Resource Software Solution

Session 01: Introduction to Java

Objectives

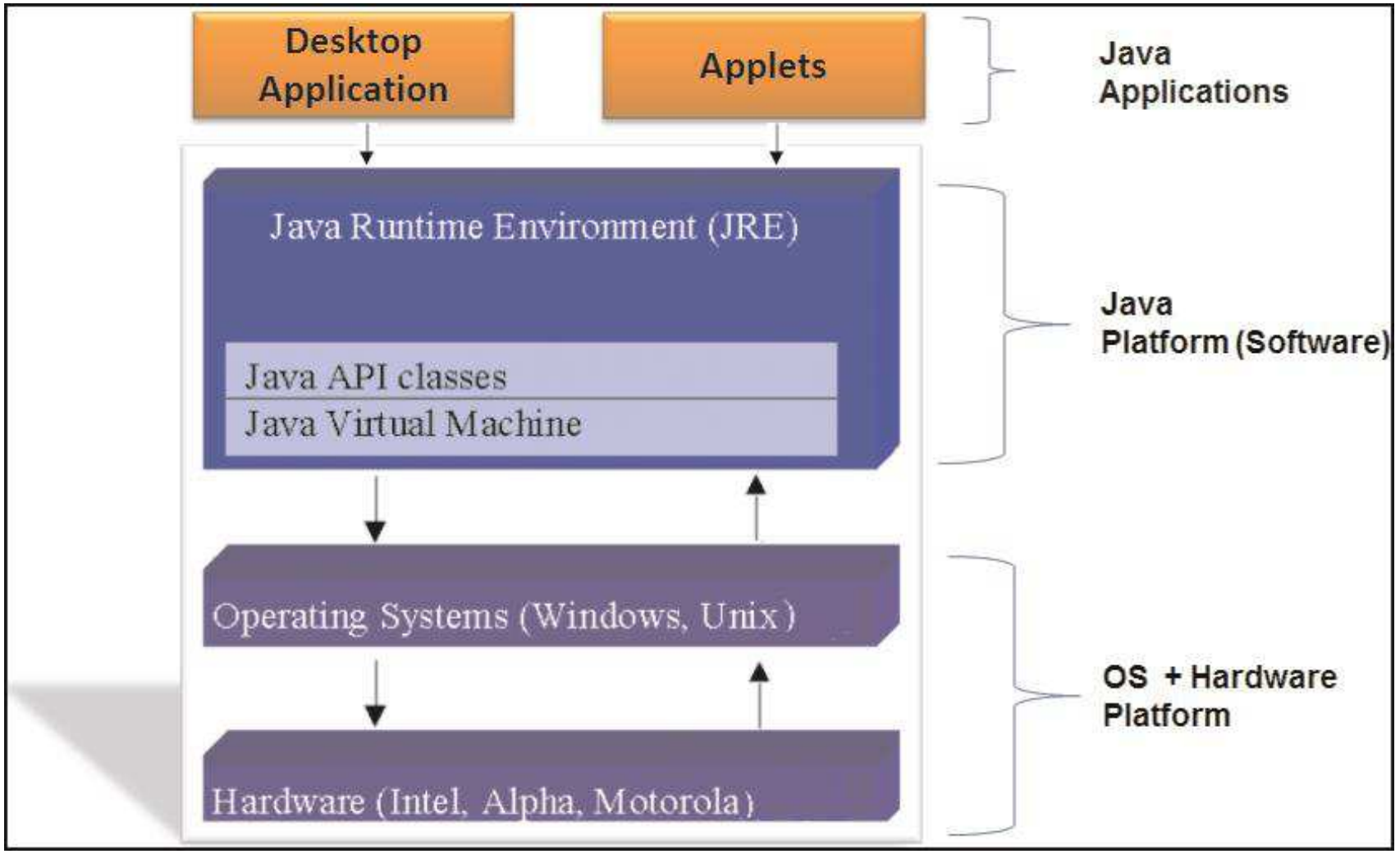
1. Overview
2. Develop, Compile and Execute
3. Variable
4. Operators
5. Standard Input and Output

Overview (1)

- It helps programmers to develop wide range of applications that can run on various hardware and Operating System (OS)
- Java applications are built on variety of platforms that range from
  - **Desktop Application**
  - **Web Application**
  - **Mobile Application**

Overview (2)

