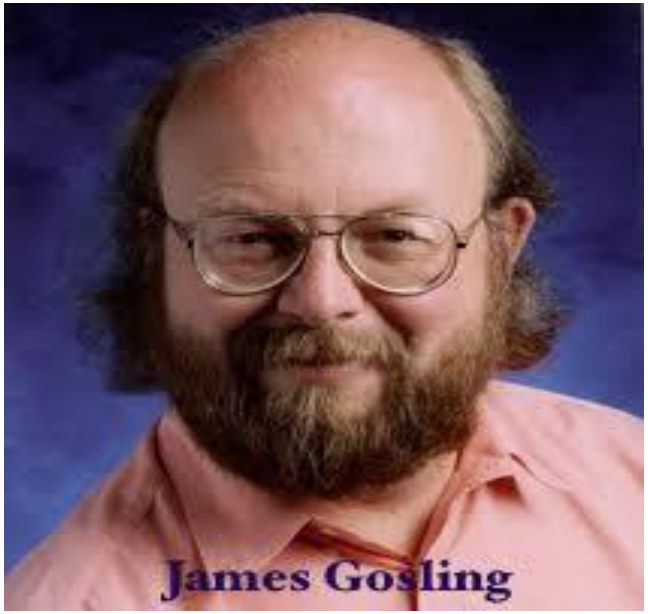
JAVA Programming Language



- In this session, you will learn to:
  - Describe the key features of Java technology
  - Write, compile, and run a simple Java application
  - Describe the function of the Java Virtual Machine
  - Define garbage collection
  - List the three tasks performed by the Java platform that handle code security
  - Define class, member, attribute, method, constructor, and package

Evolution and Need for Java

- You can use Java to develop network-oriented programs because networking features are inbuilt in Java.
- In 1991, a team of software developers at Sun Microsystems, USA, was designing a language for consumer electronic devices.
- The development team headed by James Gosling wanted to design a portable language using which programs should be developed such that they can run on computers with different platform.
- The team considered C++ as the model language for designing Java language. The team deprecated various ambiguous features from this new language.
- Initially, this developed language was called Oak, but was later renamed to Java.

Evolution and Need for Java

- The following table lists the various developments that took place in the evolution of Java:

Year	***Development***
1990	Sun Microsystems developed software to manipulate electronic devices.
1991	A new language named Oak was introduced using the most popular object-oriented language C++

Evolution and Need for Java

Year	***Development***
1993	The WWW appeared on the Internet that transformed the text-based Internet into graphical Internet.
1994	Sun Microsystems team developed a Web browser called HotJava to locate and run applet programs on the Internet.
1995	Oak was renamed as Java.
1996	Java was established as an Object-oriented programming language.

Types of Java Applications

1) Standalone Application

Standalone applications are also known as desktop applications or window-based applications. These are traditional software that we need to install on every machine. Examples of standalone application are Media player, antivirus, etc. AWT and Swing are used in Java for creating standalone applications.

2) Web Application

An application that runs on the server side and creates a dynamic page is called a web application. Currently, **Servlets,Jsp** etc. technologies are used for creating web applications in Java.

3) Enterprise Application

An application that is distributed in nature, such as banking applications, etc. is called enterprise application. It has advantages of the high-level security, load balancing, and clustering. In Java, EJB is used for creating enterprise applications.

4) Mobile Application

An application which is created for mobile devices is called a mobile application. Currently, Android and Java ME are used for creating mobile applications.

There are 4 platforms or editions of Java:

1) Java SE (Java Standard Edition)

It is a Java programming platform. It includes Java programming APIs such as java.lang, java.io, java.net, java.util, java.sql, java.math etc. It includes core topics like OOPs, String, Regex, Exception, Inner classes, Multithreading, I/O Stream, Networking, AWT, Swing, Reflection, Collection, etc.

2) Java EE (Java Enterprise Edition)

It is an enterprise platform which is mainly used to develop web and enterprise applications. It is built on the top of the Java SE platform. It includes topics like Servlet, JSP, Web Services, EJB, JPA, etc.

