Java抽象类

Java抽象类概念

Java中抽象类是指用abstract关键字修饰的类，它不能被实例化，只能被继承。抽象类通常用于定义一些公共的方法和属性，但是这些方法没有具体的实现。子类必须重写抽象类中的所有抽象方法才能被实例化。

抽象类是Java中的一个重要的概念，它可以用来定义一个抽象的模板，由子类去实现具体的细节。同时，抽象类也可以提高程序的可扩展性和可维护性。

Java抽象类示例

Animal类是一个抽象类，它有一个私有的name属性和一个构造方法来初始化这个属性。另外，Animal类中还有一个非抽象的方法getName()，它可以被所有的子类继承并使用。而eat()方法是一个抽象方法，没有具体的实现，需要子类去实现。因为这个抽象方法在Animal类中声明了，所以所有的子类都必须重写这个方法。

```java
public abstract class Animal {
    private String name;

    public Animal(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }

    public abstract void eat();
}
```

继承Animal类的子类的示例

Cat类继承了Animal类，并实现了eat()方法。注意，在Cat类中，必须实现eat()方法，否则编译器会报错。

```java
public class Cat extends Animal {
    public Cat(String name) {
        super(name);
    }

    public void eat() {
        System.out.println(getName() + " likes to eat fish.");
    }
}
```

Java抽象类详细使用方法

1、定义抽象类

在Java中，定义抽象类使用abstract修饰符，然后给出类名和花括号的内容，其中花括号里包含了抽象方法和非抽象方法。

```java
public abstract class AbstractClass {
    // 抽象方法
    public abstract void abstractMethod();

    // 非抽象方法
    public void commonMethod() {
        System.out.println("这是一个公共方法！");
    }
}
```

2、继承抽象类

如果一个类继承了抽象类，那么这个类必须实现抽象类中的全部抽象方法。如果这个类不想实现所有的抽象方法，那就只能声明为抽象类。

```java
public class ConcreteClass extends AbstractClass {
    @Override
    public void abstractMethod() {
        System.out.println("这是抽象方法的具体实现！");
    }
}
```

3、实现抽象方法

在子类中重写抽象方法并进行具体的实现。

```java
public class ConcreteClass extends AbstractClass {
    @Override
    public void abstractMethod() {
        System.out.println("这是抽象方法的具体实现！");
    }
}
```

4、完整示例代码

Java 复制代码

```java
public abstract class AbstractClass {
    // 抽象方法
    public abstract void abstractMethod();

    // 非抽象方法
    public void commonMethod() {
        System.out.println("这是一个公共方法！");
    }
}

public class ConcreteClass extends AbstractClass {
    @Override
    public void abstractMethod() {
        System.out.println("这是抽象方法的具体实现！");
    }
}

public class Main {
    public static void main(String[] args) {
        ConcreteClass concreteClass = new ConcreteClass();
        concreteClass.abstractMethod();
        concreteClass.commonMethod();
    }
}
```