Constructors - Notes with Examples

1. Introduction to Constructors

A constructor in Java is a special method used to initialize objects. It has the same name as the class and does not have a return type.

Key Characteristics:

- - Called automatically when an object is created.
  - Used to initialize object state.
  - Can be default, no-args, or parameterized.

2. No-Args Constructor

A constructor with no parameters. If no constructor is explicitly defined, Java provides a default no-args constructor.

```
public class Student {
    String name;
    int age;

    Student() {
        name = "Unknown";
        age = 0;
    }
}
```

3. Parameterized Constructor

A constructor that takes one or more parameters to initialize object values.

```
public class Student {
    String name;
    int age;

    Student(String n, int a) {
        name = n;
        age = a;
    }
}
```

4. Constructor Overloading

Multiple constructors can be defined in a class, each with a different parameter list.

```
public class Student {
    String name;
    int age;
```

```
    Student() {
        name = "Unknown";
        age = 0;
    }

    Student(String n) {
        name = n;
        age = 18;
    }

    Student(String n, int a) {
        name = n;
        age = a;
    }
}
```

5. Constructor Chaining

You can call one constructor from another using this().

```
public class Student {
    String name;
    int age;

    Student() {
        this("Unknown", 0);
    }

    Student(String n) {
        this(n, 18);
    }

    Student(String n, int a) {
        name = n;
        age = a;
    }
}
```

6. How Many Constructors Are Needed?

It depends on the flexibility required. Here are common use cases:

- - Simple object: No-args constructor
  - Custom initialization: Parameterized constructors
  - Flexible creation: Multiple overloaded constructors

7. Practical Examples

Example 1: Book Class with Overloaded Constructors

```
public class Book {
    String title;
    String author;
    double price;

    Book() {
        this("Unknown Title", "Unknown Author", 0.0);
    }

    Book(String title) {
        this(title, "Unknown Author", 0.0);
    }

    Book(String title, String author) {
        this(title, author, 0.0);
    }

    Book(String title, String author, double price) {
        this.title = title;
        this.author = author;
        this.price = price;
    }

    void displayInfo() {
        System.out.println("Title: " + title + ", Author: " + author
+ ", Price: $" + price);
    }
}
```

Usage

```
public class Main {
    public static void main(String[] args) {
        Book b1 = new Book();
        Book b2 = new Book("Java Basics");
        Book b3 = new Book("OOP Concepts", "Alice");
        Book b4 = new Book("Advanced Java", "Bob", 19.99);

        b1.displayInfo();
        b2.displayInfo();
        b3.displayInfo();
        b4.displayInfo();
    }
}
```

Example 2: Vehicle Class

```
public class Vehicle {
    String type;
    int wheels;
```

```
    Vehicle() {
        this("Unknown", 0);
    }

    Vehicle(String type, int wheels) {
        this.type = type;
        this.wheels = wheels;
    }

    void describe() {
        System.out.println("Vehicle Type: " + type + ", Wheels: " +
wheels);
    }
}
```

Test

```
public class TestVehicle {
    public static void main(String[] args) {
        Vehicle v1 = new Vehicle();
        Vehicle v2 = new Vehicle("Car", 4);
        Vehicle v3 = new Vehicle("Bike", 2);

        v1.describe();
        v2.describe();
        v3.describe();
    }
}
```

Example 3: Student Class with Optional Fields

```
public class Student {
    String name;
    int age;
    String department;

    Student() {
        this("Unknown", 18, "Undeclared");
    }

    Student(String name) {
        this(name, 18, "Undeclared");
    }

    Student(String name, int age) {
        this(name, age, "Undeclared");
    }

    Student(String name, int age, String department) {
        this.name = name;
        this.age = age;
        this.department = department;
    }

    void info() {
```

```
        System.out.println(name + ", Age: " + age + ", Dept: " +
department);
    }
}
```

Test

```
public class TestStudent {
    public static void main(String[] args) {
        Student s1 = new Student();
        Student s2 = new Student("Rafi");
        Student s3 = new Student("Maya", 22);
        Student s4 = new Student("Arif", 20, "CSE");

        s1.info();
        s2.info();
        s3.info();
        s4.info();
    }
}
```

