Chapter 03

Programming Principles

Functions & Modular Thinking

Mr. SOK Pongsametrey
metreysk@gmail.com

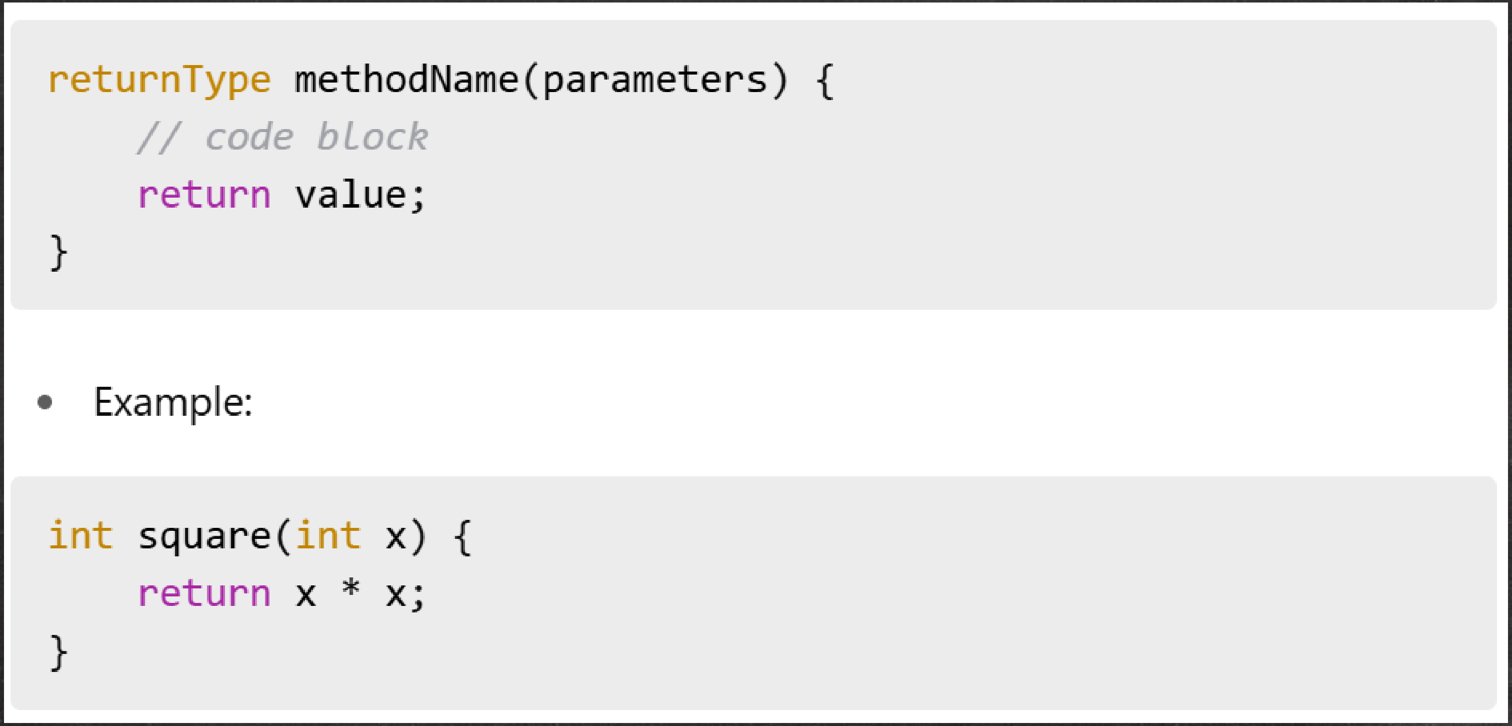
Today we explore how to break down problems into smaller, reusable units using functions — a key idea in writing maintainable code.

ABOUT

Why Use Functions?

