|  |
| --- |
| DAY 6 ASSIGNMNET |

1. **Research and find how the values of ArrayList are stored in the memory.**

* The elements of an ArrayList are stored in a chunk of contiguous memory.
* When that memory becomes full, a larger chunk of contiguous memory has to be allocated (usually twice the size) and the existing elements are copied into this new chunk.
* We call this chunk the capacity of the ArrayList object.

1. **What are the dis-advantages of ArrayList (Collections ArrayList)**

**Disadvantages of ArrayList:**

1. Unboxing
2. Runtime errors – when any other datatype is entered.
3. **In a tabular format write the differences between Collections and generics.**

|  |  |  |
| --- | --- | --- |
|  | **Collections** | **Generics** |
| 1. **namespace** | System.Collections; | System.Collections.Generic; |
| **2. Each element is of what type** | Object | Any datatype |
| **3. Do you need type casting here** | YES | NO |
|  | **ArrayList** | **List** |
| **4. Example** | ArrayList data = new ArrayList(); | List<int>data = new List<int(); |

1. **Research and find how the values of List<T> are stored in the memory.**

* The memory to store the value types is within the memory allocated for the System. Array (i.e. "Over Here").
* In an ArrayList each element is just a reference to a boxed value type, so the actual memory to store each value type is elsewhere on "The Heap", i.e. somewhere "Over There".

1. **Example programs for implicit and explicit type casting.**

|  |  |
| --- | --- |
| **Implicit Type Casting** | **Explicit Type Casting** |
| Int a = 20;  Double b = a;  Console.writeLine(b); | Float a = 8.52;  Int b = int(a);  Console.WriteLine(b); |

1. **Datatypes and it’s respective alias names.**

|  |  |
| --- | --- |
| **DataTypes** | **Alias name** |
| byte | Byte |
| ushort | UInt16 |
| Uint | UInt32 |
| Ulong | UInt64 |
| Sbyte | SByte |
| Short | Int16 |
| Int | Int32 |
| Long | Int64 |
| Float | Single |
| Decimal | Decimal |
| Double | Double |
| Bool | Boolean |
| Char | Char |
| String | String |