PT821: Object-Oriented Programming

Java Basics - Foundation for OOP

Masoud Hamad

State University of Zanzibar
BITA Second Year - Semester 1

Prerequisite Lecture

Outline

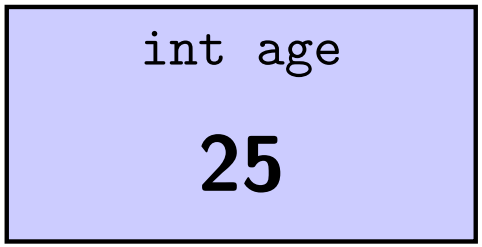
What is a Variable?

Definition

A variable is a container that holds data that can be changed during program execution.

Variable Components:

- **Name** - identifier
- **Type** - kind of data
- **Value** - stored data



Variable

Primitive Data Types

Type	Size	Range	Example
`byte`	8-bit	-128 to 127	`byte b = 100;`
`short`	16-bit	-32,768 to 32,767	`short s = 1000;`
`int`	32-bit	$\pm$2.1 billion	`int x = 42;`
`long`	64-bit	Very large	`long l = 999L;`
`float`	32-bit	7 decimal digits	`float f = 3.14f;`
`double`	64-bit	15 decimal digits	`double d = 3.14159;`
`char`	16-bit	Single character	`char c = 'A';`
`boolean`	1-bit	true/false	`boolean b = true;`

Variable Declaration and Initialization

```java
// Declaration only
int age;
String name;

// Declaration with initialization
int score = 85;
double price = 19.99;
char grade = 'A';
boolean isStudent = true;

// Multiple declarations
int x, y, z;
int a = 1, b = 2, c = 3;

// String (Reference Type)
String greeting = "Hello, Zanzibar!";
```

Type Casting

Implicit (Widening)
Automatic conversion

```
int myInt = 100;
double myDouble = myInt;
// Result: 100.0
```

Explicit (Narrowing)
Manual conversion

```
double pi = 3.14159;
int piInt = (int) pi;
// Result: 3
```

String Conversions:

```
String numStr = "42";
int num = Integer.parseInt(numStr);   // String to int
String str = String.valueOf(100);     // int to String
```

Arithmetic Operators

Operator	Name	Example	Result
+	Addition	`10 + 3`	13
-	Subtraction	`10 - 3`	7
*	Multiplication	`10 * 3`	30
/	Division	`10 / 3`	3
%	Modulus	`10 % 3`	1

Note

Integer division truncates: `10 / 3 = 3`

For decimal result: `10.0 / 3 = 3.333...`

Assignment and Compound Operators

Assignment Operators:

```
int x = 10;
x += 5;   // x = x + 5 = 15
x -= 3;   // x = x - 3 = 12
x *= 2;   // x = x * 2 = 24
x /= 4;   // x = x / 4 = 6
x %= 4;   // x = x % 4 = 2
```

Increment/Decrement:

```
int a = 5;

a++;   // a = 6 (post)
++a;   // a = 7 (pre)

a--;   // a = 6 (post)
--a;   // a = 5 (pre)
```

Comparison and Logical Operators

Comparison:

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

Logical:

&&	AND	Both true
||	OR	One true
!	NOT	Reverses

Example:

```
(age >= 18) && hasID
(x > 0) || (y > 0)
!isRaining
```

Output Methods

```
// print() - no newline
System.out.print("Hello ");
System.out.print("World");
// Output: Hello World

// println() - with newline
System.out.println("Hello");
System.out.println("World");
// Output:
// Hello
// World

// printf() - formatted output
String name = "Ali";
int age = 20;
double gpa = 3.75;
System.out.printf("Name: %s, Age: %d, GPA: %.2f%n",
                  name, age, gpa);
// Output: Name: Ali, Age: 20, GPA: 3.75
```

Format Specifiers

- `%s` – String
- `%d` – Integer
- `%f` – Float/Double
- `%.2f` – 2 decimals
- `%c` – Character

