Day 55/60 - Understanding Lazy Initialization in C#

Lazy initialization helps delay the creation of an object until it is actually needed. This can improve performance and memory usage in C# applications.

1. Why Use Lazy Initialization?

- Improves Performance: Avoids unnecessary object creation.
- Saves Memory: Only loads objects when required.
- Thread-Safe: Ensures single initialization in multi-threaded environments.

2. Example: Using Lazy<T> in C#

```
public class ExpensiveObject
{
 public ExpensiveObject()
 {
 Console.WriteLine("Expensive object created");
 }
}

public class Program
{
 static Lazy<ExpensiveObject> _lazyObject = new Lazy<ExpensiveObject>();

 static void Main()
 {
 Console.WriteLine("Program started");

 // Object is created only when accessed
 var obj = _lazyObject.Value;

 Console.WriteLine("Object accessed");
 }
}
```

3. Key Takeaways

- Lazy<T> ensures that an object is only created when needed.
- Avoids unnecessary memory allocation and improves startup time.
- Works well in multi-threaded applications.

Lazy initialization is useful for expensive resources, large datasets, or infrequently used objects.