LONDON
METROPOLITAN
UNIVERSITY

ITAHARI
INTERNATIONAL
COLLEGE

Module Code & Module Title

CC6004 – Application Development

Assessment Type

Workshop 4

Semester

2025/26 Autumn

Student Name: Abdul Razzaq Ansari

London Met ID: 23049149

College ID: NP05CP4A230196

Assignment Due Date: Tuesday, November 18, 2025

Assignment Submission Date: Tuesday, November 18, 2025

Submitted To: Mr, Kushal Tamang

Word Count:

Task 7: Research Activity

Constructors

Constructors play an important role in software development. Being a fundamental part of **OOP**, constructors ensure objects start in a valid state, improve reliability, safety, and maintainability. (ScienceDirect, 2025)

A constructor is automatically called when an **object is initialized**. The goal of this as mentioned before is to make sure the object starts with valid initial values. This helps reinforce data integrity and reduces error caused by uninitialized field. (Microsoft, 2025)

Using constructor also helps improve **code reliability** as it prevents half initialized or unsafe objects, which leads to subtle bugs that can be very annoying as a programmer. (Bloch, 2018)

Constructors also improve maintainability as good constructors allow for **overloading, chaining** and having the benefit of **single initialization point** that helps to update easily when the project evolves. (Microsoft, 2025)

The different real world use cases where constructors are utilized are provided below:

i. **Creating Secure User Accounts**
   Using Constructor, apps and websites force users to enter valid data to prevent corrupt user accounts. Example is provided below:

```
namespace Task_7;

public class User
{
    public string Email { get; set; }
    public string PasswordHash { get; set; }
    public DateTime CreatedAt { get; set; }

    // ✔ Correct Constructor
    public User(string email, string password)
    {
        Email = ValidateEmail(email);
        PasswordHash = Hash(password);
        CreatedAt = DateTime.Now;
    }

    private string ValidateEmail(string email)
    {
        // Simple example validation
        return email.Contains("@") ? email : "Invalid email";
    }

    private string Hash(string password)
    {
        // Fake hashing for example
        return "HASHED_" + password;
    }
}
```

ii. Connecting to Database

Constructors are also utilized to connect to database as they load all entities associated with the database. They throw an error if anything is missing and helps establish a safe connection. An example of this has been shown below:

```
namespace Task_7;

public class DBConnection
{
    public string ConnectionString { get; }
    public bool IsConnected { get; }

    public DBConnection(string conn)
    {
        ConnectionString = conn;
        IsConnected = ConnectToDatabase();
    }

    private bool ConnectToDatabase()
    {
        // Fake connection simulation
        Console.WriteLine("Connecting to database...");
        return true;
    }
}
```

iii. Game Engines

Games need to utilize constructors heavily due to multiple initializations done throughout it. Without using it, some bugs like enemies with no health spawning, cars not moving, guns without ammo, etc. might occur. A basic example of this has been provided below:

```
namespace Task_7
{

}
public class Game
{
    public string Name { get; }
    public int Health { get; private set; }
    public int Level { get; private set; }

    public Game(string name, int level, int health)
    {
        Name = name;
        Level = level;
        Health = health;
    }
}
```

OOP Principle (Inheritance)

Inheritance is one of the major principles in C# that allows user to create new classes that extend, reuse and modify the behaviour defined in other classes. The class whose behaviours are inherited is called the **base class** and the class that inherits those behaviours is called the **derived class**. (Microsoft, 2025) Inheritance promotes reusability, which in turn reduces code repetition making the program easier to maintain.

Classes and Objects

To understand, class is simply the **blueprint/template** whereas object is the real **instance** created from the class. An example of class with object has been provided below:

```
using System;

public class Car
{
    // Properties (variables inside the class)
    public string Brand;
    public int Year;

    // Method inside the class
    public void DisplayInfo()
    {
        Console.WriteLine("Car Brand: " + Brand);
        Console.WriteLine("Manufacturing Year: " + Year);
    }
}

public class Program
{
    public static void Main(string[] args)
    {
        // Creating an object of Car class
        Car myCar = new Car();

        // Assign values to object properties
        myCar.Brand = "Toyota";
        myCar.Year = 2020;

        // Calling the method
        myCar.DisplayInfo();
    }
}
```

Bibliography

Bloch, J., 2018. *Effective Java.* 3rd Edition ed. s.l.: Pearson Education Inc..

Microsoft, 2025. *Inheritance - derive types to create more specialized behavior.* [Online]
Available at: https://learn.microsoft.com/en-us/dotnet/csharp/fundamentals/object-oriented/inheritance
[Accessed 18 November 2025].

Microsoft, 2025. *Use constructors (C# programming guide).* [Online]
Available at: https://learn.microsoft.com/en-gb/dotnet/csharp/programming-guide/classes-and-structs/using-constructors
[Accessed 18 November 2025].

ScienceDirect, 2025. *Constructors.* [Online]
Available at: https://www.sciencedirect.com/topics/computer-science/constructors
[Accessed 18 November 2025].