# ISYS1083/1084 Object Oriented Software Design

**Topic 2 TuteLab - GRASP and CRC Cards**

**Goals**

Familiarise yourself with GRASP processes and techniques. Learn to use CRC Cards as a vehicle for those processes and techniques.

1. **GRASP**

Describe how the following GRASP principles help in assigning responsibilities?

* 1. Creator
  2. Information Expert/Holder
  3. Polymorphism
  4. Pure Fabrication
  5. Controller
     1. **Creator**

Creation of objects is one of the most common activities in an object-oriented system. Which class is responsible for creating objects is a fundamental property of the relationship between objects of particular classes. In general, a class B should be responsible for creating instances of class A if one, or preferably more, of the following apply:

* + Instances of B contain or compositely aggregate instances of A
  + Instances of B record instances of A
  + Instances of B closely use instances of A
  + Instances of B have the initializing information for instances of A and pass it on creation.
    1. **Information Expert**

**Information Expert** (also **Expert** or the **Expert Principle**) is a principle used to determine where to delegate responsibilities. These responsibilities include methods, computed fields, and so on. Using the principle of Information Expert, a general approach to assigning responsibilities is to look at a given responsibility, determine the information needed to fulfill it, and then determine where that information is stored. Information Expert will lead to placing the responsibility on the class with the most information required to fulfill it.

* + 1. **Polymorphism**

According to **Polymorphism**, responsibility of defining the variation of behaviors based on type is assigned to the types for which this variation happens. This is achieved using polymorphic operations.

* + 1. **Pure Fabrication**

A **Pure Fabrication** is a class that does not represent a concept in the problem domain, specially made up to achieve low coupling, high cohesion, and the reuse potential thereof derived (when a solution presented by the *Information Expert* pattern does not).

* + 1. **Controller**

The **Controller** pattern assigns the responsibility of dealing with system events to a non-UI class that represents the overall system or a use-case scenario. A Controller object is a non- user interface object responsible for receiving or handling a system event.

It is defined as the first object beyond the UI layer that receives and coordinates ("controls") a system operation. The controller should delegate the work that needs to be done to other objects; it coordinates or controls the activity. It should not do much work itself.