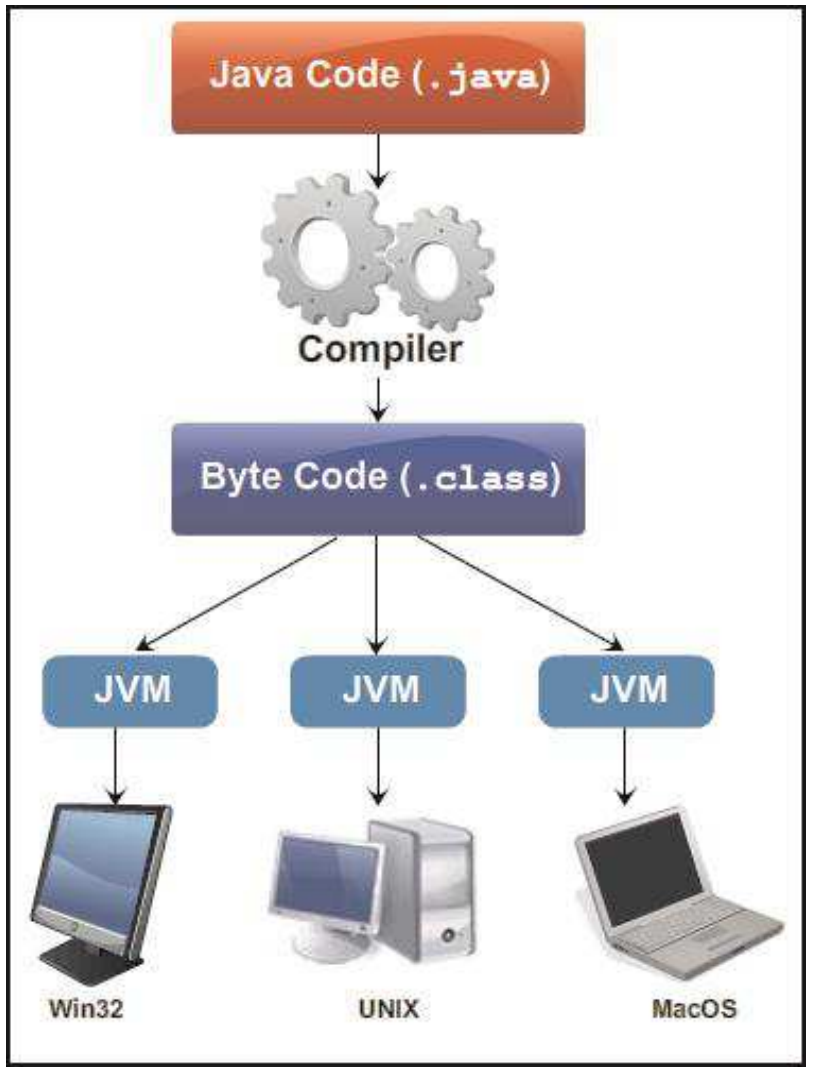
- Java platform



Overview (3)

Bytecode

- Bytecode is an intermediate form closer to machine representation
- The **same bytecode** can be **executed by different** implementations of JVM on **various platforms**



Resource Software Solution

Overview (4)

