Do’s and don’t’s

- To be eligible for attendance, you must respond clearly when your name is called. Requests for later updates will not be considered.
- During class, washroom and drinking water requests are not permitted.
- Active participation in class discussions on the subject is encouraged when prompted.
- Ensure you come prepared for the lab as per the guidelines provided in the lab manual.
- Maintaining a 200-page observation record book is mandatory.
- Take comprehensive notes during theory classes.

Chapter 1

Java Programming Fundamentals

Object-Oriented Programming or OOPs

- As the name suggests, Object-Oriented Programming or OOPs refers to languages that use objects in programming.
- Object-oriented programming aims to implement real-world entities like inheritance, hiding, polymorphism, etc in programming.
- The main aim of OOP is to bind together the data and the functions that operate on them so that no other part of the code can access this data except that function.
- **Ex:** Java, C++, Python, C#. Ruby, etc.

The Origins of Java

- Java is a high-level language conceived by James Gosling in 1991.
    - Initially called “Oak”, but was renamed “Java” in 1995.

- Java syntax is similar to the older languages C & C++.
    - It was not designed to replace C++.
- Original motivation – need for a platform-independent language for creating s/w to be embedded in various consumer electronic devices / Internet
    - The newest version is Java Platform, Standard Edition 23 (Java SE 23).

Progress Check

- Java is useful for the Internet because it can be used to produce ______ programs
- Java is the direct descendant of what languages?

Java Buzzwords

- Simple
  - easy to learn, inherits c/c++ syntax, left out difficult constructs like pointers, memory management.
- Pure Object oriented Language-
  - classes and objects
- Java is case sensitive language-
  - variable val is different from Val, VAL, VaL
- Robust-
  - Ability to create robust, reliable programs
    - Exception handling
    - Strictly typed language

      (checks for errors at both compile and run time)
    - Easy memory management

      (automatic deallocation-garbage collection)

Java Buzzwords

- Multi threaded
  - write programs that do many things simultaneously
- Architecture neutral
  - compiled java programs will run on variety of processors using various OSs (The goal- “ Write once, run any where, any time, forever”)
- Both compiled and interpreted
  - enables creation of cross-platform programs

Java Buzzwords

- Dynamic
  - Resolves type information at run time
- Distributed
  - Designed for distributed environment of internet because it handles TCP/IP protocol, RMI
- Security
  - Execution is confined to JVM

Java and Internet

- Java can be used to create two types of programs: ***applications*** and ***applets***

- An ***application*** is a program
  - that runs on your computer, under the operating system of that computer
- An ***applet*** is a tiny Java program
  - designed to be transmitted over the Internet
  - automatically downloaded across the network, just like an image, sound file, or video clip
  - executed by a Java-compatible Web browser

Applets

- Applets
  - Allow some functionality to be moved from server to client like
    - display data provided by the server
    - handle user inputs
    - provide simple functions (loan calculator)

    that execute locally rather than on server

  - Challenges: Introduces **security** and **portability** issues

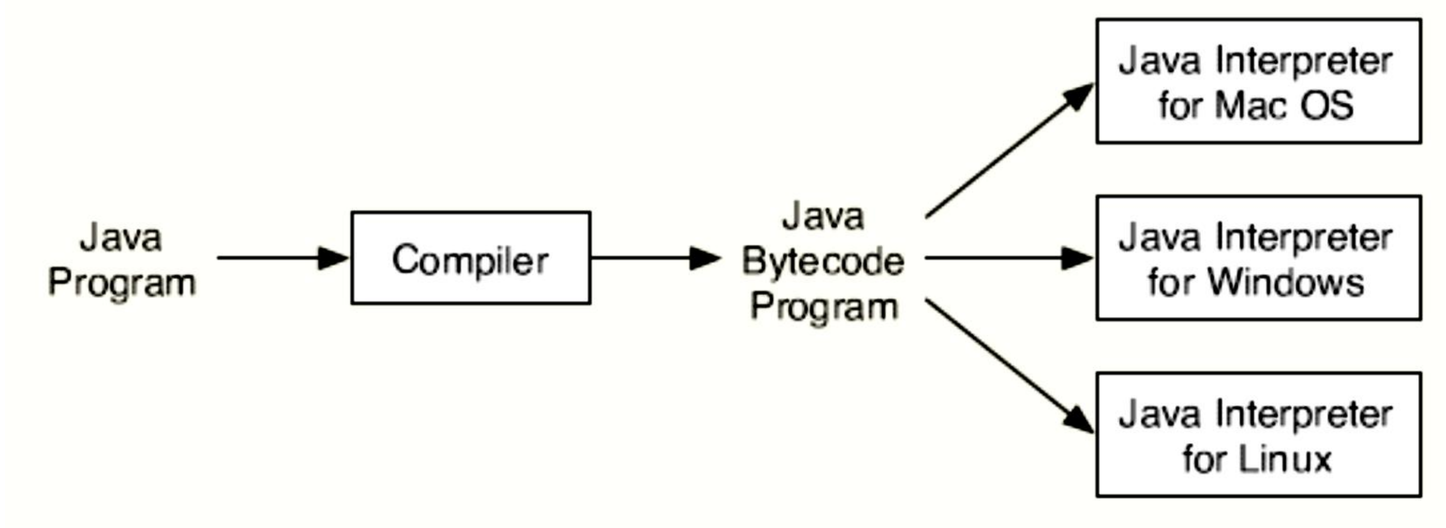
SECURITY

- Automatic downloading of applets might contain virus
  - malicious code gains unauthorized access to system resources, passwords, credit card numbers etc.
- But, when a Java-compatible Web browser used,
  - One 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 access to other parts of the computer environment

