Fundamentals of Java

Session: 1

Introduction to Java

Objectives

- Explain the structured programming paradigm
- Explain the object-oriented programming paradigm
- Explain the features of Java as a OOP language
- Describe Java platform and its components
- List the different editions of Java
- Explain the evolution of Java Standard Edition (Java SE)
- Describe the steps for downloading and installing Java Development Kit (JDK)

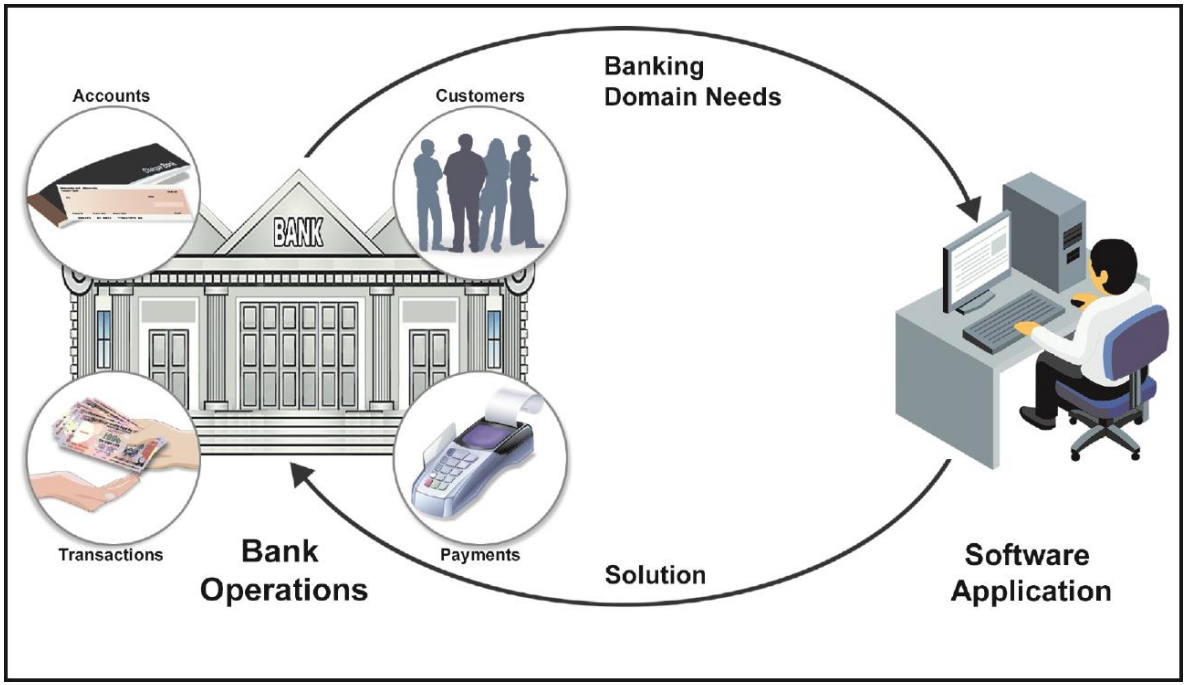
Introduction 1-3

- The most prominent use of computers is to solve problems quickly and accurately.
- The solution adopted to solve a problem is provided as a sequence of instructions or specifications of activity which enables a user to achieve the desired result.

Software Applications

- The solution for solving a problem in the field of information technology is achieved by developing software applications.
- A software application can be defined as a collection of programs that are written in high-level programming languages to solve a particular problem.

- **Knowledge of Domain:**
  - It plays an important role while developing software applications.
  - It can be defined as the field of business or technology to which a problem belongs.
  - Following figure shows the development of software application as a solution for managing various operations in a banking domain:



Introduction 3-3

- **Programming Languages:**
  - The development of software application is done using a programming language.
  - A programming language is used as a medium for communicating the instruction to the computer.
  - The programming language enforces a particular style of programming that is referred to as a programming paradigm.
  - Following are the two types of programming paradigm:

Structured Programming Paradigm

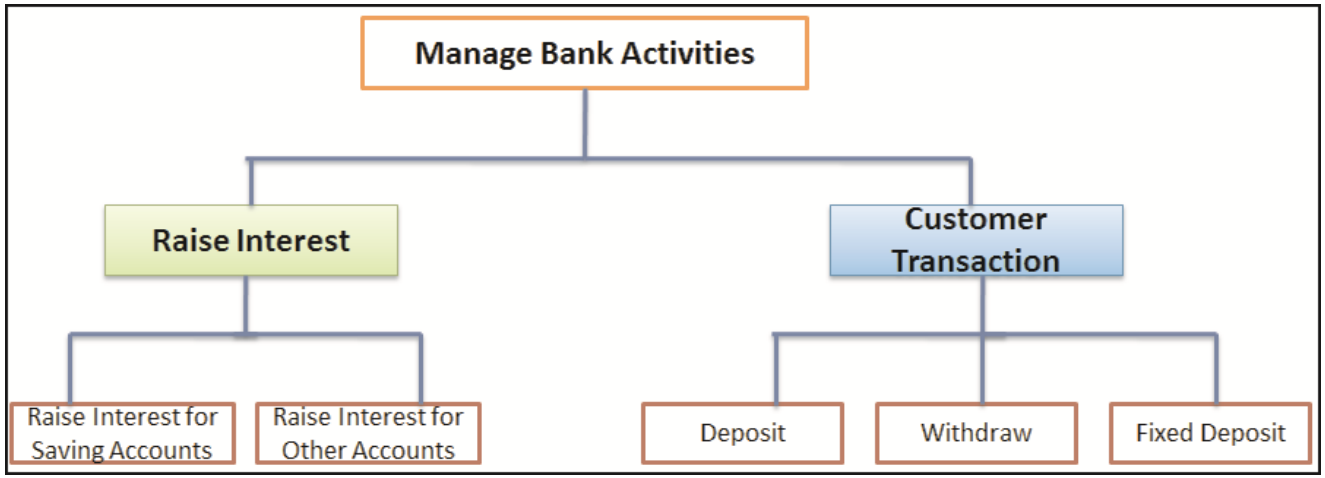
Object-oriented Programming Paradigm

Structured Programming Paradigm 1-2

Structured Programming

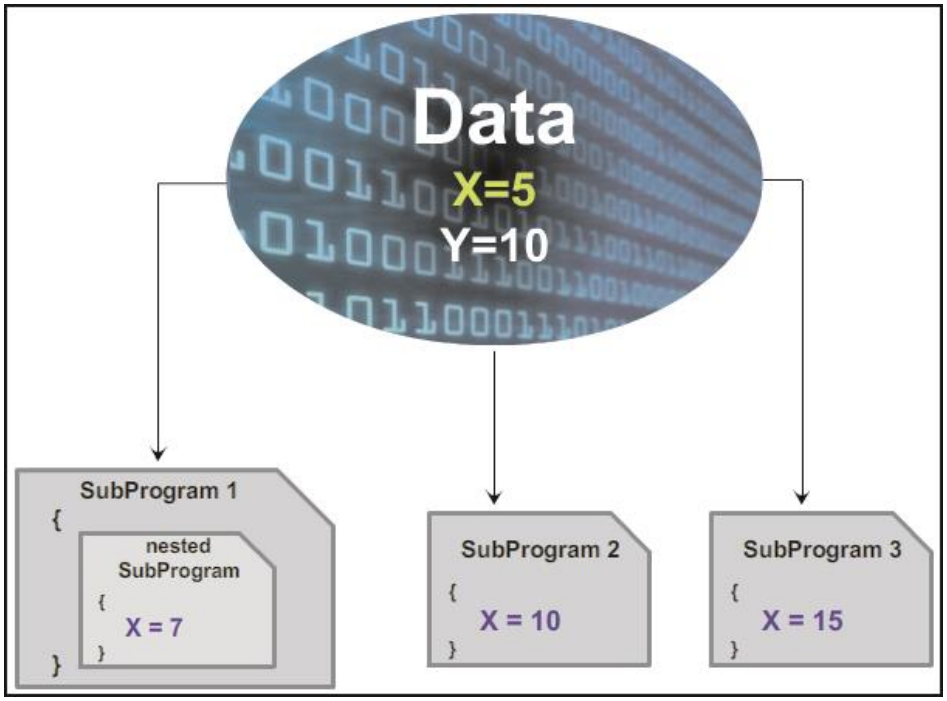
- In structured programming paradigm, the application development is decomposed into a hierarchy of subprograms.
- The subprograms are referred to as procedures, functions, or modules in different structured programming languages.
- Each subprogram is defined to perform a specific task.
- Some of structured programming languages are C, Pascal, and Cobol.

