Middle Examination Report: The differences between Java and C++

19335286 郑有为

Java programming language is a high-level language developed from C++ and becomes a new generation of object-oriented programming language. Here are six aspects of differences between Java and C++, including the process of creating an executable file & portability, object-oriented programming, inheritance, polymorphism, memory management and so on.

1. The process of creating an executable file & portability

C++ is a compiled language whose implementations are typically compliers (translators that generate machine code from source code), and not interpreters (step-by-step executors of source code, where no pre-runtime translation takes place). To generate a .exe file from .cpp file, it needs preprocessing, compiling (translates source file to assemble file), assembling and linking orderly.

However, Java is a semi-compiled and semi-interpreted language. The .java file will be compiled into .class file by javac. A .class file does not contain code that is native to your processor, it contains bytecodes — the machine language of the Java Virtual Machine. The Java launcher tool then runs your application with an instance of the Java Virtual Machine.

The difference on creating an executable file between Java and C++ leads to another difference between them. It is the portability.

C++ is platform-dependent, which means that an executive file written by C++ on a kind of environment may not be executive on another environment. Compared to C++, Java is platform-independent because it runs programs on Java Virtual Machine. As long as a computer has the Java Platform, the programs are executive no matter what its operating system is. Therefore, Java has stronger portability than C++.

2. Object-oriented Programming

Both of Java and C++ have the characteristic of object orientation, but Java is purer than C++. C++ is more like a hotchpotch. It supports procedure-oriented programming, object-oriented programming and functional programming. However, Java is a totally object-oriented programming language with all methods and data must belong to the class, which means we can't define data like global variable outside the class. It is a single root hierarchy as everything gets derived from java.lang.Object. In Java, everything is treated as an object, even the main function.

3. Inheritance

Firstly, unlike C++, Java doesn't support multi-inheritance. Java uses a single inheritance tree always because all classes are the child of Object class in java. The object class is the root of the inheritance tree in java.

Secondly, Java has the keywords that C++ doesn't have to simplify programming. Although Java doesn't have Scope Resolution Operator (::) and derived types in inheritance (public, protected, private), it has keyword *extends* to inherit other class. Besides, Java has the keyword *interface* (an interface is a group of related methods with empty bodies) and *implements* to achieve multi-inheritance in other way.

Thirdly, Java has the keyword *super* to allow access to the superclass member, which is used to reference the current object's superclass.

4. Polymorphism

Firstly, Java does not support operator overloading, which is considered a prominent feature of C++.

Secondly, in C++, a default member function is static binding. A member function with keyword virtual can be called virtual function, which is related to dynamic binding. A class with virtual function can be called abstract class. On the contrary, there is no concept of virtual function in Java. A default member function is dynamic binding in Java. Therefore, Java provides keyword *final* to define a member function with static binding. In other word, Java provides automatic polymorphism. As against, in C++ the polymorphism is explicit for each particular method.

Thirdly, compared to C++, abstract class may or may not include abstract methods. And Java provides keyword *abstract* to for user to define Abstract classes and member functions.

5. Memory Management

In C++, when we *new* an object, we need to *delete* it when we no longer need in person. Its memory management is accessible to programmers.

However, memory management is system controlled in Java. We don't need to delete the object we *new*. Java provides a garbage management mechanism to monitor all objects coming out, identifies objects that will not be referenced again, and then frees up memory space.

6. Others

There are so many differences between C++ and Java.

1. In C++, pointer is widely used to manage the memory, but pointer is not allowed in Java.

2. C++ provides preprocessing such as *#define #include*, which is not allowed in Java as well.

3. In function call, C++ supports both call by value and call by reference, while there is no call by reference in Java.

4. Structure and union in C++ is not supported by Java.

5. Lastly, to better organize classes, Java provides a *package* mechanism

for distinguishing the namespaces of class names. In C++, we use keyword *namespace* to achieve the same function. For file organization, Java uses keyword *import* to link other packages, while C++ uses preprocess (*#include*) to organize calls among source files.

In conclusion, there are quite differences between Java and C++. However, they still have so many similarities. Once you master one of them, it is very easy to learn another.