Types of Constructor & Whom to use When?

Primary Constructor

The **primary constructor** is a concise way to define and initialize properties directly when the class is instantiated. It is part of the class header and can either take arguments or be empty.

Key Features:

1. **Declared in the class header**: The primary constructor is written right after the class name.
2. **Automatic initialization**: You can declare properties directly in the constructor using `val` or `var`, and they are automatically initialized.
3. **Initializer block**: The `init` block runs any additional logic during object creation, after the primary constructor.

Example:

```
class Person(val name: String, var age: Int) {

    init {
        println("A person is created with name: $name and age: $age")
    }

}
```

In this example:

- `name` and `age` are initialized directly in the constructor.
- The `init` block runs automatically when an instance is created.

When to Use:

- Use the **primary constructor** when the class properties can be initialized directly with the constructor parameters, and no complex initialization logic is required.
- It's a good choice for most cases because it results in cleaner, more concise code.

Secondary Constructor

A **secondary constructor** is an alternative constructor that can be defined within the class body, allowing additional ways to instantiate a class. Secondary constructors are useful if:

- You need to perform more complex initialization logic.
- You want to offer multiple ways to instantiate the class with different sets of parameters.
- The primary constructor is insufficient for certain scenarios.

Key Features:

1. **Optional**: A class can have multiple secondary constructors or none at all.
2. **Calls the primary constructor**: If the class has a primary constructor, all secondary constructors must either directly or indirectly delegate to it using the `this()` keyword.
3. **Complex initialization**: Useful when different initialization logic is required for different constructors.

Example:

```
class Person {

    var name: String
    var age: Int

    // Secondary constructor #1
    constructor(name: String, age: Int) {

        this.name = name
        this.age = age
    }

    // Secondary constructor #2
    constructor() : this("Unknown", 0) {
        println("Default constructor called")
    }
}
```

In this case:

- The primary constructor takes `name` and `age`.
- The secondary constructor provides a default way to initialize the `Person` class, calling the primary constructor with default values.

When to Use:

- Use a **secondary constructor** when you need flexibility and different initialization pathways.
- Useful if there is complex initialization logic or you need multiple ways to instantiate the class with different parameter sets.