- Following figure displays bank application activities broken down into subprograms:



Structured Programming Paradigm 2-2

- Main disadvantage of structured programming languages are as follows:
  - Data is shared globally between the subprograms.
  - Efforts are spent on accomplishing the solution rather than focusing on problem domain.



- This often led to a software crisis, as the maintenance cost of complex applications became high and availability of reliable software was reduced.

Object-oriented Programming Paradigm 1-4

- Growing complexity of software required change in programming style.

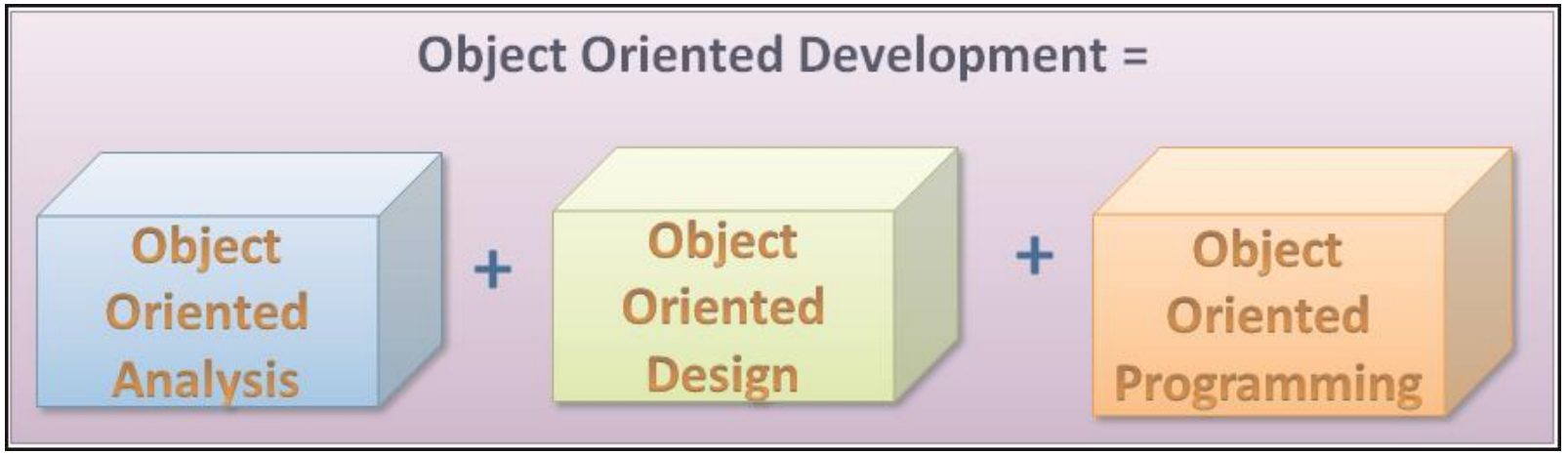
- Some of the features that were aimed are as follows:
  - Development of reliable software at reduced cost.
  - Reduction in the maintenance cost.
  - Development of reusable software components.
  - Completion of software development with the specified time interval.

- These features resulted in the evolution of ***object-oriented programming paradigm***.

Object-oriented Programming Paradigm 2-4

- The software applications developed using object-oriented programming paradigm is:
  - Designed around data, rather than focusing only on the functionalities.

- Following shows different activities involved in the object-oriented software development:



Object-oriented Programming Paradigm 3-4

Object-oriented Analysis (OOA) phase determines the functionality of the system.

Object-oriented Design (OOD) phases determines the process of planning a system in which objects interact with each other to solve a software problem.

Object-oriented Programming (OOP) deals with the actual implementation of the application.

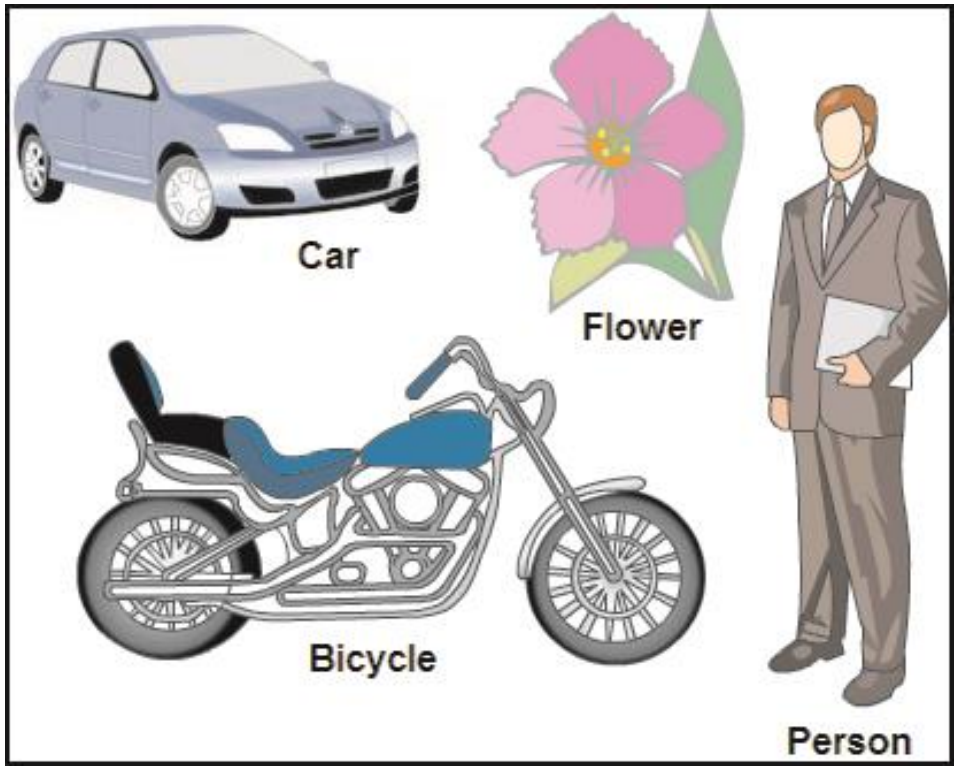
- Unified Modeling Language (UML) helps to create visual models in the system.
- The actual implementation of these visual models is done using an OOP language.

