Constructors in C# - Detailed Notes

1. What is a Constructor?

- A constructor is a special method used to initialize objects.

- Has the same name as the class and no return type.

- Automatically invoked when an object is created.

2. Types of Constructors in C#:

A. Default Constructor:

- No parameters.

- Automatically provided by compiler if not defined.

Example:

```
class MyClass {
    public MyClass() {
        Console.WriteLine("Default constructor");
    }
}
```

B. Parameterized Constructor:

- Accepts arguments to initialize the object.

Example:

```
class MyClass {
    int x;
    public MyClass(int value) {
        x = value;
```

```
    }
}
```

C. Copy Constructor (manual):

- Initializes an object using another object.

Example:

```
class MyClass {
    int x;
    public MyClass(MyClass obj) {
        x = obj.x;
    }
}
```

D. Static Constructor:

- Used to initialize static data.
- Runs once before any instance or static member is accessed.

Example:

```
class MyClass {
    static MyClass() {
        Console.WriteLine("Static constructor");
    }
}
```

E. Private Constructor:

- Restricts instance creation from outside the class.

Example:

```
class Singleton {
```

```
    private Singleton() { }
    public static Singleton instance = new Singleton();
}
```

3. Constructor Overloading:

- Multiple constructors with different signatures.

Example:

```
class MyClass {
    public MyClass() {}
    public MyClass(int x) {}
    public MyClass(string name, int age) {}
}
```

4. Keywords: this() and base()

- this(): Calls another constructor in the same class.
- base(): Calls constructor of base class.

Example:

```
class Base {
    public Base(int a) {
        Console.WriteLine("Base constructor: " + a);
    }
}

class Derived : Base {
    public Derived() : base(10) {
```

```
        Console.WriteLine("Derived constructor");
    }
}
```