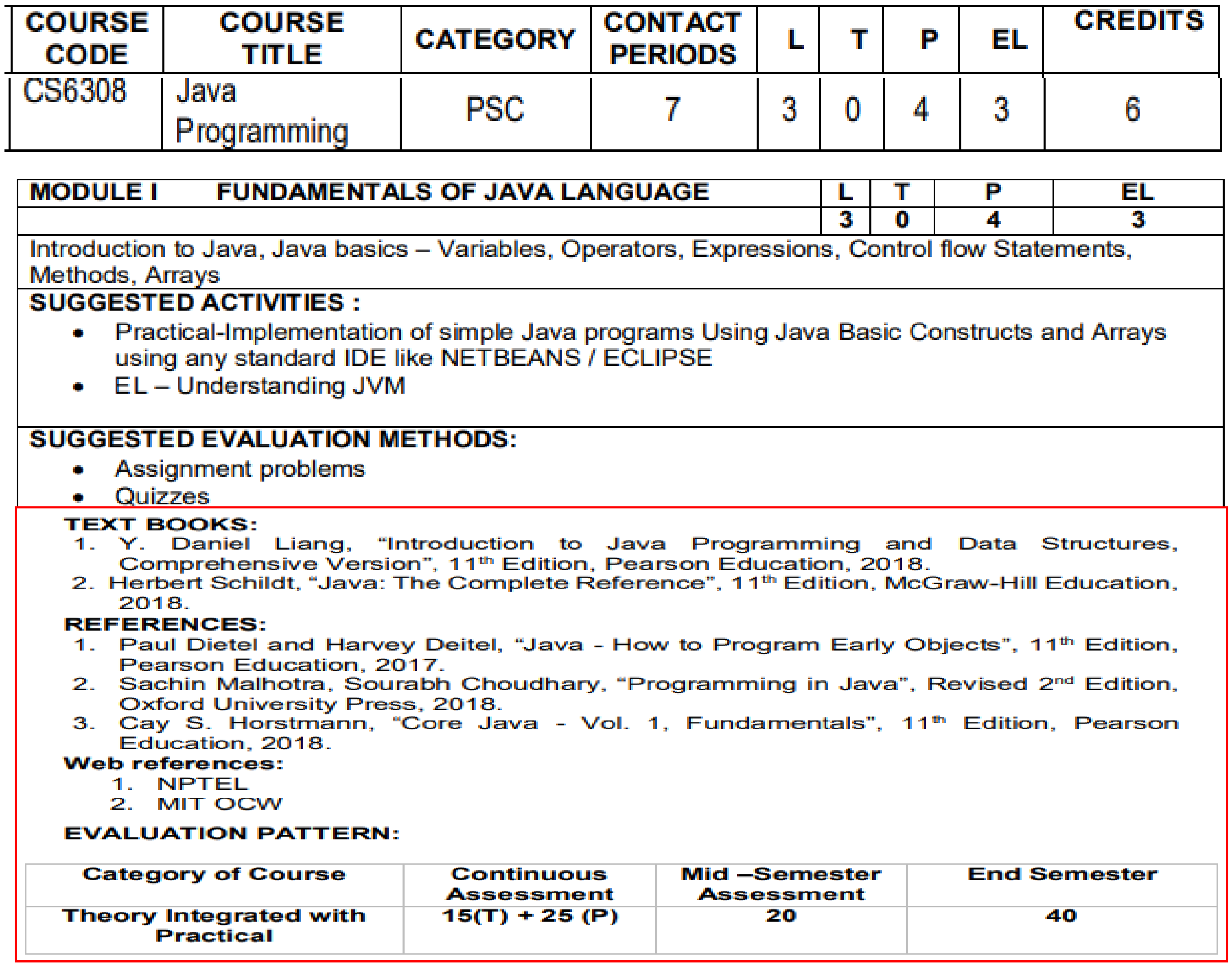
CS6308-JAVA PROGRAMMING

V P Jayachitra
Assistant Professor
Department of Computer Technology
MIT Campus
Anna University