Object-oriented Programming Paradigm 4-4

- An OOP language is based on certain principles that are as follows:
  - **Object** – Represents an entity which possesses certain features and behaviors.
  - **Class** – Is a template that is used to create objects of that class.
  - **Abstraction** – Is a design technique that focuses only on the essential features of an entity for a specific problem domain.
  - **Encapsulation** – Is a mechanism that combines data and implementation details into a single unit called class.
  - **Inheritance** – Enables the developer to extend and reuse the features of existing classes and create new classes. The new classes are referred to as derived classes.
  - **Polymorphism** – Is the ability of an object to respond to same message in different ways.

Concept of an Object 1-4

- An object represents a real-world entity.
- Any tangible or touchable entity in the real-world can be described as an object.
- Following figure shows some real-world entities:



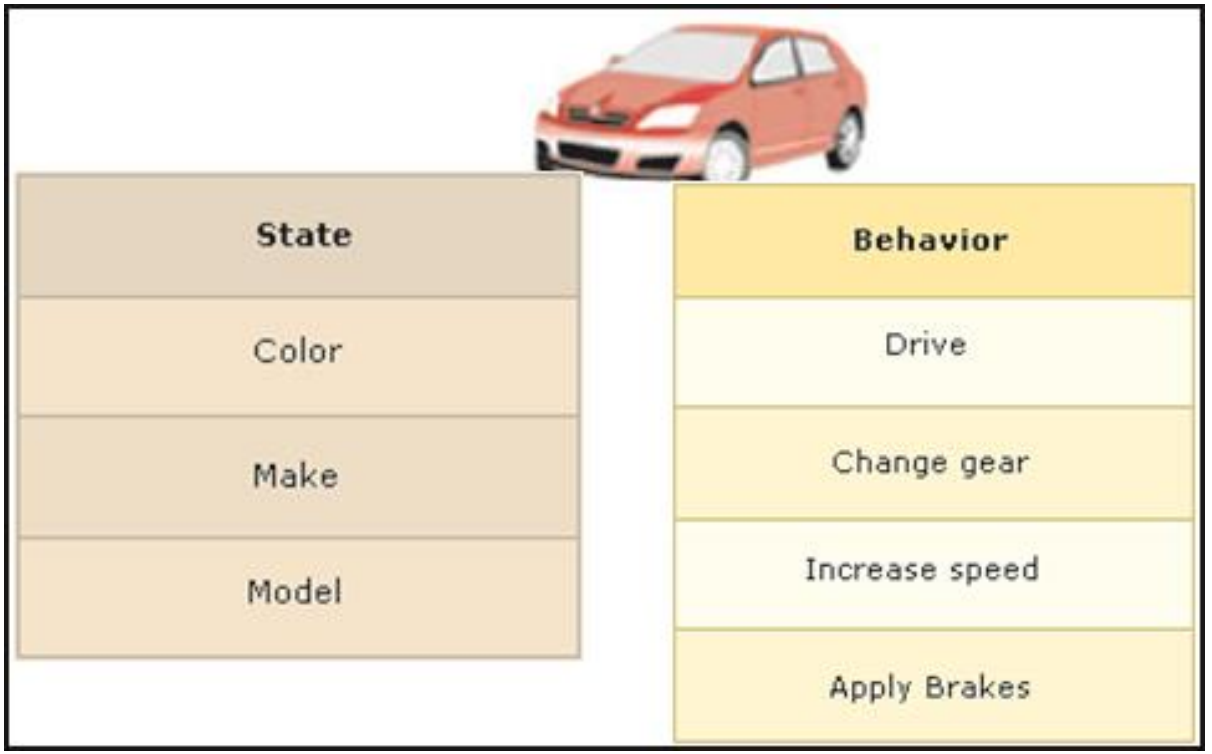
Concept of an Object 2-4

- Each object has:
  - **Characteristics** – Defined as attributes, properties, or features describing the object.
  - **Actions** – Defined as activities or operations performed by the object.
- Example of an object, **Dog**.
  - **Properties** – Breed, Color, and Age
  - **Actions** – Barking, Eating, and Running

- The concept of objects in the real-world can be extended to the programming world where software 'objects' can be defined.

Concept of an Object 3-4

- A software object has state and behavior.
- '**State**' refers to object's characteristics or attributes.
- '**Behavior**' of the software object comprises its actions.
- Following figure shows a software object, a **Car** with its state and behavior:



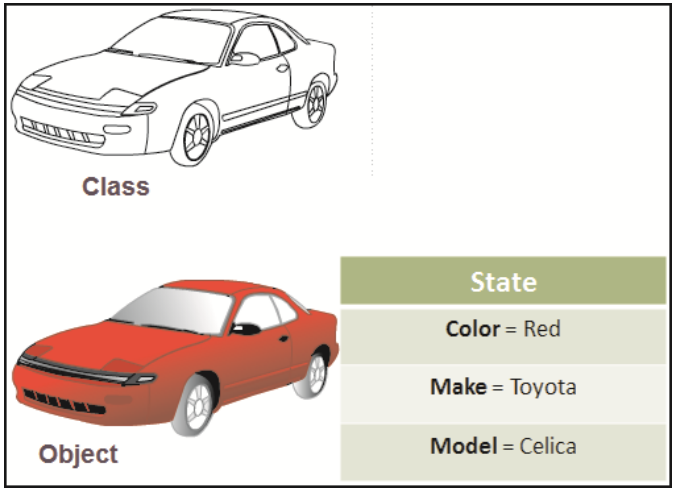
Concept of an Object 4-4

- The advantages of using objects are as follows:

1. They help to understand the real-world.
2. They map attributes and actions of real-world entities with state and behavior of software objects.

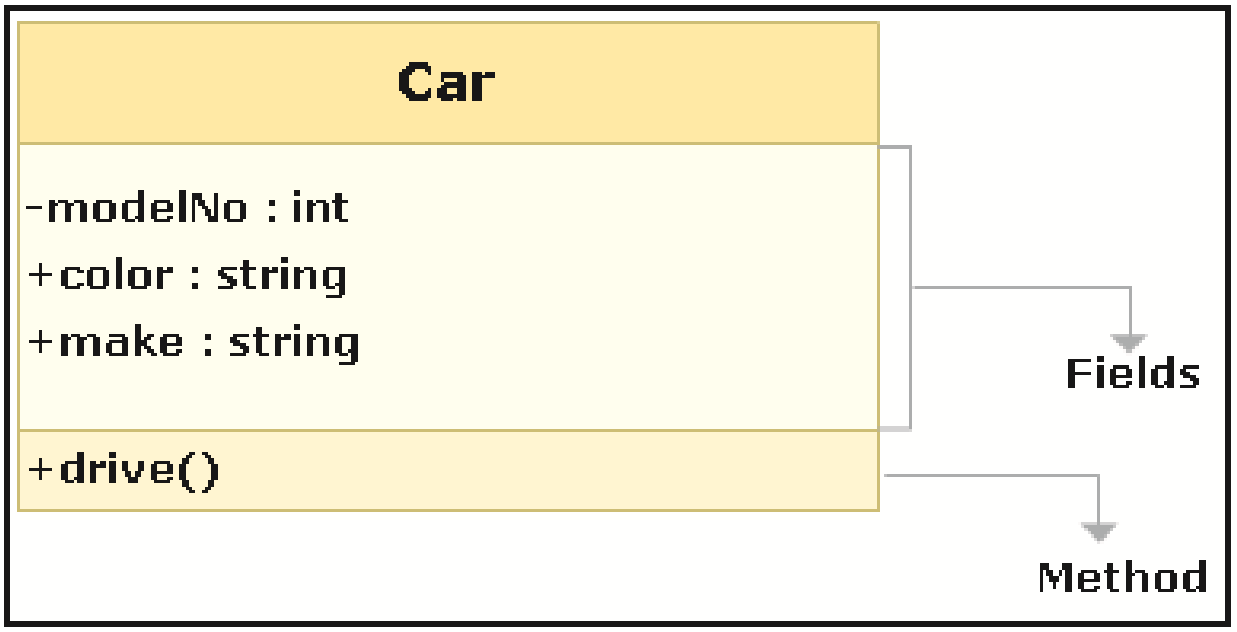
Defining a Class 1-3

- In the real-world, several objects:
  - Have common state and behavior.
  - Can be grouped under a single class.
  - Example: All car objects have attributes, such as color, make, or model.
- **Class**:
  - Can be defined that a class is a template or blueprint which defines the state and behavior for all objects belonging to that class.
  - Following figure shows a car as a template and a Toyota car as an object or instance of the class:



