This image cannot currently be displayed.

Chapter 1

Introduction

Declaration

- These slides are made for UIT, BU students only. I am not holding any copy write of it as I had collected these study materials from different books and websites etc. I have not mentioned those to avoid complexity.

Topics

- Some Basic Concept
  - A computer
  - Computer Organization
  - Evolution of Operating Systems
  - Personal, Distributed and Client/Server Computing
  - Types of programming languages
- History of C++
- History of Java
- Two Parts of Java
- Other High-Level Languages
- Structured Programming
- Primary Goals of building Java
- Process of building and running java application program
- Java Platform

Topics

- Java and C
- Java and C++: Similarities and Differences
- Overlapping of C, C + +, and Java
- Java and internet
- Java interaction with the web
- Java Environment
- Java Development Kit
- Process of building and running Java application program
- Two ways of using Java
- Layers of interactions for Java program

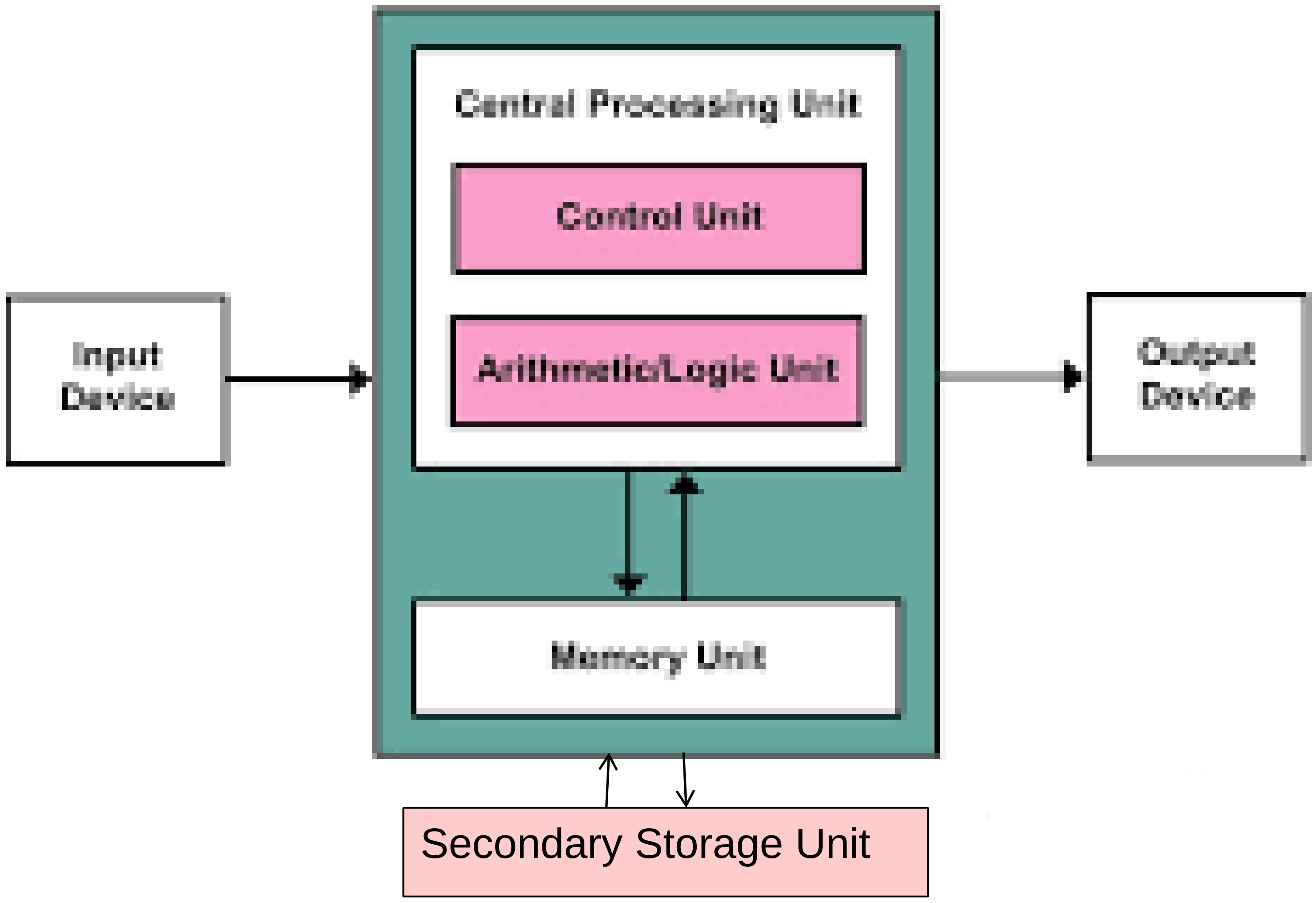
What is a Computer?

- Computer
  - A computer is a programmable electronic device that is capable of performing various operations and tasks based on instructions provided to it. It processes data and information using a combination of hardware and software components.
- Hardware
  - Various devices comprising a computer
  - Keyboard, screen, mouse, disks, memory, CD-ROM, and processing units
- Software
  - Programs that run on a computer

Computer Organization

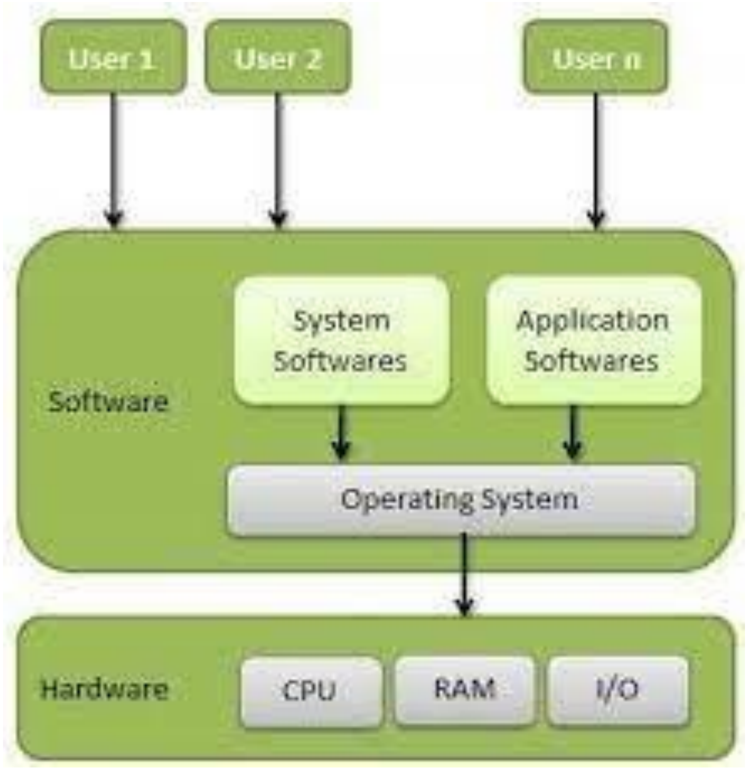
- Six *logical units* in every computer
  - Input unit
    - Gets information from input devices (keyboard, mouse)
  - Output unit
    - Gets information (to screen, to printer, to control other devices)
  - Memory unit
    - Rapid access, low capacity, stores input information
  - Arithmetic and logic unit (ALU)
    - Performs arithmetic calculations and logic decisions
  - Central processing unit (CPU)
    - Supervises and coordinates the other sections of the computer
  - Secondary storage unit
    - Cheap, long-term, high-capacity storage, stores inactive programs and data

BASIC COMPUTER ARCHITECTURE



Operating Systems

- An operating system (OS) is software that manages computer hardware and provides an interface for users to interact with the computer and run applications. It acts as an intermediary between application software and computer hardware, enabling the software to access and utilize the hardware resources effectively. Without an operating system, users would have to directly interact with the hardware, making it challenging to perform tasks and execute programs efficiently.



