Reading 15: A Second Look at Java

# Exercise 1: Summarize

Java’s subtype polymorphism, interfaces, class inheritance, and generics collectively enable flexible, reusable, and type-safe code by allowing classes to inherit behavior, implement multiple contracts, and handle various data types efficiently.

# Exercise 2: Demonstrate & Explain

// Define the interface

interface Animal {

// Method signature

void makeSound();

}

// Define the class that implements the interface

class Dog implements Animal {

// Implementing the method from interface

@Override

public void makeSound() {

System.out.println("Bark");

}

}

// Define the subclass that extends the class

class Labrador extends Dog {

// Overriding method

@Override

public void makeSound() {

System.out.println("Woof");

}

}

// Method that accepts an interface type

public void animalSound(Animal animal) {

animal.makeSound();

}

// Method that accepts a subclass type

public void dogSound(Dog dog) {

dog.makeSound();

}

// Usage

public static void main(String[] args) {

Animal myDog = new Dog();

Animal myLabrador = new Labrador();

Dog mySpecificDog = new Labrador();

animalSound(myDog); // Outputs: Bark

animalSound(myLabrador); // Outputs: Woof

dogSound(mySpecificDog); // Outputs: Woof

}

# Exercise 3: Inquire

Why are interfaces important, why not have it catch a parameter and use that like polymorphism.