Defining a Class 2-3

- Class comprises fields and methods, collectively called as members.
  - **Fields** – Are variables that depict the state of objects.
  - **Methods** – Are functions that depict the behavior of objects.



Defining a Class 3-3

- Following table shows the difference between a class and an object:

Class	Object
Class is a conceptual model	Object is a real thing
Class describes an entity	Object is the actual entity
Class consists of fields (data members) and functions	Object is an instance of a class

Java

- It is one of the most popular OOP language.
- It helps programmers to develop wide range of applications that can run on various hardware and Operating System (OS).
- It is also a platform that creates an environment for executing Java application.
- It caters to small-scale to large-scale problems across the Internet.
- Java applications are built on variety of platforms that range from:
  - Embedded devices to desktop applications
  - Web applications to mobile phones
  - Large business applications to supercomputers

History of Java 1-3

- **Java Origins: Embedded Systems**

1991

- Team of engineers from Sun Microsystems wanted to design a language for consumer devices.
- Project was named as '**Green Project**'.
- Team included: James Gosling, Mike Sheridan, and Patrick Naughton.
- Efforts were taken to produce portable and a platform independent language that can run on any machine.
- Result was evolution of Java.
- Initially called '**OAK**' and later renamed to **Java** .

History of Java 2-3

- **Java Wonder: Internet**

1995

- Internet and Web started emerging and was used worldwide.
- Sun Microsystems turned Java into an Internet programming language.
- It emerged as a Web technology that added dynamic capabilities to the Web pages.

- **Java Moved: Middle-tier**

1997

- Sun Microsystems defined Servlets API to generate dynamic HTML for Web pages and Enterprise JavaBeans for developing business logics.

2006

- Sun Microsystems released three versions for free under General Public License (GPL)
- These are: Java Platform, Standard Edition (Java SE), Java Platform, Enterprise Edition (Java EE), and Java Platform Micro Edition (Java ME)

History of Java 3-3

- **Java Ahead: Acquisition**

2009

- Sun Microsystems was acquired by Oracle Corporation.
- The project **Coin** was launched to enhance Java programming language.
- Java EE 6 was released with simplified development and deployment model.

2010

- The specification for Java 7 was approved by the Java Community Process (JCP).

Java: Object-oriented Programming Language 1-5

- Java is a high level OOP language as well a platform used for developing applications that can be executed on different platforms.
- It is characterized by following features:

Simple

- Inherits its syntax from predecessor programming languages, such as C/C++.
- Helps programmers to adapt to Java language without any extra skills and extensive trainings.
- Eliminates the use of pointers, operator overloading, and multiple inheritance features supported by predecessor languages.

Java: Object-oriented Programming Language 2-5

Object-oriented

- Java is a pure OOP language that uses classes and objects that address to the real-world problem domains.
- Even, the application development in Java starts with a class designing.

Robust

- **C and C++ languages** - Dynamic memory allocation/deallocation is done manually through pointers that resulted in memory related errors.
- Java incorporates:
  - **Strong memory management** - Handles memory allocation and deallocation using **Garbage Collection Mechanism** that destroys unused objects in memory automatically.
  - **Exception handling mechanism** - Stops abnormal termination of code at runtime.
  - **Compile-time checking** – Ensures variables are declared which contain specific type of data.
  - **Run-time checking** – Ensures purity of Java code during execution.

Secure

- Security checks applied at different layers ensures that the Java programs are protected against malicious codes.
- Java programs that are accessed on the network are known as applets.
- Java applies security to applets by placing them in a sandbox that ensures it should not have direct access to files or resources available on the local system.
- Java Virtual Machine (JVM) which is a runtime environment for executing Java programs applies its own security features to Java language.
- JVM ensures that the code loaded for execution is well-formed and conforms to Java standards.

Architecture Neutral and Portable

- Supports portability by converting application into architecture neutral bytecode during compilation.
- Java has defined language specifications, such as size of primitive data types and operators that are independent of the hardware platform.
- For example, an integer variable in Java always occupies 32 bits on whichever machine the code is executed.
- These features satisfies the major goal of Java language which is '**Write once, run anywhere**'.

Multithreaded

- Allows a single program to perform multiple tasks simultaneously with different threads.