Personal, Distributed and Client/Server Computing

- Personal computing
  - Popularized by Apple Computer in 1977
    - IBM followed suit in 1981 with the IBM Personal Computer
  - Computers economical enough for personal use
  - Stand-alone units
- Distributed computing
  - Organization has a Local Area Network (LAN)
    - Computers linked to it
  - Computing distributed over the LAN, rather than at one central location

Personal, Distributed and Client/Server Computing

- Client/Server computing
  - File servers offer common store of programs that client computers access
  - C and C++ popular for writing operating systems, networking, and distributed client/server applications
  - Java used for Internet-based applications
    - Programming in Java can be more productive than C or C++

Types of programming languages

1. Machine languages
   – Strings of numbers giving machine specific instructions
   – Example:

```
+1300042774
+1400593419
+1200274027
```

2. Assembly languages
   – English-like abbreviations representing elementary computer operations (translated via assemblers)
   – Example:

```
LOAD    BASEPAY
ADD     OVERPAY
STORE   GROSSPAY
```

Types of programming languages

3. High-level languages
   - Similar to everyday English and use mathematical notations (translated via compilers)
   - Example:

```
grossPay = basePay + overTimePay
```

History of C++

- C++ evolved from C
    - C evolved from two previous programming languages, BCPL and B
    - ANSI C established worldwide standards for C programming
- C++ “spruces up” C
    - Provides capabilities for object-oriented programming
        - Objects - reusable software components that model things in the real world
    - Object-oriented programs are easy to understand, correct and modify

History of Java

- Java
  - Based on C and C++
  - Originally developed in early 1991 for intelligent consumer electronic devices
    - Market did not develop, project in danger of being cancelled
  - Internet exploded in 1993, saved project
    - Used Java to create web pages with dynamic content
  - Java formally announced in 1995
  - Now used to create web pages with interactive content, enhance web servers, applications for consumer devices (cell phones)...

History of Java

- Jmaes Gausling initiated java language project in June 1991.
- Language originally called Ock
- Also known as green.
- Ended up with name java from a list of random word.
- Sun released the first public implementation as java 1.0. conceived by James Gosling, Patrick Naughton, Chris Warth, Ed Frank and Mike Sheridan at Sun Microsystems.
- Sun proposed WORA (Write Once, Run Anywhere).
- With the advent of Java 2 (released initially as J2SE 1.2 in December 1998) multiple configuration build for different types of platforms.
  - J2EE
  - J2ME
  - J2SE

History of Java

- On 13th November 2006, Sun released much of Java as free and open source software under the terms of the GNU (GNU's Not Unix) General public license (GPL).
- On 8th May 2007 Sun finished the process.
- Java versions

1. JDK Alpha and Beta (1995)
2. JDK 1.0 (January 23, 1996)
3. JDK 1.1 (February 19, 1997)
4. J2SE 1.2 (December 8, 1998)
5. J2SE 1.3 (May 8, 2000)
6. J2SE 1.4 (February 6, 2002)
7. J2SE 5.0 (September 30, 2004)

History of Java

8. Java SE 6 (December 11, 2006)
9. Java SE 7 (July 28, 2011)
10. Java SE 8 (March 18, 2014)
11. Java SE 9 (September, 2017)
12. Java SE 10 (March, 2018)
13. Java SE 11 (September, 2018)
14. Java SE 12 (March, 2019)
15. Java SE 13 (September, 2019)
16. Java SE 14 (March, 2020)
17. Java SE 15 (September, 2020)
18. Java SE 16 (March, 2021)
19. Java SE 17 (September, 2021)
20. Java SE 18 (March, 2022)

History of Java

21. Java SE 19 (September, 2022)
22. Java SE 20 (March, 2023)
23. Java SE 21 (September, 2023)
24. Java SE 22 (March, 2024)

Source: https://en.wikipedia.org/wiki/Java_version_history

Two Parts of Java

- Java programs
  - Consist of pieces called classes
  - Classes contain methods, which perform tasks
- Class libraries
  - Also known as Java API (Applications Programming Interface)
  - Rich collection of predefined classes, which you can use
- Two parts to learning Java
  - Learning the language itself, so you can create your own classes
  - Learning how to use the existing classes in the libraries

Other High-Level Languages

- A few other high-level languages have achieved broad acceptance
  - FORTRAN (FORmula TRANslator)
    - Scientific and engineering applications
  - COBOL (COmmon Business Oriented Language)
    - Used to manipulate large amounts of data
  - Pascal
    - Intended for academic use
  - BASIC
    - Developed in 1965 as a simple language to help novices
  - Python
    - **It** is an interpreted, high-level and general-purpose programming language.

Structured Programming

- Structured programming
  - Disciplined approach to writing programs
  - Clear, easy to test and debug, and easy to modify
  - Pascal designed to teach structured programming in universities
    - Not used in industrial or commercial applications
- Multitasking
  - Specifying that many activities run in parallel
  - C and C++ (older version) only allow one activity to be performed at a time
  - Java allows multithreading, where activities can occur in parallel