Syllabus



COURSE CODE	COURSE TITLE	CATEGORY	CONTACT PERIODS	L	T	P	EL	CREDITS
CS6308	Java Programming	PSC	7	3	0	4	3	6

MODULE I FUNDAMENTALS OF JAVA LANGUAGE	L	T	P	EL
	3	0	4	3

Introduction to Java, Java basics – Variables, Operators, Expressions, Control flow Statements, Methods, Arrays

SUGGESTED ACTIVITIES :

- Practical-Implementation of simple Java programs Using Java Basic Constructs and Arrays using any standard IDE like NETBEANS / ECLIPSE
- EL – Understanding JVM

SUGGESTED EVALUATION METHODS:

- Assignment problems
- Quizzes

TEXT BOOKS:

1. Y. Daniel Liang, "Introduction to Java Programming and Data Structures, Comprehensive Version", 11th Edition, Pearson Education, 2018.
2. Herbert Schildt, "Java: The Complete Reference", 11th Edition, McGraw-Hill Education, 2018.

REFERENCES:

1. Paul Dietel and Harvey Deitel, "Java - How to Program Early Objects", 11th Edition, Pearson Education, 2017.
2. Sachin Malhotra, Sourabh Choudhary, "Programming in Java", Revised 2nd Edition, Oxford University Press, 2018.
3. Cay S. Horstmann, "Core Java - Vol. 1, Fundamentals", 11th Edition, Pearson Education, 2018.

Web references:

1. NPTEL
2. MIT OCW

EVALUATION PATTERN:

Category of Course	Continuous Assessment	Mid –Semester Assessment	End Semester
Theory Integrated with Practical	15(T) + 25 (P)	20	40

The Origins of Java

- Java is conceived by
  - James Gosling
  - Patrick Naughton
  - Chris Warth
  - Ed Frank, and
  - Mike Sheridan

  at Sun Microsystems in 1991.
- Known as Oak initially and renamed as Java in 1995.

Motivation

- **First, to create a platform-independent language**
  - **To create software for various consumer electronic devices, such as toasters, microwave ovens, and remote controls.**
  - **Why?**
    - **computer languages were designed to be compiled into machine code that was targeted for a specific type of CPU.**
    - **compilers are expensive and time consuming to create for all kind of CPU.**
  - **What?**
    - **portable, cross-platform language that could produce code that would run on a variety of CPUs under differing environments.**

Motivation contd…

- Second focus is the World Wide Web
  - Why?
    - Web, demanded portable programs
    - Java was an obscure language for consumer electronics.
  - What?
    - Java was propelled to the forefront of computer language design due to WWW.
    - architecture-neutral programming language
    - focus switch from consumer electronics to Internet programming.

Motivation contd…

- Third, Security
  - Java’s support for networking.
- Why?
  - library of ready-to-use functionality enabled programmers to easily write programs that accessed or made use of the Internet.
  - Downloading and executing program at multiplatformed environment may harm the system (virus, Trojan etc.,)
- What?
  - Confine an application to the Java execution environment and prevent it from accessing other parts of the computer.

Java’s Lineage: C and C++

- From C, Java inherits its syntax.
- Java’s object model is adapted from C++.
  - Java is not an enhanced version of C++
  - Neither upwardly nor downwardly compatible with C++
  - C++ was designed to solve a different set of problems.
  - Java was designed to solve a certain set of problems.

Features removed from C/C++

- typedef
- #define
- struct , union
- enum
- Functions (i.e methods in java)
- Multiple inheritance
- Goto
- Operator overloading
- pointers

What is Java?

- Programming language designed to meet the challenge of application development in the context of heterogeneous, network-wide distributed environment.
- Is a language that has proven ideal for developing *secure, high performance and highly robust* application on multiplatform.

Key features of Java

- Simple
- Secure
- Portable
- Object-oriented
- Robust
- Multithreaded
- Architecture-neutral
- Interpreted
- High performance
- Distributed
- Dynamic

Key features of Java contd…

- Simple
  - Easy to learn and use effectively.
    - no pointers/stack concerns
  - Because Java inherits the C/C++ syntax and many of the object-oriented features of C++
- Object-Oriented
  - “everything is an object” paradigm
  - Only primitive types, such as integers, are kept as high-performance nonobjects.