Portability

- Portability is the major aspect of Internet
  - because many different types of computers and OSs are connected to it.
- In case of applets,
  - must be able to downloaded and must be executed by wide variety of computes and OSs.
- Because Java is architecture neutral,
  - Java programs are portable.
  - They can be run on any platform without being recompiled.

Java's solution:



Java Program
Compiler
Java Bytecode Program
Java Interpreter for Mac OS
Java Interpreter for Windows
Java Interpreter for Linux

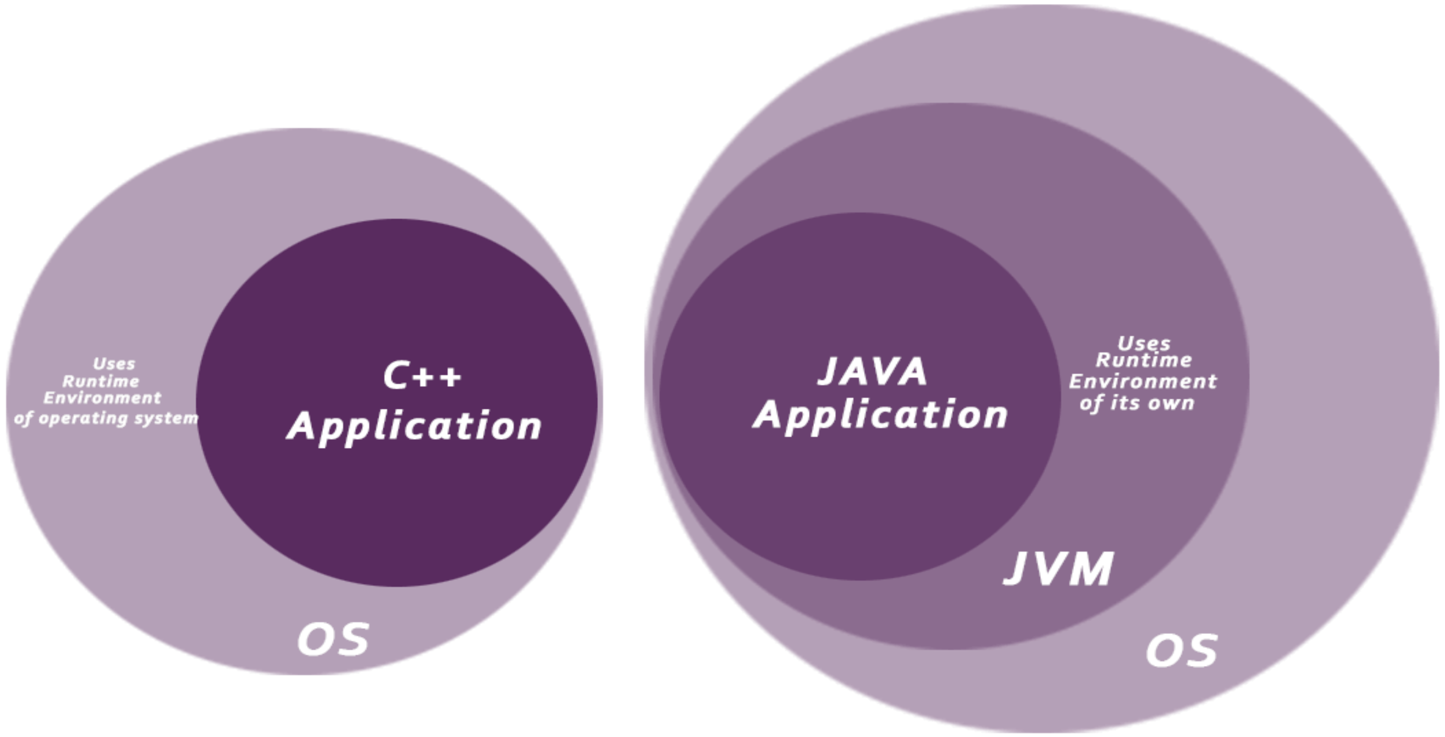
- Execution of **bytecode** by **JVM**
  - is the easiest way to create portable programs by implementing java interpreter for each platform.

