不懂，一级标题，二级标题，三级标题，注

Mit 6.031 ----01 08

3.1 Data type and Type checking 数据类型与类型检验

目录

3.1.1 Data type in programming languages

A type is a set of values, along with operations that can be performed on those values

Variables: Named location that stores a value of one particular type

Type In java

1. primitive type（基本数据类型，通常以小写开头）
2. object type（对象数据类型，通常以大写开头）

Primitives	Object Reference Types
`int, long, byte, short, char, float, double, boolean`	Classes, interfaces, arrays, enums, annotations
No identity except their value 只有值，没有ID (与其他值无法区分)	Have identity distinct from value 既有ID，也有值
Immutable 不可变的	Some mutable, some not 可变/不可变
On stack, exist only when in use 在栈中分配内存 不能实现同一的表达式	On heap, garbage collected 在堆中分配内存 可以由泛式实现同一的表达式
Can't achieve unity of expression	Unity of expression with generics
Dirt cheap 代价低	More costly代价昂贵

特殊的对象数据类型：Boxed primitives:

Definition：将基本类型包装为对象类型。

特点：1. 通常在使用容器类型的时候使用，因为容器类型操作的元素要求是对象类型

2. 一般情况下，尽量避免使用（会降低性能）
3. 一般在使用时可以进行自动转换：from primitive types to objective types

Operations:

1. Operations are functions that take inputs and produce outputs (and sometimes change the values themselves).
2. Some operations are overloaded in the sense that the same operation name is used for different types（可以重载：就是一个函数有多种实现，调用哪一个取决于函数的参数）

3.1.2 Static vs. dynamic data type checking

statically-typed language： **verify the types at compile time（在编译时检查类型）**

dynamically-typed languages： **verify the types at runtime（在程序运行时检查类型）**

Static Checking:

Definition： **the bug is found automatically before the program even runs.（静态检查）**

Characteristic（特点）

- Safe from bugs. Static checking helps with safety by catching type errors and other bugs before runtime.
- Easy to understand. It helps with understanding, because types are explicitly stated in the code.
- Ready for change. Static checking makes it easier to change your code by identifying other places that need to change in tandem. For example, when you change the name or type of a variable, the compiler immediately displays errors at all the places where that variable is used, reminding you to update them as well.

Static Checking Types：

Syntax errors 语法错误
Wrong names 类名/函数名错误
Wrong number of arguments 参数数目错误
Wrong argument types 参数类型错误
Wrong return types 返回值类型错误,

Dynamic Checking:

Definition： **the bug is found automatically when the code is executed. （动态检查）**

Dynamic Checking types：

Illegal argument values 非法的参数值
Unrepresentable return values 非法的返回值

Out-of-range indexes 越界
Calling a method on a null object reference. 空指针

No checking:

Definition：the language doesn't help you find the error at all.（无检查）

No Checking types：

Integer division: 5/2
Integer overflow :int 20000 * 20000
Special values in floating-point types:
Eg: NaN , POSITIVE_INFINITY , NEGATIVE_INFINITY

注：int 5 / int 0 会抛出异常；而 double 5.0 / double 0.0 不会抛出异常，因为 double 0.0 不是真正的 0，而是接近于 0;

注：1. 静态检查与动态检查是和静态类型语言与动态类型语言并非是一一对应的关系，后者只取决于说明类型检查所处的时间点，而前者取决于某种检查所处的时间点。
2. 静态检查：关于"类型"的检查，不考虑值；动态检查：关于"值"的检查

3.1.3 Mutability and Immutability

the distinction between changing a variable and changing a value

When you assign to a variable, you're changing where the variable's arrow points. You can point it to a different value. （改变一个变量：将该变量指向另一个值的存储空间）

When you assign to the contents of a mutable value – such as an array or list – you're changing references inside that value. （改变一个变量的值：将该变量当前指向的值的存储空间中写入一个新的值。）

Immutability 不变性

Definition（定义）

Immutable types are types whose values can never change once they have been created. 数据类型：一旦被创建，其值不能改变

Java also gives us immutable references: variables that are assigned once and never reassigned. 如果是引用类型，也可以是不变的：一旦确定其指向的对象，不能再被改变指向其他对象

Characteristic（特点）

The answer is that immutable types are safer from bugs, easier to understand, and more ready for change. 不可变类型更“安全”， 在其他质量指标上表现更好。