Key features of Java contd…

- Robust
  - Java is a strongly typed language
    - checks the code at compile time, also checks the code at run time
  - Garbage collector
    - Automatic memory management-
  - Exception handling
- Architecture-Neutral
  - JVM
    - code longevity and portability.
    - “write once; run anywhere, any time, forever.”

Key features of Java contd…

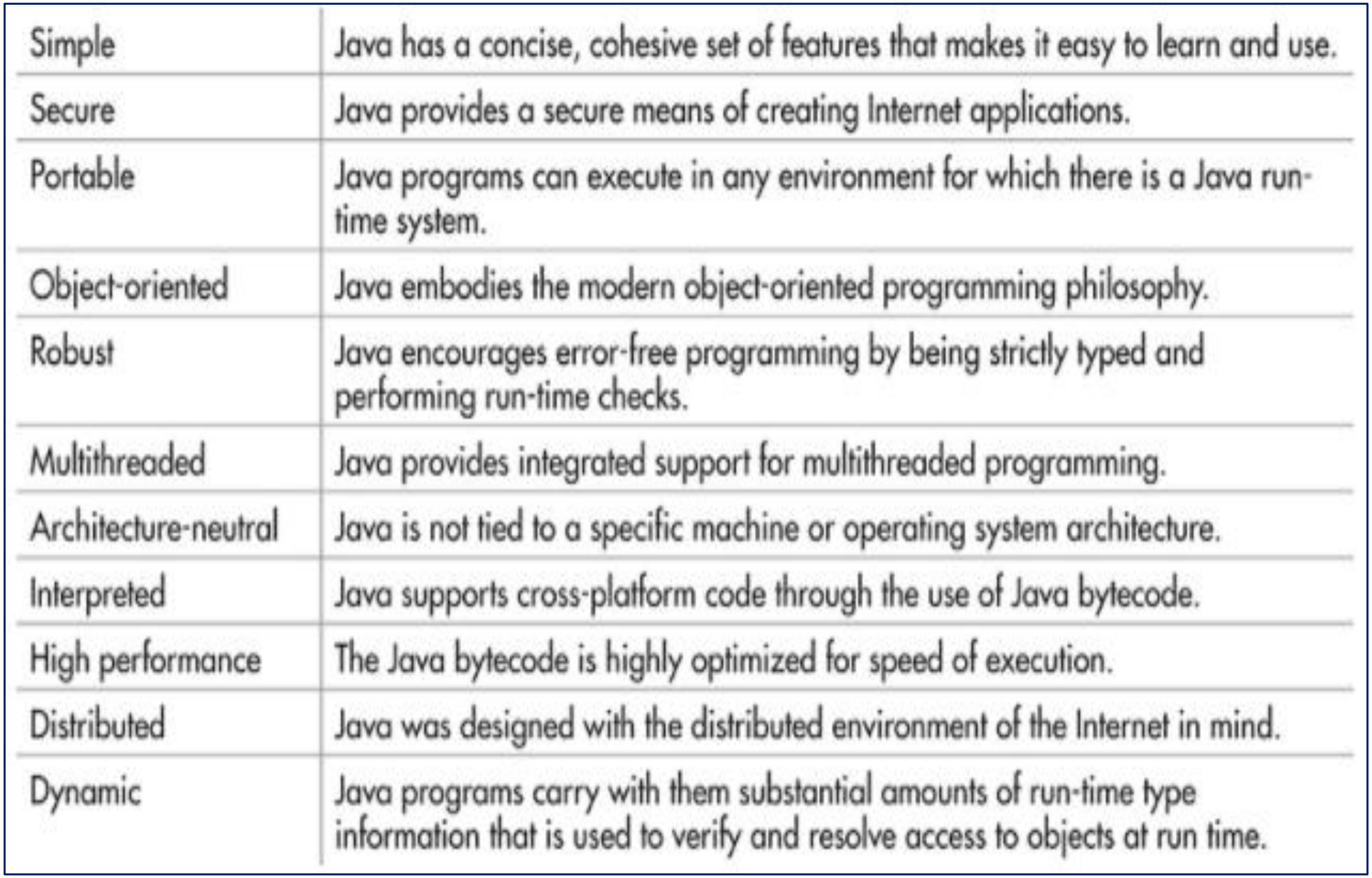
- Multithreaded
  - write programs that do many things simultaneously.
  - multiprocess synchronization -enables to construct smoothly running interactive systems
- Interpreted and High Performance
  - Java bytecode -carefully designed to translate directly into native machine code
  - For very high performance -just-in-time compiler

