1.SAM 接口

什么是函数式（SAM）接口？
只有一个抽象方法的接口称为函数式接口或 SAM（单一抽象方法）接口。函数式接口可以有多个非抽象成员，但只能有一个抽象成员。
什么是 SAM 转换？
对于函数式接口，可以通过 lambda 表达式实现 SAM 转换，从而使代码更简洁，可读性更强。
在 Kotlin 1.4 之前不支持实现 SAM 转换。

2.内联函数

在 Kotlin 中通过 inline-functions (内联函数) 实现函数内联，内联的作用：提升运行效率，调用被 inline 修饰符的函数，会把方法体内的代码放到调用的地方，其主要目的提高性能，减少对象的创建。
inline 修饰的函数适用于以下情况：
1、 inline 修饰符适用于把函数作为另一个函数的参数，例如高阶函数 filter 、 map 、joinToString 或者一些独立的函数 repeat。
2. inline 操作符适合和 reified 操作符结合在一起使用。
3. 如果函数体很短，使用 inline 操作符提高效率。
如果使用 inline 修饰符标记普通函数，Android Studio 会给一个大大大的警告，如下图所示，这是为了防止 inline 操作符滥用而带来的性能损失。

3.Inline Class

基本数据类型、String 等等运行时 JVM 会对它进行优化，但是如果将这些参数封装在一个类中，包装类不会做任何处理，依然会在堆中创建对象。
所以为了减少性能的损耗，避免对象的创建，因此 Kotlin 推出了一个内联类 inline-classes。内联类只能在构造函数中传入一个参数，参数需要用 val 声明，编译之后，会替换为传进去的。

Java 如何调用接受内联类的函数？
为了能够在 Java 中正常调用，因此添加了注解 @JvmName 更改函数名称，来解决这个问题。

Inline classes 是在 Kotlin 1.3 引入的 ，在 Kotlin 1.5 时进入了稳定版本，废弃了 inline 修饰符，引入了 Value classes。

4.Value Class

Inline classes 是 Value classes 的子集， Value classes 比 Inline classes 会得到更多优化，现阶段 Value classes 和 Inline classes 一样，只能在构造函数中传入一个参数，参数需要用 val 声明，将来可以在构造函数中添加多个参数，但是每个参数都需要用 val 声明。

因为 value class 编译后将会添加 fianl 修饰符，因此不能被继承，同样也不能继承其他的类。

虽然 value class 不能继承其他的类，但是可以实现接口。
当 value class 实现接口时，失去了内联效果，依然会在堆中创建 User 对象，除了实现接口的时候，没有内联效果，当对象为空的时候，也会失去了内联效果。