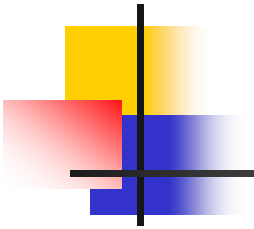
The History and Evolution of Java

Instructor: C. Pu (Ph.D., Assistant Professor)

puc@marshall.edu



Java's Lineage

- Java is related to C++, which is a direct descendant of C.
  - Much of the character of Java is inherited from these two languages.
  - From C, Java derives its syntax.
  - Many of Java's object-oriented features were influenced by C++.
  - In fact, several of Java's defining characteristics come from—or are responses to—its predecessors.

- The creation of Java was deeply rooted in the process of refinement and adaptation that has been occurring in computer programming languages for the past several decades.

The Birth of Modern Programming: C

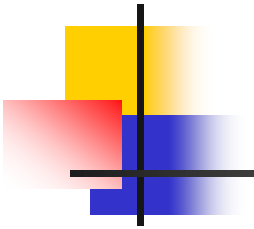
- The C language shook the computer world.
- Its impact should not be underestimated, because it fundamentally changed the way programming was approached and thought about.
- The creation of C was a direct result of the need for a structured, efficient, high-level language that could replace *assembly code* when creating systems programs.

The Birth of Modern Programming: C

- Prior to C, programmers usually had to choose between languages that optimized one set of traits or the other.
  - FORTRAN could be used to write fairly efficient programs for scientific applications, but was not very good for system code.
  - BASIC was easy to learn, it wasn't very powerful.
  - Assembly language can be used to produce highly efficient programs, but it is not easy to learn or use effectively.
  - Early computer languages were not designed around *structured principles*.
    - They relied upon the GOTO as a primary means of program control.
    - As a result, programs written using these languages tended to produce a mass of tangled jumps and conditional branches that make a program virtually impossible to understand.

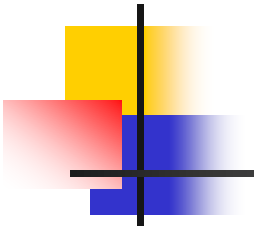
The Birth of Modern Programming: C

- C was the result of a development process of UNIX, and invented and first implemented by Dennis Ritchie.
- For many years, the de facto standard for C was the one supplied with the UNIX operating system and described in *The C Programming Lan*guage by Brian Kernighan and Dennis Ritchie.
- C was formally standardized in December 1989, when the American National Standards Institute (ANSI) standard for C was adopted.



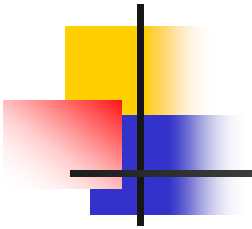
C++: The Next Step

- During the late 1970s and early 1980s, C became the dominant computer programming language, and it is still widely used today.
- Since C is a successful and useful language, you might ask why a need for something else existed.
  - The answer is *complexity*.
- Throughout the history of programming, the increasing complexity of programs has driven the need for better ways to manage that complexity.
  - C++ is a response to that need.



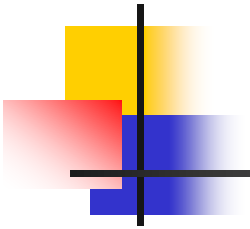
C++: The Next Step

- C++ was invented by Bjarne Stroustrup in 1979, while he was working at Bell Laboratories in Murray Hill, New Jersey.
- Stroustrup initially called the new language "C with Classes." However, in 1983, the name was changed to C++.
  - C++ extends C by adding object-oriented features.
- Because C++ is built on the foundation of C, it includes all of C's features, attributes, and benefits.
  - This is a crucial reason for the success of C++ as a language.
- The invention of C++ was not an attempt to create a completely new programming language.
  - Instead, it was an enhancement to an already highly successful one.



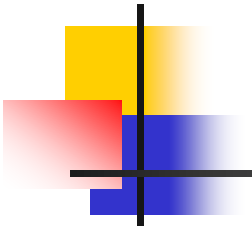
The Creation of Java

- Java was conceived by James Gosling, Patrick Naughton, Chris Warth, Ed Frank, and Mike Sheridan at Sun Microsystems, Inc. in 1991.
  - It took 18 months to develop the first working version.
  - This language was initially called "Oak," but was renamed "Java" in 1995.
- The original impetus for Java was not the Internet!
- Instead, the primary motivation was the need for a *platform-independent language*
  - That could be used to create software to be embedded in various consumer electronic devices, such as microwave ovens and remote controls.



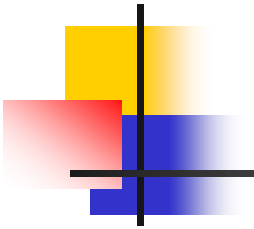
How Java Changed the Internet

- The Internet helped catapult Java to the forefront of programming, and Java, in turn, had a profound effect on the Internet.
- In addition to simplifying web programming in general, Java innovated a new type of networked program called the *applet* that changed the way the online world thought about content.
- Java also addressed some of the thorniest issues associated with the Internet: portability and security.