Key features of Java contd…

- **Distributed**
  - Java is designed for the distributed environment of the Internet as it handles TCP/IP protocols.
  - Remote Method Invocation (RMI). This feature enables a program to invoke methods across a network.

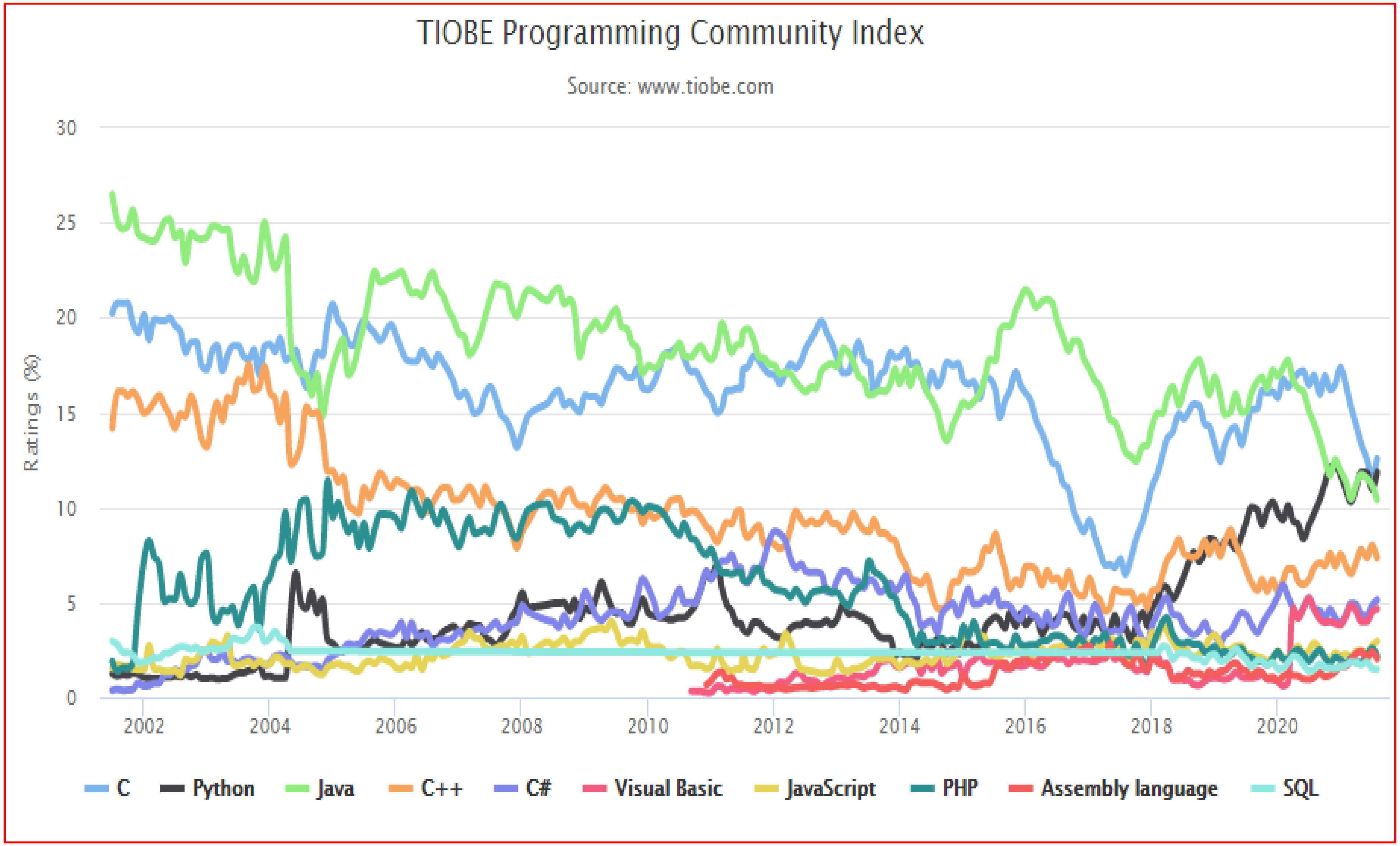
- **Dynamic**
  - run-time type information -used to verify and resolve accesses to objects at run time.