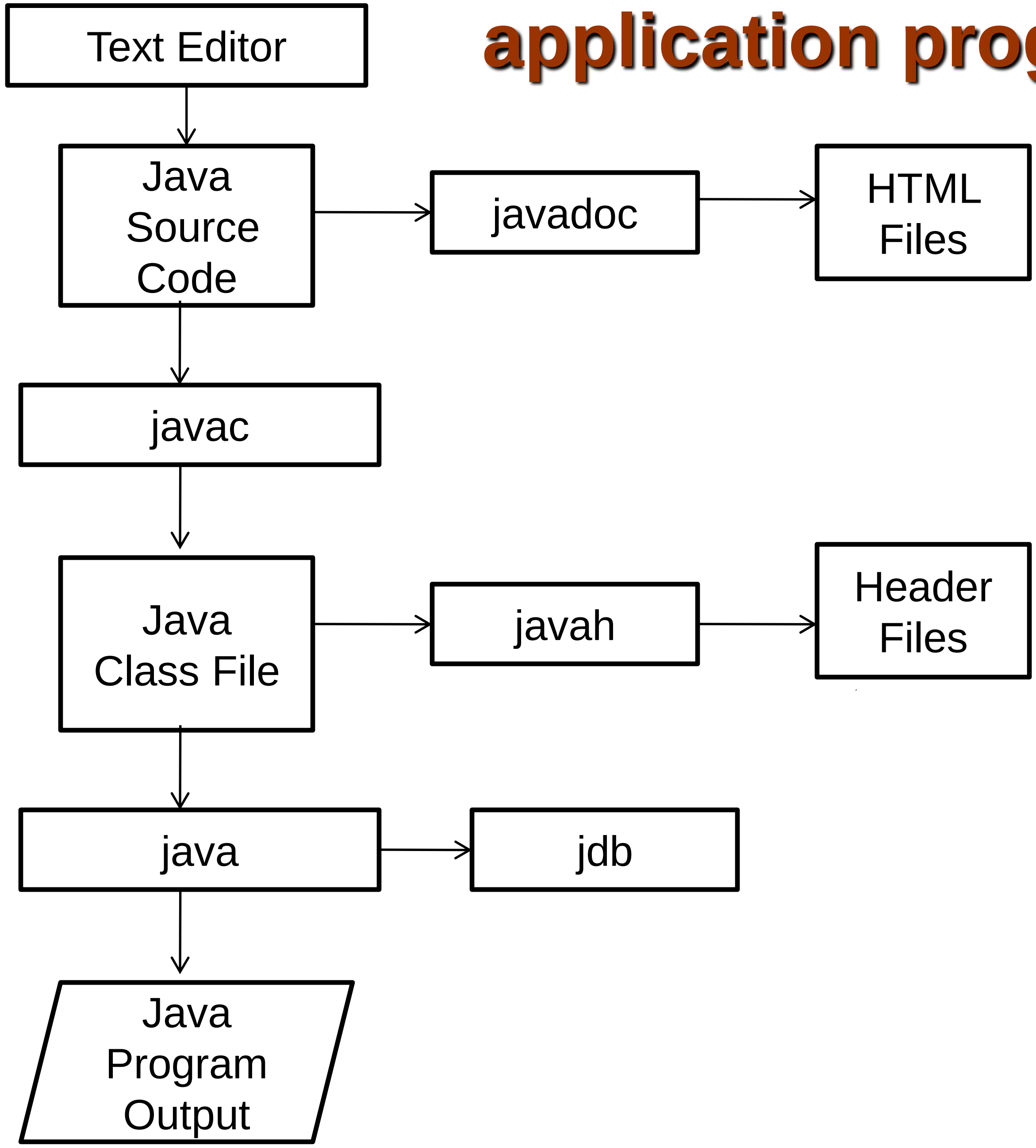
The Internet and the World Wide Web

- The Internet
  - Initially funded by the Department of Defense
  - The Internet began to take shape in the late 1960s with ARPANET, the precursor to the modern Internet, and it evolved through the 1970s and 1980s before becoming widely accessible in the 1990s.
  - Originally designed to link universities, now accessible by hundreds of millions of computers
- World Wide Web
  - Allows users to view multimedia-based documents
  - Internet has exploded
    - Mixes computing and communication
    - Changes the way business is done
    - Information instantly accessible

Primary Goals of building Java

1. It should be "simple, object-oriented and familiar"
2. It should be "robust and secure"
3. It should be "architecture-neutral and portable"
4. It should execute with "high performance"
5. It should be "interpreted, threaded, and dynamic"

Process of building and running Java application program



Java Platform

- One characteristic of Java is portability, which means that computer programs written in the Java language must run similarly on any hardware/operating-system platform. This is achieved by compiling the Java language code to an intermediate representation called Java bytecode, instead of directly to platform-specific machine code.
- Java bytecode instructions are analogous to machine code, but are intended to be interpreted by a virtual machine (VM) written specifically for the host hardware.
- End-users commonly use a Java Runtime Environment (JRE) installed on their own machine for standalone Java application, or web browser for java applets.

Java Platform

- An static or ahead-of-time (AOT) compiler is a compiler that implements ahead-of-time compilation. This refers to the act of compiling an intermediate language, such as Java bytecode.
- This achieves good performance compared to interpretation, at the expense of portability.
- A major benefit of using bytecode is porting. However, the overhead of interpretation means that interpreted programs almost always run more slowly than programs compiled to native executable would.
- Just-in-Time (JIT) compilers were introduced from an early stage that compile bytecodes to machine code during runtime.
- JIT translate java bytecode into native code the first time that code is executed then caches it.
- This results in a program that start and executes faster than pure interpreted code can, at the cost of introducing occasional compilation overhead during execution.

Java Platform

- More sophisticated VMs also use dynamic recompilation where the system may recompile some part of a program *during execution*. By compiling during execution, the system can tailor the generated code to reflect the program's run-time environment, and perhaps produce more efficient code by exploiting information that is not available to a traditional static compiler.

Java Platform

- The HotSpot execution process combines interpretation, profiling, and dynamic compilation. Rather than convert all bytescodes into machine code before they are executed, HotSpot first runs as an interpreter and only compiles the "hot" code -- the code executed most frequently. As it executes, it gathers profiling data, used to decide which code sections are being executed frequently enough to merit compilation. Only compiling code that is executed frequently has several performance advantages: No time is wasted compiling code that will execute infrequently, and the compiler can, therefore, spend more time on optimization of hot code paths because it knows that the time will be well spent. Furthermore, by deferring compilation, the compiler has access to profiling data, which can be used to improve optimization decisions, such as whether to inline a particular method call. This is known as HotSpot dynamic compilation.

Java Platform

- JIT compilation and dynamic recompilation allow java program to approach the speed of native code without loosing portability.

Java Platform

- One of the unique advantages of the concept of a runtime engine is that even the most serious errors (exceptions) in a Java program should not crash the system under any circumstances, provided the JVM itself is properly implemented.
- Moreover, in runtime engine environments such as Java there exist tools that attach to the runtime engine and every time that an exception of interest occurs they record debugging information that existed in memory at the time the exception was thrown (stack and heap values).
- These Automated Exception Handing tools provide root-cause information for exceptions in java programs that run in production, testing or development environment. Such precise debugging is much more difficult to implement without the run time support that the JVM offers.

Java and C

- C uses concept of structures (not object oriented).
- In C we use the concept of pointers whereas there are no pointers used in JAVA
- In C the programmer needs to manage memory manually. "malloc()" and "free()" are the fundamental memory allocation library calls.
- In C the declaration of variables should be on the beginning of the block.
- C supports “go to” statement, “struct” and “union” unlike Java
- C is compiled to the machines "native language" so it's execution is much faster than Java's.
- No reuse in code and by default members are public.
- C programs will have a larger memory footprint than an equivalent program written in pure machine code, but the total memory use of a C program is much smaller than the a Java program because C does not require the loading of an execution interpreter like the JVM.
- The main differences between Java and C are speed, portability, and object-orientation.

