SNS COLLEGE OF ENGINEERING

Kurumbapalayam (Po), Coimbatore – 641 107

An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A' Grade

Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

COURSE NAME : 23CSB101- OBJECT ORIENTED PROGRAMMING

I YEAR /II SEMESTER

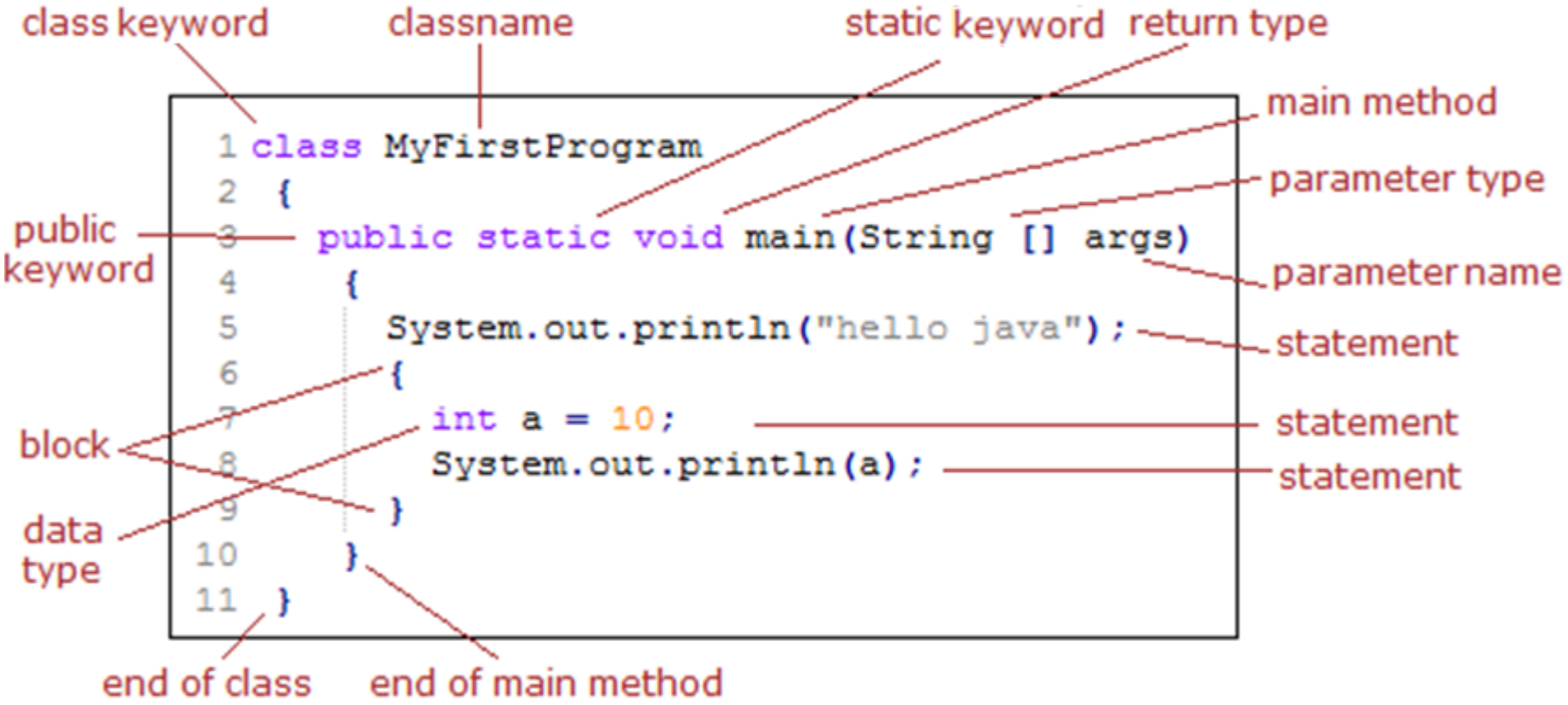
Unit I – INTRODUCTION TO OOP AND JAVA

Topic : DATATYPES – VARIABLES - ARRAY

A First Simple Program

```
/*
   This is a simple Java program.
   Call this file "Example.java".
*/
class Example {
  // Your program begins with a call to main().
  public static void main(String args[]) {
    System.out.println("This is a simple Java program.");
  }
}
```

A First Simple Program



A First Simple Program

Entering the Program

The name of the source file should be **Example.java**
the **name of the class** defined by the program is also Example
By convention, the name of **the main class** should match the name of the file that holds the program
The Java **compiler** requires that a source file use the .java filename extension.