- `%b` – Boolean
- `%n` – Newline
- `%10s` – Right-align
- `%-10s` – Left-align

Input with Scanner Class

```
import java.util.Scanner;

public class InputExample {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter your name: ");
        String name = scanner.nextLine();

        System.out.print("Enter your age: ");
        int age = scanner.nextInt();

        System.out.print("Enter your GPA: ");
        double gpa = scanner.nextDouble();

        System.out.println("Hello, " + name);
        System.out.println("Age: " + age);
        System.out.printf("GPA: %.2f%n", gpa);
```

Scanner Methods

Method	Description
`next()`	Reads single word (until space)
`nextLine()`	Reads entire line
`nextInt()`	Reads an integer
`nextDouble()`	Reads a double
`nextFloat()`	Reads a float
`nextBoolean()`	Reads true/false
`nextLong()`	Reads a long integer

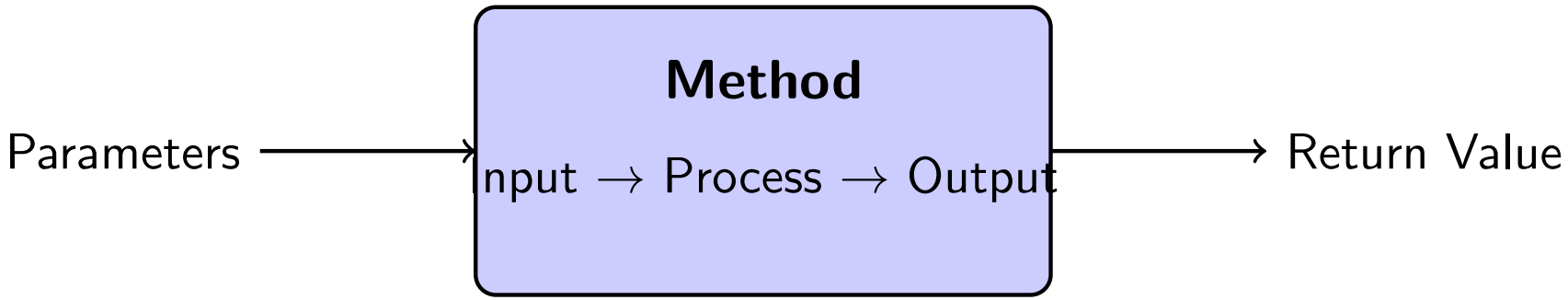
Important

After using `nextInt()`, `nextDouble()`, etc., call `nextLine()` to clear the buffer before reading a string!

What is a Method?

Definition

A method is a block of code that performs a specific task and can be called (invoked) when needed.



Benefits:

- Code reusability
- Better organization
- Easier debugging
- Modularity

Method Syntax

```
accessModifier returnType methodName(parameters) {
    // method body
    return value;  // if not void
}
```

Examples:

```
// Void method - no return value
public static void sayHello() {
    System.out.println("Hello!");
}

// Method with return value
public static int add(int a, int b) {
    return a + b;
}

// Method with parameters
public static void greet(String name) {
```

Calling Methods

```
public class MethodDemo {
    public static void main(String[] args) {
        // Calling void method
        sayHello();

        // Calling method with return value
        int result = add(10, 20);
        System.out.println("Sum: " + result);

        // Calling method with parameter
        greet("Fatma");

        // Using return value directly
        System.out.println("5 + 3 = " + add(5, 3));
    }

    public static void sayHello() {
        System.out.println("Hello!");
    }
```

