# Unit 1

## JAVA PROGRAMMING

# UNIT 1 - Contents

# Genesis of Java

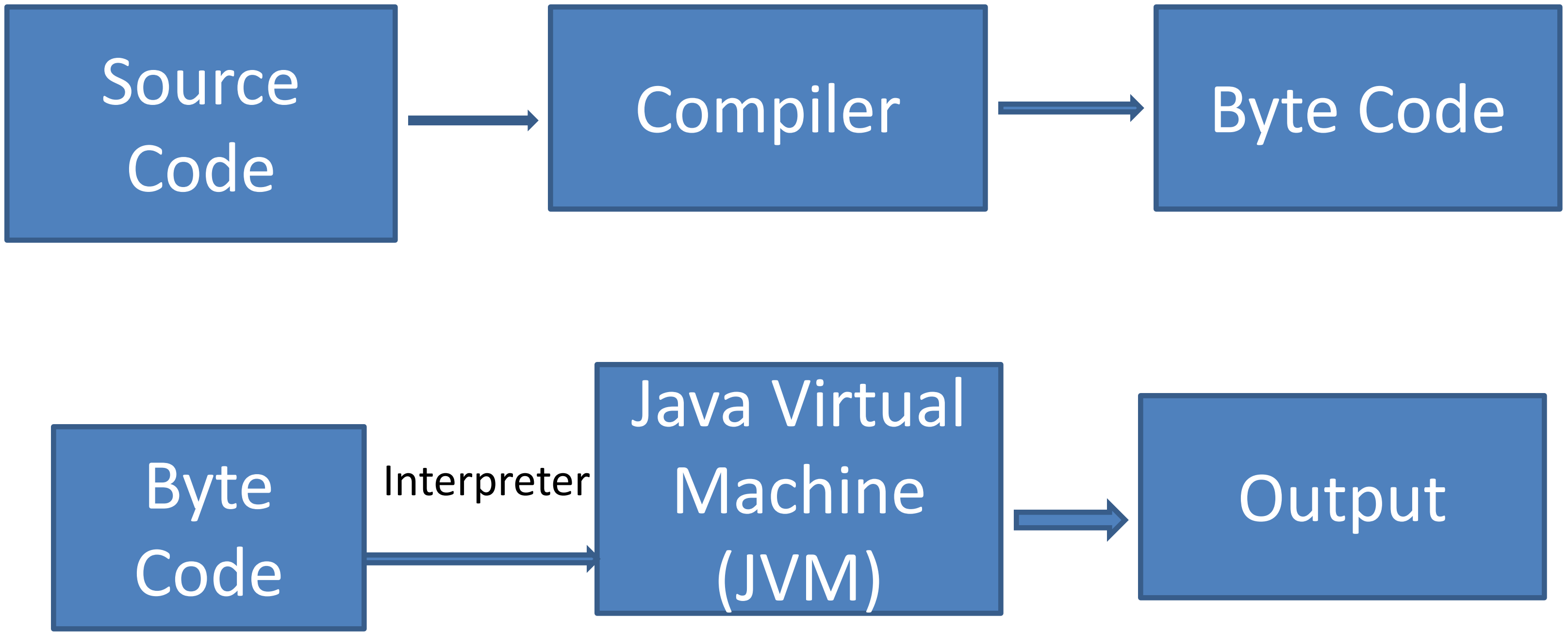
- General Purpose Object Oriented Language
- Developed by Sun Microsystems in 1991
- Originally called OAK
- Team Members – James Gosling, Patrick Naughton
- Story Behind JAVA

## Comparison with C and C++

- No typedefs, #define or pre-processor.
- No structure and unions
- No enums
- No stand-alone functions
- No multiple inheritance
- No goto statements
- No operator overloading
- No pointers

# Java Programming Language

- Java is both compiled and interpreted.



## Java Platform – 2 Components

1. Java Virtual Machine
2. Java Application Programming Interface (Java API)

API – consists of hundreds of classes and packages

## Java API Packages

- Language support package - collection of classes and methods required for implementing basic features of Java.
- Utilities package – collection of classes to provide utility functions such as date and time functions.
- Input/Output package – collection of classes required for I/O manipulation.