Highlights of Java



Simple	Java has a concise, cohesive set of features that makes it easy to learn and use.
Secure	Java provides a secure means of creating Internet applications.
Portable	Java programs can execute in any environment for which there is a Java run-time system.
Object-oriented	Java embodies the modern object-oriented programming philosophy.
Robust	Java encourages error-free programming by being strictly typed and performing run-time checks.
Multithreaded	Java provides integrated support for multithreaded programming.
Architecture-neutral	Java is not tied to a specific machine or operating system architecture.
Interpreted	Java supports cross-platform code through the use of Java bytecode.
High performance	The Java bytecode is highly optimized for speed of execution.
Distributed	Java was designed with the distributed environment of the Internet in mind.
Dynamic	Java programs carry with them substantial amounts of run-time type information that is used to verify and resolve access to objects at run time.

Java – 2020 popular language



Java Editions

- Java 2 Platform, Standard Edition (J2SE)
  - Desktop based application and networking applications
- Java 2 Platform, Enterprise Edition (J2EE)
  - Large-scale, distributed networking applications and Web-based applications
- Java 2 Platform, Micro Edition (J2ME)
  - Small memory-constrained devices, such as cell phones, pagers and PDAs

Java development environment

- **Text editor**
- **IDE (Integrated development environment)**

What Java can do?

- **Development tools**
  - Javac, Java
- **API**
  - XML, PACKAGE
- **Deployment tool**
  - JDK
- **UI tool**
  - javaFX, SWING
- **Integrated library**
  - JDBC, RMI

Java IDE

- Integrated Development Environment(IDE)
  - a GUI based application that facilitates application development such as coding, compilation or interpretation and debug programs more easily.
  - An IDE contains
    - Code editor-syntax highlights
    - Build tools -Compiler /Debugger/Interpreter
    - Auto documentation-comments
    - Libraries
  - Top 3 lightweight IDE -2021
    - IntelliJ IDEA
    - Eclipse
    - NetBeans

Overview of Java

- Two Paradigms
  - Procedural oriented paradigm
    - process-oriented model
    - code acting on data
    - problems with this approach appear as programs grow larger and more complex.
  - object-oriented programming paradigm
    - data controlling access to code.
      - organizes a program around its data (that is, objects) and a set of well-defined interfaces to that data.
    - problems with this approach appear as programs grow larger and more complex.

Overview of Java

- **Abstraction**
  - A powerful way to manage abstraction is through the use of hierarchical classifications.
  - To manage complexity

Overview of Java

- **The Three OOP Principle**
  - **Encapsulation**
  - **Inheritance, and**
  - **Polymorphism**

Overview of Java

- **Encapsulation: Implements information hiding and modularity**
  - mechanism that binds together code and the data it manipulates, and keeps both safe from outside interference and misuse.
  - protective wrapper that prevents the code and data from being arbitrarily accessed by other code defined outside the wrapper.
  - knows how to access it and thus can use it regardless of the implementation details—and without fear of unexpected side effects.
  - Class
    - To encapsulate complexity
    - A class defines the structure and behavior (data and code) that will be shared by a set of objects.
    - Members-code and data
      - Data-member variables or instance variables.
      - Code-member method or method
        - » behavior and interface of a class are defined by the methods that operate on its instance data.
  - objects are referred to as instances of a class.
  - Thus, a class is a logical construct; an object has physical reality.
  - Each method or variable in a class may be marked private or public.
  - The *public* interface of a class represents everything that external users of the class need to know, or may know.
  - The *private* methods and data can only be accessed by code that is a member of the class