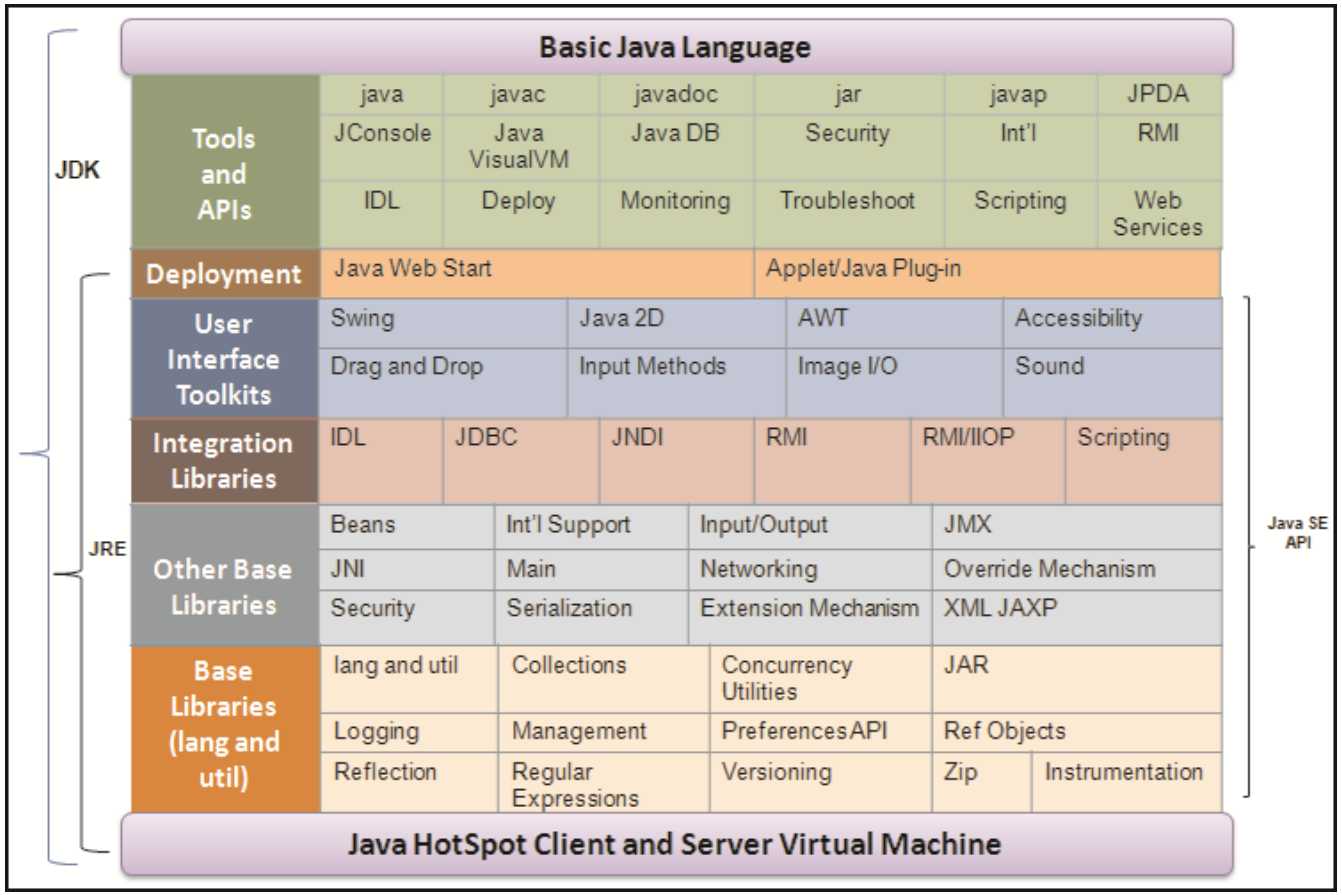
Edition of Java

- **Java Standard Edition (Java SE)** - Is a base platform and enables to develop console and networking applications for desktop computers
- **Java Enterprise Edition (Java EE)** - Is built on top of Java SE platform and provides a standard specification for developing and deploying distributed, scalable, and multi-tier enterprise applications

Overview (5)

Components of Java SE Platform

- **JRE** known as **Java Runtime Environment.** It provides JVM and Java libraries that are used to run a Java program
- **JDK** known as Java Development Kit. It contains a comprehensive set of tools, such as compilers and debuggers that are used to develop Java applications



Resource Software Solution

2

Develop, Compile and Execute

Resource Software Solution

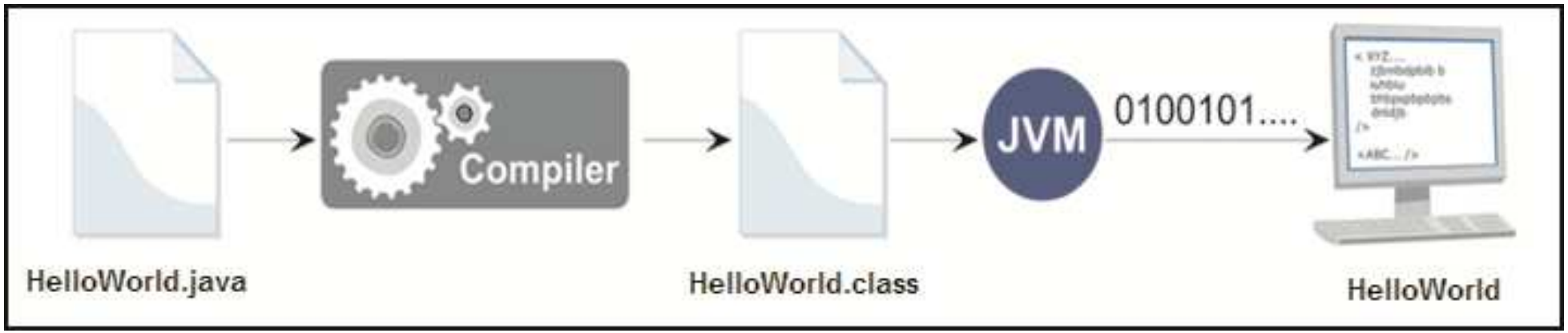
Structure of a Java Class

- The definition of the class is written in a file and is saved with a **.java extension**
- The Java programming language is designed around object-oriented features and begins with a class design