Java and C++ Similarities and Differences

- Java does not support typedefs, defines, or a preprocessor. Without a preprocessor, there are no provisions for including header files.
- Since Java does not have a preprocessor there is no concept of #define macros or manifest constants. However, the declaration of named constants is supported in Java through use of the final keyword.
- Java supports classes, but does not support structures or unions.
- All stand-alone C++ programs require a function named main and can have numerous other functions, including both stand-alone functions and functions, which are members of a class. There are no stand-alone functions in Java. Instead, there are only functions that are members of a class, usually called methods. Global functions and global data are not allowed in Java.

Java and C++
Similarities and Differences

- All classes in Java ultimately inherit from the Object class. This is significantly different from C++ where it is possible to create inheritance trees that are completely unrelated to one another.
- All function or method definitions in Java are contained within the class definition. To a C++ programmer, they may look like inline function definitions, but they aren't. Java doesn't allow the programmer to request that a function be made inline, at least not directly.
- Both C++ and Java support *class* (static) methods or functions that can be called without the requirement to instantiate an object of the class.

Java and C++ Similarities and Differences

- The interface keyword in Java is used to create the equivalence of an abstract base class containing only method declarations and constants. No variable data members or method definitions are allowed. (True abstract base classes can also be created in Java.) The interface concept is not supported by C++.
- Java does not support multiple inheritance. To some extent, the interface feature provides the desirable features of multiple inheritance to a Java program without some of the underlying problems.
- While Java does not support multiple inheritance, single inheritance in Java is similar to C++, but the manner in which you implement inheritance differs significantly, especially with respect to the use of constructors in the inheritance chain.

Java and C++ Similarities and Differences

- In addition to the access specifiers applied to individual members of a class, C++ allows you to provide an additional access specifier when inheriting from a class. This latter concept is not supported by Java.
- Java does not support the goto statement (but goto is a reserved word). However, it does support labeled break and continue statements, a feature not supported by C++. In certain restricted situations, labeled break and continue statements can be used where a goto statement might otherwise be used.
- Java does not support operator overloading.
- Java does not support automatic type conversions (except where guaranteed safe).

Java and C++
Similarities and Differences

- Unlike C++, Java has a String type, and objects of this type are immutable (cannot be modified). Quoted strings are automatically converted into String objects. Java also has a StringBuffer type. Objects of this type can be modified, and a variety of string manipulation methods are provided.
- Unlike C++, Java provides true arrays as first-class objects. There is a length member, which tells you how big the array is. An exception is thrown if you attempt to access an array out of bounds. All arrays are instantiated in dynamic memory and assignment of one array to another is allowed. However, when you make such an assignment, you simply have two references to the same array. Changing the value of an element in the array using one of the references changes the value insofar as both references are concerned.

Java and C++ Similarities and Differences

- Unlike C++, having two "pointers" or references to the same object in dynamic memory is not necessarily a problem (but it can result in somewhat confusing results). In Java, dynamic memory is reclaimed automatically, but is not reclaimed until all references to that memory become NULL or cease to exist. Therefore, unlike in C++, the allocated dynamic memory cannot become invalid for as long as it is being referenced by any reference variable.
- Java does not support pointers (at least it does not allow you to modify the address contained in a pointer or to perform pointer arithmetic). Much of the need for pointers was eliminated by providing types for arrays and strings. For example, the oft-used C++ declaration char* ptr needed to point to the first character in a C++ null-terminated "string" is not required in Java, because a string is a true object in Java.

Java and C++ Similarities and Differences

- A class definition in Java looks similar to a class definition in C++, but there is no closing semicolon. Also forward reference declarations that are sometimes required in C++ are not required in Java.
- The scope resolution operator (::) required in C++ is not used in Java. The dot is used to construct all fully-qualified references. Also, since there are no pointers, the pointer operator (->) used in C++ is not required in Java.
- In C++, static data members and functions are called using the name of the class and the name of the static member connected by the scope resolution operator. In Java, the dot is used for this purpose.

Java and C++
Similarities and Differences

- Like C++, Java has primitive types such as int, float, etc. Unlike C++, the size of each primitive type is the same regardless of the platform. There is no unsigned integer type in Java. Type checking and type requirements are much tighter in Java than in C++.
- Unlike C++, Java provides a true boolean type.
- Conditional expressions in Java must evaluate to boolean rather than to integer, as is the case in C++. Statements such as if(x+y)... are not allowed in Java because the conditional expression doesn't evaluate to a boolean.
- The char type in C++ is an 8-bit type that maps to the ASCII (or extended ASCII) character set. The char type in Java is a 16-bit type and uses the Unicode character set (the Unicode values from 0 through 127 match the ASCII character set).

Java and C++ Similarities and Differences

- Unlike C++, the >> operator in Java is a "signed" right bit shift, inserting the sign bit into the vacated bit position. Java adds an operator that inserts zeros into the vacated bit positions.
- C++ allows the instantiation of variables or objects of all types either at compile time in static memory or at run time using dynamic memory. However, Java requires all variables of primitive types to be instantiated at compile time, and requires all objects to be instantiated in dynamic memory at runtime. Wrapper classes are provided for all primitive types except byte and short to allow them to be instantiated as objects in dynamic memory at runtime if needed.
- C++ requires that classes and functions be declared before they are used. This is not necessary in Java.

Java and C++ Similarities and Differences

- The "namespace" issues prevalent in C++ are handled in Java by including everything in a class, and collecting classes into packages.
- C++ requires that you re-declare static data members outside the class. This is not required in Java.
- In C++, unless you specifically initialize variables of primitive types, they will contain garbage. Although local variables of primitive types can be initialized in the declaration, primitive data members of a class cannot be initialized in the class definition in C++.
- In Java, you can initialize primitive data members in the class definition. You can also initialize them in the constructor. If you fail to initialize them, they will be initialized to zero (or equivalent) automatically.

Java and C++ Similarities and Differences

- Like C++, Java supports constructors that may be overloaded. As in C++, if you fail to provide a constructor, a default constructor will be provided for you. If you provide a constructor, the default constructor is not provided automatically.
- All objects in Java are passed by reference, eliminating the need for the *copy constructor* used in C++.

Java and C++ Similarities and Differences

- There are no destructors in Java. Unused memory is returned to the operating system by way of a garbage collector, which runs in a different thread from the main program. This leads to a whole host of subtle and extremely important differences between Java and C++.
- Like C++, Java allows you to overload functions. However, default arguments are not supported by Java. .
- Unlike C++, several "data structure" classes are contained in the "standard" version of Java. More specifically, they are contained in the standard class library that is distributed with the Java Development Kit (JDK). For example, the standard version of Java provides the containers Vector and Hashtable that can be used to contain any object through recognition that any object is an object of type Object. However, to use these containers, you must perform the appropriate upcasting and downcasting, which may lead to efficiency problems.

