Object Oriented Programming (IT 2nd)

Department of Computing Science & Information Technology

Benazir Bhutto Shaheed University Lyari, Karachi

Fall Session 2022

Brief History of OOP

Many people believe that OOP is a product of the 1980s and the work done by Bjarne Stroustrup in moving the C language into the object-oriented world by creating the C++ language. Actually, SIMULA 1 (1962) and Simula 67 (1967) are the two earliest object-oriented languages. The work on the Simula languages was done by Ole-John Dahl and Kristen Nygaard at the Norwegian Computing Center in Oslo, Norway.

Object-oriented programming began with the Simula language (1967), which added information hiding to ALGOL. Another influential object-oriented language was Smalltalk (1980), in which a program was a set of objects that interacted by sending messages to one another. Since the 1990s, Java has been one of the most successful object-oriented languages.

While most of the advantages of OOP were available in the earlier Simula languages, it wasn't until C++ became entrenched in the 1990s that OOP began to flourish.

C was the parent language of C++ and it was often said that C was powerful enough to shoot yourself in the foot multiple times. C++, on the other hand, not only was powerful enough to shoot yourself in the foot, but you could blow your entire leg off without too much difficulty. Most programmers admit that C++ is a very powerful language, and it is still in widespread use today. However, with that power comes a lot of complexity. Language developers wanted a simpler and perhaps less complex language for OOP development.

History of Java Programming Language

Java was created at Sun Microsystems, Inc., where James Gosling led a team of researchers in an effort to create a new language that would allow consumer electronic devices to communicate with each other. Work on the language began in 1991, and before long the team's focus changed to a new niche, the World Wide Web. Java was first released in 1995, and Java's ability to provide interactivity and multimedia showed that it was particularly well suited for the Web.

The difference between the way Java and other programming languages worked was revolutionary. Code in other languages is first translated by a compiler into instructions for a specific type of computer. The Java compiler instead turns code into something called Bytecode, which is then interpreted by software called the Java Runtime Environment (JRE), or the Java virtual machine. The JRE acts as a virtual computer that interprets Bytecode and translates it for the host computer. Because of this, Java code can be written the same way for many platforms ("write once, run anywhere"), which helped lead to its popularity for use on the Internet, where many different types of computers may retrieve the same Web page.

By the late 1990s Java had brought multimedia to the Internet and started to grow beyond the Web, powering consumer devices (such as cellular telephones), retail and financial computers, and even the onboard computer of NASA's Mars exploration rovers. Because of this popularity, Sun created different varieties of Java for different purposes, including Java SE for home computers, Java ME for embedded devices, and Java EE for Internet servers and supercomputers. In 2010 the Oracle Corporation took over the management of Java when it acquired Sun Microsystems.

Despite the similarity in names, the JavaScript language that was designed to run in Web browsers is not part of Java. JavaScript was developed in 1995 at Netscape Communications Corp. and was conceived of as a companion to Java. It was originally called Mocha and then LiveScript before Netscape received a marketing license from Sun.

Computer Program

Computer program, detailed plan or procedure for solving a problem with a computer; more specifically, an unambiguous, ordered sequence of computational instructions necessary to achieve such a solution. The distinction between computer programs and equipment is often made by referring to the former as software and the latter as hardware.

Programs stored in the memory of a computer enable the computer to perform a variety of tasks in sequence or even intermittently. The idea of an internally stored program was introduced in the late 1940s by the Hungarian-born mathematician John von Neumann. The first digital computer designed with internal programming capacity was the EDVAC (acronym for Electronic Discrete Variable Automatic Computer), constructed in 1949.

A program is prepared by first formulating a task and then expressing it in an appropriate computer language, presumably one suited to the application. The specification thus rendered is translated, commonly in several stages, into a coded program directly executable by the computer on which the task is to be run. The coded program is said to be in machine language, while languages suitable for original formulation are called problem-oriented languages. A wide array of problem-oriented languages has been developed, some of the principal ones being COBOL (Common Business-Oriented Language), FORTRAN (Formula Translation), BASIC (Beginner's All-Purpose Symbolic Instruction Code), and Pascal.

Integrated Development Environment (IDE)

An IDE, or Integrated Development Environment, enables programmers to consolidate the different aspects of writing a computer program.

IDEs increase programmer productivity by combining common activities of writing software into a single application: editing source code, building executables, and debugging.

For Example: *Java NetBeans 8.2/ VSCode (Latest)

Compiler & Interpreter

We generally write a computer program using a high-level language. A high-level language is one that is understandable by us, humans. This is called source code.

However, a computer does not understand high-level language. It only understands the program written in 0's and 1's in binary, called the machine code.

To convert source code into machine code, we use either a compiler or an interpreter.

Both compilers and interpreters are used to convert a program written in a high-level language into machine code understood by computers. However, there are differences between how an interpreter and a compiler works.

Interpreter

Translates program one statement at a time.

Interpreters usually take less amount of time to analyze the source code. However, the overall execution time is comparatively slower than compilers.

No intermediate object code is generated, hence are memory efficient.

Programming languages like JavaScript, Python, Ruby use interpreters.

Compiler

Scans the entire program and translates it as a whole into machine code.

Compilers usually take a large amount of time to analyze the source code. However, the overall execution time is comparatively faster than interpreters.

Generates intermediate object code which further requires linking, hence requires more memory.

Programming languages like C, C++, Java use compilers.