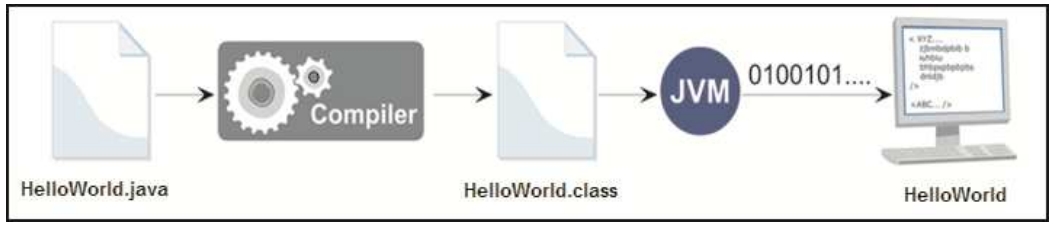
```
package <package_name>;

import <other_packages>;

public class ClassName {
    <variables(also known as fields)>;

    <constructor method(s)>;

    <other methods>;
}
```

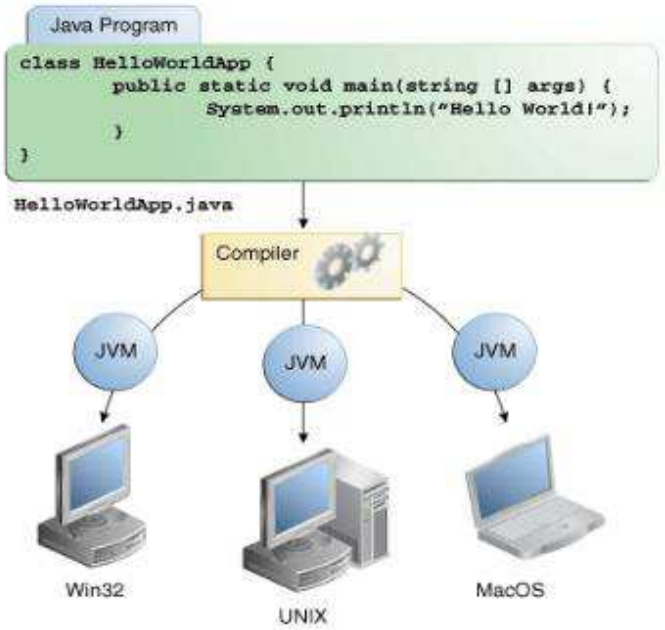
Compile .java file (1)

- The **HelloWorld.java** file is known as source code file
- It is compiled by invoking tool named **javac.exe**, which compiles the source code into a **.class file**
- The .class file contains the bytecode which is interpreted by **java.exe** tool



Compile .java file (2)





Through the Java VM, the same application is capable of running on multiple platforms.

Resource Software Solution

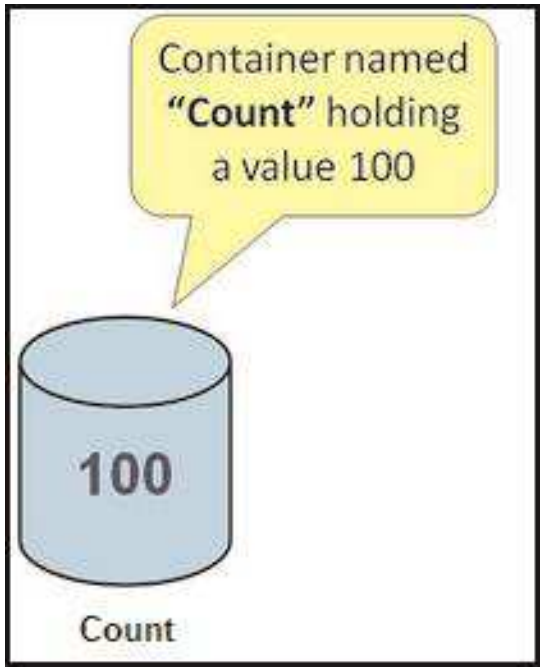
3

Explain variables and their purpose

Variables (1)

Introduction

- A variable is a **location in the computer's memory** which **stores the data** that is used in a Java program such as names, addresses, and salary details
- It used in a Java program to store data that **changes during** the execution of the program



Variables (2)