To **compile** the Example program, **execute** the compiler, **javac**,

C:\>javac Example.java

The **javac** compiler **creates** a file called **Example.class** that contains the **bytecode** version of the program

A First Simple Program

To **run** the program, use the Java application launcher called java.

Now pass the class name Example as a command-line argument, as :

C:\>java Example

When the program is run, the following **output** is displayed:

This is a simple Java program.

Note: When Java source code is compiled, each individual class is put into its own output file named after the class and using the .class extension.

A First Simple Program

- **class** keyword is used to declare a class in Java.
- **public** keyword is an access modifier that represents visibility. It means it is visible to all.
- **static** is a keyword. If we declare any method as static, it is known as the static method. The core advantage of the static method is that there is no need to create an object to invoke the static method. The main() method is executed by the JVM, so it does not require creating an object to invoke the main() method. So, it saves memory.

A First Simple Program

- **void** is the return type of the method. It means it does not return any value.
- The **main()** method represents the starting point of the program.
- **String[] args** or **String args[]** is used for command line argument.
- **System.out.println()** is used to print statement on the console. Here, System is a class, out is an object of the PrintStream class, println() is a method of the PrintStream class.
- /* …. */ is a comment statement in java

Atomic elements of Java

Java programs are a collection of

- Whitespace
- identifiers
- literals
- comments
- operators
- separators,
- keywords.

Atomic elements of Java

Whitespace - there was at least one whitespace character between each token that was not already delineated by an operator or separator. In Java, whitespace includes a space, tab, newline, or form feed.

Identifiers - Identifiers are used to name things, such as classes, variables, and methods. An identifier may be any descriptive sequence of uppercase and lowercase letters, numbers, or the underscore and dollar-sign characters.

Literals - A constant value in Java is created by using a literal representation of it.

Atomic elements of Java

Comments - there are **three** types of comments defined by Java. single-line(//), multiline (/* ... */) and documentation comment used to produce an HTML file that documents begins with a /** and ends with a */.

Separators - used to terminate statements.

Keywords - keywords cannot be used as identifiers, meaning that they cannot be used as names for a variable, class, or method. An underscore by itself is considered a keyword.

Operators - symbols that perform operations on variables and values.

Identifiers

- Identifiers are used to name things, such as classes, variables, and methods.

Rules for Valid Identifiers

- Can contain letters (A-Z, a-z), digits (0-9), underscore (_), and dollar sign ($).
- Cannot start with a digit (e.g., 2name is invalid).
- Cannot be a Java reserved keyword (e.g., int, class are invalid).
- Case-sensitive (name and Name are different).
- No spaces allowed (e.g., my name is invalid).
- Should be meaningful (recommended for readability).

Literals

- A **literal** in Java is a constant value assigned to a variable. Java supports different types of literals based on data types.
- **Types of Literals in Java**

1. **Integer Literals-** Used for whole numbers.

   Decimal (Base 10) → int x = 100;

   Binary (Base 2, starts with 0b) → int y = 0b1010; // 10 in decimal

   Octal (Base 8, starts with 0) → int z = 012; // 10 in decimal

   Hexadecimal (Base 16, starts with 0x) → int a = 0xA; // 10 in decimal

   2. Floating-Point Literals

   float f = 10.5f; // 'f' is required for float

   double d = 20.99; // Default is double

Literals

2. **Floating-Point Literals**

   float f = 10.5f; // 'f' is required for float

   double d = 20.99; // Default is double

3. **Character Literals** - Used for single characters enclosed in single quotes (' ‘).

   Escape sequence characters belongs to character literals.

   char ch = 'A';

   char unicodeCh = '\u0041'; // Unicode for 'A'

4. **String Literals** - Used for text enclosed in double quotes (" ").

   String str = "Hello, Java!";

5. **Boolean Literals** - Only two values: true and false.

   boolean isJavaFun = true;

Cont…

Literals

6. **Null Literal** - Represents no value (used for reference types).

   String s = null;

7. **Underscore in Numeric Literals** - Used to improve readability in large numbers.

   int million = 1_000_000; // Same as 1000000

   double pi = 3.14_159_265;

Literals

Escape Sequence	Description
\ddd	Octal character (ddd)
\uxxxx	Hexadecimal Unicode character (xxxx)
\'	Single quote
\"	Double quote
\\	Backslash
\r	Carriage return
\n	New line (also known as line feed)
\f	Form feed
\t	Tab
\b	Backspace

Datatypes In Java

- Java has two main categories of data types: **Primitive Data Types** and **Non-Primitive Data Types**.
- **1. Primitive Data Types -** Primitive data types are built-in, fundamental types that store simple values. There are **8 primitive data types** in Java
- **2. Non-Primitive Data Types -** Non-primitive (reference) data types store references to objects.

