Hashing Concepts *(although this refers to Java, the concepts are the same for C#)*

Hashing is used to improve efficiency when searching large, linear data structures such as Lists, ArrayLists, Maps and Sets.

Hashing - using a function/algorithm to map object data to an integer value (***Hash Code***).

Hash Code - used to narrow down our search when looking for an item.

Hash Codes may not be unique. The same Hash Code may be generated by multiple objects.

Care must be taken to minimize the generation of identical Hash Codes or handle the case when duplicate Hash Codes are generated.

Various techniques are available for generating Hash Codes.

Java provides a method called ***hash()*** which may be used by a class to easily generate Hash Codes. You, of course, can write your own method to generate a Hash Code.

Java looks for a **hashCode()** method for an object whenever it needs a Hash Code for the object. For example, when adding an element to a **HashSet** or **HashMap**.

Every class inherits an ***equals()*** and ***hashCode()*** method from the Object class. These methods use the reference to the object *(location in memory)* in thier processing. Therefore two objects are considered equal or generate the same Hash Code if and only if they have the same reference.

Every class should define a ***hashCode()*** and ***equals()*** override as only the class knows when objects instantiated with it are equal.

The same, immutable *(value does not change)*, data members must be used in both the ***equals()*** and ***hashCode()*** overrides to ensure objects of the class may be processed successfully by Java.

Immutable data members must be used to avoid the Hash Code for an object changing once generated and store.

Hash Code Collision - Occurs when the same Hash Code is created for unequal objects.

A typical method for handling Hash Code Collisions, is to store the elements in a data structure, such as an ArrayList, rather than a single object.