Mutability 可变性

Definition（定义）

值和引用都可以进行修改

Characteristic（特点）

Using immutable strings, this makes a lot of temporary copies（使用不可变类型，对其频繁修改会产生大量的临时拷贝（需要垃圾回收））

StringBuilder is designed to minimize this copying. （可变类型最少化拷贝以提高效率）

Getting good performance is one reason why we use mutable objects. （使用可变数据类型，可获得更好的性能）

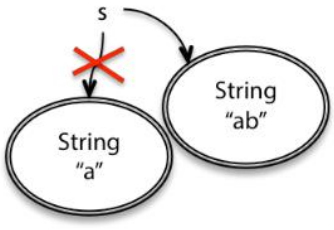
Another is convenient sharing: two parts of your program can communicate more conveniently by sharing a common mutable data structure. （也适合于在多个模块之间共享数据）

注：final 的特点：

1. A final class declaration means it cannot be inherited. （final 类无法派生子类）
2. A final variable means it always contains the same value/reference but cannot be changed （final 变量无法改变值/引用）
3. A final method means it cannot be overridden by subclasses （final 方法无法被子类重写）

注：对例子的解析：

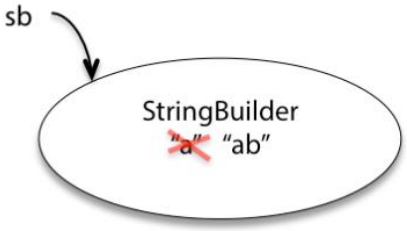
```
String s = "a";
s = s.concat("b"); // s+="b" and s=s+"b" also mean the same thing
```



快照

Note: this is a snapshot dia

```
StringBuilder sb = new StringBuilder("a");
sb.append("b");
```