3) Java ME (Java Micro Edition)

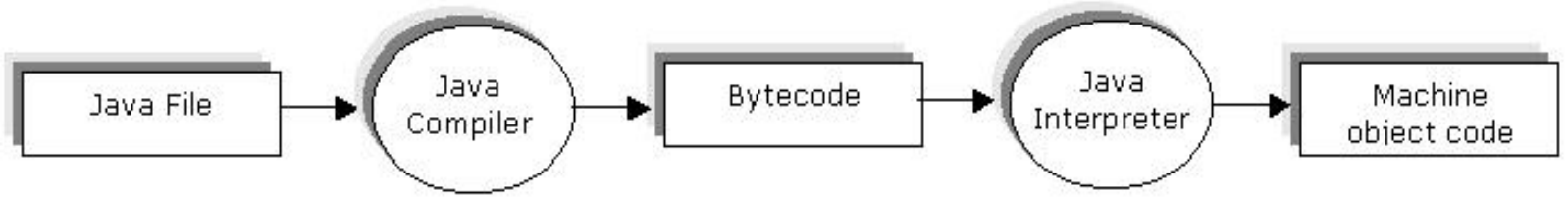
It is a micro platform which is mainly used to develop mobile applications.

4) JavaFX

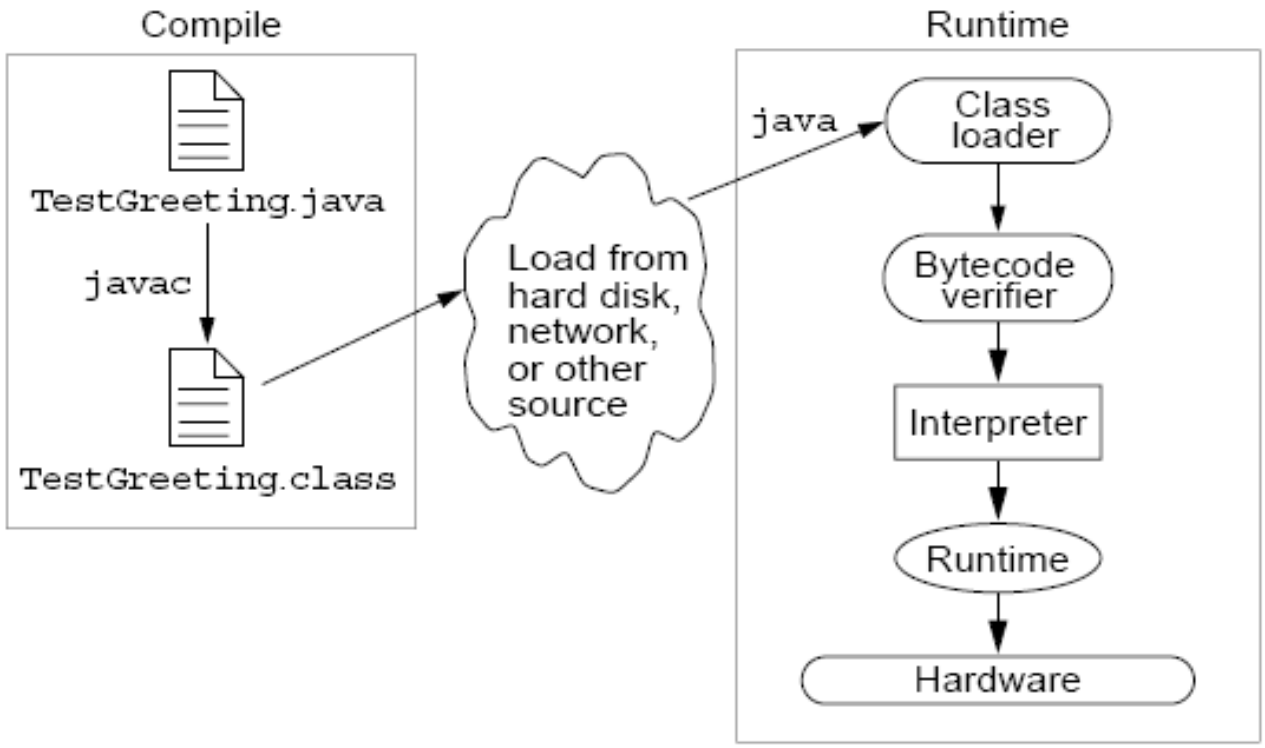
It is used to develop rich internet applications. It uses a light-weight user interface API.

