Java Learning Journey

Chapter 2: Elementary Programming

✍ **Prepared by:** Ahmed Elsifi

2.1 Introduction to Elementary Programming

- Learn to write programs that perform computations and process input
- Understand variables, data types, operators, and expressions
- Apply problem-solving techniques to programming tasks

2.2 Writing a Simple Program

```java
public class ComputeArea {
    public static void main(String[] args) {
        double radius = 20;
        double area = radius * radius * 3.14159;
        System.out.println("The area is " + area);
    }
}
```

- Programs follow algorithms (step-by-step problem-solving procedures)
- Variables store data in memory locations
- Use descriptive names for readability

2.3 Reading Input with Scanner Class

```java
import java.util.Scanner;  // Import statement

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

        System.out.print("Enter a value: ");  // Prompt user
        double value = input.nextDouble();    // Read input

        // Other input methods:
        // input.nextByte(), input.nextShort(), input.nextInt()
        // input.nextLong(), input.nextFloat(), input.nextBoolean()
    }
}
```

2.4 Output Methods

```
System.out.println("Hello World");    // Prints with newline
System.out.print("Hello ");           // Prints without newline
System.out.print("World");

// Formatted output
String name = "John";
int age = 25;
System.out.format("Name: %s, Age: %d%n", name, age);
// %s - string, %d - integer, %f - float, %n - newline
```

2.5 Variables and Constants

```
// Variable declaration
int count;
double radius;

// Variable initialization
count = 1;
radius = 5.5;

// Declaration and initialization in one step
int number = 10;

// Constants (use final keyword)
final double PI = 3.14159;
final int MAX_SIZE = 100;
```

- `final` creates constants (similar to `const` in other languages)
- Constants cannot be changed after initialization

2.6 Naming Conventions

- **camelCase**: variables and methods (`myVariable`, `calculateArea`)
- **PascalCase**: class names (`ComputeArea`, `Scanner`)
- **UPPER_CASE**: constants (`MAX_VALUE`, `PI`)
- Identifiers can contain letters, digits, _, $
- Cannot start with digit or be a reserved keyword

2.7 Numeric 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 = 500;`
`int`	32-bit	~ -2.1 billion to 2.1 billion	`int i = 100000;`
`long`	64-bit	Very large numbers	`long l = 10000000000L;`
`float`	32-bit	6-9 significant digits	`float f = 3.14f;`
`double`	64-bit	15-17 significant digits	`double d = 3.14159;`

2.8 Numeric Operations

```
// Arithmetic operators
int sum = 5 + 3;       // Addition
int difference = 5 - 3; // Subtraction
int product = 5 * 3;    // Multiplication
int quotient = 5 / 2;   // Division → 2 (integer division)
double result = 5.0 / 2; // → 2.5 (floating-point division)
int remainder = 5 % 2;   // Modulus → 1
// Important: Integer vs floating-point division
System.out.println(5 / 2);     // Output: 2
System.out.println(5.0 / 2);   // Output: 2.5
System.out.println(5f / 2);    // Output: 2.5
```

2.9 Math Class Methods

```
// Common Math methods
Math.pow(2, 3);        // 8.0 - Exponentiation
Math.sqrt(25);         // 5.0 - Square root
Math.abs(-5);          // 5 - Absolute value
Math.max(10, 20);      // 20 - Maximum value
Math.min(10, 20);      // 10 - Minimum value
Math.round(3.6);       // 4 - Round to nearest integer
Math.ceil(3.2);        // 4.0 - Round up
Math.floor(3.8);       // 3.0 - Round down
Math.random();         // Random number between 0.0 and 1.0
// Trigonometric functions
Math.sin(Math.PI/2);   // 1.0 - Sine
Math.cos(0);           // 1.0 - Cosine
Math.tan(Math.PI/4);   // ~1.0 - Tangent
// Constants
Math.PI;               // 3.141592653589793
Math.E;                // 2.718281828459045
```

2.10 Numeric Literals

```
// Integer literals
int decimal = 100;         // Base 10
int octal = 0144;          // Base 8 (prefix with 0)
int hexadecimal = 0x64;    // Base 16 (prefix with 0x)
int binary = 0b1100100;    // Base 2 (prefix with 0b)
long bigNumber = 10000000000L; // Long literal (suffix with L)

// Floating-point literals
double normal = 3.14159;
double scientific = 3.14159e0; // 3.14159
double largeNumber = 1.234e5;  // 123400.0

// Using underscores for readability
int million = 1_000_000;
double pi = 3.141_592_653;
```

2.11 Type Casting

```
// Widening (automatic)
int myInt = 10;
double myDouble = myInt;   // Automatic casting: int to double

// Narrowing (manual)
double myDouble = 9.78;
int myInt = (int) myDouble; // Manual casting: double to int → 9

// Casting in expressions
double result = (double) 5 / 2; // → 2.5
```

2.12 Augmented Assignment Operators

```
int x = 10;
x += 5;  // Equivalent to x = x + 5
x -= 3;  // Equivalent to x = x - 3
x *= 2;  // Equivalent to x = x * 2
x /= 4;  // Equivalent to x = x / 4
x %= 3;  // Equivalent to x = x % 3
```

2.13 Increment and Decrement Operators

```
int i = 5;
int j = i++;  // Post-increment: j = 5, i = 6
int k = ++i;  // Pre-increment: k = 7, i = 7

int m = 5;
int n = m--;  // Post-decrement: n = 5, m = 4
int o = --m;  // Pre-decrement: o = 3, m = 3
```

2.14 Getting Current Time

```
long totalMilliseconds = System.currentTimeMillis();
// Returns milliseconds since Unix epoch (Jan 1, 1970)

// Convert to seconds, minutes, hours
long totalSeconds = totalMilliseconds / 1000;
long currentSecond = totalSeconds % 60;
long totalMinutes = totalSeconds / 60;
long currentMinute = totalMinutes % 60;
long totalHours = totalMinutes / 60;
long currentHour = totalHours % 24;
```

2.15 JShell (Java REPL)

- Interactive tool for testing Java code without full class structure
- Launch with `jshell` command
- Useful for quick experiments and learning
- Supports expressions, statements, and variable declarations

2.16 Software Development Life Cycle

1. **Requirements Specification**: Define what the program should do
2. **System Analysis**: Determine input, processing, output (IPO)
3. **System Design**: Create algorithm and plan structure
4. **Implementation**: Write code following the design
5. **Testing**: Verify program works correctly
6. **Deployment**: Make program available to users
7. **Maintenance**: Update and improve program over time

2.17 Common Errors and Pitfalls

1. **Undeclared or uninitialized variables**
2. **Integer overflow** (values exceeding type limits)
3. **Round-off errors** in floating-point calculations
4. **Unintended integer division** (5/2 = 2 instead of 2.5)
5. **Redundant Scanner objects** (create only one)

2.18 Case Studies

- Computing loan payments with interest
- Converting between measurement systems
- Calculating geometric properties
- Processing monetary amounts
- Displaying and formatting current time

Important Notes :

- Use appropriate data types for your values
- Remember integer division behavior (5/2 = 2)
- Always initialize variables before use
- Use descriptive names following naming conventions
- Test programs with various input values
- Understand operator precedence in expressions
- Use casting when converting between types