Core Java 17 Refresher Workbook

4-Hour Guided Practice with Labs

This workbook is designed to help you quickly refresh and practice modern Core Java concepts with labs, exercises, and examples.

Hour 1 – Java Basics & OOP Foundations

1. Java Overview & Setup

Java is an object-oriented language. Modern Java (17+) emphasizes immutability, streams, and records.

2. Hello World

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, Java 17!");
    }
}
```

■ Lab 1: Compile and run HelloWorld.java

• Open VS Code → New File → Save as HelloWorld.java

• Compile: javac HelloWorld.java

• Run: java HelloWorld

Hour 2 – Inheritance, Interfaces & Enums

Java supports single inheritance and multiple interface implementation.

```
interface Animal {
    void makeSound();
}

class Dog implements Animal {
    public void makeSound() {
        System.out.println("Woof");
    }
}

public class Main {
    public static void main(String[] args) {
        Animal a = new Dog();
        a.makeSound();
    }
}
```

■ Lab 2: Add another class Cat that also implements Animal and prints 'Meow'.

Hour 3 – Collections, Generics & Streams

```
import java.util.*;
import java.util.stream.*;

public class StreamExample {
    public static void main(String[] args) {
        List<Integer> nums = Arrays.asList(1, 2, 3, 4, 5);
        int sum = nums.stream().filter(n -> n % 2 == 1).mapToInt(n -> n).sum();
        System.out.println("Sum of odd numbers: " + sum);
    }
}
```

■ Lab 3: Try replacing filter condition with n > 2 and see the result.

Hour 4 – Modern Java, Exceptions & File I/O

```
import java.io.IOException;
import java.nio.file.*;

public class FileDemo {
    public static void main(String[] args) throws IOException {
        String data = "Java File I/O Example";
        Path path = Paths.get("output.txt");
        Files.writeString(path, data);
        System.out.println("File written!");
    }
}
```

■ Lab 4: Modify the above to read and print from output.txt

Mini Project – Library Management

Build a small console app managing books with title, author, and availability.

```
import java.util.*;

class Book {
    String title, author;
    boolean available = true;

    Book(String title, String author) {
        this.title = title;
        this.author = author;
    }
}

public class Library {
    public static void main(String[] args) {
        List<Book> books = new ArrayList<>();
        books.add(new Book("Java 17", "Oracle"));
        books.add(new Book("Effective Java", "Joshua Bloch"));

        for (Book b : books) {
            System.out.println(b.title + " by " + b.author);
        }
    }
}
```

■ Challenge: Add methods to borrow and return books.