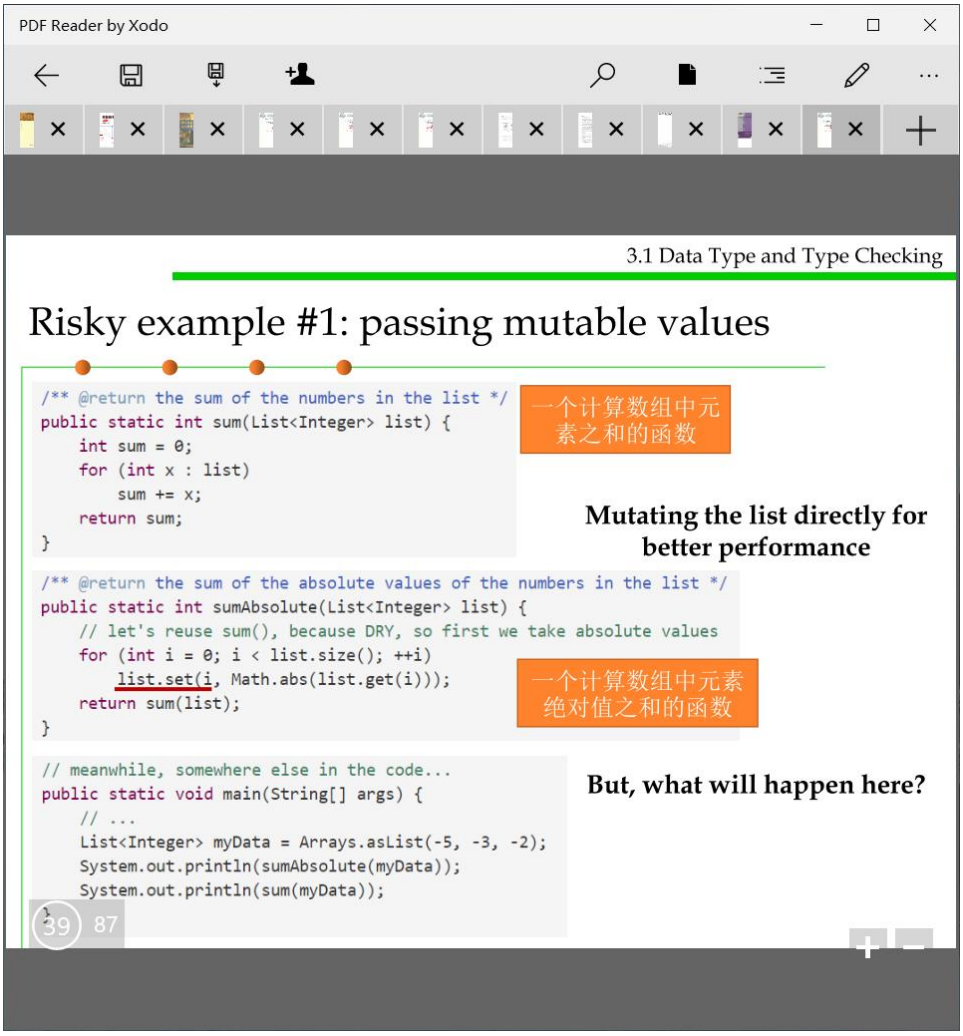


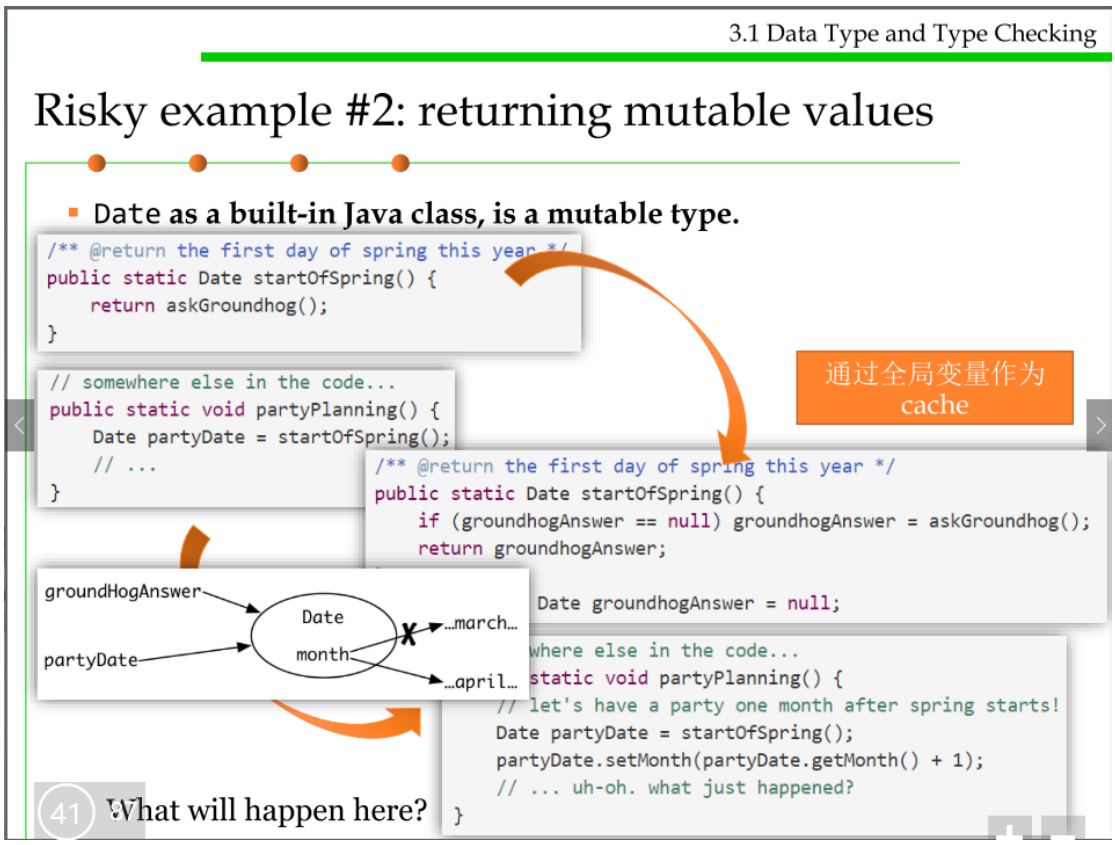
需要指出的是：在 java 数据类型中，所有的变量（a variable）均是引用类型，其引用一个值，而该引用的值可以是不可变值类型，即例子中的 String，也可以是可变数据类型，即例子中的 StringBulider。