Java Programming Language

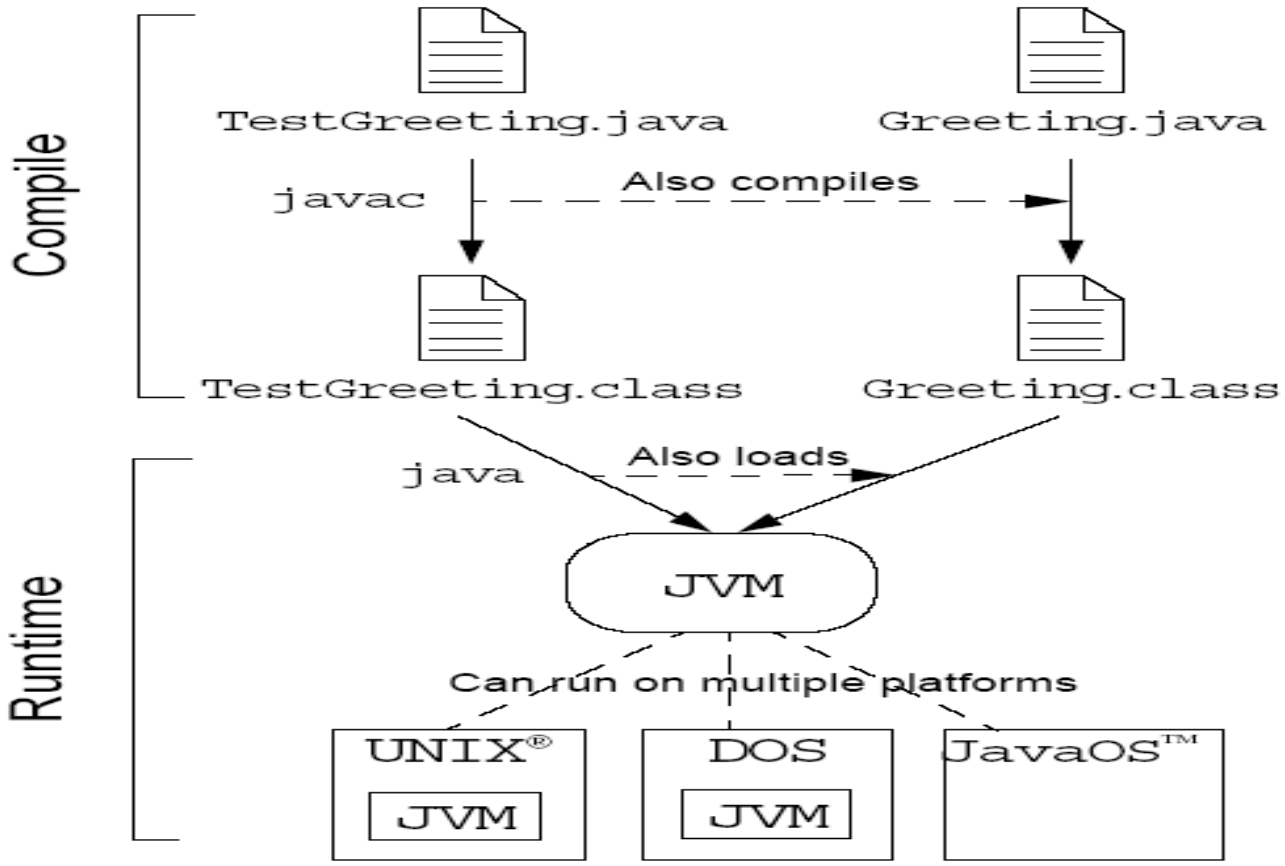
- Java Programming Language and class File
  - The Java programming environment



- The Java application environment performs as follows:



Java Technology Runtime Environment



Java Programming Language

- Java Architecture
  - Various components of Java Architecture are:
    - Java programming language
    - Java class file
    - Java Virtual Machine (JVM)
    - Java Application Programming Interface (API)

- The JVM performs three main tasks:
  - Loads code – Performed by the class loader.
  - Verifies code – Performed by the bytecode verifier.
  - Executes code – Performed by the runtime interpreter.