Datatypes In Java

- Java defines **eight primitive types(**simple types) of data: byte, short, int, long, char, float, double, and boolean. These can be put in four groups:

1. **Integers** This group includes byte, short, int, and long, which are for whole-valued signed numbers.

2. **Floating-point numbers** This group includes float and double, which represent numbers with fractional precision.

3. **Characters** This group includes char, which represents symbols in a character set, like letters and numbers.

4. **Boolean** This group includes boolean, which is a special type for representing true/false values.

Datatypes In Java

```
int age = 25;  // Integer variable

double price = 99.99;  // Floating-point variable

char grade = 'A';  // Character variable

boolean isJavaFun = true;  // Boolean variable

byte b, c;   //byte type variable

short t;    //short type variable

long  days; //long type variable
```

Primitive Datatypes In Java

Data Type	Size	Default Value	Description
byte	1 byte	0	Stores small integers (-128 to 127)
short	2 bytes	0	Stores integers (-32,768 to 32,767)
int	4 bytes	0	Stores whole numbers (-2^{31} to $2^{31}-1$)
long	8 bytes	0L	Stores large integers (-2^{63} to $2^{63}-1$)
float	4 bytes	0.0f	Stores decimal numbers with single precision
double	8 bytes	0.0d	Stores decimal numbers with double precision
char	2 bytes	'\u0000'	Stores a single character (Unicode)
boolean	1 bit	false	Stores true or false values

Variables In Java

A **variable** is a container that holds data. Java variables must be declared with a specific type.

Declaring Variables

The basic/general form of a variable declaration

type identifier [= value][, identifier [= value] …];

type is one of Java's atomic types, or the name of a class or interface. The **identifier** is the name of the variable. And the value for variable specified by an equal sign and a value.

Variables In Java

Declaring Variables - examples:

- int age = 25; // Integer variable
- double price = 99.99; // Floating-point variable
- char grade = 'A'; // Character variable
- boolean isJavaFun = true; // Boolean variable
- float hightemp, lowtemp;

Variables In Java

Types of Variables

- Local Variables → Declared inside a method and used only within that method.
- Instance Variables → Declared inside a class but outside any method (associated with an object).
- Static Variables → Declared with the static keyword; shared among all instances of a class.

Variables In Java

Example:

```
class Example {
    int instanceVar = 10;  // Instance variable
    static int staticVar = 20;  // Static variable
    void method() {
        int localVar = 30;  // Local variable
    }
}
```

Variables In Java

The Scope and Lifetime of Variables:

A block begins with an opening curly brace and ended by a closing curly brace. A **block defines a scope**. Thus, each time you start a new block, you are creating a new scope. A **scope determines** what objects are **visible** to other parts of your program. It also determines the lifetime of those objects. A variable declared within a block is called a **local variable**.

In **Java**, the **two** major **scopes** are those defined by a **class** and those defined by a **method**.

Type Conversion and Casting

Java's Automatic Conversions:

When one type of data is assigned to another type of variable, an automatic type conversion will take place if the following two conditions are met:

- The two types are compatible.
- The destination type is larger than the source type.

When these two conditions are met, a widening conversion takes place. **Example** , the int type is always large enough to hold all valid byte values, so no explicit cast statement is required.

Type Conversion and Casting

Java’s No-Automatic Conversions:

However, there are **no automatic conversions** from the numeric types to char or boolean. Also, char and Boolean are not compatible with each other.

Type Conversion and Casting

Casting:

To create a conversion between two incompatible types, you must use a cast. A cast is simply an explicit type conversion.

General form:

(target-type) value

Here, target-type specifies the desired type to convert the specified value to.

Remember: when a floating-point value is assigned to an integer type results in truncation.

Arrays In Java

```
// Declaration
int[] numbers;

// Initialization
numbers = new int[5];

// Declaration + Initialization
int[] numbers = {10, 20, 30, 40, 50};
```

Arrays In Java

b) Accessing Elements

```
System.out.println(numbers[0]);  // Output: 10
```

c) Iterating Through an Array

```
for (int i = 0; i < numbers.length; i++) {
    System.out.println(numbers[i]);
}
```

(OR)

```
for (int num : numbers) {
    System.out.println(num);
}
```

Arrays In Java

d) Multi-Dimensional Arrays

```
int[][] matrix = {
{1, 2, 3},
{4, 5, 6}
};

// Accessing elements
System.out.println(matrix[0][1]);  // Output: 2
```

Summary

Datatypes forms the basics of any programming language.

Java has two main categories of data types:

Primitive Data Types and **Non-Primitive Data Types**.


Thank
you