对于可变性与不可变性两者安全性差异的分析（reason）

主要存在于不可变值与可变值的多个引用上。

Eg：

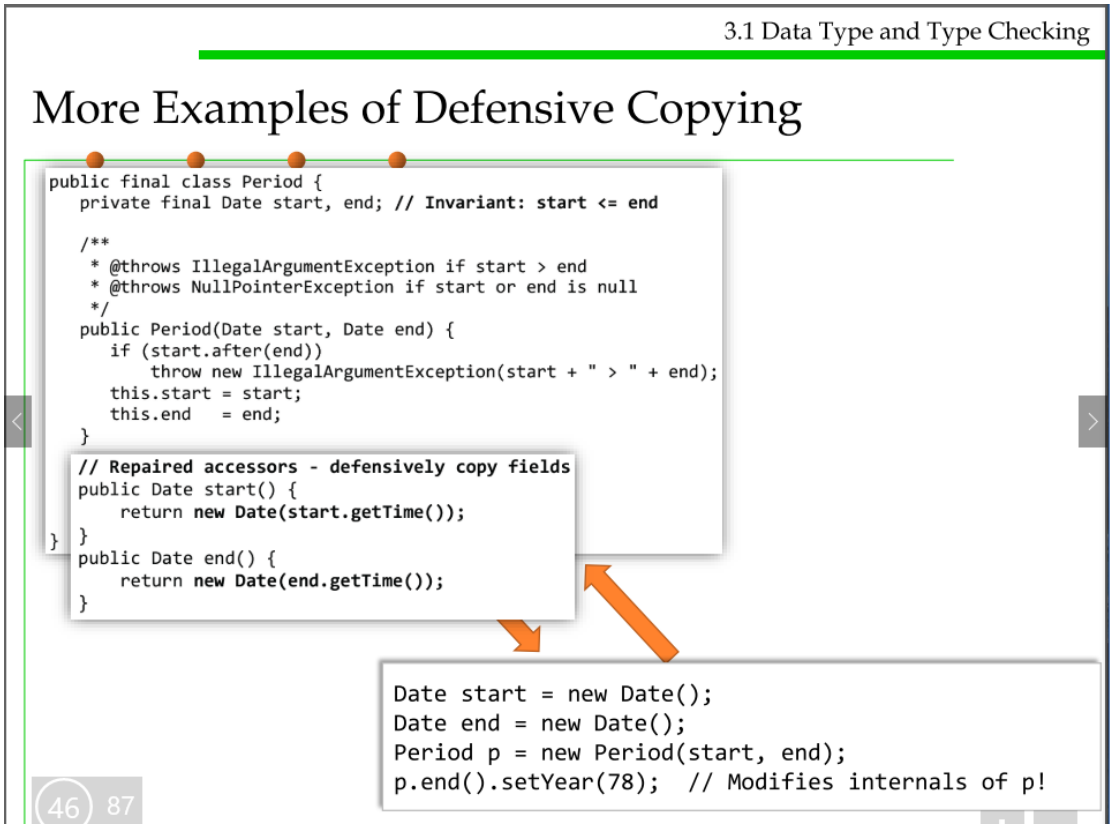


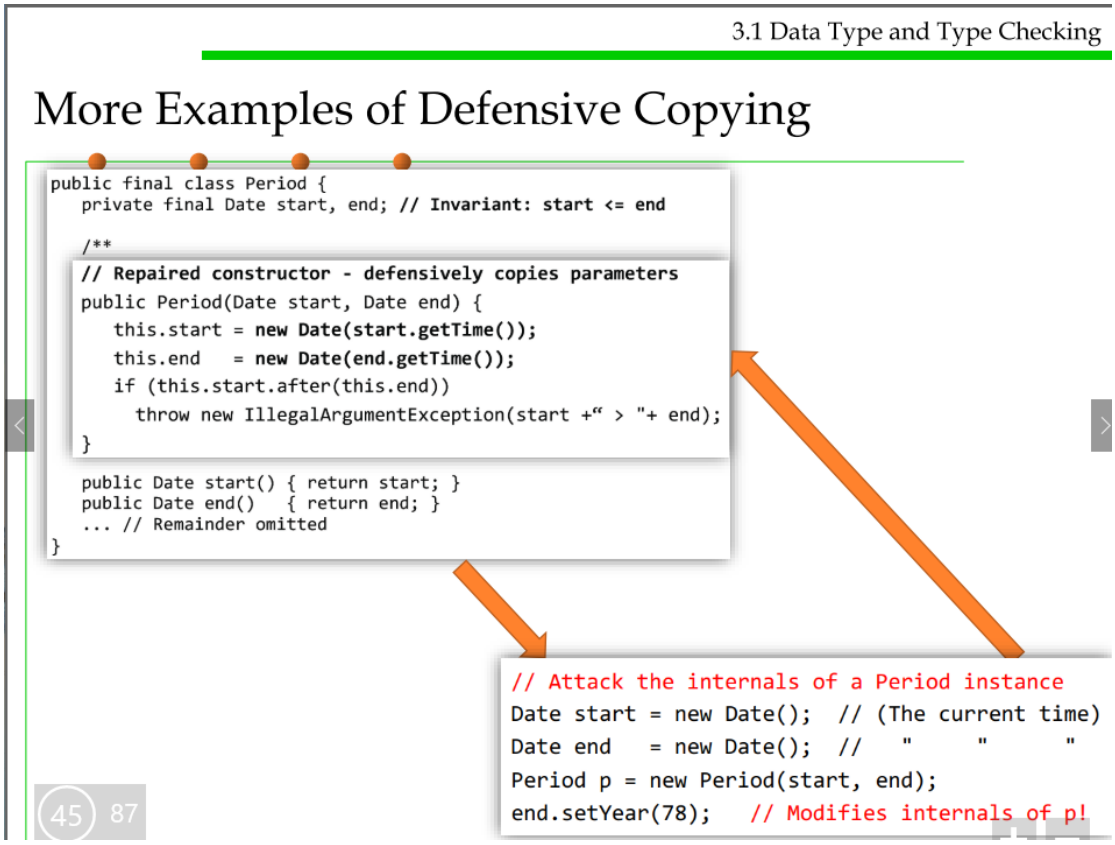


注：箭头表示扩展，不需要每次都进行计算，使用一个 cache 进行保存数据。

对于一个可变型的数据，可以进行进行防御式拷贝，来解决上述问题。

Eg：





注：外部客户端的修改数据在我们的意料之外。

3.1.4 Snapshot diagram as a code-level, run-time, and moment view

Definition

Snapshot diagrams represent the internal state of a program at runtime – its stack (methods in progress and their local variables) and its heap (objects that currently exist). （用于描述程序运行时的内部状态）

Characteristic

1. To talk to each other through pictures. 便于程序员之间的交流