Java bytecode

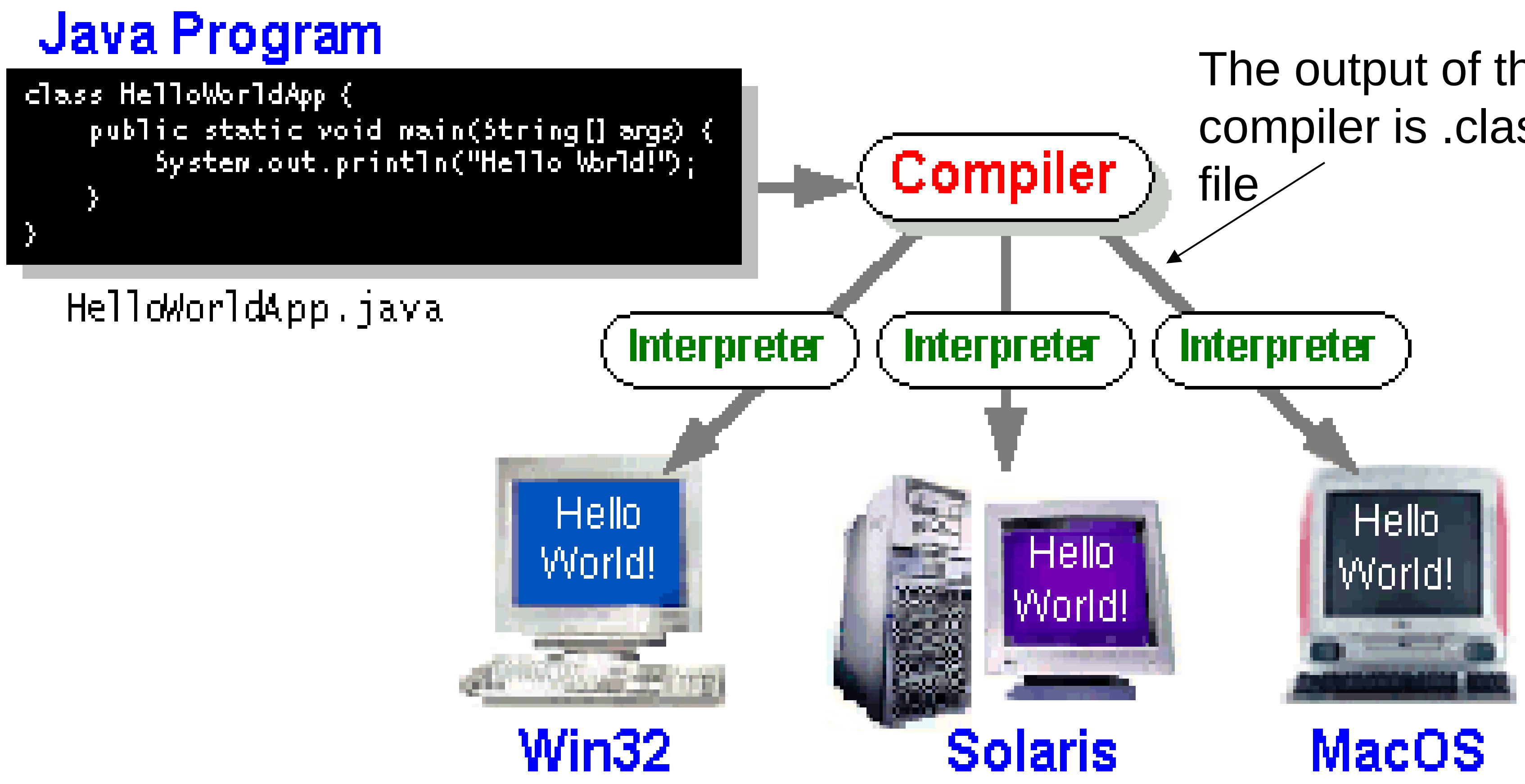
- The output of a Java compiler(javac) is
  - not executable code ; rather, it is ***bytecode***

- ***Bytecode*** is a highly optimized set of instructions
  - designed to be executed by the Java run-time system, ***Java Virtual Machine*** (*JVM*)

Java Virtual Machine(JVM)


Uses Runtime Environment of operating system
C++ Application
OS
JAVA Application
Uses Runtime Environment of its own
JVM
OS

