A Short History of Programming Languages

The first computers were built in the late 1930's and 1940's.

Remember that a computer is just wires and switches. All data is represented by wires that are ON or OFF. (1 or 0) So, all data is just a number in binary.

This includes the commands that determine which operation to do. They are also ON or OFF states of wires, and so are also just numbers in binary.

On all computers, the programs to run the computer are "machine code", and each operation is a specific number.

Every computer make will have a different machine code - not easy to code and almost impossible for a human to read.

The first high level programming languages were Fortran, LISP, and Cobol (late 1950's).

These were English and Math like. A program called a compiler converted the high level program down into machine code.

To run a high level language on a computer, we need either a compiler or an interpreter.

A compiler takes the high level language program and converts it into a machine code program file.

An interpreter is a separate program that runs on the computer. It reads the high level program file and executes it.

Compiled programs will run faster than interpreted programs because the compiled programs are in the machine code while the interpreted programs have that interpreter program running them.

Interpreted languages often have more complex features because it is easier to write an interpreter for the features than to figure out how to represent them in machine code.

Note that a language does not have to be only compiled or only interpreted. Every language can be compiled and we can easily write interpreters for every language. Instead, specific languages tend

to be compiled or tend to be interpreted as standard practice.

A History of Java

C was a language developed by AT&T in 1970 for the sole purpose of creating the Unix operating system.

C became very popular because AT&T (a phone monopoly) gave the language and Unix away for free, and universities jumped at it.

C is very good for low level coding of machines, but does not scale well to very large projects.

No good data organization routines or safety features to limit bugs.

C++ (1980's) added "objects" to C. Objects are a nice way to organize data in a program. C++ still had no safety features so code tended to be buggy.

(C and C++ were designed before safety became an important issue in programming languages.)

Java (1990's) was built by Sun Microsystems as a safe version of C++. The original purpose was to write code for devices that could not be easily debugged (like microwaves, cars, etc.),

but it became very popular for building web applications.

Java adds safety features and it uses a more elegant and simpler object model than C++ has.

Java is both compiled and interpreted.

The Java compiler compiles the Java program to Java Byte Code instead of to machine code.

Then an interpreter called the Java Virtual Machine interprets the bytecode.

Because it is interpreted, we only have to compile our Java code once. Each machine make will have its own Java Virtual Machine to interpret the program.

Unlike C or C++ where we have to separately compile a different version for every machine.

(The Java Virtual Machine comes standard on all Windows and Apple machines.)

Because it is partially compiled, the interpreter runs Java faster than other "normally" interpreted languages.

C# (2000) was built by Microsoft as an update to Java. It kept the safety features of Java but added back in many of the conveniences of C++ that programmers missed.

Objective-C (1980's) is C with a the Smalltalk object model (the same object model used by Java). Objective-C is used for iOS development. (Jave is used for Android development.)

Data Types:

A data type is what a piece of data represents. It is specified by the programmer.

Java is a "strongly typed" language. That means that every piece of data will have a well defined type

The programmer must set the type of every expression (either explicitly or implicitly), and the Java compiler

will verify that each type is used correctly. If a type is used incorrectly, an error is given.

For example, Java will not allow you to play a string of text as a sound file. (C does not have such a safety restriction.)

There are two kinds of types in Java:

1. primitive types : these are pre-defined in the language

2. compound types (called reference types in your book) : there are some pre-defined, but the programmer can define new types

Primitive Types in Java

The following are the primitive types:

boolean: 1-bit of data (really stored as 8-bits). Represents either true or false.

int: 32-bits of data. Represents an integer between -2^31 and +2^31 - 1.

All whole numbers in your program default to the type int.

example: 5, -30, 0 are all ints

short: 16-bits of data. Represents an integer between -2^15 and +2^15 - 1.

byte: 8-bits of data. Represents an integer between -2^7 and +2^7 - 1.

char: 16-bits of data. Represents a character. Can be converted to a number.

char values are single characters inside single quotes.

example: 'a', 'x', ' ', '?', '3' are all char values.

(Note that '3' and 3 are not the same thing! The first is a binary value that Java interprets as the character 3, and

the second is a binary value that Java interprets as the whole number 3.)

long: 64-bits of data. Represents an integer between -2^63 and +2^63 - 1.

To enter a long value, you use the letter L (upper or lower case).

example: 10L, 3000000L are all long values

Note: int, short, byte, long, and char are all considered "integer types". Java treats all equivalently as integers for the types of legal operations.

The only difference with char is how it is displayed.

double: 64-bits of data. Represents a "floating point" number (a number with a decimal point) in scientific notation.

Roughly 52 bits for the mantissa, 11 bits for the exponent, 1 sign bit.

All numbers with decimal points or in scientific notation in your program default to the type double.

example: 3.1415, -0.3, 10., 3e10 are all doubles

float: 32-bits of data. Binary, scientific notation. Roughly 23 bits for the mantissa, 8 bits for the exponent, 1 sign bit.

To enter a float, you use F (upper or lower case).

example: 3.14F is a float value

We will use int, double, char, and boolean in this class.

Please note that the number of bits of the type determines how many different values it can store.

Even though a long stores numbers between -2^63 and +2^63, and double stores values between +/- 4.9 x 10^-324 and +/- 1.7 x 10^308,

the double stores the same number of different values as the long does. The long can store every integer in that range, the double

stores only some of the possible floating point numbers in that range. Thus double cannot store all the integer values long can.