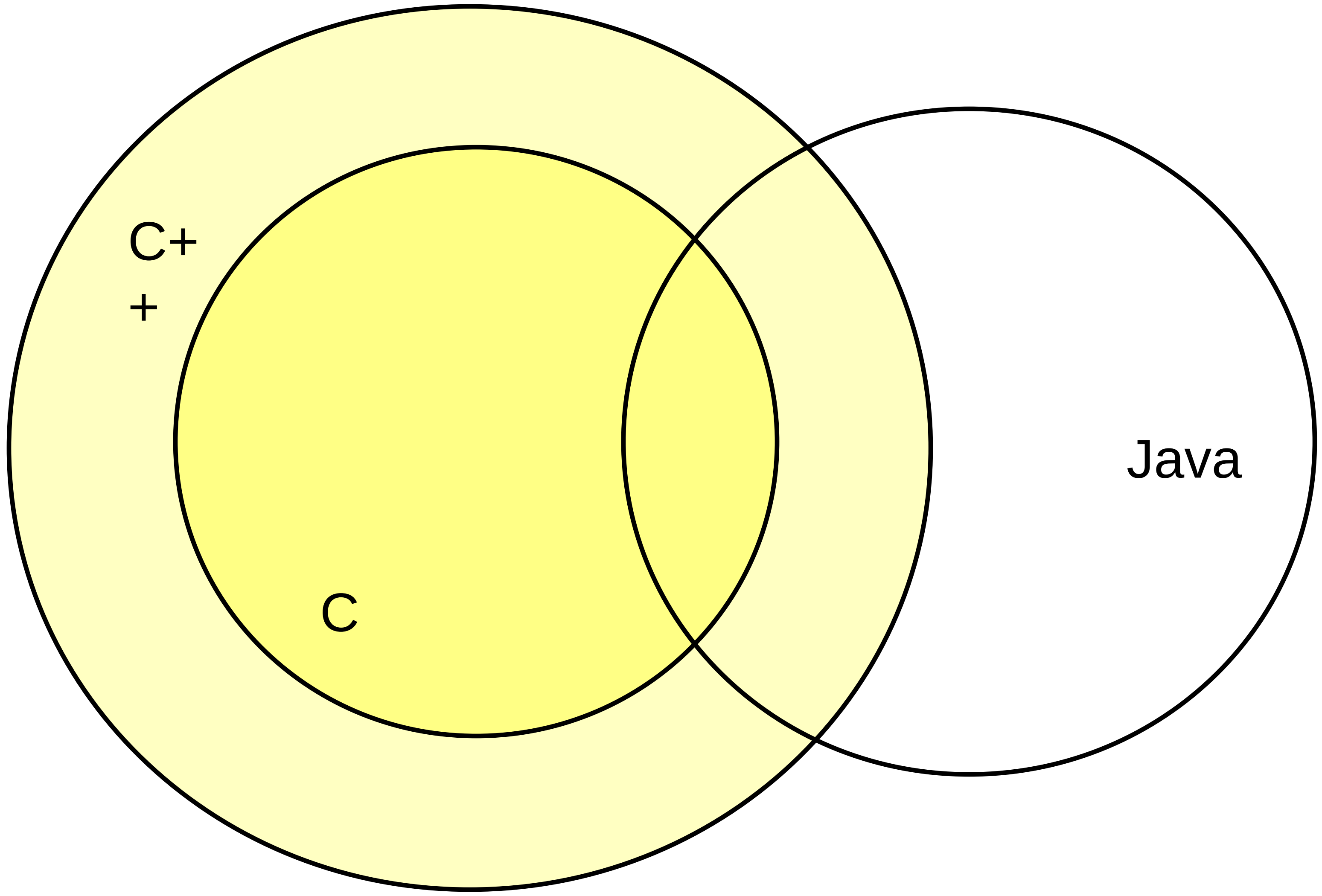
Java and C++
Similarities and Differences

- Multithreading is a standard feature of the Java language.
- Although Java uses the same keywords as C++ for access control: private, public, and protected, the interpretation of these keywords is significantly different between Java and C++.
- There is no virtual keyword in Java. All non-static methods always use dynamic binding, so the virtual keyword isn't needed for the same purpose that it is used in C++.
- Java provides the final keyword that can be used to specify that a method cannot be overridden and that it can be statically bound. (The compiler may elect to make it inline in this case.)
- The detailed implementation of the exception handling system in Java is significantly different from that in C++.

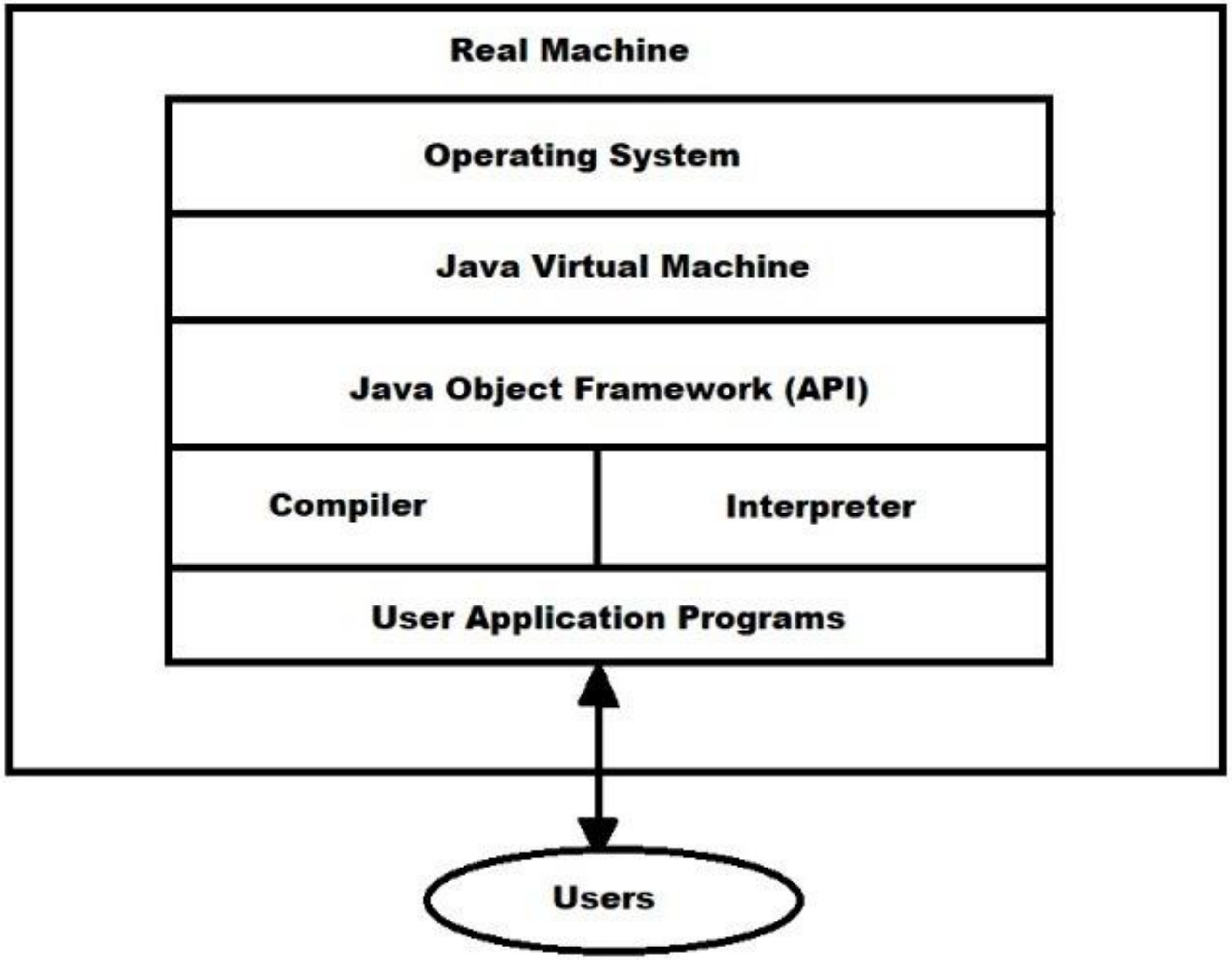
Java and C++ Similarities and Differences