2. To illustrate concepts like primitive types vs. object types, immutable values vs. immutable references, pointer aliasing, stack vs. heap, abstractions vs. concrete representations. 便于刻画各类变量随时间变化
3. To help explain your design for your team project (with each other and with your TA). 便于解释设计思路
4. To pave the way for richer design notations in subsequent courses

Draw：

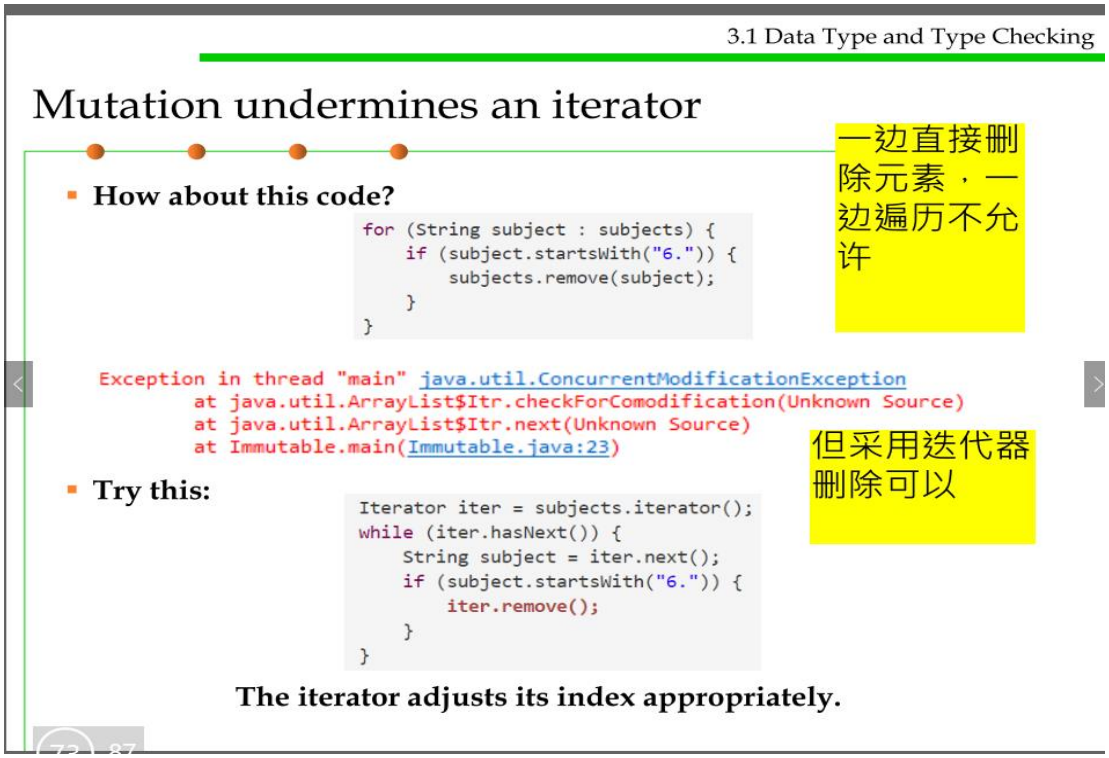
1. Primitive values（基本数据类型值）：bare constants（直接写内容）
2. Immutable Object values（不可变的对象数据类型值）：constants with a circle which has a double border（内容加双圈）
3. mutable Object values（可变的对象数据类型值）：constants with a circle which has a singer border（内容加单圈）
4. immutable references （不可变的引用）：a double arrow（双线箭头）
5. mutable references（可变的引用）：a singer arrow（单线箭头箭头）