A Picture is Worth...



```
class HelloWorldApp {
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

The output of the compiler is .class file

The Interpreter's are sometimes referred to as **Java Virtual Machines**

Java Compilers and the JVM

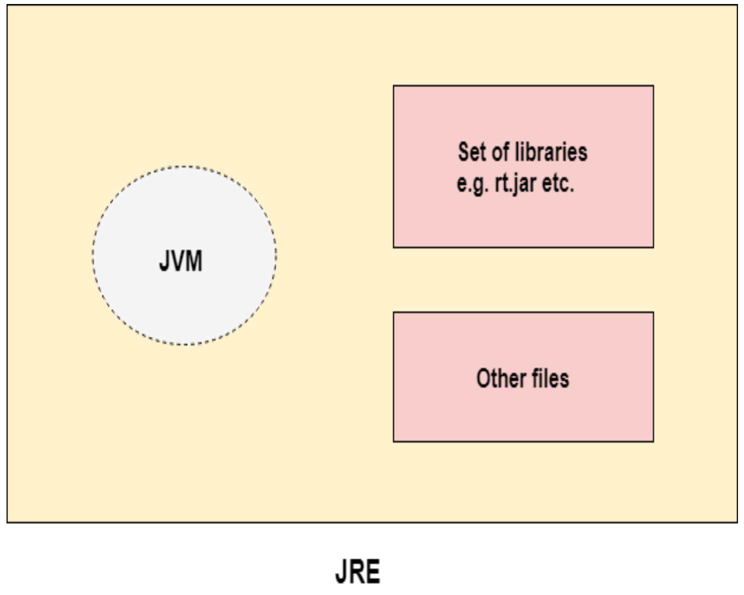
- Java compilers do not generate machine code for a CPU.
- Java compilers generate machine code for the JVM (Java Virtual Machine).
- The JVM machine code (called *bytecode*) is executed by a JVM interpreter program on each computer.

Just –in-time Compiler(JIT)

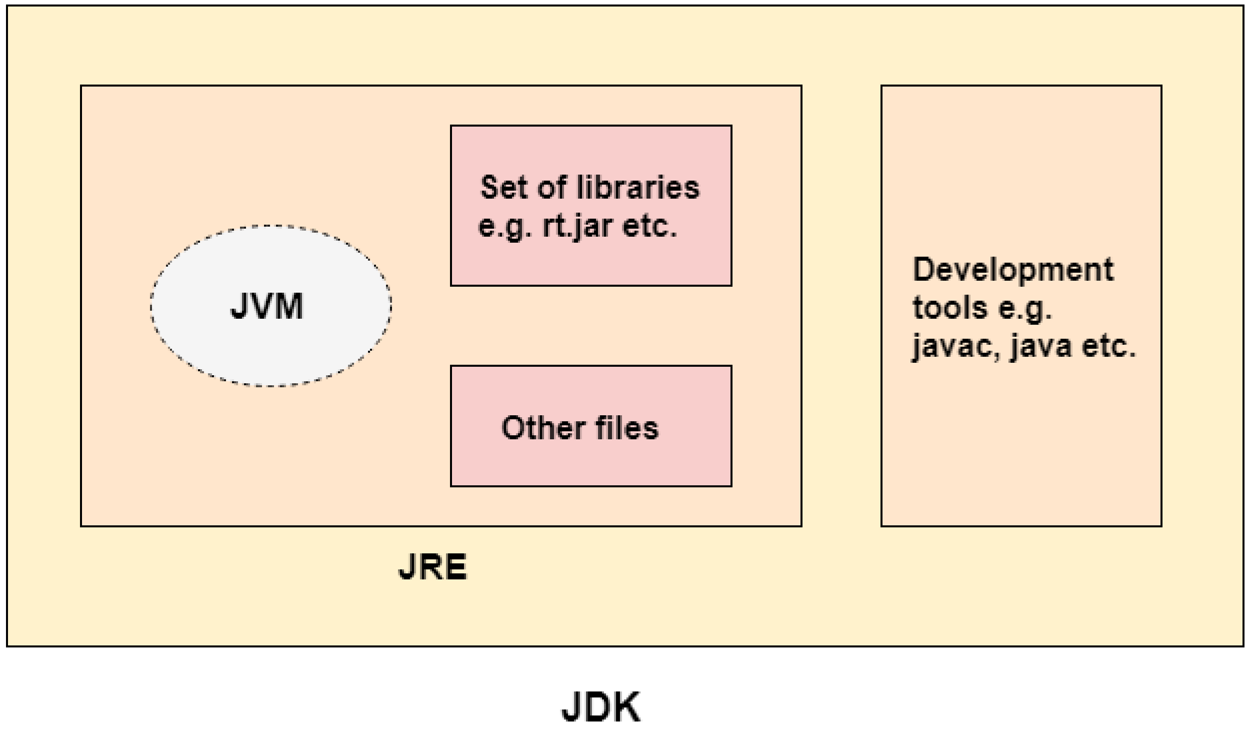
- When JIT is part of JVM
    - Selected portions of bytecode are compiled into executable code at run time
        - On a piece-by-piece, demand basis
    - It does not compile an entire java program
    - Furthermore, not all sequences of bytecode are compiled
        - only those that will benefit from compilation

Java Runtime Environment (JRE)

- a set of programming tools for developing Java applications:
    - Java Virtual Machine (JVM),
    - core classes (libraries),
    - supporting files.



