Java Notes Book

A concise guide to understanding and using Java for robust, object-oriented programming.

Table of Contents

1. Introduction to Java
2. Core Concepts
3. Object-Oriented Programming
4. Collections Framework
5. Exception Handling
6. Best Practices
7. Example: Simple Library System

Introduction to Java

Java is a high-level, object-oriented, platform-independent programming language developed by Sun Microsystems (now Oracle). Known for its "write once, run anywhere" philosophy, it runs on the Java Virtual Machine (JVM).

- **Key Features**:
  - Platform independence via JVM
  - Strong type system
  - Automatic memory management (garbage collection)
  - Rich standard library
- **Use Case**: Enterprise applications, Android development, web servers, and more.

Core Concepts

Variables and Data Types

Java is statically typed; variables must be declared with a type.

```
int age = 25;            // Integer
double price = 19.99;    // Floating-point
String name = "Alice";   // String
boolean isActive = true; // Boolean
```

Control Flow

Use `if`, `for`, `while`, and `switch` for decision-making and looping.

```
// If statement
if (age >= 18) {
    System.out.println("Adult");
} else {
    System.out.println("Minor");
}

// For loop
for (int i = 0; i < 5; i++) {
    System.out.println(i); // Outputs 0 to 4
}
```

Methods

Define reusable code blocks with methods.

```
public String greet(String name) {
    return "Hello, " + name + "!";
}
```

Object-Oriented Programming

Java is built around OOP principles: encapsulation, inheritance, polymorphism, and abstraction.

Class and Object

```
public class Dog {
    private String name;

    public Dog(String name) {
        this.name = name;
    }

    public String bark() {
        return name + " says Woof!";
    }
}

// Usage
Dog dog = new Dog("Buddy");
System.out.println(dog.bark()); // Outputs: Buddy says Woof!
```

Inheritance

Extend classes to reuse code.

```
public class Puppy extends Dog {
    public Puppy(String name) {
        super(name);
    }
}
```

Interfaces

Define contracts for classes to implement.

```
public interface Animal {
    String makeSound();
}
```

Collections Framework

Java’s Collections Framework provides data structures like lists, sets, and maps.

ArrayList

Dynamic, resizable array.

```
import java.util.ArrayList;

ArrayList<String> fruits = new ArrayList<>();
fruits.add("Apple");
fruits.add("Banana");
System.out.println(fruits.get(0)); // Outputs: Apple
```

HashMap

Key-value pairs.

```
import java.util.HashMap;

HashMap<String, Integer> ages = new HashMap<>();
ages.put("Alice", 25);
System.out.println(ages.get("Alice")); // Outputs: 25
```

Exception Handling

Handle errors using `try-catch` blocks.

```
try {
    int result = 10 / 0;
} catch (ArithmeticException e) {
    System.out.println("Cannot divide by zero!");
} finally {
    System.out.println("Cleanup code");
}
```

- **Checked vs Unchecked**: Checked exceptions (e.g., `IOException`) must be declared or handled; unchecked (e.g., `NullPointerException`) are runtime errors.

Best Practices

1. **Follow Naming Conventions**: Use CamelCase for classes (`MyClass`), camelCase for methods/variables (`myMethod`).

2. **Use Access Modifiers**: Prefer `private` fields with public getters/setters for encapsulation.
3. **Avoid Magic Numbers**: Use constants (`final int MAX_USERS = 100`).
4. **Write Unit Tests**: Use JUnit or TestNG for reliable code.
5. **Leverage Generics**: Ensure type safety in collections (`List<String>`).
6. **Keep Methods Short**: Aim for single-responsibility methods.

Example: Simple Library System

Below is a simple console-based library system using Java.

```
import java.util.ArrayList;
import java.util.Scanner;

public class Library {
    private ArrayList<String> books;

    public Library() {
        books = new ArrayList<>();
    }

    public void addBook(String title) {
        books.add(title);
        System.out.println("Added: " + title);
    }

    public void viewBooks() {
        if (books.isEmpty()) {
            System.out.println("No books available!");
            return;
        }
        for (int i = 0; i < books.size(); i++) {
            System.out.println((i + 1) + ". " + books.get(i));
        }
    }

    public void removeBook(int index) {
        if (index >= 1 && index <= books.size()) {
            String title = books.remove(index - 1);
            System.out.println("Removed: " + title);
        } else {
            System.out.println("Invalid book number!");
        }
    }

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

        while (true) {
            System.out.println("\n1. Add Book\n2. View Books\n3. Remove 
Book\n4. Exit");
            System.out.print("Choose an option (1-4): ");
            String choice = scanner.nextLine();
            if (choice.equals("1")) {
                System.out.print("Enter book title: ");
                String title = scanner.nextLine();
                library.addBook(title);
            } else if (choice.equals("2")) {
                library.viewBooks();
            } else if (choice.equals("3")) {
                library.viewBooks();
                System.out.print("Enter book number to remove: ");
                try {
                    int index = Integer.parseInt(scanner.nextLine());
                    library.removeBook(index);
                } catch (NumberFormatException e) {
                    System.out.println("Please enter a valid number!");
                }
            } else if (choice.equals("4")) {
                System.out.println("Goodbye!");
                break;
            } else {
                System.out.println("Invalid option!");
            }
        }
        scanner.close();
    }
}
```

Steps to Use

1. Save the code as `Library.java`.
2. Compile it: `javac Library.java`.
3. Run it: `java Library`.
4. Follow the menu to add, view, or remove books.

Additional Resources

- **Official Docs**: docs.oracle.com/javase
- **Tutorials**: JavaTpoint, Baeldung, Oracle's Java Tutorials
- **Community**: Stack Overflow, Java subreddit

Java Notes Book

A concise guide to understanding and using Java for robust, object-oriented programming.