# Java API Packages

- Networking package – collection of classes for communicating with other computers via Internet.
- AWT package – Abstract Window Toolkit Package contains classes that implements GUI.
- Applet package – Includes a set of classes that allows to create java applets.

# Java Environment

- Java API
- Java Development Kit (JDK)

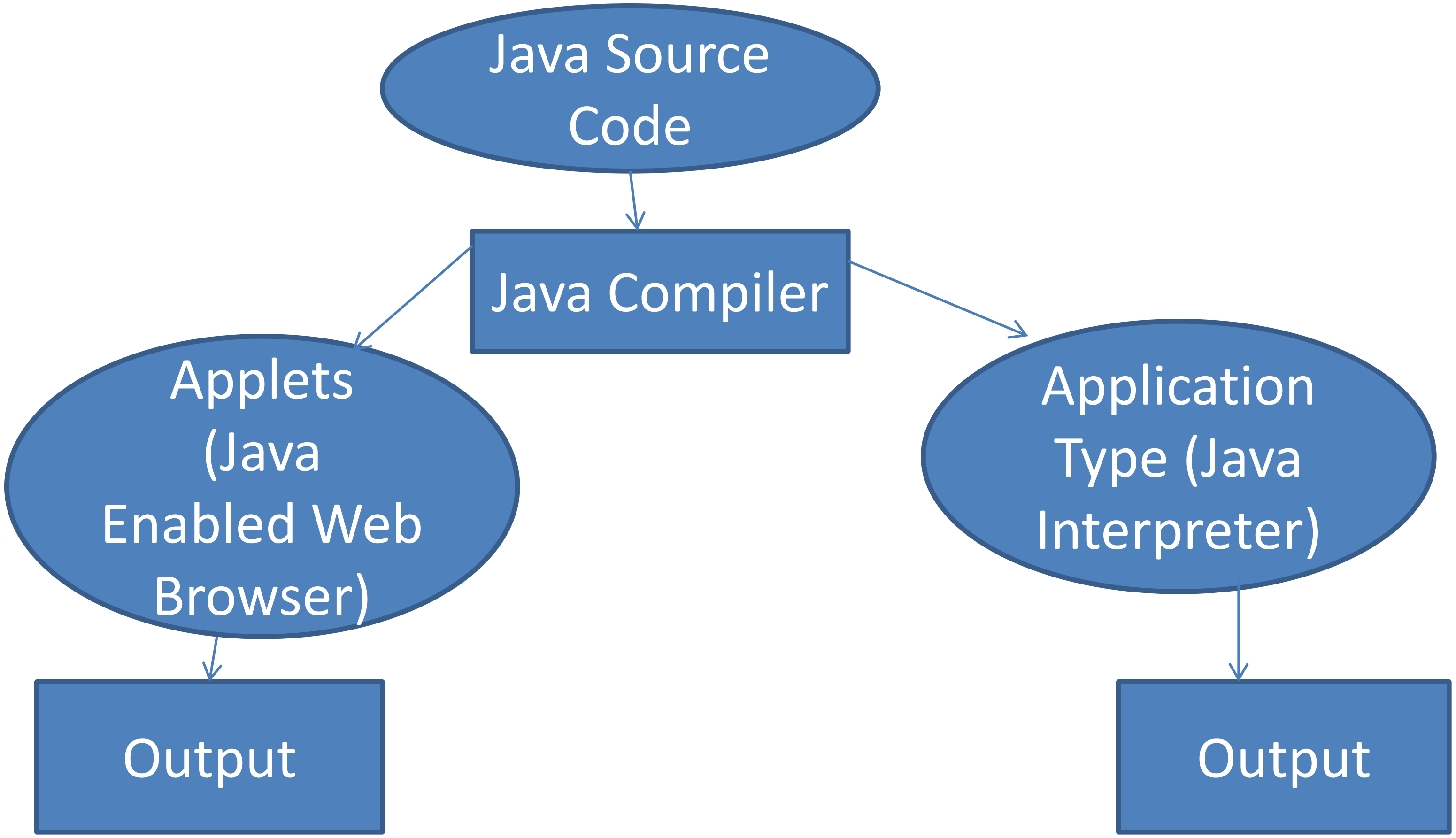
JDK

- ❑ javac – compiler which translates java source code to bytecode
- ❑ java – interpreter that interprets bytecode into machine instructions.
- ❑ appletviewer – for viewing java applets.