Overview of Java

- **Inheritance:** Code reuse and code organization
  - *Inheritance* is the process by which one object acquires the properties of another object.
  - Without the use of hierarchies, each object would need to define all of its characteristics explicitly.
  - an object need only define those qualities that make it unique within its class.
  - It can inherit its general attributes from its parent

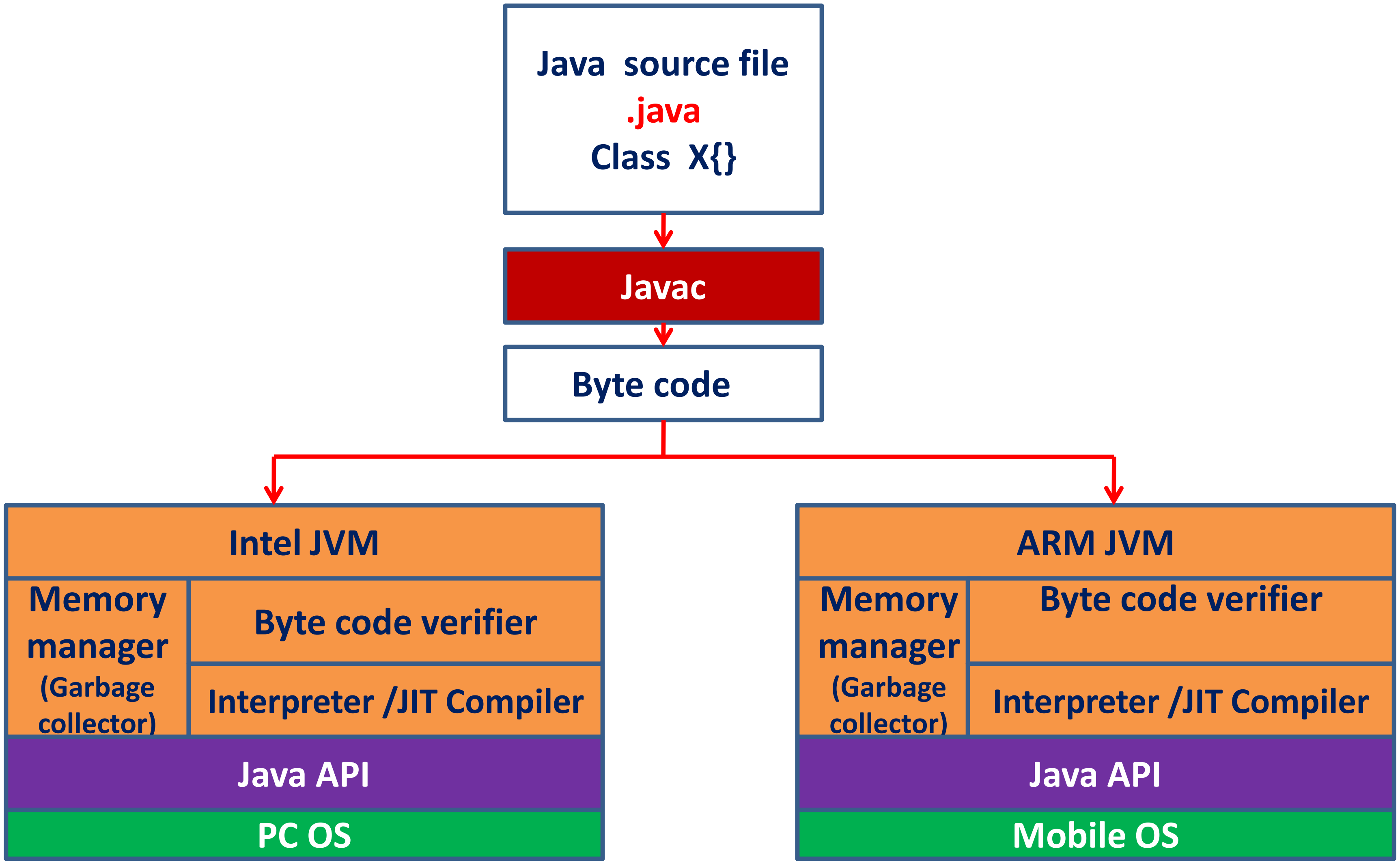
Overview of Java

- **Polymorphism: defined in many forms**
  - is a feature that allows one interface to be used for a general class of actions.
  - The specific action is determined by the exact nature of the situation

