COMP2000: Software Engineering 2 - Introduction to Java

Outline

1. Introduction to Java

Key Points:

- Java is a "very popular" programming language
- Runs on a "virtual machine" (JVM)
- More complex than some languages (e.g., Python)
- Simpler than others (e.g., C++)

Java Compilation Process:

1. Source Code (.java) → Compiler (javac) → Byte Code (.class)
2. Byte Code → Java Interpreter → Machine Code

Java Development Kit (JDK):

- Provides environment to develop and execute Java programs
- Includes:
  - Development Tools
  - Java Runtime Environment (JRE)
  - Java Virtual Machine (JVM)

2. Java Program Structure

Basic Structure:

```
class ClassName {
    public static void main(String[] arguments) {
        // Program execution begins here
        // STATEMENTS
    }
}
```

Key Components:

- `public`: Method can be accessed from outside the class
- `static`: Method belongs to the class itself
- `void`: Method does not return any value
- `main`: Starting point of the program

Output:

- Use `System.out.println(some String)` to output to the console

3. Object-Oriented Programming Concepts

Class:

- Template/blueprint describing behavior/state of objects

Object:

- Instance of a class
- Has states (attributes) and behaviors (methods)

Methods:

- Behaviors of a class
- Where logic is written and actions are executed

Instance Variables:

- Unique set of variables for each object
- Represent the object's state

4. Java Modifiers and Data Types

Access Modifiers:

- default: Visible to the package
- public: Visible to the world
- private: Visible to the class only
- protected: Visible to the package and all subclasses

Non-Access Modifiers:

- static: For creating class methods and variables
- final: For finalizing implementations
- abstract: For creating abstract classes and methods
- synchronized and volatile: Used in multithreading

Data Types:

1. Primitive Types:
    - byte, short, int, long
    - float, double
    - char
    - boolean
2. Non-primitive Types:
    - String

    - Arrays
    - Classes
    - Interfaces

Type Conversion:

- Implicit conversion: Automatic (e.g., int to double)
- Explicit conversion: Using casting (e.g., (int)18.7)

5. Loops and Conditions in Java

Loops:

1. while loop:

   ```
   while (condition) {
       // code block
   }
   ```

2. for loop:

   ```
   for (initialization; condition; increment/decrement) {
       // code block
   }
   ```

3. do...while loop:

   ```
   do {
       // code block
   } while (condition);
   ```

4. Enhanced for loop (for-each):

   ```
   for (declaration : expression) {
       // code block
   ```

```
}
```

Control Statements:

- break: Terminates the loop or switch statement
- continue: Skips the remainder of the loop body and retests the condition

Decision Making:

1. if statement:

```
if (condition) {
    // code block
}
```

2. if...else statement:

```
if (condition) {
    // code block
} else {
    // code block
}
```

3. nested-if statement: if or else if statements inside other if or else if statements

4. switch statement: Tests a variable for equality against a list of values

6. Exception Handling

Try-Catch Block:

```
try {
    // Protected code
} catch (ExceptionName e) {
    // Catch block
}
```

Multiple Catch Blocks:

```
try {
    // Protected code
} catch (ExceptionType1 e1) {
    // Catch block
} catch (ExceptionType2 e2) {
    // Catch block
}
```

7. File I/O and Wrapper Classes

File I/O:

- Involves reading from and writing to files
- Requires exception handling

Wrapper Classes:

- Provide a way to use primitive data types as objects
- Examples: Integer, Double, Boolean, Character
- Useful for:
  - Collections (which only work with objects)
  - Utility methods (e.g., parsing strings)

Wrapper Class Methods:

Where xxx represents the primitive type name (always lowercase):

1. parseXxx() methods: Convert strings to primitive types

   - Examples:
     - Integer.parseInt()
     - Double.parseDouble()
     - Boolean.parseBoolean()

    - Note: String doesn't have a parse method as it's already a reference type

2. valueOf() methods: Return wrapper objects

    - Examples:
        - Integer.valueOf()
        - Double.valueOf()
        - Boolean.valueOf()
        - String.valueOf() (works differently, converts other types to String)

3. xxxValue() methods: Return primitive values from wrapper objects

    - Examples:
        - intValue()
        - floatValue()
        - doubleValue()
        - booleanValue()
    - Note: String is not a primitive type, so it doesn't have an xxxValue() method

Important notes:

- The xxx in xxxValue() always refers to the lowercase primitive type name (int, float, double, boolean, etc.)
- String is a special case as it's not a primitive type:
    - It doesn't have parseString() or stringValue() methods
    - It has valueOf() method, but it works differently from primitive wrappers

Additional common methods:

- toString(): Converts the wrapper object to a String
- compare(xxx a, xxx b): Compares two values of the type

Conclusion

This lecture provided a comprehensive introduction to Java programming, covering fundamental concepts such as program structure, object-oriented programming, data types, control flow, exception handling, and more advanced topics like file I/O and wrapper classes. Understanding these concepts is crucial for developing robust and efficient Java applications.