The Class Loader

- Loads all classes necessary for the execution of a program.
- Maintains classes of the local file system in separate namespaces.
- Avoids execution of the program whose bytecode has been changed illegally.

Significance of the Java Class File

- The Java class file contains the Java Bytecode.
- The class files are platform independent therefore you can run a Java program by loading the class file on any system with the Java Runtime Environment (JRE).
- The following command shows how to view the contents of a class file:

```
javap -c <class_filename>
```

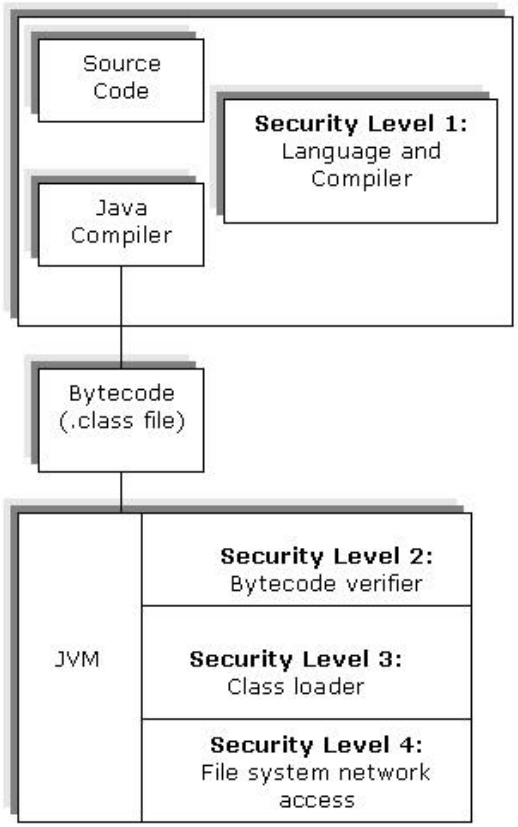
- The javap command prints the instructions that comprise the Java Bytecode, for each of the methods in a class.

Java Programming Language

- Java Virtual Machine (JVM)
  - Components of the JVM:
    - Class loader
    - Execution engine
    - Just In Time(JIT) compiler

Java Programming Language

- Security levels in Java architecture:



Java Programming Language

- **Bytecode is verified in two phases:**
  - In the first phase, the verifier checks for the structure of the .class file.
  - The second level phase occurs when the Bytecode is run.The Bytecode verifier checks the validity of classes, variables, and methods used in a program.
  - JVM is an interpreter to java bytecode. It converts bytecode into machine language and executes line by line.

Garbage Collection in JVM

- Garbage collection is the process that is used to free the memory of the objects that are no longer in use.
- When a program stops referencing an object, it is not required any more and can be deleted.
- The space that is used by the object is released for use by another objects.
- The garbage collection feature implies that the new objects are created and all the unreferenced objects are deallocated from the memory.

Class and Object

- **In object-oriented programming technique, we design a program using objects and classes.**

- **What is class in Java?**
  - A class is a group of objects which have common properties. It is a template or blueprint from which objects are created.
  - A class in Java can contain:
    - Fields
    - Methods
    - Constructors
    - Nested class and interface

Class and Object(Contd…)

- **What is an object in Java?**
  - An object is the instance(result) of a class.
  - **An object has three characteristics:**
    - **State**: represents the data (value) of an object.
    - **Behavior**: represents the behavior (functionality) of an object such as deposit, withdraw, etc.
    - **Identity**: An object identity is typically implemented via a unique ID. The value of the ID is not visible to the external user. However, it is used internally by the JVM to identify each object uniquely.
- **Example**: Pen is an object. Its name is Reynolds; color is white, known as its state. It is used to write, so writing is its behavior.

