Scaler Companion beta

Comprehensive Revision Notes: Learning Object-Oriented Programming with Java

---

Introduction to Object-Oriented Programming (OOP)

Why OOP?

Object-Oriented Programming is a paradigm that provides a way of structuring programs so that properties and behaviors are bundled into individual objects. This allows for improved scalability, maintainability, and modularity. In OOP, everything can be modeled as objects, which is akin to real-world entities【6:16†source】.

Basic Concepts: Classes and Objects

- **Class**: A class is a blueprint or template from which objects are created. It defines properties (attributes) and behaviors (methods) .
- **Object**: An object is an instance of a class. It is a real-world entity created based on the blueprint provided by a class .

Analogy: Consider a class as a "cookie cutter," and objects as the "cookies" made from that cutter. The cutter defines the shape, while each cookie is an individual instance of that shape.

Creating Classes and Objects in Java

Syntax for Java Class and Object:

1. **Defining a Class**:

```
class Book {
    String title;
    void read() {
        System.out.println("Reading a book");
    }
}
```

- **Attributes:** `title` , `author` , `pages`
- **Methods:** `read()` 【6:2†source】

2. **Creating an Object:**

```
public class Main {
    public static void main(String[] args) {
        Book javaBook = new Book();
    }
}
```

3. **Setting Object Attributes:**

```
javaBook.title = "Java Programming";
javaBook.author = "Dennis Ritchie";
```

Constructors in Java

Constructors are special methods used to initialize objects. They have the same name as the class and no return type .

Types of Constructors:

1. **Default Constructor:**
   - Automatically provided by Java if no constructor is defined.
   - Initializes object attributes with default values:
     - `null` for strings, `0` for integers, `0.0` for doubles .

```
class Student {
    String name;
    int age;
    double psp;
    String email;
}
```

2. **Parameterized Constructor:**

```
public Student(String name, int age, double psp, String email)
    this.name = name;
    this.age = age;
    this.psp = psp;
    this.email = email;
}
```

3. **Manual Constructor** (Non-Parameterized):
    - Explicitly defined by the programmer.
    - If a manual constructor is defined, Java will not provide a default constructor .

Important Keywords

- `this` **Keyword in Java**:
    - Used to distinguish between object attributes and parameter variables with the same name.
    - Enhances clarity and avoids naming conflicts .

Memory Management in Java

- **Stack Memory**: Used for static memory allocation and execution of threads.
- **Heap Memory**: Used for dynamic memory allocation, where objects are stored .

Process:

1. Creation of an object involves allocation of memory on the heap, identified by an address.
2. Java's `new` keyword triggers this allocation, and the constructor initializes the object .

Deep Copy vs Shallow Copy

- **Shallow Copy**:

- **Deep Copy**:
  - Creates a new instance with the same values, ensuring independence from the original .

Example:

- Shallow copy can be illustrated by `Student ST3 = ST1;` , which only copies the reference .

These notes provide a comprehensive overview of the foundational aspects of Object-Oriented Programming as taught in the lecture with a focus on Java; exploring the nature of classes, objects, and constructors, alongside how memory and copying work in an OOP context.