Java: Object-oriented Programming Language 5-5

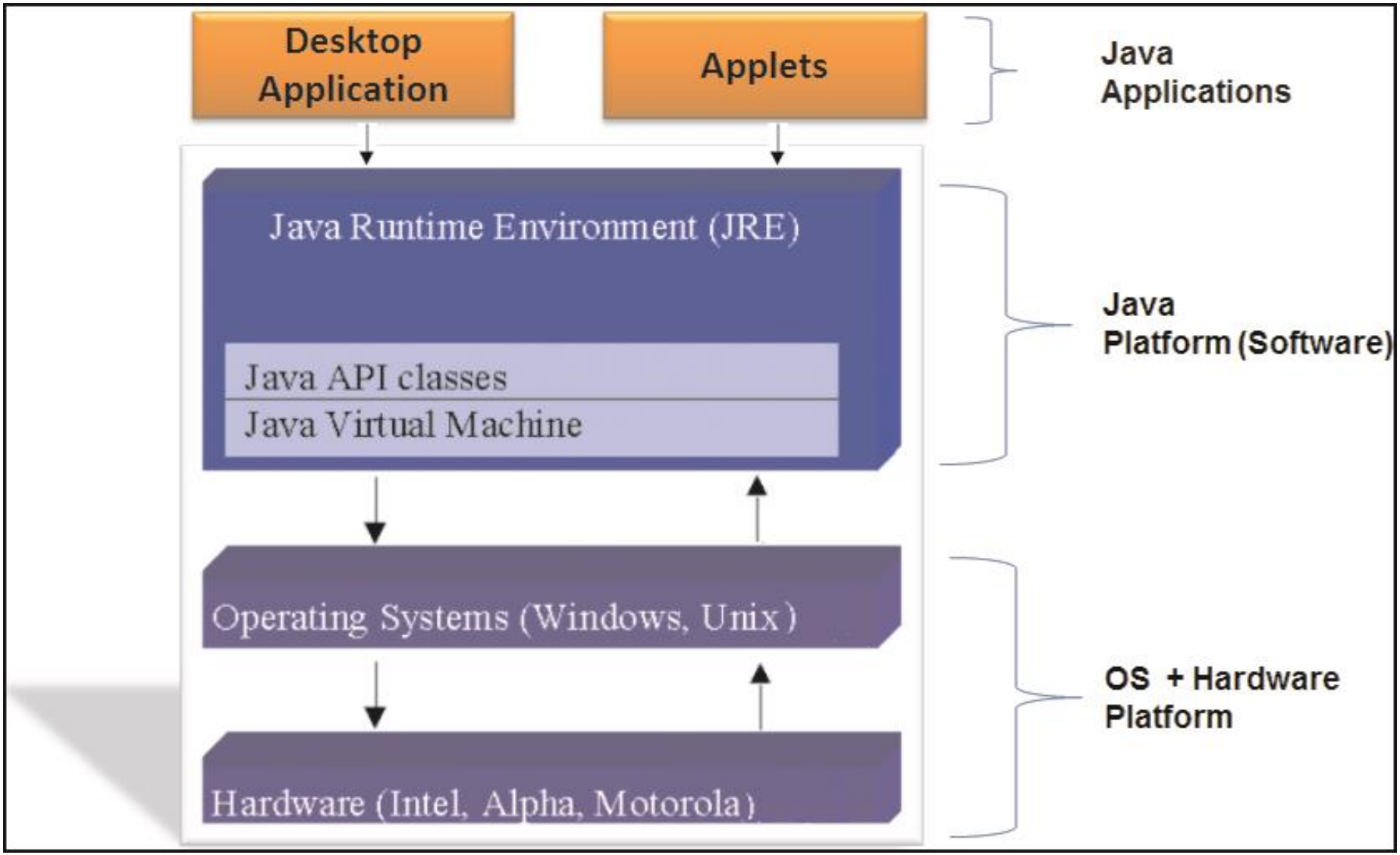
Distributed

- Supports distributed programming in which resources can be accessed across the network.

Dynamic

- At runtime an application can dynamically decide which classes it requires and loads them accordingly.
- This gives new perspective to Java for designing and developing applications.

Java Platform



Java Platform

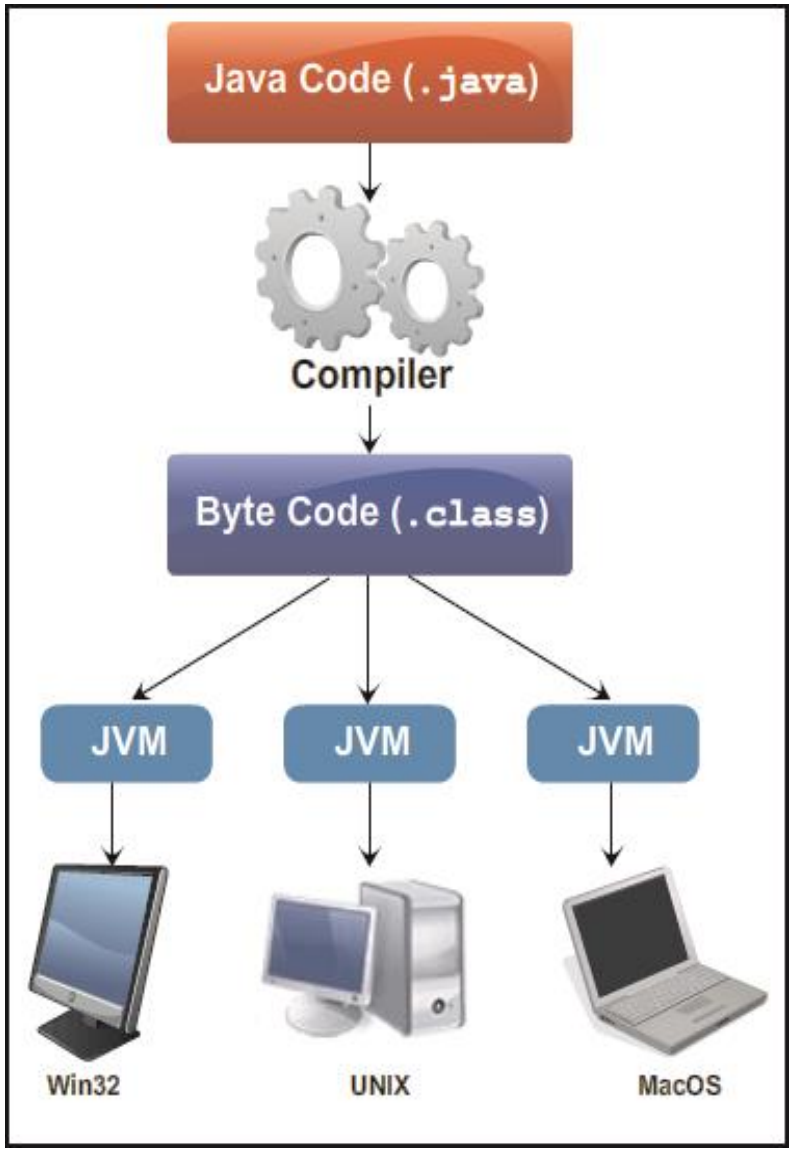
- Is a software-only platform that runs on top of other hardware-based platforms.
- Contains **Java Runtime Environment (JRE)** with components namely:
  - Java Virtual Machine (JVM)
  - Java class library also referred to as Java Programming Interface (Java API)

Java Virtual Machine (JVM) 1-2

- It is an executable engine that creates an environment for executing Java compiled code, that is, bytecode.
- It is known as a virtual machine because it is an imitation of a Java processor on the physical machine.
- There are different implementations of JVM available for different platforms, such as Windows, Unix, and Solaris.

Java Virtual Machine (JVM) 2-2

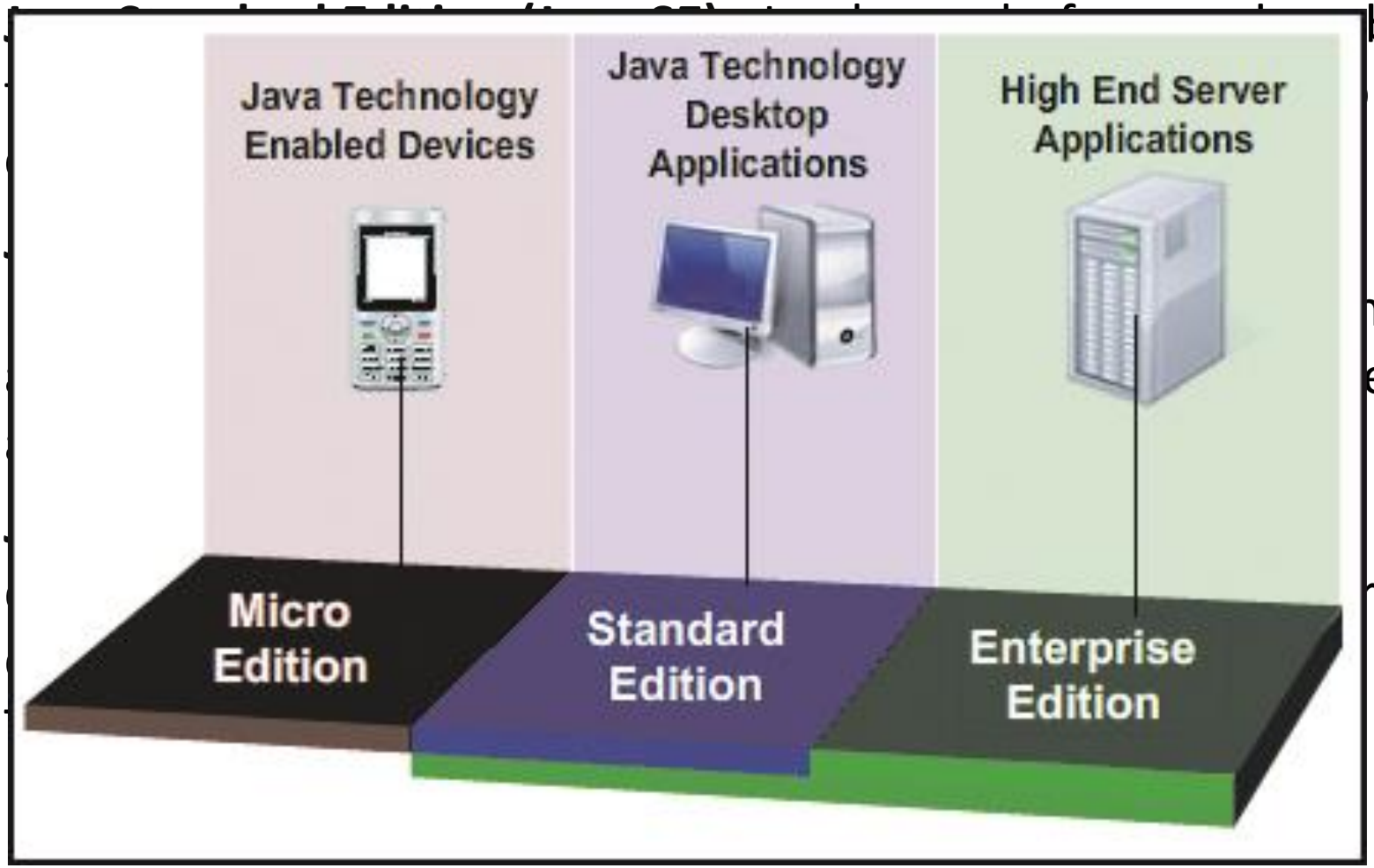
- **Bytecode**:
  - Is an intermediate form closer to machine representation.
  - Is an optimized set of instructions executed by the Java runtime environment.
  - This environment is known as JVM.
- The same bytecode can be executed by different implementations of JVM on various platforms.



