📓 Java Learning Journey

Chapter 2: Elementary Programming

✍️ **Prepared by:** Ahmed Elsifi

2.1 Introduction

- This chapter focuses on writing programs to perform computations using variables, operators, and input/output.

2.2 Writing a Simple Program

- Programs are designed using algorithms.
- Example: Compute area of a circle:

  ```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);
      }
  }
  ```

- Variables (e.g., `radius`, `area`) store data in memory.
- Use descriptive names for variables.

2.3 Reading Input from the Console

- Use the `Scanner` class to read input:

  ```java
  import java.util.Scanner;
  Scanner input = new Scanner(System.in);
  double radius = input.nextDouble();
  ```

- Prompt the user before reading input.
- Multiple inputs can be read in one line or separately.

2.4 Identifiers

- Rules for identifiers:
  - Can contain letters, digits, `_`, `$`
  - Cannot start with a digit
  - Cannot be a keyword
  - Case-sensitive
- Use meaningful names (e.g., `numberOfStudents` instead of `numStuds`).

2.5 Variables

- Declare variables: `dataType variableName;`
- Initialize: `int count = 1;`
- Variables must be declared and initialized before use.
- Scope: Part of the program where a variable can be accessed.

2.6 Assignment Statements

- Syntax: `variable = expression;`
- Example: `x = x + 1;`
- Chained assignment: `i = j = k = 1;`
- The right-hand side must be compatible with the left-hand side type.

2.7 Named Constants

- Use `final` keyword: `final double PI = 3.14159;`
- Benefits:
  - Avoid repeating values
  - Easy to change
  - Improve readability

2.8 Naming Conventions

- Variables/methods: `camelCase`
- Classes: `PascalCase`
- Constants: `UPPER_CASE_WITH_UNDERSCORES`

2.9 Numeric Data Types and Operations

Data Types:

Type	Size	Range
`byte`	8-bit	-128 to 127
`short`	16-bit	-32768 to 32767
`int`	32-bit	~ -2.1e9 to 2.1e9
`long`	64-bit	~ -9.2e18 to 9.2e18
`float`	32-bit	6-9 significant digits
`double`	64-bit	15-17 significant digits

Operators:

- `+`, `-`, `*`, `/`, `%`
- Integer division truncates: `5 / 2 = 2`
- Use `Math.pow(a, b)` for exponentiation.

Reading Numbers:

- `nextByte()`, `nextShort()`, `nextInt()`, `nextLong()`, `nextFloat()`, `nextDouble()`

2.10 Numeric Literals

- Integer literals: `int` by default. Use `L` for `long`: `2147483648L`
- Floating-point: `double` by default. Use `f` for `float`: `100.2f`
- Scientific notation: `1.23456e+2`
- Underscores for readability: `232_455_199`

2.11 JShell

- REPL tool for quick Java code testing.
- Launch: `jshell`
- Commands: `/vars`, `/edit`, `/exit`

2.12 Evaluating Expressions

- Operator precedence:
    1. Parentheses
    2. `*`, `/`, `%`
    3. `+`, `-`
- Example:
  `(3 + 4 * 4 + 5 * (4 + 3) - 1) = 53`

2.13 Case Study: Displaying Current Time

- Use `System.currentTimeMillis()` to get milliseconds since Unix epoch (Jan 1, 1970).
- Convert to seconds, minutes, hours using `/` and `%`.

2.14 Augmented Assignment Operators

- `+=`, `-=`, `*=`, `/=`, `%=`
- Example: `x += 2;` is equivalent to `x = x + 2;`

2.15 Increment/Decrement Operators

- `++i` (preincrement), `i++` (postincrement)
- `--i` (predecrement), `i--` (postdecrement)
- Example:

```java
int i = 10;
int newNum = 10 * i++; // newNum = 100, i = 11
```

2.16 Numeric Type Conversions

- Widening (automatic): `int` to `double`
- Narrowing (requires casting): `double` to `int`
- Syntax: `(targetType) value`
- Example:

```java
double d = 4.5;
int i = (int)d; // i = 4
```

2.17 Software Development Process

1. Requirements Specification
2. System Analysis (IPO: Input, Process, Output)
3. System Design
4. Implementation (Coding)
5. Testing
6. Deployment
7. Maintenance

2.18 Case Study: Counting Monetary Units

- Convert dollars to cents to avoid floating-point errors.
- Use integer division and remainder to break into coins.

2.19 Common Errors and Pitfalls

1. **Undeclared/uninitialized variables**
2. **Integer overflow**:
   `int value = 2147483647 + 1; // becomes -2147483648`
3. **Round-off errors**:
   `1.0 - 0.1 - 0.1 - 0.1 - 0.1 - 0.1 != 0.5`
4. **Unintended integer division**:
   `(1 + 2) / 2 = 1` but `(1 + 2) / 2.0 = 1.5`
5. **Redundant input objects**: Use one `Scanner` object.

Key Terms

- Algorithm, Variable, Constant, Identifier
- Primitive data types, Operator, Operand
- Casting, Overflow, Round-off error
- IPO, REPL, JShell

Example Programs to Remember:

1. Compute area of circle
2. Read input with `Scanner`
3. Convert Fahrenheit to Celsius
4. Display current time
5. Compute loan payments
6. Count monetary units

This summary covers all key concepts, syntax, and common pitfalls from Chapter 2. Review the code examples and practice writing programs to reinforce your understanding.