Java Development Kit(JDK)


Set of libraries e.g. rt.jar etc.
JVM
Other files
Development tools e.g. javac, java etc.
JRE
JDK

Progress Check

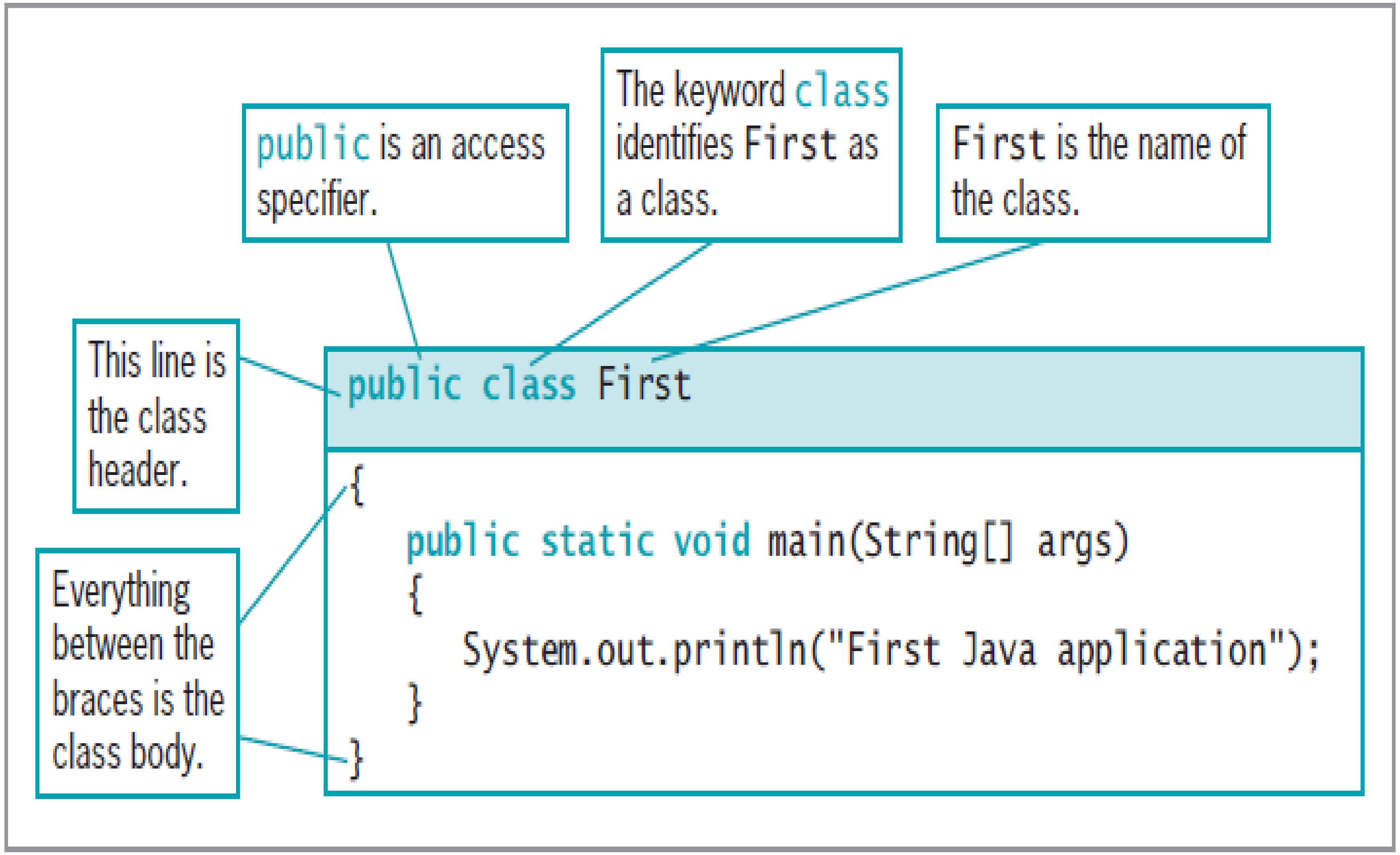
- What is an applet?
- What is a Java bytecode?
- Which two internet programming problems can be solved using bytecode?

The Java Development Kit (JDK)

- To compile and run java programs JDK must be installed in the m/c, which can be downloaded using the below link
- www.oracle.com/technetwork/java/javase/downloads/index.html
- Provide two primary programs
  - ✓ Java compiler: javac
  - ✓ Application launcher or java interpreter : java
- It runs in the command prompt environment and use command line tools

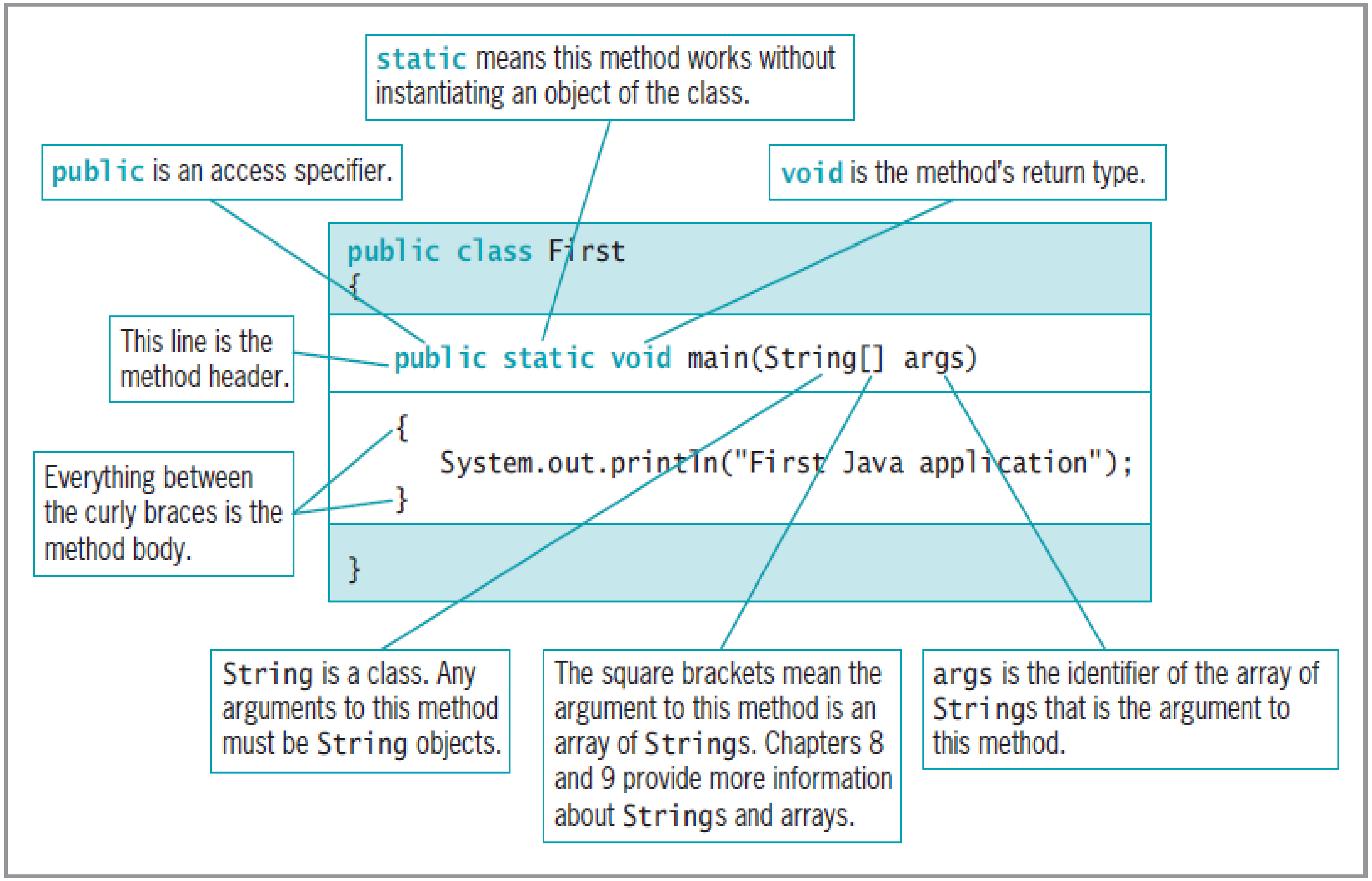
```
public class First
{
   public static void main(String[] args)
   {
      System.out.println("First Java application");
   }
}
```

Figure 1-4 The First class


public is an access specifier.
The keyword class identifies First as a class.
First is the name of the class.
This line is the class header.
Everything between the braces is the class body.