Java API

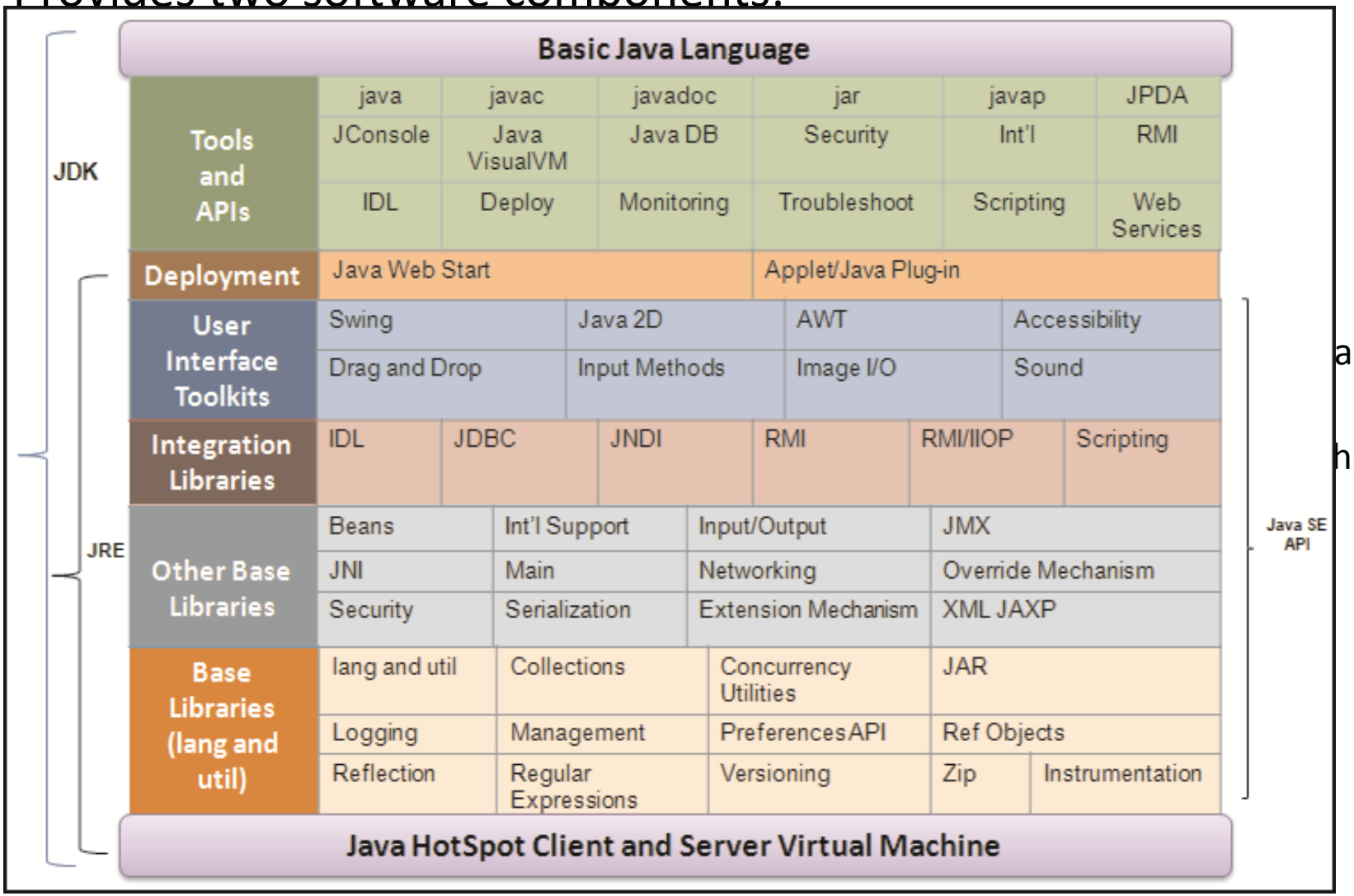
- It is a large collection of ready-made software components.
- These components are classes and interface grouped into libraries referred to as packages in Java.
- Example:
  - The **Swing** library provides classes for User Interface (UI) components.
  - The **Input/Output (I/O)** library provides the standard interface for reading and writing data into files stored in the system.

Editions of Java



Components of Java SE Platform 1-2

- Provides two software components:



as

hat

Components of Java SE Platform 2-2

- **Development Tools** – Include tools used for compiling, running, debugging, and documenting a Java application.

java	javac	javadoc	jar	javap	JPDA

- **API** - Provides the core functionality of the Java programming language.
- **Deployment Tools** – Provides software for deploying the developed applications to end-users.
- **User Interface Toolkits** - Enables the developer to create graphical interfaces in a Java application.
- **Integration libraries** - Enable developers to access and manipulate database and remote objects in an application.