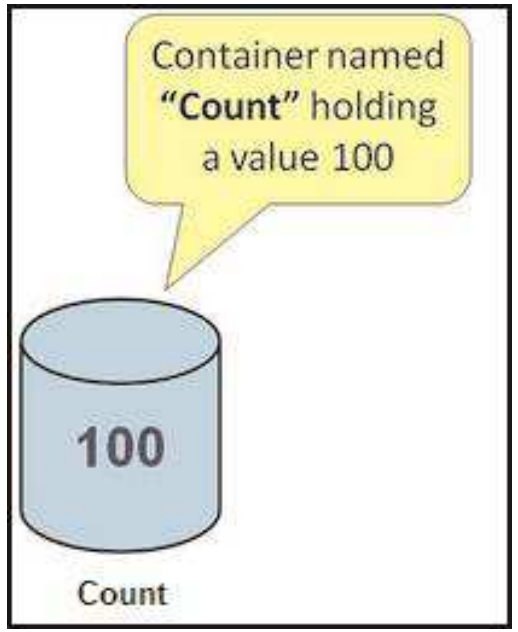
Syntax

- The syntax to declare a variable in a Java program is as follows

```
datatype variableName [=value][, variableName [=value]...];
```

- Where
  - datatype: Is a valid data type in Java
  - variableName: Is a valid variable name
- Following code snippet demonstrates

```
int count;
```



Resource Software Solution

Variables (3)

Rule for Naming variables

- Variable names must **not be a keyword**, and must **begin with a letter**
- Variable names in Java are **case-sensitive**. For example, the variable names number and Number refer to two different variables
- If a variable name comprises a single word, the name should be in **lowercase**. For example, **age**
- If the variable name consists of more than one word, the first letter of each subsequent word should be capitalized. For example, **employeeNumber**

Variables (4)

Assigning value to a variable

- Values can be assigned to variables by using = operator
- Following code snippet demonstrates the initialization of variables at the time of declaration

```
int rollNumber = 101;
char gender = 'M';
```

- Where
  - In the code, variable **rollNumber** is an integer variable, so it has been initialized with a numeric value **101**
  - Similarly, variable **gender** is a character variable and is initialized with a character '**M**'

Resource Software Solution

Variables (5)

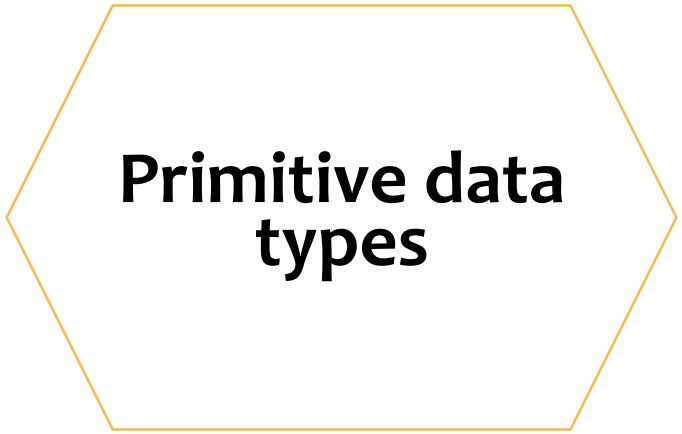
Data types (1)

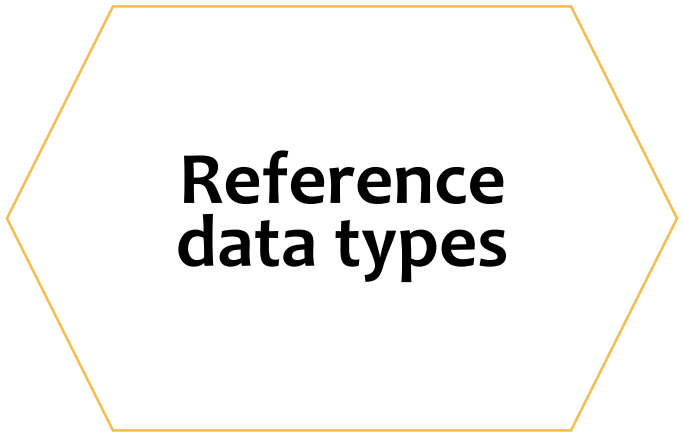
- When you define a variable in Java, you must inform the compiler what **kind of a variable** it is
- That is, whether it will be expected to store an **integer**, a **character**, or **some other kind** of data
- This information tells the **compiler how much space to allocate in the memory** depending on the data type of a variable
- Thus, the data types determine the type of data that can be stored in variables and the operation that can be performed on them

Variables (6)

Data types (2)

