1. Reuse memory
   1. Create new Windows forms application named Demo
   2. Add button with text “Reuse memory Test”
   3. Add below code on the click event of button

for (int i=0; i<100;i++)

{

var numbers = new int[10];

MessageBox.Show(numbers.Sum());

}

* 1. Open DotMemory profiler
  2. Start Application through profiler
  3. Click “Reuse memory Test” button
  4. Take profiler snapshot
  5. Add another button with text “Updated Reuse memory Test”
  6. Add below code on the click event of button

var numbers = new int[10];

for (int i=0; i<100;i++)

{

MessageBox.Show(numbers.Sum());

}

* 1. Open DotMemory profiler
  2. Start Application through profiler
  3. Click “Updated Reuse memory Test” button
  4. Take profiler snapshot
  5. Compare output from both snapshots for number of objects created memory

Utilization

1. For vs. For..Each loop
   1. Add another button with text “For Each Test” in above solution
   2. Add below code on the click event of button

var list = new List<int>(100);

int count = 1;

foreach (var item in list)

{

Item = count \* 10;

Count++;

}

* 1. Open DotMemory profiler
  2. Start Application through profiler
  3. Click “For Each Test” button
  4. Take profiler snapshot
  5. Add another button with text “For Test” in above solution
  6. Add below code on the click event of button

var list = new List<int>(100);

int length = list.Length;

int count = 1;

for (in I = 0;i< list. Length ; <i++)

{

List[i] = count \* 10;

Count++;

}

* 1. Open DotMemory profiler
  2. Start Application through profiler
  3. Click “For Each Test” button
  4. Take profiler snapshot
  5. Compare output from both snapshots for number of objects created memory

Utilization

1. Parsing through TryParse
   1. Add another button “Parse Test”
   2. Add below code to click event of this button

Var value = “10”;

MessageBox.Show(Int.Parse(value) \* 10);

* 1. Run the application & observer output 100
  2. Set value = “Hello”
  3. Run the application & observer error
  4. Modify above code as

Var value = “Hello”;

Int result;

Int32.TryParse(value, out result);

MessageBox.Show(result \* 100);

* 1. Run the application & observer output without error

1. Mutable & Immutable strings
   1. Add button with text “Immutable String”
   2. Add below code

MessageBox.Show (“{0} {1} {2}”, “Welcome”, “Sam”, ”To the C# Training”);

* 1. Run application to show message
  2. Add below code to same button click event

String message = “Welcome”;

String.concat(message, “ “, “Ram”);

String.concat(message,” “,”To the C# Training”);

MessageBox.Show(message);

* 1. Run the application & observe that message text show only word “Hello”
  2. Modify above code as

String message = “Welcome”;

message = String.concat(message, “ “, “Ram”);

message = String.concat(message,” “,”To the C# Training”);

message = MessageBox.Show(message);

* 1. Run the application & observe complete message
  2. Add button with text “Mutable String”
  3. Add below code to click handler of this button

StringBuilder sb = new StringBuilder();

Sb.Append(“Welcome”);

Sb.Append(“ “ );

Sb.Append(“Naveen”);

Sb,Append(“ “);

Sb.Append(“To the C# Training);

Messagebox.Show(sb.ToString());

* 1. Run the application & observe complete message

1. Using Right type of collection
   1. Add button with text “List”
   2. Add below code the click event handler

Var List<string> names = new List<string>();

names.Add(“John”);

names.Add(“Sam”);

names.Add(“Lynda”);

foreach(string item in names)

MessageBox.Show(item);

* 1. Run the application & hit button “List”
  2. Observe that the names are being displayed in the order of addition
  3. Add button with text “Sorted List”
  4. Add below code the click event handler

Var SortedList<string,string> names = new SortedList <string,string>();

names.Add(“John”,”John”);

names.Add(“Sam”,”Sam”);

names.Add(“Lynda”,”Lynda”);

foreach(string item in names.Keys)

MessageBox.Show (names[item]);

* 1. Run the application & hit button “List”
  2. Observe that the names are being displayed in the sorted order
  3. Add button with text “Dictionary”
  4. Add below code the click event handler

Var Dictonary<string,string> names = new SortedList <string,string>();

names.Add(“John”,”John”);

names.Add(“Sam”,”Sam”);

names.Add(“Lynda”,”Lynda”);

MessageBox.Show (“Record for Lynda {0}”, names.ContainsKey(“Lynda”) ? “exists”, “does not exist”)

1. Jagged Arrays –
   1. Add button with text “2-D Array”
   2. Add below code to the button

DateTime startdate = DateTime.Now;

String[,] topics = new string[3,2];

topics [0,0] = “.Net”;

topics [1,0] = “SQL Server”;

topics [2,0] = “HTML”;

topics [0,1] = “.Net”;

topics [1,1] = “SQL Server”;

topics [2,1] = “HTML5”;

For (int i =0;i< 3; i++)

{

For (int j =0;i< 2; j++)

MessageBox.Show (topics[i][j]);

}

MessageBox.Show(“Time Taken: “ + (end date – startdate).ToString());

* 1. Add button with text “Jagged Array”
  2. Add below code to the button

DateTime startdate = DateTime.Now;

String[][] topics = new string[3][];

Topics[0] = new string[2];

Topics[1] = new string[2];

Topics[2] = new string[2];

topics [0][1] = “.Net”;

topics [1][1] = “SQL Server”;

topics [2][1] = “HTML5”;

For (int i =0;i< 3; i++)

{

For (int j =0;i< 2; j++)

MessageBox.Show (topics[i][j]);

}

MessageBox.Show(“Time Taken: “ + (end date – startdate).ToString());