3.1.5 Complex data types: Arrays and Collections

Array（定长数组）：前学之述备已，只需明白 array 是一个对象，需用对象的方式。
List（变长数组）：1. List is an interface（接口）
2. members in a List must be an object.（list 内的数据类型必须是对象）
Set(集合)： A Set is an unordered collection of zero or more *unique* objects。(独一无二)
Map(键值对)：

注：Iterator as a mutable type 迭代器

Iterator is an object that steps through a collection of elements and returns the elements one by one. （迭代器是一个对象，它遍历一组 元素并逐个返回元素。）
方法：1. next() returns the next element in the collection --- this is a mutator method!
2. hasNext() tests whether the iterator has reached the end of the collection.
Eg：迭代器不允许一边遍历元素，一边对元素进行直接修改。而使用迭代器可以



Eg：增强 For 循环本质上就是一个迭代器。

```
List<String> lst = ...;
for (String str : lst) {
    System.out.println(str);
}
```

```
List<String> lst = ...;
Iterator iter = lst.iterator();
while (iter.hasNext()) {
    String str = iter.next();
    System.out.println(str);
}
```

3.1.6 Useful immutable types

1. The primitive types and primitive wrappers are all immutable. （基本类型及其封装对象类型都是不可变的）
2. The Collections utility class has methods for obtaining unmodifiable views of these mutable collections。

Eg：

- `Collections.unmodifiableList`
- `Collections.unmodifiableSet`
- `Collections.unmodifiableMap`

a wrapper around the underlying list/set/map

注：1 .这种包装器得到的结果是不可变的：只能看

2. The downside is that you get immutability at runtime, but not at compile time. （但是这种“不可变”是在运行阶段获得的，编译阶段无法据此进行静态检查）

Eg：

```
List<String> list = new ArrayList<>();

list.add("ab");

List<String> listCopy = Collections.unmodifiableList(list);

listCopy.add("c"); 不允许

list.add("c");     允许

System.out.println(listCopy.size());
```

2

```
Exception in thread "main" java.lang.UnsupportedOperationException
at java.util.Collections$UnmodifiableCollection.add(Collections.java:1055)
at Immutable.main(Immutable.java:33)
```

3.To allow certain clients read-only access to your data structures. You keep a reference to the backing collection but hand out a reference to the wrapper. In this way, clients can look but not modify, while you maintain full access.（开发人员可以为客户设置一个不可变的包装，使得其只能读，而为开发人员设置一个可变的 collection，使其可以对之进行修改。）

3.1.7 Null references

Definition：

In Java, references to objects and arrays can also take on the special value Null , which means that the reference doesn't point to an object. Null values are an unfortunate hole in Java's type system.（意味着一个对象不指向任何内存）

注：1. Primitives cannot be null.

2. Don't call any methods or use any fields with one of the null references (otherwise ， throws NullPointerExceptions)（）
3. Null values are implicitly disallowed in parameters and return values. If a method allows null values for a parameter, it should explicitly state it, or if it might return a null value as a result, it should explicitly state it. But these are in general not good ideas. Avoid null.