- In Java, data types fall under two categories that are as follows

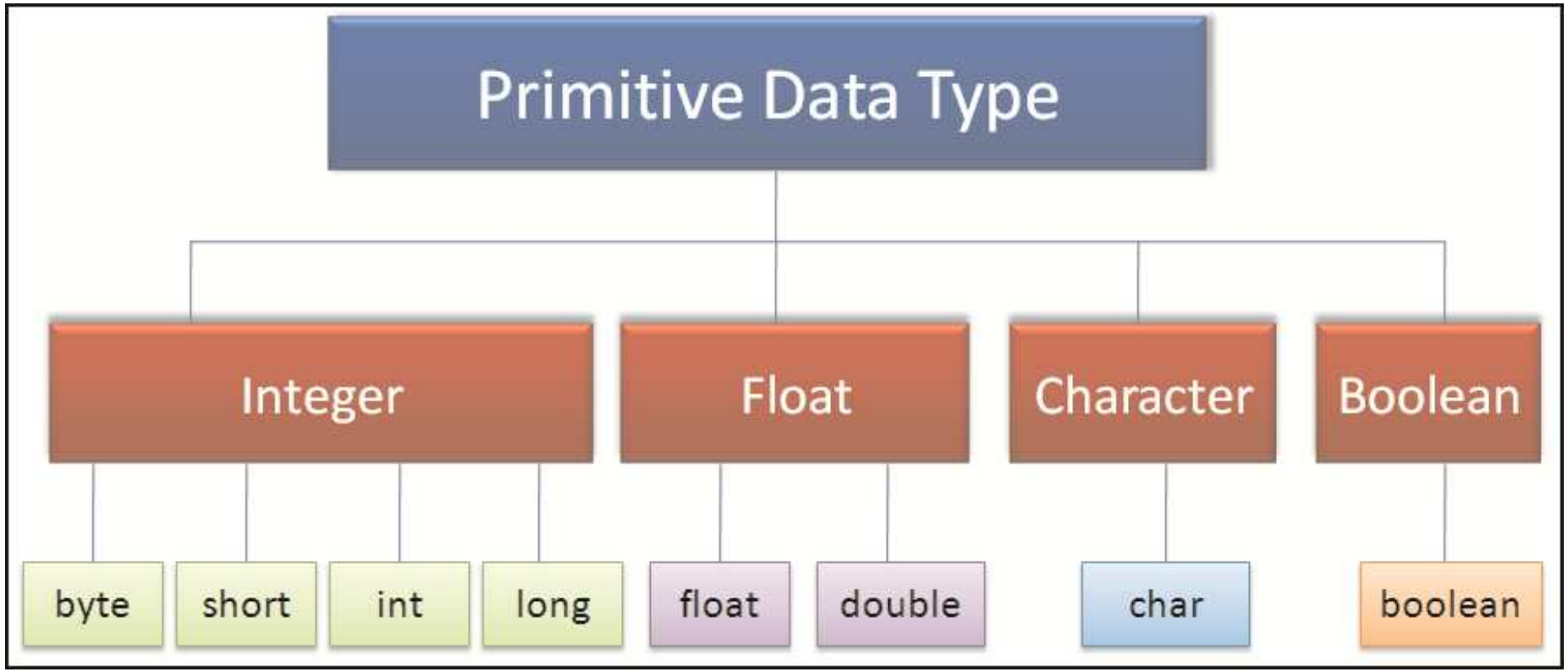




Variables (7)

Data types (3)

- Primitive data types



Resource Software Solution

Variables (8)

Data types (4)

- Primitive data types - Integer
  - **byte**: The byte data type is an **8-bit** signed two's complement integer. It has a minimum value of **-128** and a maximum value of **127** (inclusive)
  - **short**: The short data type is a 16-bit signed two's complement integer. It has a minimum value of **-32,768** and a maximum value of **32,767** (inclusive)
  - **int**: The int data type is a 32-bit signed two's complement integer. It has a minimum value of **-2,147,483,648** and a maximum value of **2,147,483,647** (inclusive)
  - **long**: The long data type is a 64-bit signed two's complement integer. It has a minimum value of **-9,223,372,036,854,775,808** and a maximum value of **9,223,372,036,854,775,807** (inclusive)

Resource Software Solution

Variables (9)

Data types (5)

- Primitive data types – Float, Character, and Boolean
  - **float:** The float data type is a single-precision 32-bit IEEE 754 floating point. Its range of values is from $3.4E^{-45}$ to $3.4E^{38}$
  - **double:** The double data type is a double-precision 64-bit IEEE 754 floating point. Its range of values is from $1.7E^{-324}$ to $1.79769313486231 57E^{308}$
  - **char:** The char data type is a single 16-bit Unicode character
  - **boolean:** The boolean data type has only two possible values: **true** and **false**

Resource Software Solution

Variables (10)

Data types (6)

