Your First Steps with Java

This introduction into objectoriented programming with Java is primarly intended for readers already possessing fundamental programming experience – at least to a certain extent. Basic knowledge of procedural programming techniques is required to understand and follow the course. Since many textbooks dive into object-oriented programming rather directly, this unusual didactic approach requires a justification. A short look at the paradigmatic development of programming languages may be helpful.

The fundamental techniques of programming historically have been developed as a practical branch of software development covering a fairly long period of time - "long" means: in comparison to the usual breathless rapidity in computer science. Nevertheless it is possible to distinguish several successive development sections, in each of which different principles ('paradigms') prevailed.

Early programming languages (up to around 1940) followed a *sequential* programming scheme in which each line of code is following the other and is processed accordingly. Coding is performed in a special manner which matched to the particular type of the processor (CPU). This rather complex programming in machine dependent (binary) form demanded high programming skills. Subsequently a more human readable form was developed, especially for self-contained subroutines (procedures or functions) to define and control the execution of programs through checking of conditions and parameters. The *procedural* paradigm states that a program consists of a number of functions which may be called from the main program section (the main function). It is a modular principle which defines the structure of the program into sub-tasks (functions) and also determines the automatical returning to the correct call location, just as dynamic function call is possible. Developed in the early seventies, C is a typical example of such a procedural programming language.

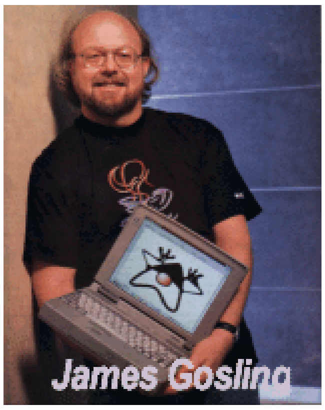
The *object-oriented* programming languages have emerged as a response to the international "software crisis" in the 60's, when it became clear what software development actually is about - mapping a particular segment of reality (the relevant business-related tasks) in software. They must be based on the most important principles of reality that are empowered to act as individual selcontained units (and not just as abstract data structures). The traditional procedural languages proved to be increasingly inadequate for software development in such complex reality cutouts. A new approach was required to encapsulate independently operating units (objects), whose data values (properties, attributes) are associated with specific behaviors (methods, object-typical functions). Attributes and methods are defined in classes, objects represent real forms of the characteristics of their class that are filled with individual data values.

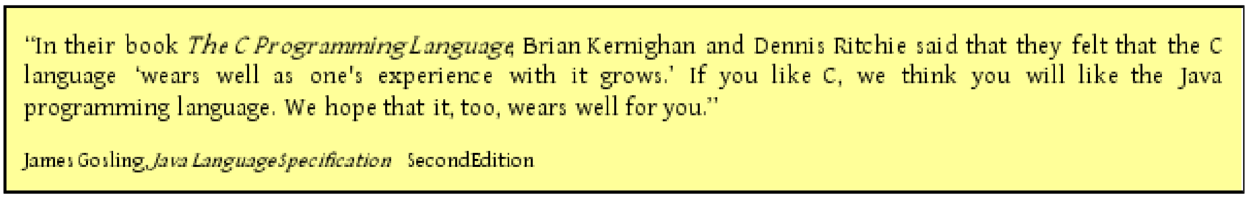
This brief historical outline shows that the object-oriented paradigm is an extension and completion of the older procedural paradigm, rather than a complete new start. Thus the basic syntax and grammar of Java differ only marginally form C, C++, or C#. These languages are closely related syntactically and grammatically and are using up to 98% identical language components (commands, instructions, operations).

Learning Java cannot be separated from learning the fundamental principles of object-orientation. As Java is object oriented in its inner structure from ground up – denying this would ultimately end in abusing the language and divert from its roots. Nevertheless the basic grammar and syntax of Java (data types, control structures, loops, functions, user-defined data types) is derived directly from C. Therefore in the following chapters reference will be given only to differences; omission means syntactic identity with C.

History of the Java programming language

The similarity of C and Java has historical roots. Java was developed by a group of experienced C-programmers (James Gosling, Patrick Naughton, Mike Sheridan, et al.) With the strong support of Bill Joy several precursors were developed in the early 90s. These fellows were employees of SUN Microsystems in California and had set out to overcome some of the known weaknesses of C and intended to develop a language that can be used for mobile devices and networked applications. The new language should be independent of hardware and operating-system. It should have a high level of abstraction and support all principles of object-oriented programming.




"In their book *The C Programming Language*, Brian Kernighan and Dennis Ritchie said that they felt that the C language 'wears well as one's experience with it grows.' If you like C, we think you will like the Java programming language. We hope that it, too, wears well for you."

James Gosling, *Java Language Specification* Second Edition

Web pages on the history and development of Java:

http://www.wired.com/wired/archive/3.12/java.saga.html
http://www.wired.com/wired/archive/3.12/java.saga.html?pg=2
http://www.blinkenlights.com/classiccmp/javaorigin.html

Bill Joy:

http://www.wired.com/wired/archive/8.04/joy.html
http://news.cnet.com/8301-13860_3-20005814-56.html

The historic development of the Java programming language took place in small steps and in the course of time Java experienced several renamings (Oak, Green, Java). For a long time it was not clearly visible on which question it could be an answer. The developers tried to provide household appliances with an embedded operating system environment or tried to go into business with interactive TV (Star-7, set-top-boxes, ITV). But all such initiatives ultimately failed and the developers were already close to giving up. Due to the surge in popularity of the Internet by 1993, an unexpected niche opened up and the advantages of Java suddenly were practically usable. The programming of a web browser (WebRunner, HotJava) was very successful and the importance of the Java language has become increasingly clear. Ultimately, there were three factors that led to the breakthrough brought about:

- Provide portable compiled intermediate code (class code, byte code)
- Interpretative execution - platform independence
- Strict object-centeredness

The timeline of the development of Java can be traced by using Internet ressources of Oracle, the current owner of Java:

http://oracle.com.edgesuite.net/timeline/java/