- Divide and conquer: split big problems into small parts
- Reusability: write once, use many times
- Improves readability and testing
- Promotes modular programming

Basic Method Structure in Java



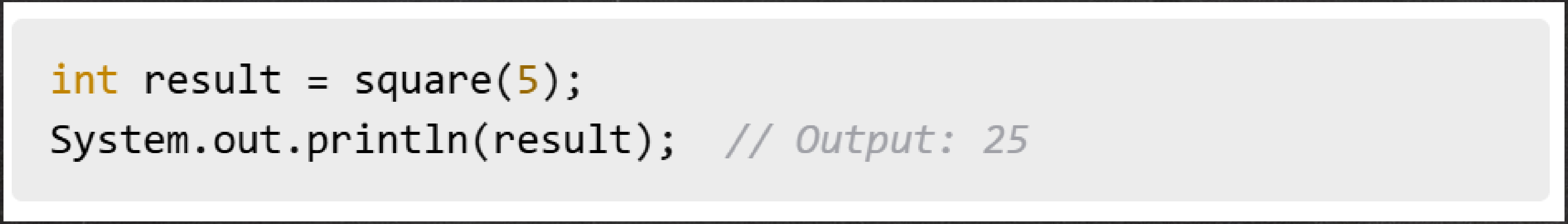
```java
returnType methodName(parameters) {
    // code block
    return value;
}
```

- Example:

```java
int square(int x) {
    return x * x;
}
```

Calling Methods

- Call method using its name and provide arguments
- Result can be stored or printed



```java
int result = square(5);
System.out.println(result);  // Output: 25
```

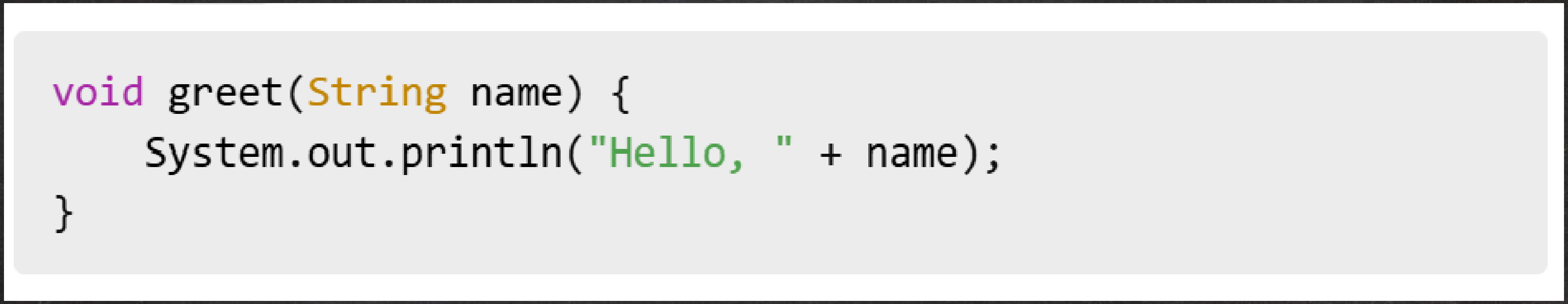
Method Overloading

- Overloading allows flexibility. You can have multiple versions of the same function name tailored for different data types or parameter counts.
- Multiple methods can have the same name if they differ in:
  - Number of parameters
  - Types of parameters

```
int add(int a, int b) {}
double add(double a, double b) {}
```

void Methods (No Return Value)

- Use void when method performs an action but returns nothing



```java
void greet(String name) {
    System.out.println("Hello, " + name);
}
```

Modular Thinking in Real Systems

- In large systems like AI apps or e-commerce sites, modularity is key. Each function should do one job well.
- Think of your program as building blocks
- Separate logic into meaningful units:
  - Input functions
  - Processing functions
  - Output functions
- Modular code is easier to read, test, and maintain

Example – Menu-Driven Application

- Options:
  1. Add numbers
  2. Subtract numbers
  3. Exit
- Each option handled by its own method

- We’ll build a menu-based app where each choice triggers a separate method. It’s a great way to apply logic, conditions, and modularity together.

Lab Activity

- Write a calculator program with:
  - add(), subtract(), multiply(), divide() methods
  - Main method presents a looped menu
- Use Scanner for input

Exercise

- Task: Create a Java program with the following features:
  - isPrime(int n) method
  - reverseNumber(int n) method
  - Menu to select operation
- Due: Next class

*Technically explain how to do it and show how to run and result