- Default Values

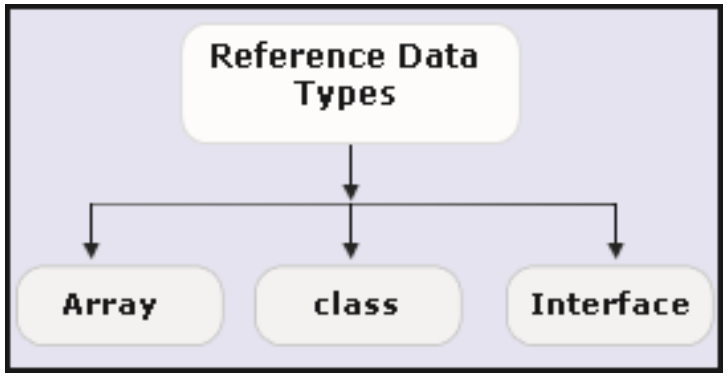
Data Type	Default Value (for fields)
byte	0
short	0
int	0
long	0L
float	0.0f
double	0.0d
char	'\u0000'
String (or any object)	null
boolean	false

Variables (11)

Data types (7)

- Reference Data Types

Data Type	Description
Array	It is a collection of several items of the same data type. For example, names of students in a class can be stored in an array.
Class	It is encapsulation of instance variables and instance methods.
Interface	It is a type of class in Java used to implement inheritance.



Resource Software Solution

Variables (12)

Type Casting (1)

- In type casting, a data type is converted into another data type
- Automatic Type Promotion in Expressions

```
public class AutomaticTypePromotion {
  public static void main(String[] args) {
    byte a = 40, b = 50, c = 100;
    int d = a * b / c;
    b = b * 2; // Error! Cannot assign an int to a byte!
    System.out.println("Value d: " + d);
  }
}
```

Variables (13)

Type Casting (2)

- In type casting, a data type is converted into another data type
- Automatic Type Promotion in Expressions

```
public class AutomaticTypePromotion {
  public static void main(String[] args) {
    byte a = 40, b = 50, c = 100;
    int d = a * b / c;
    b = b * 2; // Error! Cannot assign an int to a byte!
  }
}
```

- **Widening**[an toàn/mở rộng]**conversions:**
  - byte->short->int->long->float->double

Resource Software Solution

Variables (14)

Type Casting (3)

- **Type casting in Expressions:** Casting is used for explicit type conversion. It loses information above the magnitude of the value being converted
- **Example**

```
float f = 34.89675f;
int d = (int) (f + 10);
```

Resource Software Solution

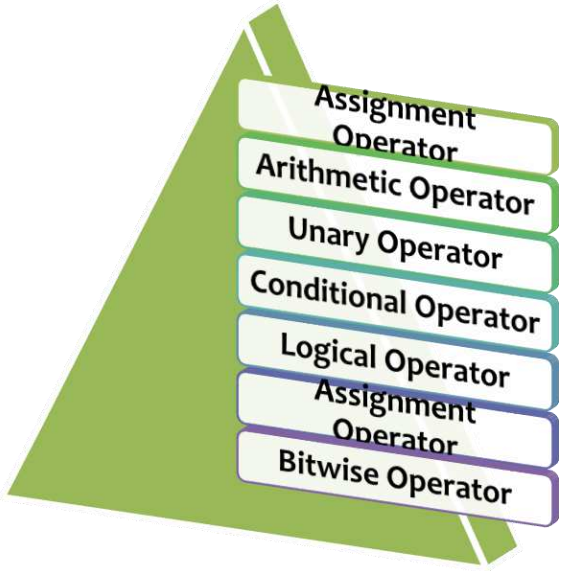
4

Operators

Operators (1)

Introduction

- Operators are set of symbols used to indicate the kind of operation to be performed on data
- Java provides several categories of operators and they are as follows



Operators (2)

Arithmetic Operators (1)

- Arithmetic operators manipulate numeric data and perform common arithmetic operations on the data

Operator	Description
+	Addition - Returns the sum of the operands
-	Subtraction - Returns the difference of two operands
*	Multiplication - Returns the product of operands
/	Division – Returns the result of division operation
%	Remainder - Returns the remainder from a division operation

Operators (3)

Arithmetic Operators (2)

```
public class ArithmeticOperator {
  public static void main(String[] args) {
    double number1 = 12.5, number2 = 3.5, result;

    // Using addition operator
    result = number1 + number2;
    System.out.println("number1 + number2 = " + result);

    // Using remainder operator
    result = number1 % number2;
    System.out.println("number1 % number2 = " + result);
  }
}
```

Output:

number1 + number2 = 16.0

number1 % number2 = 2.0

Operators (4)