# Types of Java Programs

- Stand alone applications
- Web applets
- Servlets

# Types of Java Program


Java Source Code
Java Compiler
Applets (Java Enabled Web Browser)
Application Type (Java Interpreter)
Output
Output

# Characteristics of Java

- Compiled and interpreted
- Platform independent and portable- can run on any machine or Operating System.
- Object-oriented
- Robust and secure – strict runtime and compile time checking for data types. Supports exception handling.

# Characteristics of Java

- Distributed – java applications can access and open remote objects as they do in a local system.
- Simple, small and familiar – many features in C and c++ which are unreliable are not in java. To make it familiar, it was modelled on C and C++.
- Multithreaded and interactive – handling multiple tasks simultaneously.

# Characteristics of Java

- High performance- Due to intermediate bytecode and multithreading
- Dynamic – capable of linking new class libraries, methods and objects.

## Program Structure

public – any class can see the main method.

static – this allows to call a method without creating the object of that class. main() method should be declared as static since it is called by the interpreter before any instances are made.

void- not returning any value

main- method name

String args[] – array of instances of the class String.

## Program Structure

- System- built-in class used for output
- out- object of the built-in class System
- println() – built-in method used for printing the output. It also prints the newline character “\n”. (If print() is used instead of println(), then newline character “\n” will not be printed.)

# Reserved Keywords

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

## Identifiers

- Used for naming classes, methods, variables, labels, packages and interfaces.

Rules

1. Can have alphabets, digits, underscore and $.
2. Must not begin with a digit.
3. Uppercase and lowercase are distinct.
4. Can be of any length

## Identifiers

- Example for invalid identifiers

1more, 3$, a:b, #1, @abc

- Java Naming Conventions

1. Class names begin with an uppercase letter and each subsequent word with an uppercase letter. Eg: JavaTest, PrimeNum
2. All public methods and instance variables begin with a lowercase letter and each subsequent word by uppercase letter. Eg: nextItem, getMarks()

## Literals(Constants)

- Sequence of characters, digits, letters and other characters that represent constant values to be stored.

1. Integer literals
2. Floating point literals
3. Boolean literals
4. Character literals
5. String literals

# Literals- Integer literals

- Any integral numeric value is an integer literal.

1. Decimal Integer Literal – consists of digits from 0 to 9. Eg: 123, -81
2. Octal Integer Literal – consists of digits from 0 to 7, with a leading zero. Eg: 053, 041
3. Hexadecimal Integer Literal – consists of digits from 0 to 9, alphabets a to f. The sequence of digits are preceded by 0x or 0X. Eg: oxbca, ox345

# Literals- Floating point literals

- Represent decimal values with a fractional part.
- Can be expressed in standard or scientific notation.

1. Example for standard notation - 3.1419, 0.6667.
2. Example for scientific notation – 2.1565e2 means 2.1565 x $10^2$

# Literals- Boolean literals

- Only 2 logical values – true or false
- True is not equal to 1 and false not equal to 0

# Literals- Character literals

- Constants that contain a single character enclosed within a pair of single quotation mark.
- Eg: ’a’, ‘2’ etc.
- Backslash character constants – used for escape sequences and characters that cannot be entered directly.
- \’ – single quote
- \”- double quote
- \\ - backslash
- \n – New line
- \t – Tab space
- \b - Backspace

# Literals- String literals

- A sequence of characters enclosed within double quotes.
- Example – “Java”

## Variables

- Basic unit of storage in a Java program.
- Defined as a combination of an identifier, type and scope.

❑ Rules

1. Must not begin with a digit.
2. Uppercase and lowercase are distinct.
3. It should not be a keyword.
4. Whitespace is not allowed.
5. Can be of any length.