Method Overloading

Definition

Method overloading allows multiple methods with the same name but different parameters.

```
// Same name, different parameters
public static int add(int a, int b) {
    return a + b;
}

public static int add(int a, int b, int c) {
    return a + b + c;
}

public static double add(double a, double b) {
    return a + b;
}

// Usage
```

if Statement

Simple if:

```
if (condition) {
    // code to execute
}
```

if-else:

```
if (condition) {
    // if true
} else {
    // if false
}
```

Example:

```
int age = 20;

if (age >= 18) {
    System.out.println(
        "You are an adult")
            ;
} else {
    System.out.println(
        "You are a minor");
}
```

if-else if-else Ladder

```
int score = 75;

if (score >= 90) {
    System.out.println("Grade: A");
} else if (score >= 80) {
    System.out.println("Grade: B");
} else if (score >= 70) {
    System.out.println("Grade: C");
} else if (score >= 60) {
    System.out.println("Grade: D");
} else if (score >= 50) {
    System.out.println("Grade: E");
} else {
    System.out.println("Grade: F");
}
```

switch-case Statement

```
int day = 3;

switch (day) {
    case 1:
        System.out.println("Monday");
        break;
    case 2:
        System.out.println("Tuesday");
        break;
    case 3:
        System.out.println("Wednesday");
        break;
    case 4:
        System.out.println("Thursday");
        break;
    case 5:
        System.out.println("Friday");
        break;
    default:
```

switch with Multiple Cases

```
String month = "March";

switch (month) {
    case "January":
    case "February":
    case "March":
        System.out.println("Q1 - First Quarter");
        break;
    case "April":
    case "May":
    case "June":
        System.out.println("Q2 - Second Quarter");
        break;
    // ... and so on
    default:
        System.out.println("Invalid month");
}
```

Remember

Ternary Operator

Syntax

```
condition ? valueIfTrue : valueIfFalse
```

```java
int x = 10, y = 20;

// Using if-else
int max;
if (x > y) {
    max = x;
} else {
    max = y;
}

// Using ternary operator (same result)
int max = (x > y) ? x : y;

// More examples
String result = (score >= 50) ? "PASS" : "FAIL";
String type = (num % 2 == 0) ? "even" : "odd";
```

for Loop

```java
// Syntax
for (initialization; condition; update) {
    // code to repeat
}

// Example: Count 1 to 5
for (int i = 1; i <= 5; i++) {
    System.out.println(i);
}

// Count backwards
for (int i = 5; i >= 1; i--) {
    System.out.println(i);
}

// Step by 2 (even numbers)
for (int i = 2; i <= 10; i += 2) {
    System.out.print(i + " ");  // 2 4 6 8 10
}
```

while Loop

```
// Syntax
while (condition) {
    // code to repeat
}

// Example: Count 1 to 5
int count = 1;
while (count <= 5) {
    System.out.println(count);
    count++;
}

// Example: Find how many times divisible by 2
int number = 64;
int divisions = 0;
while (number > 1) {
    number = number / 2;
    divisions++;
}
```

do-while Loop

```
// Syntax - executes AT LEAST ONCE
do {
    // code to repeat
} while (condition);

// Example
int num = 1;
do {
    System.out.println(num);
    num++;
} while (num <= 5);
```

Difference from while

`do-while` always executes the body at least once, even if condition is false initially.

break and continue

break - exit loop:

```
for (int i = 1; i <= 10; i
    ++) {
     if (i == 5) {
          break;  // exit
              loop
     }
     System.out.print(i + "
         ");
}
// Output: 1 2 3 4
```

continue - skip iteration:

```
for (int i = 1; i <= 5; i
    ++) {
     if (i == 3) {
          continue;  // skip
     }
     System.out.print(i + "
         ");
}
// Output: 1 2 4 5
```

Nested Loops

```java
// Multiplication table
for (int i = 1; i <= 5; i++) {
    for (int j = 1; j <= 5; j++) {
        System.out.print(i * j + "\t");
    }
    System.out.println();
}

// Triangle pattern
for (int i = 1; i <= 5; i++) {
    for (int j = 1; j <= i; j++) {
        System.out.print("* ");
    }
    System.out.println();
}
// Output:
// *
// * *
// * * *
```

Array Declaration and Initialization

```java
// Method 1: Declare, then assign
int[] numbers = new int[5];
numbers[0] = 10;
numbers[1] = 20;

// Method 2: Declare with values
int[] scores = {85, 92, 78, 95, 88};

// Method 3: new keyword with values
String[] names = new String[]{"Ali", "Fatma", "Hassan"
    };

// Accessing elements (0-indexed)
System.out.println(scores[0]);  // 85 (first)
System.out.println(scores[4]);  // 88 (last)

// Array length
System.out.println(scores.length);  // 5
```