Unary Operators (1)

- They increment/decrement the value of a variable by 1, negate an expression, or invert the value of a boolean variable

Operator	Description
+	Unary plus - Indicates a positive value
-	Unary minus - Negates an expression
++	Increment operator - Increments the value of a variable by 1
--	Decrement operator - Decrements the value of a variable by 1
!	Logical complement operator - Inverts a boolean value

Operators (5)

Unary Operators (2)

```
public class UnaryOperator {
 public static void main(String[] args) {
   double number = 5.2;
   boolean flag = false;

   // ++number is equivalent to number = number + 1
   System.out.println("number = " + ++number);

   // -- number is equivalent to number = number - 1
   System.out.println("number = " + --number);

    System.out.println("!flag = " + !flag);
   }
}
```

Output:

number = 6.2

number = 5.2

!flag = true

Operators (6)

Conditional Operators (1)

- The conditional operators test the relationship between two operands. An expression involving conditional operators always evaluates to a boolean value (that is, either true or false)

Operator	Description
==	Equal to – Checks for equality of two numbers
!=	Not Equal to - Checks for inequality of two values
>	Greater than - Checks if value on left is greater than the value on the right
<	Less than - Checks if the value on the left is lesser than the value on the right
>=	Greater than or equal to - Checks if the value on the left is greater than or equal to the value on the right
<=	Less than or equal to – Checks if the value on the left is less than or equal to the value on the left

Resource Software Solution

Operators (7)

Conditional Operators (2)

```
public class ConditionalOperator {
  public static void main(String[] args) {
    int number1 = 5, number2 = 6;

    if (number1 > number2) {
      System.out.println("number1 is greater than number2.");
    } else {
      System.out.println("number2 is greater than number1.");
    }
  }
}
```

Output:

number2 is greater than number1.

Operators (8)

Logical Operators (1)

- Logical operators (&& and ||) work on two boolean expressions

Operator	Description
&&	Conditional AND - Returns true only if both the expressions are true
||	Conditional OR - Returns true if either of the expression is true or both the expressions are true

Resource Software Solution

Operators (9)

Logical Operators (2)

```
public class LogicalOperator {
  public static void main(String[] args) {
    int number1 = 1, number2 = 2, number3 = 9;
    boolean result;

    result = (number1 > number2) || (number3 > number1);
    System.out.println(result);

    result = (number1 > number2) && (number3 > number1);
    System.out.println(result);
  }
}
```

Output:
true false

Resource Software Solution

Operators (10)

Ternary Operator

- The ternary operator (?:) is a shorthand operator for an if-else statement
- Syntax

```
expression1 ? expression2 : expression3
```

- Where
  - expression1: Represents an expression that evaluates to a boolean value of true or false
  - expression2: Is executed if expression1 evaluates to true
  - expression3: Is executed if expression1 evaluates to false

Resource Software Solution

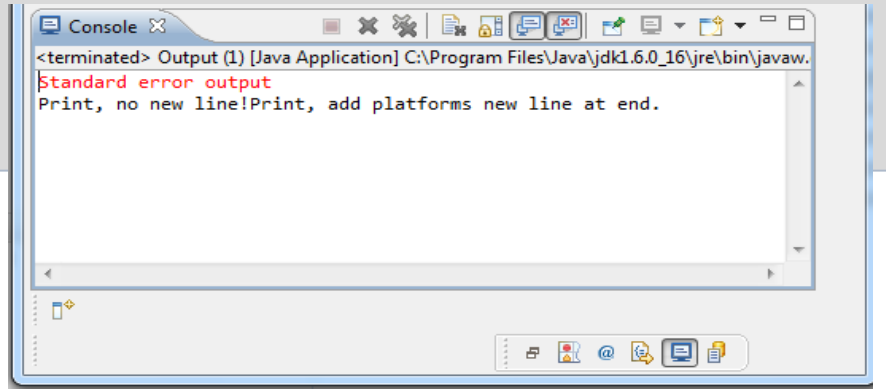
5

Input and Output

Standard Java Output

- System.out is standard out in Java
- System.err is error out in Java
- Example

```
public class Output {
  public static void main(String[] args) {
     System.err.println("Standard error output");
      System.out.print("Print, no new line!");
      System.out.println("Print, add platforms new line at end.");
  }
}
```



Standard Java Input

- Scanner is standard out in Java
- Example

```
public class Input {
 public static void main(String[] args) {
   Scanner s = new Scanner(System.in);


   int n = s.nextInt();
   float m = s.nextFloat();
   String str = s.nextLine();
 }
}
```

Resource Software Solution

Thankyou!