Table of Contents

1. Introduction to Java
2. Core Concepts
3. Object-Oriented Programming
4. Collections Framework
5. Exception Handling

6. Best Practices
7. Example: Simple Library System

Introduction to Java

Java is a high-level, object-oriented, platform-independent programming language developed by Sun Microsystems (now Oracle). Known for its "write once, run anywhere" philosophy, it runs on the Java Virtual Machine (JVM).

- **Key Features**:
  - Platform independence via JVM
  - Strong type system
  - Automatic memory management (garbage collection)
  - Rich standard library
- **Use Case**: Enterprise applications, Android development, web servers, and more.

Core Concepts

Variables and Data Types

Java is statically typed; variables must be declared with a type.

```
int age = 25;            // Integer
double price = 19.99;    // Floating-point
String name = "Alice";   // String
boolean isActive = true; // Boolean
```

Control Flow

Use `if`, `for`, `while`, and `switch` for decision-making and looping.

```
// If statement
if (age >= 18) {
    System.out.println("Adult");
} else {
    System.out.println("Minor");
}

// For loop
for (int i = 0; i < 5; i++) {
    System.out.println(i); // Outputs 0 to 4
}
```

Methods

Define reusable code blocks with methods.

```
public String greet(String name) {
    return "Hello, " + name + "!";
}
```

Object-Oriented Programming

Java is built around OOP principles: encapsulation, inheritance, polymorphism, and abstraction.

Class and Object

```
public class Dog {
    private String name;

    public Dog(String name) {
        this.name = name;
    }

    public String bark() {
        return name + " says Woof!";
    }
}

// Usage
Dog dog = new Dog("Buddy");
System.out.println(dog.bark()); // Outputs: Buddy says Woof!
```

Inheritance

Extend classes to reuse code.

```
public class Puppy extends Dog {
    public Puppy(String name) {
        super(name);
    }
}
```

Interfaces

Define contracts for classes to implement.

```
public interface Animal {
    String makeSound();
}
```

Collections Framework

Java’s Collections Framework provides data structures like lists, sets, and maps.

ArrayList

Dynamic, resizable array.

```
import java.util.ArrayList;

ArrayList<String> fruits = new ArrayList<>();
fruits.add("Apple");
fruits.add("Banana");
System.out.println(fruits.get(0)); // Outputs: Apple
```

HashMap

Key-value pairs.

```
import java.util.HashMap;

HashMap<String, Integer> ages = new HashMap<>();
ages.put("Alice", 25);
System.out.println(ages.get("Alice")); // Outputs: 25
```

Exception Handling

Handle errors using `try-catch` blocks.

```
try {
    int result = 10 / 0;
} catch (ArithmeticException e) {
    System.out.println("Cannot divide by zero!");
} finally {
    System.out.println("Cleanup code");
}
```

- **Checked vs Unchecked**: Checked exceptions (e.g., `IOException`) must be declared or handled; unchecked (e.g., `NullPointerException`) are runtime errors.

Best Practices

1. **Follow Naming Conventions**: Use CamelCase for classes (`MyClass`), camelCase for methods/variables (`myMethod`).
2. **Use Access Modifiers**: Prefer `private` fields with public getters/setters for encapsulation.
3. **Avoid Magic Numbers**: Use constants (`final int MAX_USERS = 100`).
4. **Write Unit Tests**: Use JUnit or TestNG for reliable code.
5. **Leverage Generics**: Ensure type safety in collections (`List<String>`).
6. **Keep Methods Short**: Aim for single-responsibility methods.

Example: Simple Library System

Below is a simple console-based library system using Java.

```
import java.util.ArrayList;
import java.util.Scanner;

public class Library {
    private ArrayList<String> books;

    public Library() {
        books = new ArrayList<>();
    }

    public void addBook(String title) {
        books.add(title);
        System.out.println("Added: " + title);
    }

    public void viewBooks() {
        if (books.isEmpty()) {
            System.out.println("No books available!");
            return;
        }
        for (int i = 0; i < books.size(); i++) {
            System.out.println((i + 1) + ". " + books.get(i));
        }
    }

    public void removeBook(int index) {
        if (index >= 1 && index <= books.size()) {
            String title = books.remove(index - 1);
            System.out.println("Removed: " + title);
        } else {
            System.out.println("Invalid book number!");
        }
    }

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

        while (true) {
            System.out.println("\n1. Add Book\n2. View Books\n3. Remove
Book\n4. Exit");
            System.out.print("Choose an option (1-4): ");
            String choice = scanner.nextLine();

            if (choice.equals("1")) {
                System.out.print("Enter book title: ");
                String title = scanner.nextLine();
                library.addBook(title);
            } else if (choice.equals("2")) {
                library.viewBooks();
            } else if (choice.equals("3")) {
                library.viewBooks();
                System.out.print("Enter book number to remove: ");
                try {
                    int index = Integer.parseInt(scanner.nextLine());
                    library.removeBook(index);
                } catch (NumberFormatException e) {
                    System.out.println("Please enter a valid number!");
                }
            } else if (choice.equals("4")) {
                System.out.println("Goodbye!");
                break;
            } else {
                System.out.println("Invalid option!");
            }
        }
        scanner.close();
    }
}
```

Steps to Use

1. Save the code as `Library.java`.
2. Compile it: `javac Library.java`.
3. Run it: `java Library`.
4. Follow the menu to add, view, or remove books.

Additional Resources

- **Official Docs**: docs.oracle.com/javase
- **Tutorials**: JavaTpoint, Baeldung, Oracle's Java Tutorials
- **Community**: Stack Overflow, Java subreddit