Example 4: BankAccount Class

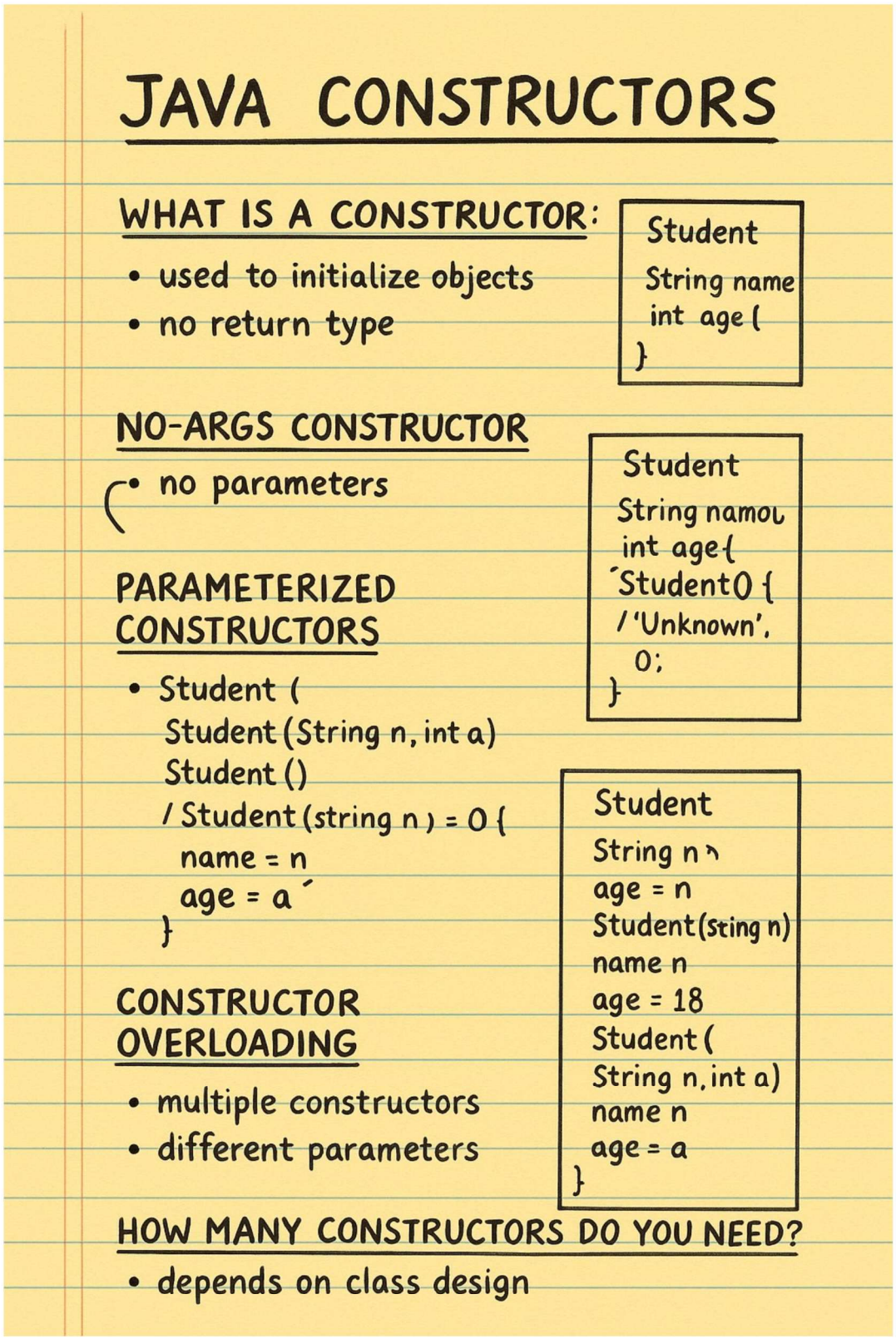
```
public class BankAccount {
    String accountHolder;
    double balance;

    BankAccount() {
        this("Unknown", 0.0);
    }

    BankAccount(String accountHolder, double balance) {
        this.accountHolder = accountHolder;
        this.balance = balance;
    }

    void deposit(double amount) {
        balance += amount;
    }

    void display() {
        System.out.println("Account Holder: " + accountHolder + ",
Balance: $" + balance);
    }
}
```

Test

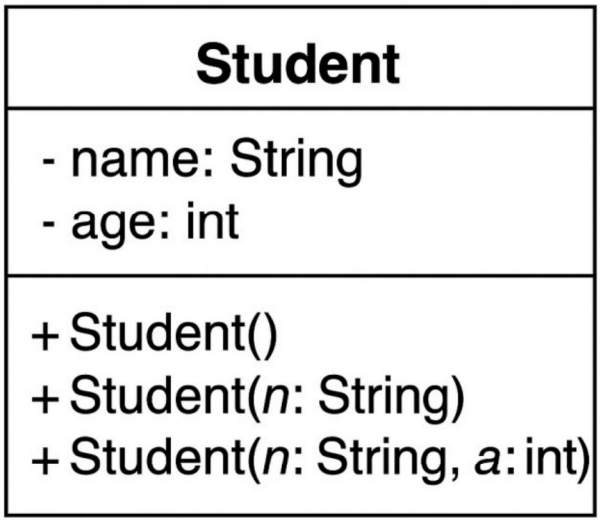
```
public class TestBank {
    public static void main(String[] args) {
        BankAccount a1 = new BankAccount();
        BankAccount a2 = new BankAccount("Sara", 1000);

        a1.deposit(500);
        a2.deposit(150);
```

```
        a1.display();
        a2.display();
    }
}
```

Diagram: Java Constructors Overview



UML Diagram: Student Class with Constructors



Quiz: Java Constructors

1. What is a constructor in Java and how is it different from a method?
2. What will happen if no constructor is defined in a Java class?
3. Explain constructor overloading with an example.
4. What is the use of the ***"this()"*** keyword in constructor chaining?
5. Write a class `Employee` with at least two overloaded constructors.
6. How many constructors can a Java class have?
7. Is it possible to overload constructors with the same number of parameters but different types? Explain.
8. Why might a framework like Spring require a no-args constructor?
9. Create a `Laptop` class with no-args, single-parameter, and full-parameter constructors. Demonstrate overloading.
10. Which constructor is called in the following code and why?
    "Car c = new Car("Toyota");" (Assume multiple constructors exist in Car class)