created by copying
- God object: an object that knows or does too much (it is an example of an anti-pattern).
- Singleton object: an object that is the only instance of its class during the lifetime of the program.
- Filter object.

Question NO 10: The following sections present the first five GRASP patterns:

Information Expert

Creator

High Cohesion

Low Coupling

Controller

There are others, introduced in a later chapter, but it is worthwhile mastering these five first because they address very basic, common questions and fundamental design issues. Please study the following patterns, note how they are used in the example interaction diagrams, and then apply them during the creation of new interaction diagrams. Start by mastering Information Expert, Creator, Controller, High Cohesion, and Low Coupling. Later, learn the remaining patterns.