```
public class First
{
   public static void main(String[] args)
   {
      System.out.println("First Java application");
   }
}
```

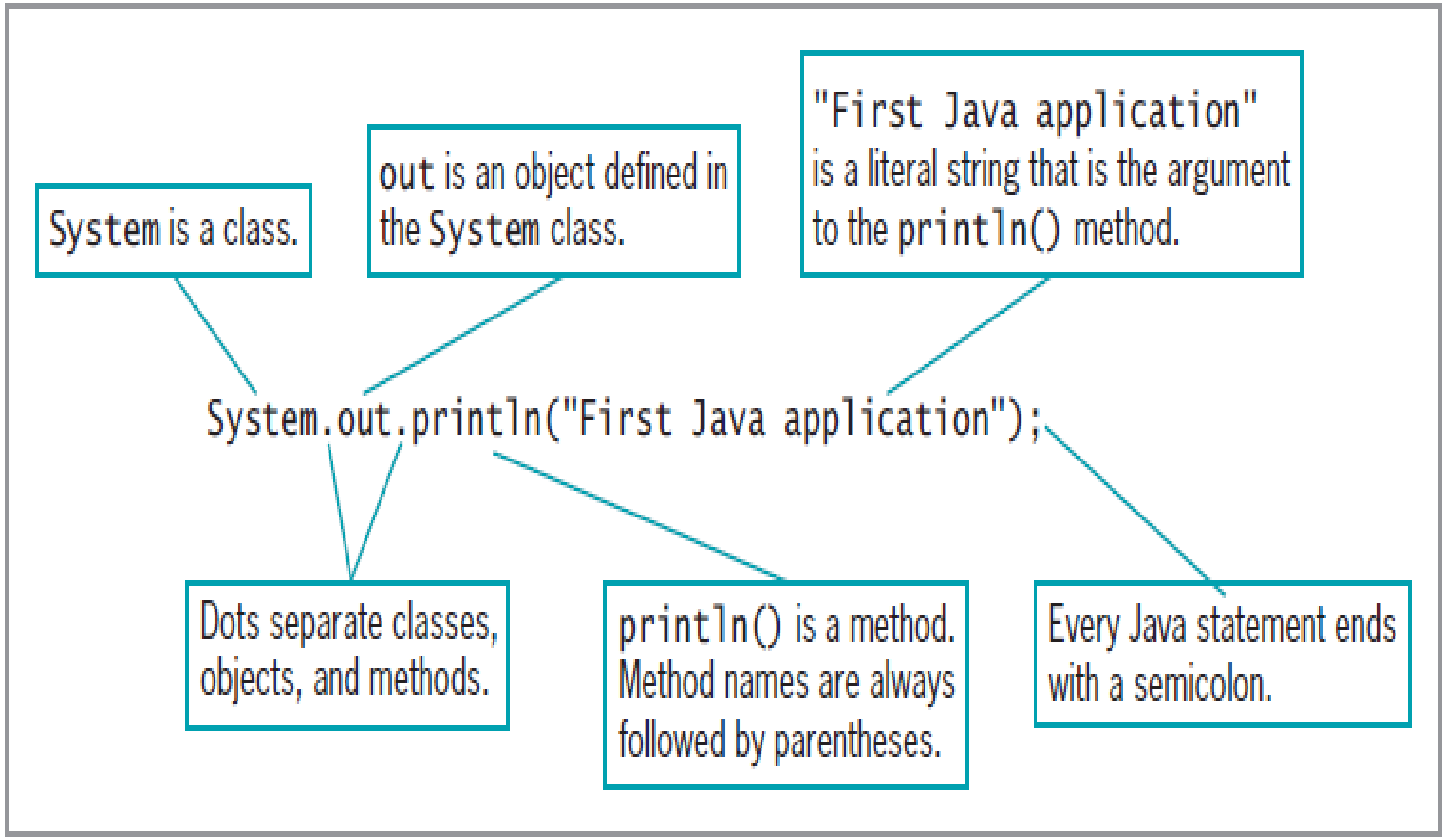
Figure 1-6 The parts of a typical class



