JAVA PROGRAM STRUCTURE AND BASIC CONCEPTS

Syntax, Types, Control Flow, and OOP Basics

AI Assisted Programming

Week 2 | 60 minutes

LEARNING OUTCOMES

By the end of this lecture, you will be able to:

- Explain Java program structure: classes, methods, and the `main` entry point.
- Write, compile, and run a simple Java program from the command line.
- Use variables, primitive types, operators, and strings appropriately.
- Apply control flow (if/switch/loops) and define methods with parameters and returns.
- Work with arrays and create simple classes and objects.

AGENDA

- Hello World and program structure
- Compile and run: `javac` and `java`
- Variables, primitive types, operators, and strings
- Control flow: if, switch, loops
- Methods, arrays, and simple objects

HELLO, WORLD!

```java
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, world!");
    }
}
```

- File name must match public class: `HelloWorld.java`
- `main` is the entry point: `public static void main(String[] args)`

COMPILE AND RUN

COMMANDS

```
javac HelloWorld.java
java HelloWorld
```

NOTES

- `javac` compiles source to `.class` bytecode
- `java` runs the class by name (no `.class`)
- Ensure JDK is installed and on PATH

VARIABLES AND TYPES

PRIMITIVES

- `byte`, `short`, `int`, `long`
- `float`, `double`
- `char`, `boolean`

EXAMPLE

```
int count = 3;
double price = 19.99;
char grade = 'A';
boolean done = false;
String name = "Ada"; // reference type
```

OPERATORS AND STRINGS

- Arithmetic: `+ - * / %`; Assignment: `= += -=`
- Comparison: `== != < > <= >=`; Logical: `&& || !`
- String concatenation: `"Hello, " + name`
- Use `.equals()` to compare strings, not `==`

CONTROL FLOW

IF / ELSE

```
if (score >= 50) {
    System.out.println("Pass");
} else {
    System.out.println("Fail");
}
```

SWITCH

```
switch (day) {
    case 1: System.out.println("Mon"); break;
    case 2: System.out.println("Tue"); break;
    default: System.out.println("Other");
```

```
}
```

LOOPS

```
for (int i = 0; i < 3; i++) {
    System.out.println(i);
}

int j = 0;
while (j < 3) {
    j++;
}
```

METHODS

```java
static int add(int a, int b) {
    return a + b;
}

public static void main(String[] args) {
    int sum = add(2, 3);
    System.out.println(sum);
}
```

- Methods can be `static` or instance
- Parameters, return types, and overloading

ARRAYS

```java
int[] nums = {1, 2, 3};
System.out.println(nums.length);
for (int n : nums) {
    System.out.println(n);
}
```

- Fixed-size, zero-indexed sequences
- Use enhanced for-loop for iteration

OBJECTS AND CLASSES

```
class Person {
    String name;
    Person(String name) { this.name = name; }
}

public class Main {
    public static void main(String[] args) {
        Person p = new Person("Ada");
        System.out.println(p.name);
    }
}
```

- Classes define state (fields) and behavior (methods)
- Create objects with new; constructors initialize state

TIPS AND COMMON ERRORS

- File/class name mismatch: `PublicClass.java` must match `public class PublicClass`
- Missing semicolons and braces
- Using == instead of `.equals()` for strings
- Package and directory structure must align

SUMMARY

- Java programs are class-based with a `main` entry point
- Compile with `javac`, run with `java`
- Use primitives, strings, operators, and control flow to model logic
- Organize code with methods, arrays, and simple classes

NEXT STEPS

- Practice: write a program that reads a name and greets
- Implement FizzBuzz and a simple calculator using methods
- Explore arrays and loops with small exercises

THANK YOU

Questions & Discussion

AI Assisted Programming

Next: Prompting for Programmers

Speaker notes