- Unlike C++, Java does not support operator overloading. However, the (+) and (+=) operators are automatically overloaded to concatenate strings, and to convert other types to string in the process.
- As in C++, Java applications can call functions written in another language. This is commonly referred to as native methods. However, applets cannot call native methods.
- Unlike C++, Java has built-in support for program documentation. Specially written comments can be automatically stripped out using a separate program named javadoc to produce program documentation.
- Generally Java is more robust than C++ due to the following:
  - Object handles (references) are automatically initialized to null.
  - Handles are checked before accessing, and exceptions are thrown in the event of problems.
  - You cannot access an array out of bounds.
  - Memory leaks are prevented by automatic garbage collection.

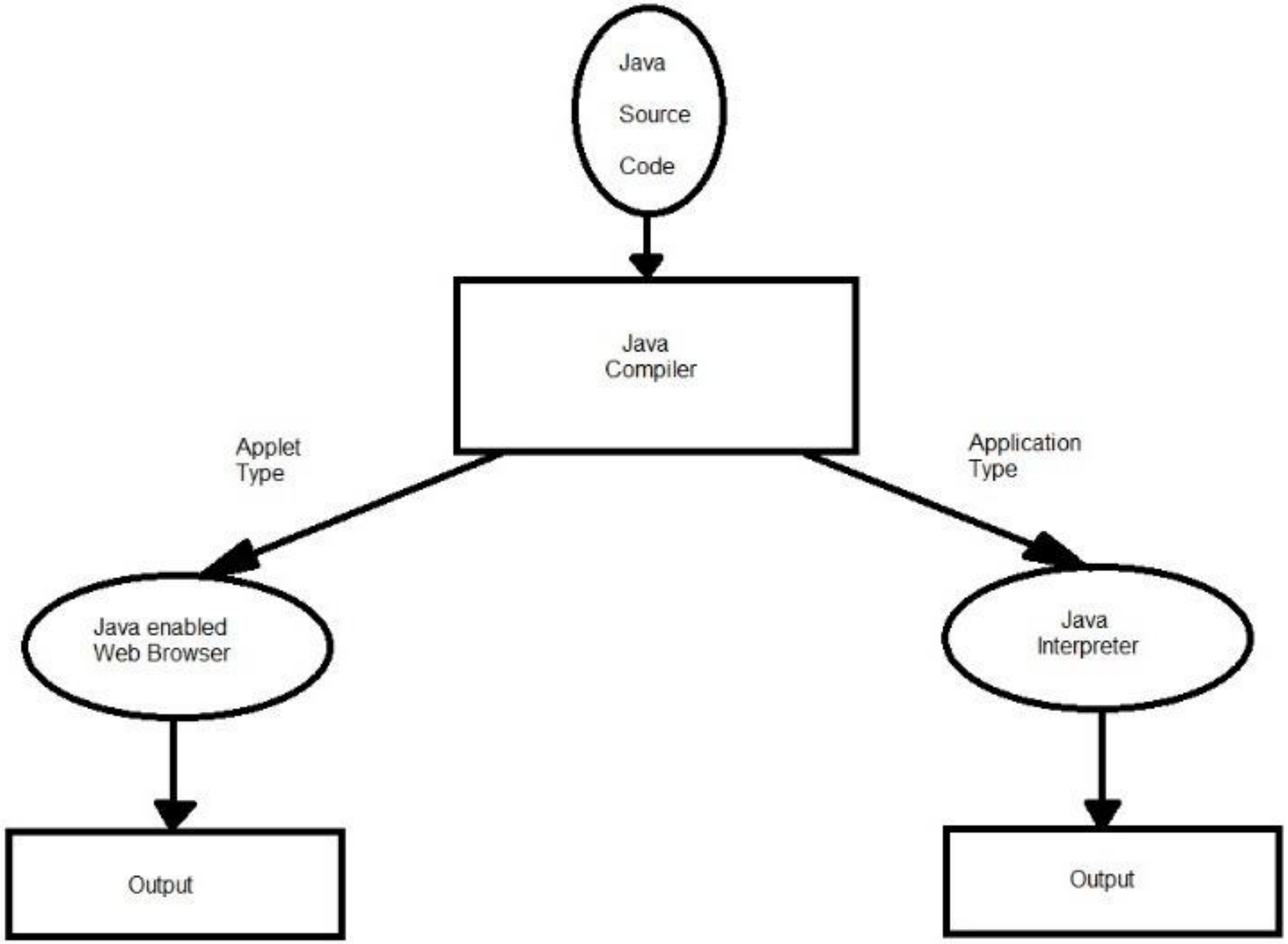
Overlapping of C, C + +, and Java



Layers of interactions for java program



Two ways of using java



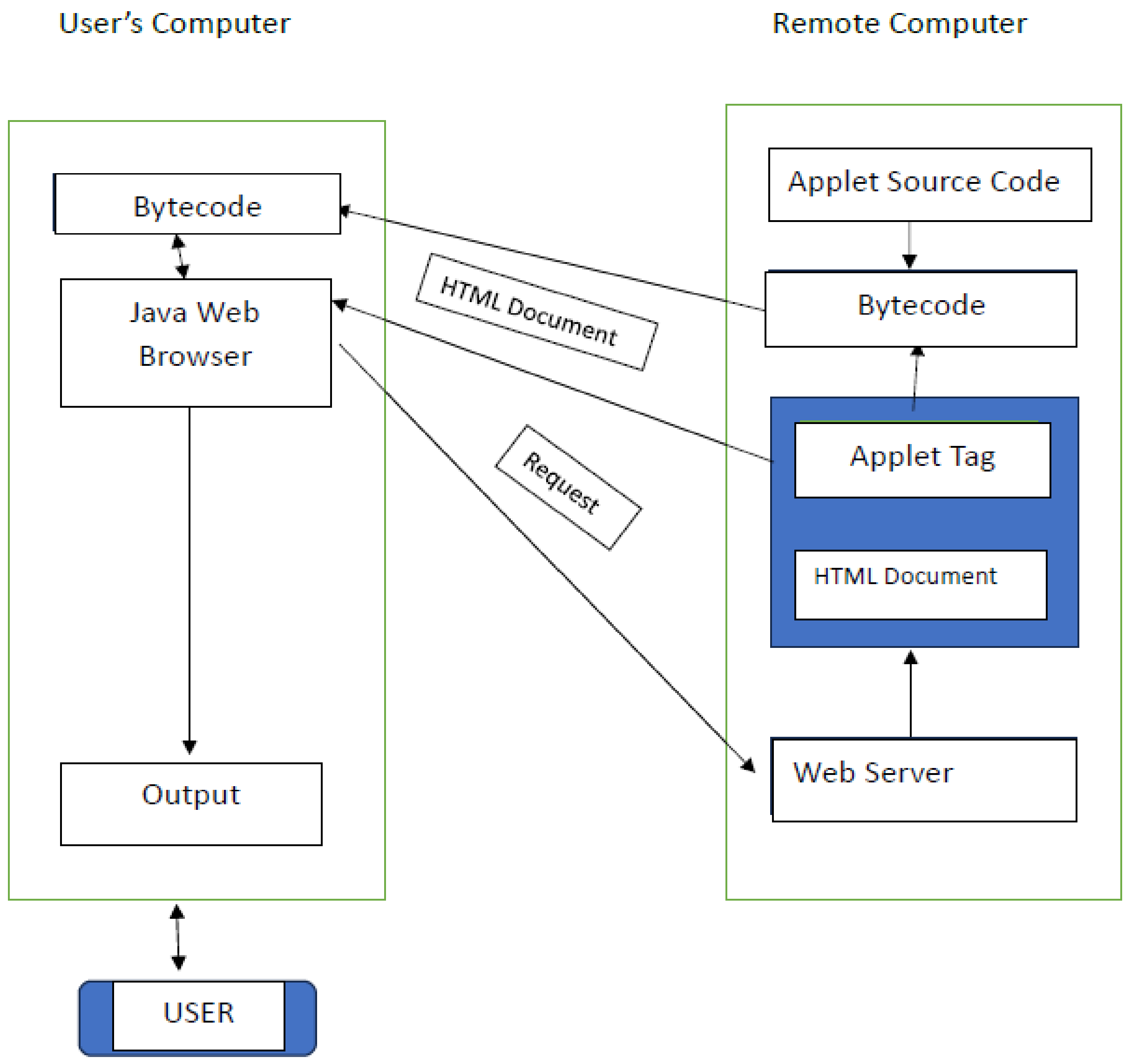
Java and internet

- In a network two very broad categories of objects are transmitted between the server and your personal computer.
    - Passive information: email