Evolution of Java

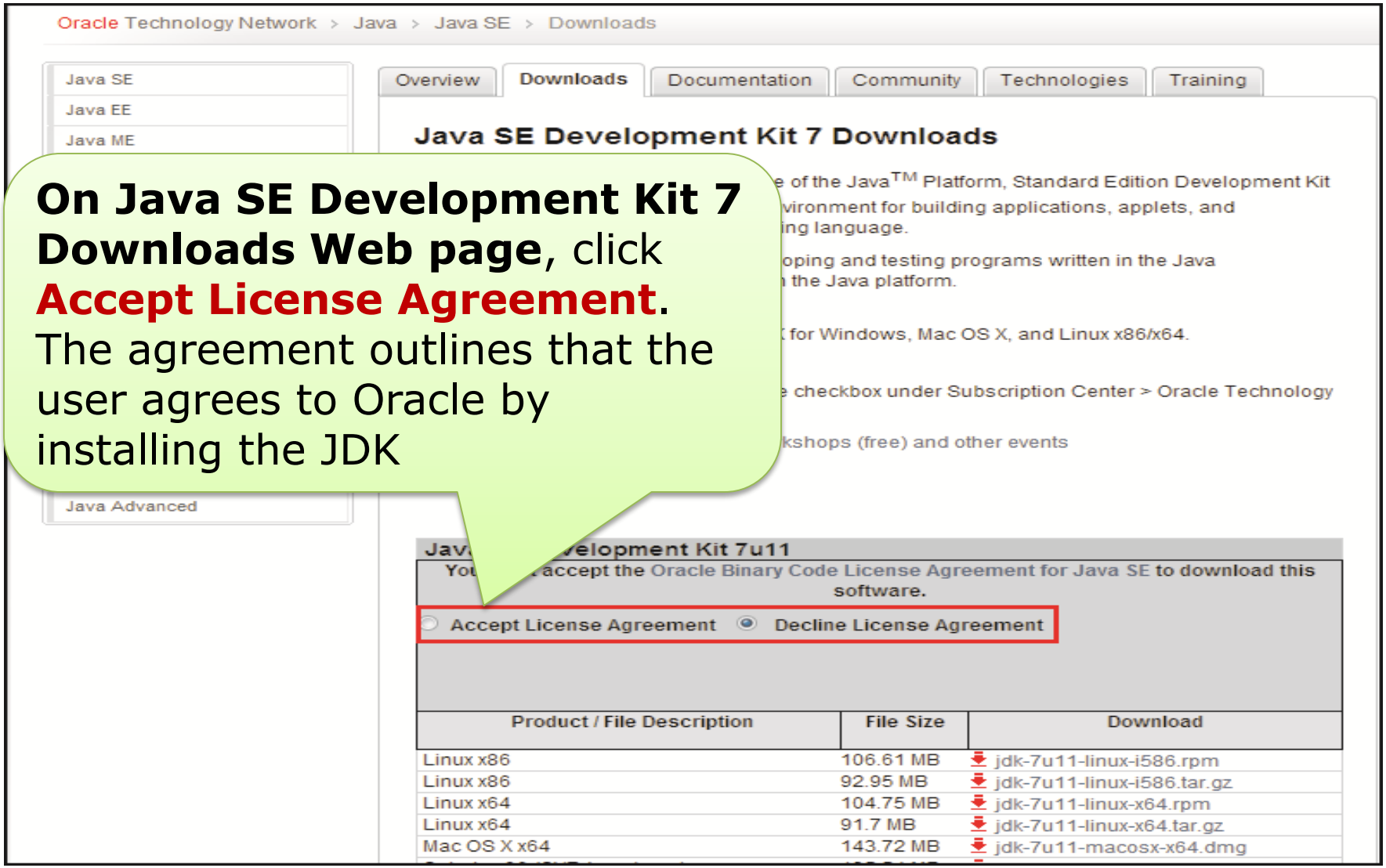
Releases	Implementation
JDK 1.0	Creation of packages with classes in the standard library
JDK 1.1	Included an event delegation model for Graphical User Interface (GUI) package AWT, JavaBeans, and Java Database Connectivity (JDBC) API
JDK 1.2 (Java 2)	Included a new graphical API, named Swing. Also, added APIs for reflection and collection framework (based on data structure)
JDK 1.3	Included a directory interface to lookup for components, named, Java Naming and Directory Interface (JNDI)
JDK 1.4	Included regular expression API, assertions, exception chaining, channel-based I/O API, an XML API for parsing and processing
JDK 1.5	Included new features in the language such as for-each loop, generics, annotations, and auto-boxing
JDK 1.6	Included script language, visual basic language support, and improvements in the GUI

Java SE 7

- Java SE 7 is the new major release of Java with internal version number as 1.7.
- The lists of new features that are incorporated in the language are as follows:
  - Supports the use of `String` class in the `switch` decision-making construct.
  - Integer types can be assigned with a binary number value.
  - Supports the use of underscore character (_) between the digits of a numeric value.
  - An expandable `try` statement called `try-with-resources` statement used for automatic resource management.
  - Constructor of a generic class declaration can be replaced with an empty set of parameters (<>).
  - Enhancement in exception handling mechanism. In exception handling code, a single catch can be used to handle one or more exceptions.
  - Compiler warnings generated for `varargs` methods has been improved.

Downloading and Installing JDK 1-6

Downloading and Installing JDK 2-6



Downloading and Installing JDK 3-6

Java SE Development Kit 7u11

You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

Thank you for accepting the Oracle Binary Code License Agreement for Java SE; you may now download this software.

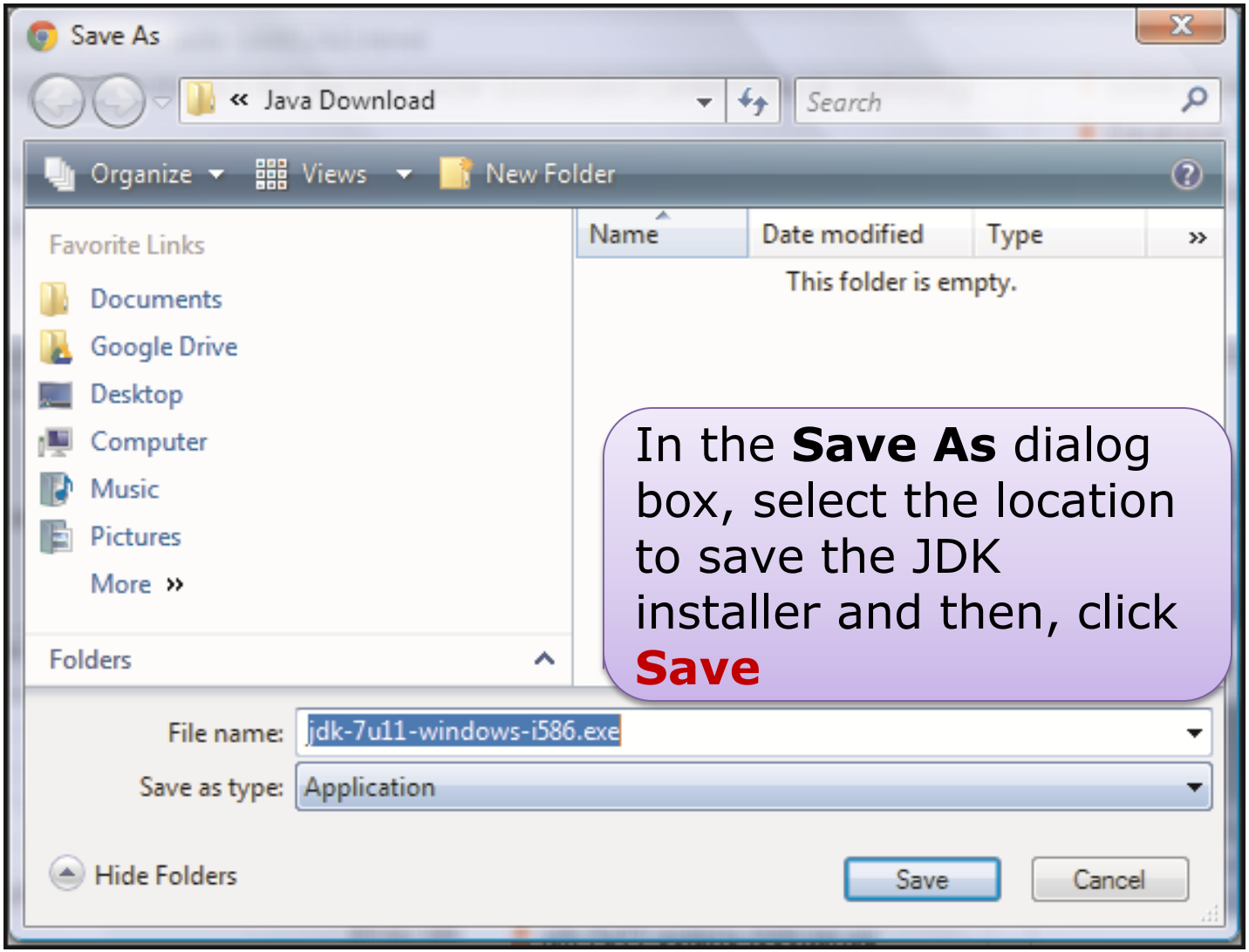
Product / File Description	File Size	Download
Linux x86	106.61 MB	jdk-7u11-linux-i586.rpm
	92.95 MB	jdk-7u11-linux-i586.tar.gz
	104.75 MB	jdk-7u11-linux-x64.rpm
	91.7 MB	jdk-7u11-linux-x64.tar.gz
	143.72 MB	jdk-7u11-macosx-x64.dmg
	135.54 MB	jdk-7u11-solaris-i586.tar.Z
	91.92 MB	jdk-7u11-solaris-i586.tar.gz
	22.52 MB	jdk-7u11-solaris-x64.tar.Z
	14.95 MB	jdk-7u11-solaris-x64.tar.gz
	135.87 MB	jdk-7u11-solaris-sparc.tar.Z
Solaris SPARC	95.25 MB	jdk-7u11-solaris-sparc.tar.gz
Solaris SPARC 64-bit (SVR4 package)	22.77 MB	jdk-7u11-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	17.49 MB	jdk-7u11-solaris-sparcv9.tar.gz
Windows x86	88.77 MB	jdk-7u11-windows-i586.exe
Windows x64	90.41 MB	jdk-7u11-windows-x64.exe

