C# and .NET Containers and Constructors

1. Types of Containers in C# and .NET

In C# and .NET, containers refer to structures that manage dependencies, hold collections, or isolate environments.

1.1 Dependency Injection (DI) Container

Manages service lifetimes and injections.

Types:

- **Transient** – New object per call.
- **Scoped** – One per HTTP request.
- **Singleton** – One for the whole app.

Usage: Used in ASP.NET Core for injecting services.

1.2 IoC Container

Implements the Inversion of Control pattern. Automatically handles object creation and dependencies.

Usage: Frameworks like Autofac, Unity use this.

1.3 Collection Containers (C# Data Containers)

Store and manage data:

- `List<T>`, `Array`, `Dictionary<TKey, TValue>`, `Queue<T>`, `Stack<T>`, `HashSet<T>`

Usage: Used to handle data in memory.

1.4 Docker Containers

Package apps and run them in isolated environments. Cross-platform and lightweight.

Usage: Used in deployment and DevOps.

2. Types of Constructors in C#

A constructor initializes a class object. It runs when the object is created.

2.1 Default Constructor

No parameters. Sets default values.

```
public Student() { Name = ”Unknown”; }
```

2.2 Parameterized Constructor

Takes arguments to assign values.

```
public Student(string name) { Name = name; }
```

2.3 Copy Constructor

Copies values from another object.

```
public Student(Student s) { Name = s.Name; }
```

2.4 Static Constructor

Used to initialize static members. Runs only once.

```
static Student() { School = ”ABC”; }
```

2.5 Private Constructor

Restricts object creation. Used in Singleton Pattern.

```
private Student() { }
```