    - Dynamic active programs: active agent on the client computer, yet is initiated by the server.
- Active programs present serious problems in the areas of security and portability.
- Prior to java, cyberspace was effectively closed to half the entities that now live there.
- Java address those concerns and open the door to an exiting new form of program: the applet.
- An applet is an application designed to be transmitted over the internet and executed by a java compatible web browser.
- Prior to java, most users did not download executable programs frequently, and those who did scanned them for viruses prior to execution.

Java and internet

- When you use a java compatible web browser, you can safely download java applets without fear of viral infection or malicious intent.
- Java achieves this protection by confining a java program to the java execution environment and not allowing it access to other parts of the computer.

Java interaction with the web



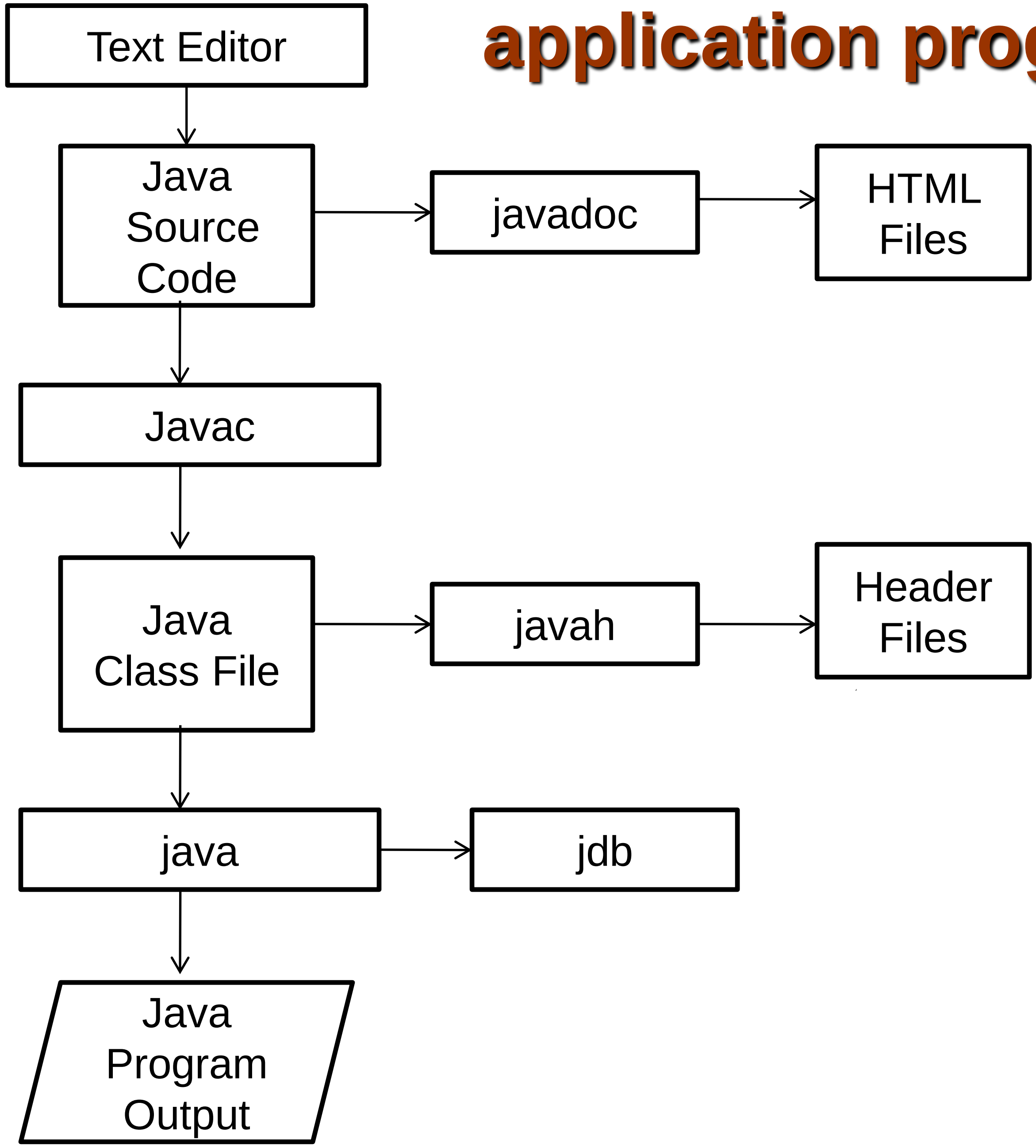
Java Environment

- Java Environment includes a large number of development tools and hundreds of classes and methods. The development tools are part of the system known a Java Development Kit (JDK) and the classes and methods are part of the Java Standard Library (JSL) and also known as the Application Programming Interface(API).

