**Week 4 Term Definitions**

**aggregate object**

One of two objects in the aggregation (whole/part) relationship, referring to the object that is the 'whole' (the other object is 'part of' the aggregate object).

**aggregation (class relationship)**

A relationship between classes whereby the objects of one class will be part of the objects of the other class. The lifetimes of the two objects are usually loosely coupled, e.g., Book objects listed in an Order object.

**association (class relationship)**

A general relationship between classes that allows the objects of one class to invoke the methods defined for the objects of the other class.

**automatic variable**

A variable whose memory is "automatically" allocated and deallocated on the call stack from when the variable is defined until it is no longer required, e.g., a local variable in a method that no longer exists when the method terminates.

**call stack (process concept)**

A region of memory in a process that is used for temporary storage of data including variables local to a method, intermediate calculation results, and recording data between method calls (hence why it is also relevant to debugging).

**class relationship**

Some kind of connection or dependency between two classes in an object-oriented application. An instance of a class relationship is an object relationship.

**client object**

An object that requests the services (invokes the methods) of a supplier object.

**composition (class relationship)**

A tighter form of the aggregation relationship whereby the part object can only ever be part of one whole. The lifetimes of the two objects are usually tightly coupled, e.g., the body of an email message.

**copy constructor**

A type of constructor that contains a single parameter - an object that of the same class - whose state is copied entirely into the new object.

**custom constructor**

A constructor that has been defined by the programmer, as opposed to the default constructor which is defined by the compiler. The term custom constructor can also be used to refer to a constructor with one or more parameters.

**deep copy**

The process of copying an object and also all of the objects referred to by that object to overcome the shallow copy problem.

**default constructor**

The constructor that is generated by the compiler for a class when no other constructors are defined by the programmer. The default constructor has no parameters and no method body. The term can also be used to refer to a parameter-less constructor.

**delegate**

A reference to a method that allows, similar to a reference to an object, but allows the method to be invoked through that reference. Commonly used for event handlers for GUI.

**dispose method**

A method that can be defined to allow the programmer to directly invoke the functionality of a finaliser without the need to wait for the garbage collector to invoke the finaliser.

**duplicate code**

Where identical lines of code exist in a program unnecessarily. This can lead to problems where if a bug is corrected in the code it may not be corrected in all duplicates.

**dynamic memory allocation**

The creation of an object by allocating memory from the heap at run-time, as opposed to an automatic variables.

**finaliser**

Another name for a destructor, which is a method that is invoked automatically when an object is to be deallocated.

**garbage collector**

A program that runs .

**heap (process concept)**

A region of memory in a process that is used for dynamic memory allocation.

**indexer**

A means through which it is possible to define the indexer operator, similar to accessing an array element, for custom classes, e.g., myObject[index].

**link (object relationship)**

A connection between two objects whereby a client object can request the services (invoke the methods) of a supplier object.

**memory leak**

A common error in applications where explicit memory deallocation is required, where memory is no longer used but it has not been deallocated, and has usually been forgotten (all references lost/overwritten).

**navigability**

A link is considered navigable if an object is able to request the services/invoke methods of the other object, i.e., the link from the client object to the supplier object is considered navigable, but not vice-versa (a supplier cannot invoke the methods of a client, unless there is another link.

**object initialiser**

An object initialiser allows one constructor to invoke another constructor, either in the same class (using the keyword this) or in a base class (using the keyword base).

**object relationship**

An instance of a class relationship, known as a link.

**parameter-less constructor**

A custom constructor written by the programmer that does not have any parameters.

**shallow copy problem**

Where an object has been copied but other objects that are linked to that object are not copied, which can lead to unexpected behaviour of the program, e.g., data corruption.

**string concatenation**

The action of joining two strings using the addition operator (+).

**supplier object**

An object that provides services (contains methods that are claled) by client object/s.

**text region (process concept)**

A region of memory in a process that stores the compiled instructions (in machine code) of a program.

**unmanaged resource**

A logical entity, such as an open file, network connection, database connection, or GUI component, that takes up memory outside of the managed (Microsoft.Net Framework) resources.

**unreachable object**

An object previously allocated by an application that is no longer being referenced by any part of the active program, or an object that is only being referenced by other objects that are no longer referenced by the active program.