About Java

- Case sensitive
- Statically typed
- Strongly typed
- Multiparadigm

JVM Architecture



JVM architecture

A First Simple Program

```
class System{
 static PrintStream out ;}

class PrintStream{
            println(){}
            print(){}
}
```

```
/* First Java program
    call this file Example.java
*/
class Example{
        public static void main(String args[]){
            System.out.println("This is java template");
}
```

Keyword

Identifier

Access modifier

Keyword

cannot declare a top-level class as private or protected. It can be either public or default (no modifier).

- The keyword class to declare that a new class is being defined.
- Example is an *identifier* that is the name of the class.
- The entire class definition, including all of its members, will be between the opening curly brace ({) and the closing curly brace (}).
- The public keyword is an access modifier, which allows the programmer to control the visibility of class members.
- The keyword static allows main() to be called without having to instantiate a particular instance of the class.
- Necessary, since main( ) is called by the Java Virtual Machine before any objects are made.
- The keyword void simply tells the compiler that main( ) does not return a value.
- Java compiler will compile classes that do not contain a main( ) method. But java has no way to run these classes
- String args[ ] declares a parameter named args, which is an array of instances of the class String.
- args receives any command-line arguments present when the program is executed.
- A complex program will have dozens of classes, only one of which will need to have a main( ) method to get things started.
- System is a predefined class that provides access to the system, and out is the

Comment

- The contents of a comment are ignored by the compiler.
- Java supports three styles of comments.
  - multiline comment.
    - begin with /* and end with */
  - Single line comment
    - //
  - Documentation comment
    - begin with /** and end with */.

```
/**
* <h1> sum of two numbers !</h1>
* program finds the sum
*and gives the output on
*the screen.
*
*/
```

A First Simple Program

```
/* First Java program
     call this file Example.java
*/
class Example{
   int a=40; //non static
   public static void main(String args[]){
     System.out.println(“the value of value of a is”+ a);
}
```

Static method:
Used for class and method identifier
Belong to the class
No instance is required to access
Can access static data memeber
Cannot access non static members or methods directly
This and super cannot be used

compile time error

A First Simple Program

```
/* First Java program
     call this file Example.java
*/
class Example{
      static {
     System.out.println(“the value of value of a is”+ a);
       }
}
```

Static method:
Used for class and method identifier
Belong to the class
No instance is required to access
Can access static data memeber
Cannot access non static members
or methods directly
This and super cannot be used

compile time error : supported upto jdk 7 but later version need main method

Class

```
class identifier {
   //class body
}
```

```
class identifier implements identifier1, identifier2 {

}
```

```
class identifier  extends superclass identifier implements interface identifier{

}
```

Modifier: public , private
Class name: first letter capital
Class can extend one parent
Class can implement more than one interface

Member variables: fields
Variables in methods: local variables
Variables in method declaration: parameter

attribute declaration: noun, state
Modifier: 0 or more modifiers
Field type
Field name

method name: first word verb

```
modifier return type method name(parameter list){
//method body
}
```

Java modifiers

- **public: access across package**
- **protected: members**
  - **Used by members of subclass of same or different package**
  - **Not used for class ,interface**
- **default: only classes of same package**
- **Private: members within in the class**
  - **No private class or interface**

Java program

- **In Java, a source file is officially called a compilation unit.**
- **The Java compiler requires that a source file use the .java filename extension.**
- **In Java, all code must reside inside a class.**
- **Java is case-sensitive.**
- **Filename should match the class name.**

Compiling the Program

- Execute the compiler, javac, specifying the name of the source file on the command line
- C:\>javac Example.java
- The javac compiler creates a file called Example.class that contains the bytecode version of the program.
- Java bytecode is the intermediate representation of your program that contains instructions the Java Virtual Machine will execute.
- The output of javac is not code that can be directly executed.

Run the program

- Use Java application launcher called java.
- Pass the class name Example as a command-line argument
- C:\>java Example
- Output:
  - This is java template