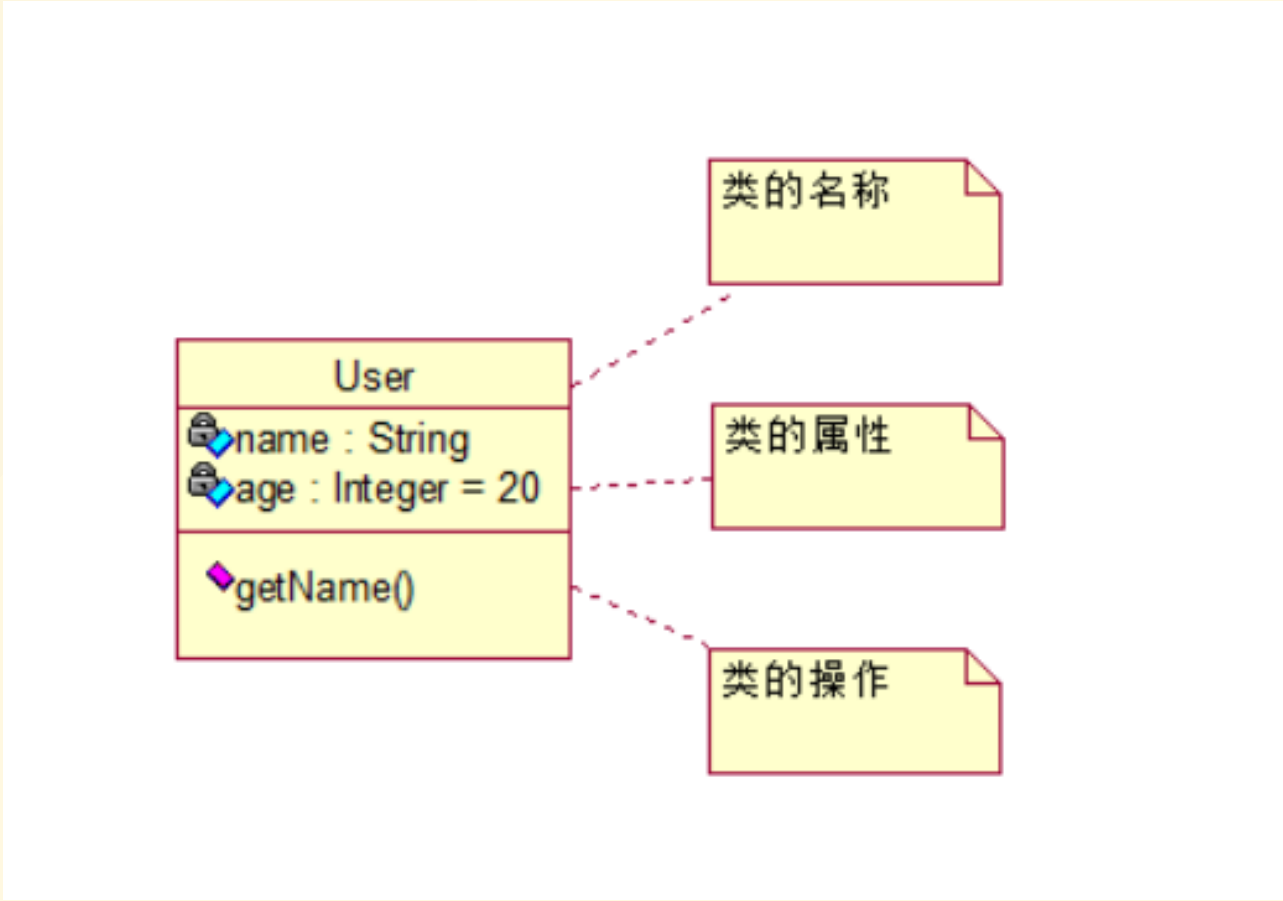
Kotlin中的类

kotlin类

普通类	开放类	抽象类	嵌套类
枚举类	密封类	泛型类	注解类
委托类	内联类	Nothing类	接口



1.普通类与开放类

在Kotlin中，类默认是不可继承的（final），这意味着您必须明确地将类声明为可继承的，才能被其他类继承。在这个方面，Kotlin提供了两种类型的类：普通类和开放类。

1.1 普通类

原代码

```
class MyClass {
    fun myFunction() {
        // ...
    }
}
```

反编译后代码

```
public final class MyClass {
   public final void myFunction() {
      // ...
   }
}
```

如上，类默认是public final的，不可继承

1.2 开放类

```
open class MyBaseClass {
    open fun myFunction() {
        // ...
    }
}

class MyDerivedClass : MyBaseClass() {
    override fun myFunction() {
        // ...
    }
}
```

如上，添加open关键字让类可继承

为什么Kotlin的类默认设计为final?

* 编译时优化，例如内联函数（inline function）等
* 运行时开销，不需要进行继承检查和动态分派
* 开发者设计素养

2.抽象类

抽象类定义

```
abstract class Shape {
    abstract fun draw()
}
```

抽象类具体的实现

```
class Circle : Shape() {
    override fun draw() {
        println("Drawing a circle")
    }
}
```

3.嵌套类与内部类

Kotlin有内部类和嵌套类，但它们不是同一个东西。

- 嵌套类是定义在其他类里面的类，它不能访问外部类的成员。
- 内部类是用inner关键字标记的嵌套类，它能够访问外部类的成员，并且持有一个对外部类的对象的引用

3.1 嵌套类

原代码

```
class Outer {
    private val message: String = "Hello, world!"

    class Nested {
        fun foo() = "Welcome to Kotlin"
    }
}

fun main() {
    val nested = Outer.Nested()
    println(nested.foo())
}
```

嵌套类反编译后代码

```
public final class Outer {
   private final String message = "Hello, world!";

   public static final class Nested {
      @NotNull
      public final String foo() {
         return "Welcome to Kotlin";
      }
   }
}
```

如上，嵌套类是静态内部类，不能访问外部类的成员message。

3.2 内部类

原代码

```
class Outer {
    private val message: String = "Hello, world!"

    inner class Inner {
        fun foo() = message
    }
}

fun main() {
    val outer = Outer()
    val inner = outer.Inner()
    println(inner.foo())
}
```

内部类反编译后代码

```
public final class Outer {
   private final String message = "Hello, world!";

   public final class Inner {
      @NotNull
      public final String foo() {
         return Outer2.this.message;
      }
   }
```

如上，内部类不是静态的，所以要先构造外部类对象，可以访问外部类的成员message。 ---