Iterating Through Arrays

```
int[] numbers = {10, 20, 30, 40, 50};

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

// Using for-each loop (enhanced for)
for (int num : numbers) {
    System.out.println(num);
}

// Find sum
int sum = 0;
for (int num : numbers) {
    sum += num;
}
System.out.println("Sum: " + sum);
```

Common Array Operations

```java
import java.util.Arrays;

int[] arr = {64, 34, 25, 12, 22};

// Print array
System.out.println(Arrays.toString(arr));

// Sort array
Arrays.sort(arr);
System.out.println(Arrays.toString(arr));
// [12, 22, 25, 34, 64]

// Find max
int max = arr[0];
for (int num : arr) {
    if (num > max) max = num;
}

// Find min
```

2D Arrays (Matrices)

```
// Declaration
int[][] matrix = {
    {1, 2, 3},
    {4, 5, 6},
    {7, 8, 9}
};

// Accessing elements
System.out.println(matrix[0][0]);  // 1
System.out.println(matrix[1][2]);  // 6

// Iterating through 2D array
for (int i = 0; i < matrix.length; i++) {
    for (int j = 0; j < matrix[i].length; j++) {
        System.out.print(matrix[i][j] + " ");
    }
    System.out.println();
}
```

ArrayList vs Array

Feature	**Array**	**ArrayList**
Size	Fixed	Dynamic
Type	Primitives + Objects	Objects only
Syntax	`int[] arr`	`ArrayList<Integer>`
Access	`arr[i]`	`list.get(i)`
Length	`arr.length`	`list.size()`
Add element	Not possible	`list.add(item)`
Remove	Not directly	`list.remove(i)`

Note

ArrayList uses wrapper classes: `Integer`, `Double`, `Boolean`, etc.

ArrayList Basic Operations

```java
import java.util.ArrayList;

ArrayList<String> names = new ArrayList<>();

// Add elements
names.add("Ali");
names.add("Fatma");
names.add("Hassan");

// Access element
String first = names.get(0);  // "Ali"

// Modify element
names.set(0, "Ahmed");

// Remove element
names.remove(1);              // by index
names.remove("Hassan");       // by value
```

ArrayList Iteration and Sorting

```
import java.util.ArrayList;
import java.util.Collections;

ArrayList<Integer> numbers = new ArrayList<>();
numbers.add(64);
numbers.add(34);
numbers.add(25);

// Iterate with for loop
for (int i = 0; i < numbers.size(); i++) {
    System.out.println(numbers.get(i));
}

// Iterate with for-each
for (int num : numbers) {
    System.out.println(num);
}

// Sort
```

Java Basics Summary

Data & Variables:

- 8 primitive types
- String (reference)
- Type casting

Operators:

- Arithmetic: +, -, *, /, %
- Comparison: ==, !=, <, >
- Logical: &&, ||, !

Input/Output:

- Scanner for input
- print, println, printf

Control Structures:

- if, else if, else
- switch-case
- for, while, do-while
- break, continue

Data Structures:

- Arrays (fixed size)
- ArrayList (dynamic)
- 2D Arrays

Methods:

- Parameters & return
- Method overloading

Practice Exercises

Try these on your own:

1. Create a calculator using switch-case
2. Build a grade calculator with if-else
3. Print multiplication table using loops
4. Create a number guessing game with while loop
5. Store and analyze student scores with arrays
6. Build a student list manager with ArrayList

Available on Course Website

All examples and exercises are available at:
`https://massoudhamad.github.io/pt821/`

Questions?

Any Questions?

Contact Information:
Instructor: Masoud Hamad
Email: massoud.hamad@suza.ac.tz
Office Hours: Thursday 08:00AM-12:00PM

"Master the basics, and you can build anything!"

Thank You!

Now you're ready for OOP!

Practice the exercises before next class!