Java Development Kit

- Appletviewer: Enables us to run Java applets (without actually using a Java-compatible browser).
- Javac: The Java compiler, which translates Java sourecode to bytecode.
- Java: Java interpreter, which runs applets and applications by reading and interpreting bytecode files.
- Javap: Java disassembler – convert bytecode files into program description.
- Javah: Produces header *files* for *use with native method.*
- Javadoc: Creates HTML - format documentation for .java source code files
- Jdb: Java debugger, which helps us to find errors in our program.

Process of building and running java application program



Process of building and running java application program

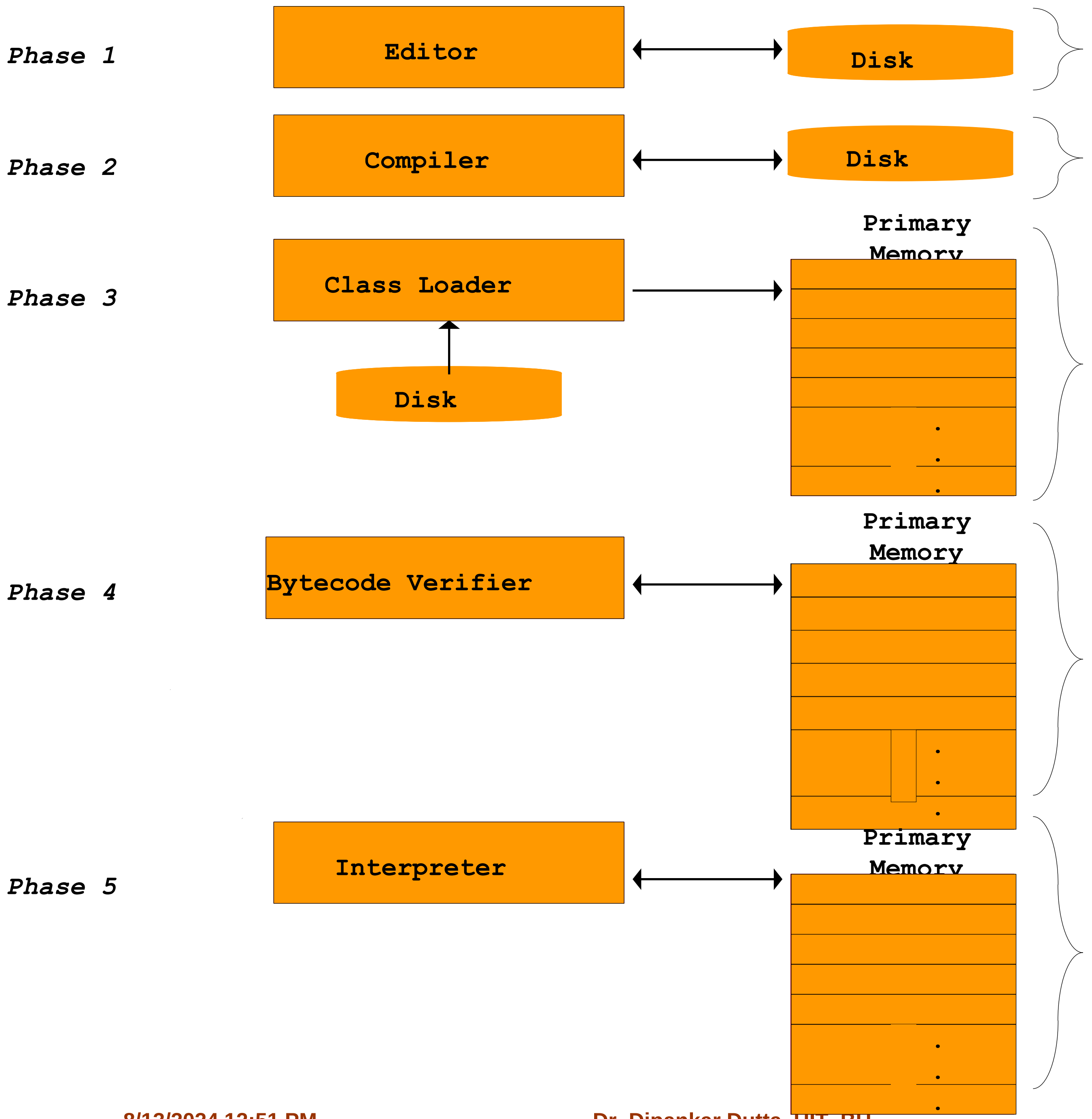
- Java programs have five phases
  - Edit
    - Use an editor to type Java program
    - **`vi`** or **`emacs`**, notepad, Jbuilder, Visual J++, Eclips, Neatneans
    - **`.java`** extension
  - Compile
    - Translates program into bytecodes, understood by Java interpreter
    - **`javac`** command: **`javac myProgram.java`**
    - Creates **`.class`** file, containing bytecodes (**`myProgram.class`**)

Process of building and running java application program

- Loading
  - Class loader transfers **`.class`** file into memory
    - Applications - run on user's machine
    - Applets - loaded into Web browser, temporary
  - Classes loaded and executed by interpreter with **`java`** command
  - **`java Welcome`**
  - HTML documents can refer to Java Applets, which are loaded into web browsers. To load,
  - **`appletviewer Welcome.html`**
    - **`appletviewer`** is a minimal browser, can only interpret applets

Process of building and running java application program

- Verify
  - Bytecode verifier makes sure bytecodes are valid and do not violate security
  - Java must be secure - Java programs transferred over networks, possible to damage files (viruses)
- Execute
  - Computer (controlled by CPU) interprets program one bytecode at a time
  - Performs actions specified in program
- Program may not work on first try
  - Make changes in edit phase and repeat



Program is created in the editor and stored on disk.

Compiler creates bytecodes and stores them on disk.

Class loader puts bytecodes in memory.

Bytecode verifier confirms that all bytecodes are valid and do not violate Java’s security restrictions.

Interpreter reads bytecodes and translates them into a language that the computer can understand, possibly storing data values as the program executes.

This image cannot currently be displayed.

End of Chapter 1

Questions?