Differences Between C++ and Java

数据科学与计算机学院 软件工程 郑卓民 18342138

Preface

We know that there are quite a few differences between C++ and Java, both of which are programming languages. In this article, I will briefly analyze the differences between them and give examples and explanations.

Variables and Types

First of all, in terms of variables and types, they have the following differences.

First, Java has no unsigned integers. (Unsigned right shift is forced in Java with three right angle brackets). the number of bits in the C++ integer randomizer changes, but Java does not. (C++ int is 16-bit on 16-bit machines, 32-bit on 32-bit level. Long is 32-bit on 32-bit and below, and 64-bit on 64-bit machines.)

```
//In c++, 'int' has two types: signed and unsigned
unsigned int u_a=0;
signed int s_a=0;
//In Java, Int type is fixed 32 bit
System.out.println("max int ="+Integer.MAX_VALUE);
System.out.println("min int ="+Integer.MIN_VALUE);
```

Second, Java has a built-in type String, which can not be changed, but C++ doesn't. C++'s std::string is mutable, similar to Java's StringBuffer.

When you look at the java.lang.String method, you can see:

```
/**
 * The String class represents character strings. All string literals in Java programs, such as "abc",
 * are implemented as instances of this class.
*/
public final class String implements java.io.Serializable, Comparable<String>, CharSequence {
    /** The value is used for character storage. */
    private final char value[];
   ...
}
```

The member field value of the String class is a char[] array, and it is also modified with the final keyword. The field modified by the final keyword is immutable after creation, but only the reference address of the value is immutable, but Array The value of the array is mutable.

Third, Java strings are stored in Unicode in memory. C++ is the same code as the source code.

Fourth, there is no pointer in Java. Java references are pointers to weakened functions, and can only be used as operations for "calling methods of the pointed object."

Fifth, Java Stream object (except PrintStream) has a single function, can only read and write by byte, and needs the help of Reader or Writer. Any stream in C++ can be read or written in bytes, strings, or integers.

Sixth, Java forces local variable initialization.

In order to improve code security, java specifies: The member variables defined in the class will automatically initialize for you if you have not initialized java. If the variable is a number, it will be automatically initialized to 0. The variable is initialized to 'a', and the variable is an object. The reference will be initialized to null, and the variable will be boolean, then it will be initialized to false automatically. If you define a local variable to be used later (to extract values from there), then it must be initialized at the time of declaration, otherwise compile Will report an error.

```
public class VariableTest{
     int a;
     char ch;
     boolean flag;
     public static void main(String[] args){
          VariableTest variableTest = new VariableTest();
          System.out.println("a: "+ variableTest.a);
          System.out.println("ch: "+ variableTest.ch);
          System.out.println("flag: "+ variableTest.flag);
     }
}
//a: 0
//ch:
//flag: false
```

Class Mechanism

Then, we talk something about the class mechanism between C++ and Java.

First, Java is completely object-oriented, and all methods must be written in the class. And Java is single root inheritance, all objects are inherited from Object. And provide an interface mechanism.

```
// in java:
public class Penguin extends Animal{}
// in c++:
class Waiter {};
class Singer  {};
class SingerWaiter:public Waiter,public Singer {};
```

Second, Java defaults to a virtual function when overridden.
Third, Java has no default parameters.
Fourth, there is no operator overloading, and conversion functions in Java.

```cpp
// In C++, it has member function like this:
Box operator+(const Box& b)
{
     Box box;
     box.length = this->length + b.length;
     box.breadth = this->breadth + b.breadth;
     box.height = this->height + b.height;
     return box;
}
// In java, it has not.
```

Fifth, Java has no reference value. (The standard library is generally replaced by Boxing.)
Sixth, Java does not have Struct or Union.
Seventh, Java objects (including arrays) are stored on the heap and instantiated with new. C++ can be optionally stored on the stack or on the heap.

```
// In C++
Student stu1; // in stack
Student *stu2 = new Student(); // in heap

// In Java
A a=new A();  // in heap
```

Eighth, Java forces a file to have only one public class/interface, and the file path is forced to its package name and class name for the reason that each compilation unit can only have one public interface, and this interface is represented by its public class.

Other Language Features

In addition, let us talk about other differences.
First, Java does not have a delete operator. Since there is gc hosting, there is no need to delete. But in C++, new and delete must correspond one-to-one.

```cpp
// In C++, new must have a corresponding delete, but in Java this is not needed.
int *lis = new int[100];
......
delete lis;
```

Second, Java does not have macros. (On the other hand, this is also a benefit, macro is another source of all evils).

```
// In C++, you can do things like those:
//#define <name> <string>
#define PI 3.1415926
//#define <name> (<parameter>) <Macroscopic>
#define A(x) x
// but in Java, you cann't
```

Third, Java uses the dot operator to do things with C++::operators. The Java::operator can turn a static function or a dynamic function with an object into a closure (Java8).

Fourth, C++ only looks for the line in the current code file when looking up the name. However. The scope of Java search is the entire project. (This is why C++ requires “#include” and Java does not.)

Running Speed

Finally, I want to talk about the most intuitive difference between C++ and Java: the gap in runtime.

For one thing, because Java is semi-compiled and semi-interpreted, it runs slightly slower. The jvm startup speed is not generally slow. Java can't be directly compiled into machine language execution, but compiled into a binary .class bytecode file, which is interpreted and executed by the java virtual machine. In general, java is much slower than C++. But this is not a flaw in Java. On the contrary, because of the binary bytecode and virtual machine, Java can achieve cross-platform features. The same bytecode can be executed only by the java virtual machine in different operating systems.

For another, because Java is a dynamic high-level language and has a powerful tool for Java virtual machines, Java sacrifices some performance for simplicity and security. For example, Java does not require manual GC, nor does the programmer need to care about array out-of-bounds. Instead, the Java virtual machine dynamically monitors the overflow at runtime and is GC-enabled by the Java virtual machine at the appropriate time. And Java will monitor the reference situation during the running process to prevent abnormal phenomena such as null pointer exceptions; it also needs to perform type monitoring at runtime, and throws a ClassCastException if the type is incorrect. It is these handy features that sacrifice some of the performance of Java. But this is why Java is accepted by most programmers and is widely popular.

Fortunately, with the development of JVM virtual machine technology, language features have a lower impact on high-performance computing performance. The understanding of computer architecture, compilation principles, and virtual machine compilation mechanisms has become more important. The automatic optimization of the JVM is very powerful. In addition, programmers can significantly improve the performance of Java programs by optimizing the code, such as avoiding the use of static variables as much as possible, using basic data types instead of objects and trying to avoid creating too many Java objects.

Now, let's take a look at the difference in performance of the virtual machine's optimization of the Java code.

```java
// In Java:
public static void testArray1(int[] array){
    for(int k=0; k<5000; k++){
        for(int i=1; i<array.length; i++){
            array[i] = k;
        }
    }
} // run 0.654s (Expand C2 Optimization)

public static void testArray2(int[] array){
    for(int k=0; k<5000; k++){
        for(int i=1; i<array.length; i++){
            array[i] = array[i-1];
        }
    }
} // run 1.755s (Expand C2 Optimization)

public static void testArray2(int[] array, int stripe){
    for(int k=0; k<5000; k++){
        for(int i=1; i<array.length; i+=stripe){
            array[i] = array[i-1];
        }
    }
} // run 6.163s (Without expand C2 Optimization)
```

Conclusion

All in all, there are quite a few differences between C++ and Java. No one is better or worse, each has its own advantages and each has its own characteristics. What we have to do is to get what we need and choose the programming language that suits our project according to your needs.