Example: int x,

float a,b

## Variables - Scope

```
class  scope{
public static void main(String args[ ])
{
     int b=1;
{    int b=2; //creates compile time error
}
}
}
```

## Data Types

- Integers
  - byte – 1 byte
  - short – 2 bytes
  - int – 4 bytes
  - long – 8 bytes
- Floating-point numbers
  - float – 4 bytes
  - double – 8 bytes
- Characters – 2 bytes
- Boolean – 1 byte

## Data Types

```
int three=3;
char one='1';
char four = (char)(three+one);
```

Result

four='4'

# Casting

- Automatic conversion – when the destination variable has enough precision to hold the source value (widening). Eg: byte a;

  int b=a

- Narrowing – storing an integer variable into a byte variable. In this case casting is required.

Example: int a=100

byte b =(byte) a

## Operators

1. Arithmetic operators
2. Relational operators
3. Logical operators
4. Assignment operators
5. Increment & Decrement operators
6. Conditional operators
7. Bitwise operators
8. Special operators

## Arithmetic Operator

- + Addition
- - Subtraction
- * Multiplication
- / Division
- % Modulus

## Relational Operator

- <   Less than
- <=  Less than or Equal to
- >    Greater than
- >=  Greater than or Equal to
- ==  Equal to
- !=  Not Equal to

## Logical Operator

- && - Logical AND
- || - Logical OR
- ! – Logical Not

Example: if(age>55 && salary >10000)

## Assignment Operator

- =
- Short hand assignment operator
- +=
- -=
- *=
- /=

Example : a+=1

## Increment & Decrement Operator

- ++
- --

Example : a++, ++a

# Conditional Operator

Syntax:

exp1 ? exp2:exp3 where exp1, exp2 and exp3 are expressions.

exp1 is evaluated first and if it is True, exp2 is evaluated and becomes the value of the conditional expression.

If exp1 is False, exp3 is evaluated and its value becomes the conditional expression.

Example: x=(a>b)?a:b     a=10, b=15

## Bitwise Operators

- Used for testing the bits or shifting them to the right or left.
- & - Bitwise AND
- | - Bitwise OR
- ^ - Bitwise Exclusive OR
- ~ - 1's Complement (NOT Operator)
- << - Shift left
- >> - Shift right

## Bitwise Operators

Bitwise AND operator(&)

00101010

00001111

00001010 --------Result

Bitwise OR operator(|)

00101010

00001111

00101111---------Result

# Bitwise Operators

Bitwise Exclusive OR operator(^)

00101010

00001111

00100101 --------Result

Bitwise NOT operator(~)

00101010

11010101---------Result

# Bitwise Operators

Bitwise shift LEFT operator(<<)

00101010 <<2

01010100

10101000 --------Result

Bitwise shift RIGHT operator(>>)

00101010 >>2

00010101

00001010---------Result

# Special Operators

- instanceof

Example: if (Student instanceof(person))

- Member selection operator(.)

Example: Student.age

## Operator Precedence – Highest to Lowest

1. (), []
2. ++, -- ,~ , !
3. *, /, %
4. +, -
5. >>, <<
6. >,  >=, <, <=
7. ==, !=
8. &
9. ^
10. |
11. &&
12. ||
13. ?:
14. =, op=

## SAMPLE PROGRAMS

- Command Line Programs

1. Find the sum of two numbers.

## Assignment

- Control Statements

✓ Branching statements

1. If statement
2. If…else statement
3. Nested if statement
4. Switch statement

# Assignment

- Control Statements

✓ Looping statements

1. While loop
2. Do while loop
3. For loop

✓ Jump statements

1. Break
2. Continue
3. return

✓ Labelled loops (with continue)

## Assignment

- Control Statements

✓ Labelled loops (with continue)

**Example Program**

```
class ContinueLabel{
public static void main(String args[])
{
outer: for(int i=0;i<5;i++){
      for(int j=0;j<5;j++){
      if(j>i)
      { System.out.println(" ");
        continue outer;
      }
System.out.print(" "+(i*j));
}
}}}
```

## End of Unit 1