```
public class First
{
   public static void main(String[] args)
   {
      System.out.println("First Java application");
   }
}
```

static means this method works without instantiating an object of the class.
public is an access specifier.
void is the method's return type.
This line is the method header.
Everything between the curly braces is the method body.
String is a class. Any arguments to this method must be String objects.
The square brackets mean the argument to this method is an array of Strings. Chapters 8 and 9 provide more information about Strings and arrays.
args is the identifier of the array of Strings that is the argument to this method.

Figure 1-7 The parts of a typical `main()` method


System is a class.
out is an object defined in the System class.
"First Java application" is a literal string that is the argument to the println() method.
Dots separate classes, objects, and methods.
println() is a method. Method names are always followed by parentheses.
Every Java statement ends with a semicolon.

```
System.out.println("First Java application");
```

Figure 1-5 Anatomy of a Java statement

First Simple Java Program

```
/*
   This is a simple Java program.
   Call this file Example.java.
*/
Public class Example {
  // A Java program begins with a call to main().
  public static void main(String[] args) {
    System.out.println("Java drives the Web.");
  }
}
```

- When a class member is preceded by **public**, then that member may be accessed by code outside the class in which it is declared

- The keyword **static** allows **main( )** to be called without having to instantiate a particular instance of the class

- The keyword **void** simply tells the compiler that **main( )** does not return a value

Progress Check

- Where does a Java program begin execution?
- What does System.out.println() do
- What is the name of the java compiler? What do you use to run java program?
- Why main is declared as public?
- Why main is declared as static?
- If I run a Program in commandline as **java Example 1 2 3** what is args[0]?
- What is the command to compile java in commandline?
- Name some primitive data types you studied in java.
- What is bytecode? What extension of file name it has?
- Write a Program to convert 10 gallons to liters.(1Gallon=3.7854l)

A Second Example

```
class Example2 {
  public static void main(String[] args) {
    int var1; // this declares a variable
    int var2; // this declares another variable

    var1 = 1024; // this assigns 1024 to var1
    System.out.println("var1 contains " + var1);

    var2 = var1 / 2;
    System.out.print("var2 contains var1 / 2: ");
    System.out.println(var2);
  }
}
```

A Third Example

```
class Example3 {
  public static void main(String[] args) {
    int w; // declare an int variable
    double x; // declare a floating-point variable

    w = 10; // assign w the value 10
    x = 10.0; // assign x the value 10.0
    System.out.println("Original value of w: " + w);
    System.out.println("Original value of x: " + x);
    System.out.println(); // print a blank line

    // now, divide both by 4
    w = w / 4;
    x = x / 4;
    System.out.println("w after division: " + w);
    System.out.println("x after division: " + x);
  }
}
```

Converting Gallons to Liters

```
class GalToLit {
  public static void main(String[] args) {
    double gallons; // holds the number of gallons
    double liters; // holds conversion to liters

    gallons = 10; // start with 10 gallons

    liters = gallons * 3.7854; // convert to liters
    System.out.println(gallons + " gallons is " +
                       liters + " liters.");
  }
}
```

The **if** Statement

- Simplest form:

  if (*condition*) *statement*;

- Example:

```
if(3 < 4) System.out.println("yes");
```

- Relational operators:

  <, >, <=, >=, ==, !=

Example of **if** Statements

```
class IfDemo {
  public static void main(String[] args) {
    int a, b;
    a = 2;
    b = 3;

    if(a < b)
       System.out.println("a is less than b");

