Task 7

A **constructor** is a special method in a class that is automatically called when an object of the class is created. Its main purpose is to initialize the object's state by setting up initial values for its fields or performing any setup if required.

How Constructors Contribute to Software Development

a. Object Initialization

Constructors ensure that objects are created in a valid state. Without constructors, developers would have to manually initialize each property, which increases the chance of errors.

```
class BankAccount
{
   public string AccountNumber;
   public double Balance;

   public BankAccount(string accNum, double initialBalance)
   {
      AccountNumber = accNum;
      Balance = initialBalance;
   }
}

var account = new BankAccount("12345", 1000.0);
```

Here, the BankAccount constructor guarantees that every account has a number and a starting balance when it is created.

b. Code Reliability

Constructors help enforce consistency and rules in object creation. By controlling how objects are initialized, you can prevent invalid data and reduce runtime errors.

```
class User
{
   public string Username { get; }
   public string Email { get; }

   public User(string username, string email)
   {
      if (string.IsNullOrEmpty(username) ||
string.IsNullOrEmpty(email))
         throw new ArgumentException("Username and email must
be provided");
      Username = username;
      Email = email;
   }
}
```

This ensures all User objects are valid from the start, improving reliability.

c. Maintainability

Using constructors centralizes initialization logic. If the rules for object creation change, you only need to update the constructor rather than multiple places in the code.

```
class Rectangle
{
   public double Width { get; set; }
   public double Height { get; set; }

   public Rectangle(double width, double height)
   {
      Width = width;
      Height = height;
   }
}
```

Later, if a minimum width/height restriction is required, it can be added inside the constructor, and all object creations will automatically comply.

3. Real-World Use Cases of Constructors

Database Connections

```
class DatabaseConnection
{
   public string ConnectionString { get; }
   public DatabaseConnection(string connStr)
   {
      ConnectionString = connStr;
      // Initialize connection here
   }
}
```

Constructors ensure every connection is properly initialized before use.

E-commerce Shopping Cart

```
class ShoppingCart
{
   public List<string> Items { get; } = new List<string>();
   public ShoppingCart(List<string> initialItems)
   {
      Items = initialItems;
   }
}
```

This avoids null lists and ensures a cart is ready to use when created.

Game Characters in Video Games

```
class Player
{
   public string Name { get; }
   public int Health { get; set; }

   public Player(string name, int health)
   {
      Name = name;
      Health = health;
   }
}
```

Constructors initialize characters with starting health and names, preventing invalid gameplay scenarios.

OOP Principles: Encapsulation

Encapsulation is the concept of restricting direct access to some of an object's components and exposing only necessary functionality. This protects the internal state and allows controlled modification through methods.

Class: A blueprint for creating objects. It defines properties, fields, and methods.

Object: An instance of a class, representing real-world entities.·

```
class BankAccount
{
    private double balance; // hidden from outside
    public void Deposit(double amount)
    {
        if (amount > 0) balance += amount;
    }
    public double GetBalance() => balance;
}
```

Use Cases of OOP Principles:

Encapsulation in Banking Software: Protects account balances from unauthorized modifications.

Encapsulation in E-commerce: Restricts direct modification of product inventory, enforcing proper business rules.

References

Microsoft docs - https://learn.microsoft.com/en-us/dotnet/csharp/programming-guide/classes-and-structs/constructors

Geeks for geeks - https://www.geeksforgeeks.org/c-sharp/encapsulation-in-c-sharp/

programiz - https://www.programiz.com/csharp-programming/constructors