Java Applets

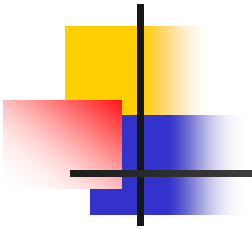
- An *applet* is a special kind of Java program that is designed to be transmitted over the Internet and automatically executed by a Java-compatible web browser.
- Furthermore, an applet is downloaded on demand, without further interaction with the user.
  - If the user clicks a link that contains an applet, the applet will be automatically downloaded and run in the browser.
  - Applets are intended to be small programs.
  - They are typically used to display data provided by the server, handle user input, or provide simple functions, such as a loan calculator, that execute locally, rather than on the server.
- In essence, the applet allows some functionality to be moved from the server to the client.



Security

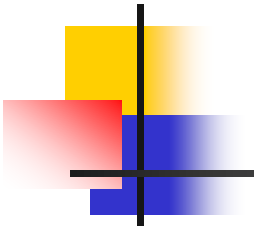
- As you are likely aware, every time you download a "normal" program, you are taking a risk, because the code you are downloading might contain a virus, Trojan horse, or other harmful code.
  - At the core of the problem is the fact that malicious code can cause its damage because it has gained unauthorized access to system resources.

- For example,
  - a virus program might gather private information, such as credit card numbers, bank account balances, and passwords, by searching the contents of your computer's local file system.



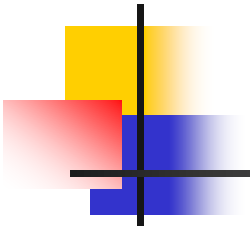
Security

- In order for Java to enable applets to be downloaded and executed on the client computer safely, it was necessary to prevent an applet from launching such an attack.
  - Java achieved this protection by confining an applet to the Java execution environment and not allowing it access to other parts of the computer.
  - The ability to download applets with confidence that no harm will be done and that no security will be breached may have been the single most innovative aspect of Java.



Portability

- Portability is a major aspect of the Internet because there are many different types of computers and operating systems connected to it.
- If a Java program were to be run on virtually any computer connected to the Internet, there needed to be some way to enable that program to execute on different systems.
- For example,
  - In the case of an applet, the same applet must be able to be downloaded and executed by the wide variety of CPUs, operating systems, and browsers connected to the Internet.
  - It is not practical to have different versions of the applet for different computers.
  - The same code must work on all computers.
  - Therefore, some means of generating portable executable code was needed.



Java's Magic: The Bytecode

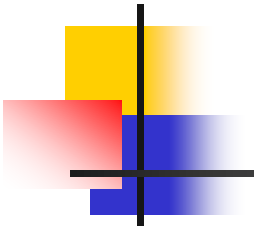
- The key that allows Java to solve both the security and the portability problems just described is that the output of a Java compiler is not executable code. Rather, it is *bytecode*.
- *Bytecode* is a highly optimized set of instructions designed to be executed by the Java run-time system, which is called the *Java Virtual Machine* (JVM).
- In essence, the original JVM was designed as an interpreter for bytecode.

Java's Magic: The Bytecode

- Translating a Java program into bytecode makes it much easier to run a program in a wide variety of environments because only the JVM needs to be implemented for each platform.
- Once the run-time package exists for a given system, any Java program can run on it.
- Remember, although the details of the JVM will differ from platform to platform, all understand the same Java bytecode.
- If a Java program were compiled to native code, then different versions of the same program would have to exist for each type of CPU connected to the Internet.
  - This is, of course, not a feasible solution.
- Thus, the execution of bytecode by the JVM is the easiest way to create truly portable programs.

Java's Magic: The Bytecode

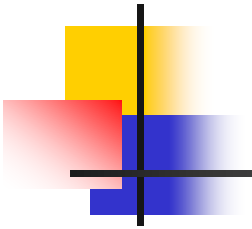
- The fact that a Java program is executed by the JVM also helps to make it secure.
  - Because the JVM is in control, it can contain the program and prevent it from generating side effects outside of the system.
- In general, when a program is compiled to an intermediate form and then interpreted by a virtual machine, it runs slower than it would run if compiled to executable code.
- However, with Java, the differential between the two is not so great.
  - Because bytecode has been highly optimized, the use of bytecode enables the JVM to execute programs much faster than you might expect.



Simple

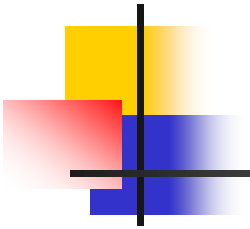
- Java was designed to be easy for the professional programmer to learn and use effectively.
  - Assuming that you have some programming experience, you will not find Java hard to master.
  - If you already understand the basic concepts of object-oriented programming, learning Java will be even easier.

- Best of all, if you are an experienced C++ programmer, moving to Java will require very little effort.
  - Because Java inherits the C/C++ syntax and many of the object-oriented features of C++, most programmers have little trouble learning Java.



Object-Oriented

- Although influenced by its predecessors, Java was not designed to be source-code compatible with any other language.
- This allowed the Java team the freedom to design with a blank slate.
- One outcome of this was a clean, usable, pragmatic approach to objects.
- The object model in Java is simple and easy to extend, while primitive types, such as integers, are kept as high-performance non-objects.



Interpreted and High Performance

- Java enables the creation of cross-platform programs by compiling into an intermediate representation called Java bytecode.
- This code can be executed on any system that implements the Java Virtual Machine.
- Most previous attempts at cross-platform solutions have done so at the expense of performance.
- As explained earlier, the Java bytecode was carefully designed so that it would be easy to translate directly into native machine code for very high performance by using a just-in-time compiler.
- Java run-time systems that provide this feature lose none of the benefits of the platform-independent code.