- **Defining Variables:**
  - A variable is the name that refers to a memory location where some data value is stored.
  - Each variable that is used in a program must be declared.
  - **int** data=50; //Here data is variable

- **There are three types of variables in java:**
  - local variable
  - instance variable
  - static variable

1) Local Variable:

A variable declared inside the body of the method is called local variable. You can use this variable only within that method and the other methods in the class aren't even aware that the variable exists.

A local variable cannot be defined with "**static**" keyword.

2) Instance Variable:

A variable declared inside the class but outside the body of the method, is called instance variable. It is not declared as static.

It is called instance variable because its value is instance specific and is not shared among instances.

3) Static variable:

A variable which is declared as static is called static variable. It cannot be local. You can create a single copy of static variable and share among all the instances of the class.

/*Example to understand the different variables in Java:*/

```
class  A
{
int data=50;   //instance variable
static int m=100;       //static variable
void method()
{
int n=90;         //local variable
}
}          //end of class
```

Declaring Variables(Contd...)

- Naming Conventions for Variables:
  - The name of a variable needs to be meaningful, short, and without any embedded space or symbol.
  - A variable name must be unique.
  - A variable name must begin with a letter, an underscore (_), or the dollar symbol ($), which can be followed by a sequence of letters or digits (0 to 9), '$', or '_' .
  - A variable name should not start with a digit.
  - A variable name should not contain embedded white spaces .
  - A variable name should not consist of a keyword.
  - A variable name in Java is case sensitive.

Java Keywords:

Java keywords are also known as **reserved words**. Keywords are particular words which acts as a key to a code. These are predefined words by Java so it cannot be used as a variable or object name.

Java Keywords(Contd…)

- **Keywords available in Java:**

abstract	boolean	break	byte
case	catch	char	class
const	continue	default	do
double	else	extends	final
finally	float	for	goto
if	implements	Import	instanceof

- **Keywords available in Java: (Contd.)**

int	interface	long	native
new	package	private	protected
public	return	short	static
strictfp	super	switch	synchronized
this	throw	throws	transient
try	void	volatile	while

Declaring Java Classes

- Basic syntax of a Java class:

```
<modifier>* class <class_name>
{ <attribute_declaration>*
  <constructor_declaration>*
  <method_declaration>*
}
```

Example:

```
public class MyFirstClass
{ private int age;
  public void setAge(int value)
  { age = value;
  }
}
```

Declaring Attributes

- Basic syntax of an attribute:

```
<modifier>* <type> <name> [ = <initial_value>];
```

Example:

```
public class MyFirstClass
{
    private int x;
    private float y = 10000.0F;
    private String name = "CDAC";
}
```

Declaring Methods

- Basic syntax of a method:

```
<modifier>* <return_type> <name> (
<argument>* ){    <statement>*    }
```

Example:

```
public class Dog
{  private int weight;
   public int getWeight()
   {  return weight;
   }
   public void setWeight(int newWeight)
   {  if ( newWeight > 0 )
      {  weight = newWeight;
   } } }
```

Accessing Object Members

- To access object members, including attributes and methods, dot notation is used
  - The dot notation is: <object>.<member>

    Examples:

```
d.setWeight(42);
d.weight = 42;        // only permissible if weight is public
```

// A SIMPLE JAVA PROGRAM:

```
class TestGreetings
{
        public static void main(String args[])
        {

System.out.println("Welcome to the Exciting World of Java");

        }
}
```

Valid java main method signature:

- ✓ public static void main(String[] args)
- ✓ public static void main(String []args)
- ✓ public static void main(String args[])
- ✓ public static void main(String... args)
- ✓ static public void main(String[] args)
- ✓ public static final void main(String[] args)
- ✓ final public static void main(String[] args)
- ✓ final strictfp public static void main(String[] args)