**面向对象编程 (OOP)** 是一种基于对象和类概念的编程范式。OOP 的主要概念包括：

1. **对象**：类的实例，包含数据（字段）和用于处理这些数据的函数（方法）。
2. **继承**：一种机制，允许基于现有类创建新类，继承其属性和行为。
3. **多态**：不同类的对象可以对相同方法做出不同响应的能力。
4. **封装**：将对象的内部实现隐藏起来，通过公共方法提供访问。

**类的概念。Java 中的类和对象。**

**类** 是一个模板或蓝图，用于创建对象。类定义对象的特性（字段）和行为（方法）。  
类的示例：

java

复制代码

class Pokemon {

String name;

int level;

Pokemon(String name, int level) {

this.name = name;

this.level = level;

}

void attack() {

System.out.println(name + " 攻击！");

}

}

**对象** 是类的实例，通过 new 操作符创建：

java

复制代码

Pokemon pikachu = new Pokemon("皮卡丘", 5);

pikachu.attack();

**类的成员：字段、方法、构造函数**

* **字段**：类中的变量，用于存储对象的数据。
* **方法**：描述对象行为的函数。
* **构造函数**：用于初始化对象的特殊方法。

构造函数的示例：

java

复制代码

Pokemon(String name, int level) {

this.name = name;

this.level = level;

}

**访问修饰符**

访问修饰符决定了类成员的可见性：

* public — 在任何地方可访问。
* private — 仅在类内部可访问。
* protected — 在类、包及其子类中可访问。
* 默认（包访问权限）— 仅在包内部可访问。

示例：

java

复制代码

class Pokemon {

private String name; // 仅在类内部可访问

protected int level; // 在子类和包内可访问

}

**创建和初始化对象。方法调用。**

对象通过关键字 new 创建，其方法通过点号调用：

java

复制代码

Pokemon bulbasaur = new Pokemon("妙蛙种子", 10);

bulbasaur.attack();

**变量的作用域**

* **局部变量**：仅在代码块或方法内部可用。
* **类字段**：在整个类中都可访问。

示例：

java

复制代码

class Example {

int field; // 类字段

void method() {

int localVariable = 10; // 局部变量

}

}

**final 和 static 修饰符**

* final — 被该修饰符修饰的变量在初始化后不可更改。final 修饰的方法和类不可被重写或继承。

示例：

java

复制代码

final int MAX\_HP = 100; // 常量

* static — 静态字段和方法属于类，而不属于类的实例。它们可以在不创建对象的情况下调用。

示例：

java

复制代码

class MathUtil {

static int square(int num) {

return num \* num;

}

}

int result = MathUtil.square(5);

**包、import 语句**

**包** 是一种将类进行分组的方式，帮助组织代码并避免名称冲突。

包声明的示例：

java

复制代码

package com.example.pokemon;

**import 语句** 用于从其他包中导入类：

java

复制代码

import com.example.pokemon.Pokemon;

包创建了程序的逻辑结构，并有助于提高模块化。