Differences between Java and C / C++

The fundamental description of Java and the reasons for maintaining or altering properties of the previous language C / C++ was written by *James Gosling*. It is a publication from 1996 and many later language-extensions are not yet mentioned. The document is still valuable and worth reading as it has achieved the status of a classic reader: The **Java Language Environment** (White Paper):

http://www.oracle.com/technetwork/java/langenv-140151.html

On the first pages Gosling founded the syntactic similarity between Java and C / C + +:

```
"Another major design goal is that Java looks familiar to a
majority of programmers in the personal computer and
workstation arenas, where a large fraction of system
programmers and application programmers are familiar with C and
C++.
Thus, Java "looks like" C++. Programmers familiar with C,
Objective C, C++, Eiffel, Ada, and related languages should
find their Java language learning curve quite short—on the
order of a couple of weeks."
```

Another (albeit rather abstract) description of the Java programming language was developed by Sun in the **Java Language Specification** document *(James Gosling, Bill Joy, Guy Steele, Gilad Bracha).*

http://docs.oracle.com/javase/specs/jls/se6/html/j3TOC.html

Java is easy to learn for an experienced C and C++ programmer because the basic language syntax and grammar is very similar. Nevertheless, there are some fundamental differences that need to be observed.

Strictly object oriented (Klassenmodularität)
No unbound classes (strict class hierarchy)
No multiple inheritance
No static (stack) objects
No global declarations and definitions
No inclusions of external files
Strict distinction between value and object data types
No operators for addresses (* and &)
No choice between call-by-value / call-by-reference
No implicit conversion (type Control)
No variables overlap in methods

Some important differences between Java and C / C + +

A class is the minimum unit of which a Java program exist. Each class is an automatic part of the Java class hierarchy and inherits basic attributes and methods of the highest class `<Object>`.

Objects must always be allocated with `new()`.The object name is a reference (pointer) to the memory location of the object‘s data. There are no static (stack) objects but only dynamic (heap) objects. Objects can not be explicitly destroyed (there is no delete() or destructor-method), instead there is an automatic garbage collection discarding objects when there is no valid reference existing. Reference operaor is the dot.

Methods must be implemented within the class. It can be distinguished object and class methods. An object method is called on an object and uses its data, while a class methods can not work with object data, but serves as a substitute for the functions not existing (global).

Attributes of a class can be initialized immediately.

There are few types of instructions that can be left outside of a class, suchB. `import` (notice external classes) or `package` (included in a class module).The import statement inserts (in contrast to C / C + + <include>) no code, but only makes the precise location of an existing class declaration known. Other directories can not be specified explicitly. External files can not be included.

There are no global type definitions or macros allows, apply to several classes.

Only elementary data types (numbers / letters) are value types.My name is identical to its value must be a single. All data types that consist of multiple values are generally treated as an object.All arrays, even if its elements consist of value data types are objects and must be allocated with `new`.For array declarations, the [] brackets are also allowed for the data type.

There is no pointer <*> and operators for address manipulation <&>.So many sources of error are eliminated. Internally, the virtual machine (runtime) continues to work with stack and heap addresses, but there are no direct memory accesses allowed.

In the argument passing methods, there is no choice between call-by-value and call-by-reference.All data types are passed by call-by-value principle. For objects have all the changes apply only to the object name, the method of leaving no lasting effects - the access to object data, however, is permanent.

There is no automatic (implicit) type conversion in assignments, but it must be explicitly (typ) be cast, even the return value of methods.

Local variable names must not be redeclared (overwritten) in an inner block of a method.

Main takes over when calling an array of strings (without program name) and can not return a value back to the operating system level.Main must be static method as part of an (arbitrary) class.

Declarations of data types (variables or constants) have not quite at the beginning of a valid block (as in C / C + +), but can occur at any point where they are needed.Counter variables can be declared within the `for` statement and then apply only within the loop.