**Day 6 - Morning**

**Assignment By**

**Manoj Yekolla**

**31-Jan-2022**

|  |
| --- |
| **1. Create a simple program to declare ArrayList and assign some values**  **and find sum ?** |
| Code : |
| using System;  using System.Collections;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace ArrayList\_find\_Sum  {  internal class Program  {  static void Main(string[] args)  {  ArrayList data = new ArrayList();  int sum = 0;  data.Add(10);  data.Add(30);  data.Add(25);  data.Add(66);  foreach(var d in data)  {  sum = sum + (int)d;  }  Console.WriteLine(sum);  Console.ReadLine();  }  }  } |
| Output:  Screenshot (85) |

|  |
| --- |
| **2. Research and find how the values of ArrayList are stored in the memory ?** |
| **Ans :**  For storing ArrayList in memory JVM allocates contiguous memory in heap just like an array, but the difference is an array has a fixed size whereas an ArrayList can grow.  Both array and arraylist are stored in the same way ie contiguous memory locations in heap. Your selection of array vs arraylist depends on if you know the size of the data already. Array is fixed and ArrayList is dynamic.  In an ArrayList each element is just a reference to a boxed value type, |

|  |
| --- |
| **3. What are the dis-advantages of ArrayList (Collections ArrayList) ?** |
| **Ans :**    A possible disadvantage of ArrayList is that **it holds only object types and not primitive types (eg, int )**. To use a primitive type in an ArrayList, put it inside an object or use of the wrapper classes (eg, Integer, Double, Character, ...).  Some of Datatypes are mostly occured in runtime errors ,The Developer can’t  Resolved them. |

|  |
| --- |
| **4. Create a simple program to declare List<int> and assign some values**  **and find sum ?** |
| Code : |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Project\_4\_List\_int\_Find\_Sum  {  internal class Program  {  static void Main(string[] args)  {  List<int> data = new List<int>();  int sum = 0;  data.Add(10);  data.Add(19);  data.Add(53);  data.Add(49);  foreach(var d in data)  {  sum = sum + d;  Console.WriteLine(sum);  }  Console.ReadLine();  }  }  } |
| Output :  Screenshot (87) |

**5. In a tabular format write the differences between Collections and generics.**

**1. namespace**

**2. Each element is of what type**

**3. do you need type casting here**

**4. Example - ArrayList, List<T>**

**Ans :**

|  |  |  |
| --- | --- | --- |
|  | ArrayList | List |
| NameSpace | System.Collections namespace | System.Collections.Generic  namespace |
| Type | Object -data type | With out Specific data types |
| Type Casting | Yes | No |
| Example | using System.Collections;  ArrayList arlist = new ArrayList(); | using System.Collections.Generic;  List<int> data = new List<int>(); |

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

Ans :

a) In a List<T> , the memory to store the value types is **within the memory allocated for the System**.

b) Instances of reference types are stored on **the heap**. Since the list in your example is a List<int> it can only hold int s and no boxing occurs.

|  |
| --- |
| 1. **WACP to declare List<String> and add 5 values and print the values using?** |
| Code : |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace List\_String\_\_for\_foreach\_lamda\_  {  internal class Program  {  static void Main(string[] args)  {  List<string> data = new List<string>();  data.Add("Manoj Y");  data.Add("Gabbar");  data.Add("Murali");  data.Add("Nithesh");  //print values using for loop  for (int i=0;i<data.Count;i++)  {  Console.WriteLine(data[i]);  }  //print values using foreach loop  foreach (var d in data)  {  Console.WriteLine(d);  }  //print values using lamda experssions  data.ForEach(y=> Console.WriteLine(y));  Console.ReadLine();  }  }  } |
| Output:  Screenshot (89) |
| **8. WACP to declare List<int> and read 5 values from user and find sum using ?**  **a. for loop**  **b. foreach loop**  **c. Lamdba Expression** |
| Code: |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace List\_Int\_Find\_Sum\_for\_foreach\_lamda\_  {  internal class Program  {  static void Main(string[] args)  {  List<int> data = new List<int>();  int temp;  int sum1 = 0, sum2 = 0, sum3 = 0;  //Read 3 Num from User  for(int i=1;i <= 3;i++)  {  Console.WriteLine("enter any value :");  temp = Convert.ToInt32(Console.ReadLine());  data.Add(temp);  }  //find sum using for loop  for(int i=0;i<data.Count;i++)    sum1 = sum1 + data[i];  //find sum using foreach loop  foreach (var d in data)  sum2 = sum2 + d;  //find sum using lamda experssion  data.ForEach(d => sum3 = sum3 + d);  Console.WriteLine(sum1);  Console.WriteLine(sum2);  Console.WriteLine(sum3);  Console.ReadLine();  }  }  } |
| Output :  Screenshot (92) |

**9. In a tabular format write all data types in C# and write the respective alias name ?**

|  |  |  |
| --- | --- | --- |
|  | Data Type | Alias Name |
| 1 | byte | System.Byte |
| 2 | ushort | System.UInt16 |
| 3 | uint | System.UInt32 |
| 4 | ulong | System.UInt64 |
| 5 | sbyte | System.SByte |
| 6 | short | System.Int16 |
| 7 | int | System.Int32 |
| 8 | long | System.Int64 |
| 9 | float | System.Single |
| 10 | double | System.Double |
| 11 | decimal | System.Decimal |
| 12 | char | System.Char |
| 13 | boolean | System.Boolean |
| 14 | string | System.String |

**10. Write example programs for implicit and explicit type casting ?**

* ****Implicit Casting**** (automatically) - converting a smaller type to a larger type size  
  char -> int -> long -> float -> double

EX :

|  |
| --- |
| Code : |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace TypeCasting\_\_implicit\_explicit\_  {  internal class Program  {  static void Main(string[] args)  {  int myInt = 9;  double myDouble = myInt; // Automatic casting: int to double  Console.WriteLine(myInt); // Outputs 9  Console.WriteLine(myDouble);  Console.ReadLine();  }  }  } |
| Output :  Screenshot (94) |

* ****Explicit Casting**** (manually) - converting a larger type to a smaller size type  
  double -> float -> long -> int -> char

EX :

|  |
| --- |
| Code : |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace TypeCasting\_Explicit  {  internal class Program  {  static void Main(string[] args)  {  double myDouble = 9.78;  int myInt = (int)myDouble; // Manual casting: double to int  Console.WriteLine(myDouble); // Outputs 9.78  Console.WriteLine(myInt); // Outputs 9  Console.ReadLine();  }  }  } |
| Output :  Screenshot (96) |