OOAE Test 3 Revision

Polymorphism is the ability for an object to assume many different forms. E.g. A car object could have many derived classes, such as saloon, SUV, coupe etc. and these would be a different form of the car object with the ability to redefine methods from the base class.

Inheritance is where a subclass can inherit methods and variables from it’s superclass.

Abstract classes cannot be instantiated because it isn’t guaranteed that all methods have been implemented while a concrete class can as it’s guaranteed. Abstract classes are meant to be extended.

Interfaces are a contract or blueprint with only method signatures with functionality being realised by classes implementing the interface.

**Creational patterns**Abstract the instantiation process, meaning the client is independent of how classes and objects are created, composed or represented. I.e the builder, or factor are responsible for the creation of objects.

Singleton Pattern  
Intent: Dictates that only one instance of an object exists at any one time.

Factory Pattern  
Intent: Defines an interface for creating objects, but let subclasses decide which class to instantiate.

Builder Pattern  
Intent: Separate the construction of a complex object from it’s representation so that the same construction process can create different representations. E.g. instead of having an opaque constructor with lots of parameters, construction can be ordered by the builder to build it up step by step.

**Structural patterns**Structural patterns define how classes and objects are composed to form larger structures.

Bridge Pattern  
Intent: Decouple an abstraction from it’s implementation so that the two can vary independently.