Java SE Development Kit 7u10

You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

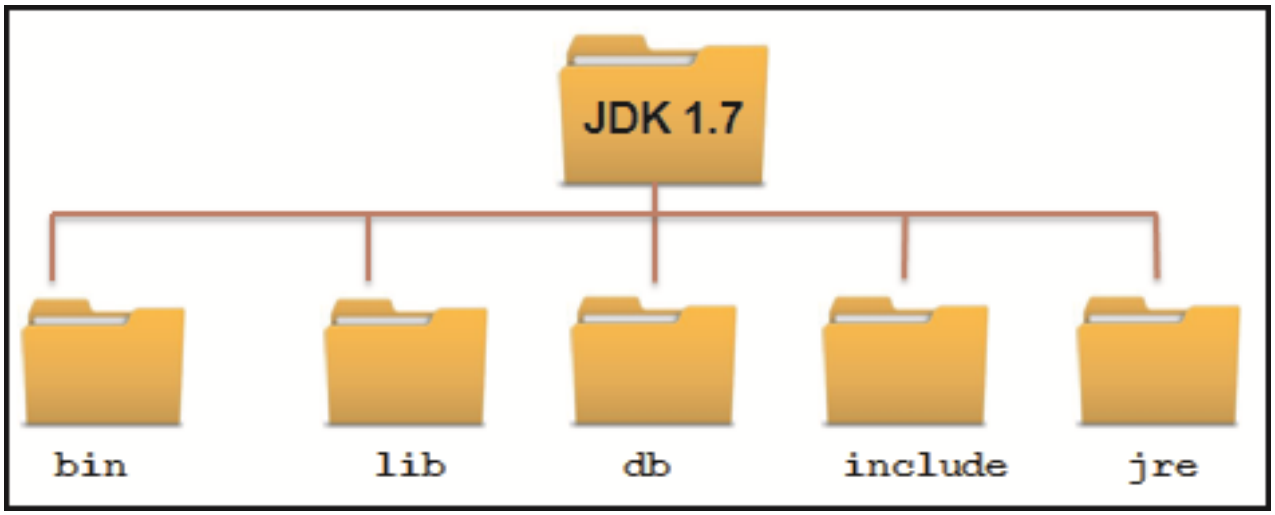
To download JDK 7 for Windows 32-bit OS, click **jdk-7u11-windows-i586.exe** under Download

Downloading and Installing JDK 4-6



Downloading and Installing JDK 5-6

- Double-click the installer icon and follow the instructions provided by the JDK installer.
- The installer installs development tools, source code, and the JRE in the default directory, `C:\Program Files\Java`.
- Following figure shows the directory structure of the installed JDK on the system:



Downloading and Installing JDK 6-6

- Following table lists the common directories that are part of the typical JDK installation:

Directory	Description
bin	Contains tools that are used for developing a Java application, such as compiler and JVM
db	Contains a relational database named Apache Derby
include	Contains header files that are used to interact with C applications
jre	Represents the JRE used by the JDK

Configuring JDK 1-3

- To work with JDK and Java programs, certain settings need to be made to the environment variables .
- Environment variables are pointers pointing to programs or other resources. The variables to be configured are as follows:
  - **PATH** - Set to point to the location of Java executables (javac.exe and java. exe).
  - **CLASSPATH** - Specifies the location of the class files and libraries needed by the Java compiler to compile applications.

Configuring JDK 2-3

- To set the value for the `PATH` variable, perform the following steps in Windows 7:
  - Right-click **My Computer** icon on the desktop and then, click **Properties** from the context menu.
  - Click **Advanced system** settings link on the left tab.
  - Under **Advanced** tab, click **Environment Variables**.
  - In the System variables area, select the **PATH** variable and then, click **Edit** button to enter the JDK installation folder path.
  - Type path of the bin folder in the **Variable Value** text box.
  - For example, the path can be:

    ```
    C:\ WINDOWS\system32;C:\WINDOWS;C:\Program Files\Java\jdk1.7.0\bin
    ```

Configuring JDK 3-3

- To set `CLASSPATH` variable in Windows 7, perform the following steps:
  - Right-click **My Computer** icon on the desktop and click **Properties** from the context menu.
  - Click **Advanced system** settings link on the left tab.
  - Under **Advanced** tab, click **Environment Variables.**
  - In the System variables area, click **New** button.
  - Type **CLASSPATH** in **Variable Name** and then, type `C:\<jdk _ installation _ folder>` in the **Variable Value**.

Summary 1-2

- The development of application software is performed using a programming language that enforces a particular style of programming, also referred to as programming paradigm.
- In structured programming paradigm, the application development is decomposed into a hierarchy of subprograms.
- In object-oriented programming paradigm, applications are designed around data, rather than focusing only on the functionalities.
- The main building blocks of an OOP language are classes and objects. An object represents a real-world entity and a class is a conceptual model.

Summary 2-2

- Java is an OOP language as well a platform used for developing applications that can be executed on different platforms. Java platform is a software-only platform that runs on top of the other hardware-based platforms.
- The editions of Java platform are Java SE, Java EE, and Java ME.
- The components of Java SE platform are JDK and JRE. JRE provides JVM and Java libraries that are used to run a Java program. JDK includes the necessary development tools, runtime environment, and APIs for creating Java programs.