    // this won't display anything
    if(a == b)
       System.out.println("you won't see this");
  }
}
```

Code Block

- A **code block** in Java (and many other programming languages) is a set of one or more statements enclosed within curly braces {}. It groups multiple statements together so they can be treated as a single unit, typically used with control structures like if, for, while, methods, or classes.
- All statements inside the code block are enclosed within an opening { and a closing } brace. The braces define the boundaries of the block.
- Unlike individual statements, a code block itself does not need to end with a semicolon (;). Each statement **inside** the block, however, must follow Java's syntax rules (e.g., statements must end with a semicolon).

Example:

```
if(w < h) {
   v = w*h;
   w = 0;
}
```

Comments in Java

- Java supports **single line and multi-line** comments very similar to C++.

Example 1

//This is an example of single line comment

Example 2

/*This is my first java program.

This will print 'Hello World‘

as the output

*/

Indentation Practices

- The Java compiler doesn’t care about indentation.
- Use indentation to make your code more readable.
  - ***Indent one level*** for each opening brace and
  - ***move back out*** after each closing brace.

Progress Check

- How is a block of code created? What does it do?
- In Java statements are terminated by a ____
- All Java statements must start and end on one line. T/F

Java Keywords

abstract	asset	boolean	break	byte	case	catch	char	class	const
continue	default	do	double	else	enum	extends	final	finally	float
for	goto	if	implements	import	instanceof	int	interface	long	native
new	package	private	protected	public	return	short	static	strictfp	super
switch	synchronized	this	throw	throws	transient	try	void	volatile	while

const and goto are reserved but not used.

Java Identifiers

- An identifier is a name given to
  - a method, variable, class or other user-defined item.
- Identifiers are one or more characters long.

- The dollar sign(**$**), the underscore(**-**), any letter of the alphabet(**a-z,A-Z**), and any digit(**0-9**) can be used in identifiers.
- Should not be a keyword or reserved word

Java Identifiers

- The **first character** in an identifier cannot be a digit.
  - e.g. **12a** is an **invalid** identifier
- **Java is case sensitive:** Upper case and lower case are different.
  - **myvar** and **MyVar** are different identifiers.
- Legal identifiers
  - Test , x, y2, maxLoad, sample34, $up, _top

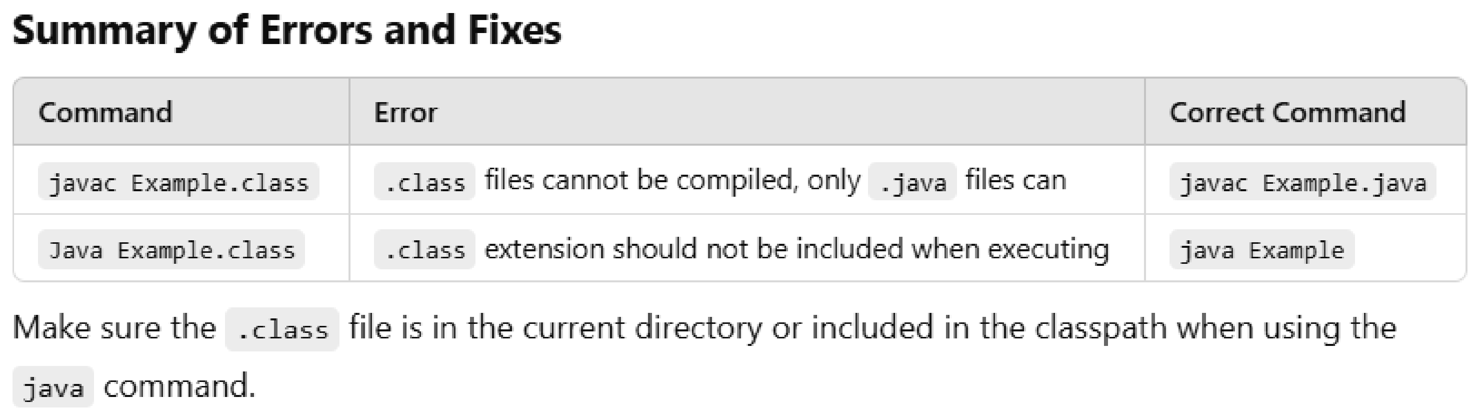
Java class Libraries

- A package which gets imported automatically to all the java programs
  - ‘java.lang’
- java.lang’ package has
  - many built-in classes and methods (System and String classes) and
  - println and print methods.
- Different packages for
  - I/O , Applets , GUI, Event handling, networking, multithreading, exception handling etc.

Exercises

- Which of the following variable names is invalid?
  - **A.** count
  - **B.** $count
  - **C.** count27
  - **D.** 67count

- What is wrong with each of the following commands
  - javac Example.class
  - Java Example.class



Summary of Errors and Fixes

Command	Error	Correct Command
`javac Example.class`	`.class` files cannot be compiled, only `.java` files can	`javac Example.java`
`Java Example.class`	`.class` extension should not be included when executing	`java Example`

Make sure the `.class` file is in the current directory or included in the classpath when using the `java` command.

Exercises

- Assume x is a variable that is declared as type int. What is wrong with each of the following statements?
  - **A.** x=3.5;
  - **B.** if(x=3) x=4;
  - **C.** x= “34”