NAME : DERRICK WAGOKI M

REG NO: SCT221-0997/2021

APPLICATION PROGRAMMING ASSIGNMENT.

BIT2323.

1a. public class Calculator

{

public static double CalculateAverage(int[] numbers)

{

if (numbers.Length == 0)

return 0;

int sum = 0;

foreach (int number in numbers)

{

sum += number;

}

return (double)sum / numbers.Length;

}

}

b. Constructors in Class Instantiation

Role of Constructors:

Constructors are special methods used to initialize objects. They differ from regular methods in that they have no return type and their name matches the class name.

public class Car

{

public string Model { get; set; }

public int Year { get; set; }

// Default constructor

public Car()

{

Model = "Unknown";

Year = 0;

}

// Overloaded constructor

public Car(string model, int year)

{

Model = model;

Year = year;

}

}

Explanation:

The default constructor initializes Model to "Unknown" and Year to 0.

The overloaded constructor allows setting both properties at initialization.

c. Employee Management System

public class Employee

{

public string Name { get; set; }

public int ID { get; set; }

public string Department { get; set; }

public double Salary { get; set; }

// Constructor

public Employee(string name, int id)

{

Name = name;

ID = id;

}

// Overloaded constructor with optional parameters

public Employee(string name, int id, string department = "Unknown", double salary = 0)

{

Name = name;

ID = id;

Department = department;

Salary = salary;

}

}

Creating Instances:

Employee emp1 = new Employee("John Doe", 123);

Employee emp2 = new Employee("Jane Smith", 456, "HR", 50000);

2. Comparing String Inputs

Explanation

== Operator: Compares references for objects. For strings, it compares values.

Equals() Method: Compares values, considering the string content.

Code Output and Explanation

string str1 = "Hello";

string str2 = "Hello";

string str3 = new string(new char[] { 'H', 'e', 'l', 'l', 'o' });

Console.WriteLine(str1 == str2); // True: Both refer to the same string instance

Console.WriteLine(str1 == str3); // True: Strings have the same content

Console.WriteLine(str1.Equals(str3)); // True: Content comparison

3. .NET Framework Components

CLR and BCL

CLR (Common Language Runtime): Manages the execution of .NET programs, providing services like garbage collection, exception handling, and security.

BCL (Base Class Library): Provides fundamental classes and types used in .NET applications, such as collections, file handling, and basic types.

File Operations

using System;

using System.IO;

class LibraryManager

{

static void Main()

{

string filePath = "books.txt";

// Write to file

File.WriteAllLines(filePath, new string[] { "Book1", "Book2", "Book3" });

// Read from file

string[] lines = File.ReadAllLines(filePath);

foreach (string line in lines)

{

Console.WriteLine(line);

}

}

}

4. Value Types vs. Reference Types

Explanation

Value Types: Store data directly. Examples: int, float, bool.

Reference Types: Store references to data. Examples: string, arrays, class instances.

Program Demonstrating Types:

using System;

class Program

{

static void Main()

{

int x = 10;

int y = x;

y = 20;

Console.WriteLine(x); // 10

string[] arr1 = { "a", "b", "c" };

string[] arr2 = arr1;

arr2[0] = "z";

Console.WriteLine(arr1[0]); // "z"

}

}

5. Encapsulation

Explanation

Encapsulation hides internal state and requires all interaction to be performed through an object's methods or properties.

Person Class Example

public class Person

{

private string name;

private int age;

public string Name

{

get { return name; }

set { name = value; }

}

public int Age

{

get { return age; }

set

{

if (value < 0)

throw new ArgumentException("Age cannot be negative.");

age = value;

}

}

}

6. Arrays and Enums

Single-Dimensional vs. Jagged Arrays

Single-Dimensional Array: A basic array with a single index.

Jagged Array: An array of arrays, where each sub-array can have different lengths.

Two-Dimensional Array Sum:

public class ArrayOperations

{

public static int SumTwoDimensionalArray(int[,] array)

{

int sum = 0;

for (int i = 0; i < array.GetLength(0); i++)

{

for (int j = 0; j < array.GetLength(1); j++)

{

sum += array[i, j];

}

}

return sum;

}

}

Enum Example:

public enum Color

{

Red,

Green,

Blue

}

public class Shape

{

public class Circle

{

public Color ShapeColor { get; set; }

}

}

7. Exception Handling

Explanation

try: Block where exceptions are expected.

catch: Block to handle exceptions.

finally: Block that executes regardless of an exception.

Exception Handling Program:

public class ListManager

{

static void Main()

{

int[] numbers = { 1, 2, 3 };

try

{

try

{

Console.WriteLine(numbers[5]);

}

catch (IndexOutOfRangeException ex)

{

Console.WriteLine("Index out of range: " + ex.Message);

}

}

catch (Exception ex)

{

Console.WriteLine("General exception: " + ex.Message);

}

finally

{

Console.WriteLine("Execution completed.");

}

}

}

8. Conditional Statements and Loops

Check Number Type

public class NumberCheck

{

static void Main()

{

int number = int.Parse(Console.ReadLine());

if (number > 0)

Console.WriteLine("Positive");

else if (number < 0)

Console.WriteLine("Negative");

else

Console.WriteLine("Zero");

}

}

// while loop

int i = 0;

while (i < 5)

{

Console.WriteLine(i);

i++;

}

// do-while loop

int j = 0;

do

{

Console.WriteLine(j);

j++;

} while (j < 5);

// for loop

for (int k = 0; k < 5; k++)

{

Console.WriteLine(k);

}

Factorial Calculation

csharp

public class FactorialCalculator

{

public static void Main()

{

for (int i = 1; i <= 10; i += 2) // Only odd numbers

{

int factorial = 1;

for (int j = 1; j <= i; j++)

{

factorial \*= j;

}

Console.WriteLine($"Factorial of {i} is {factorial}");

}

}

}

Asterisk Pattern

public class PatternPrinter

{

public static void Main()

{

// Right-angled triangle

for (int i = 1; i <= 5; i++)

{

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

{

Console.Write("\*");

}

Console.WriteLine();

}

// Inverted triangle

for (int i = 5; i > 0; i--)

{

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

{

Console.Write("\*");

}

Console.WriteLine();

}

}

}

9. Threads and Tasks

Thread Class Example

using System;

using System.Threading;

public class ThreadExample

{

static void Main()

{

Thread newThread = new Thread(() =>

{

for (int i = 0; i < 5; i++)

{

Console.WriteLine("New thread: " + i);

Thread.Sleep(1000);

}

});

newThread.Start();

newThread.Join(); // Wait for the thread to finish

Console.WriteLine("Main thread finished.");

}

}

10. HTTP Client and File Operations

HttpClient

using System;

using System.Net.Http;

using System.Threading.Tasks;

public