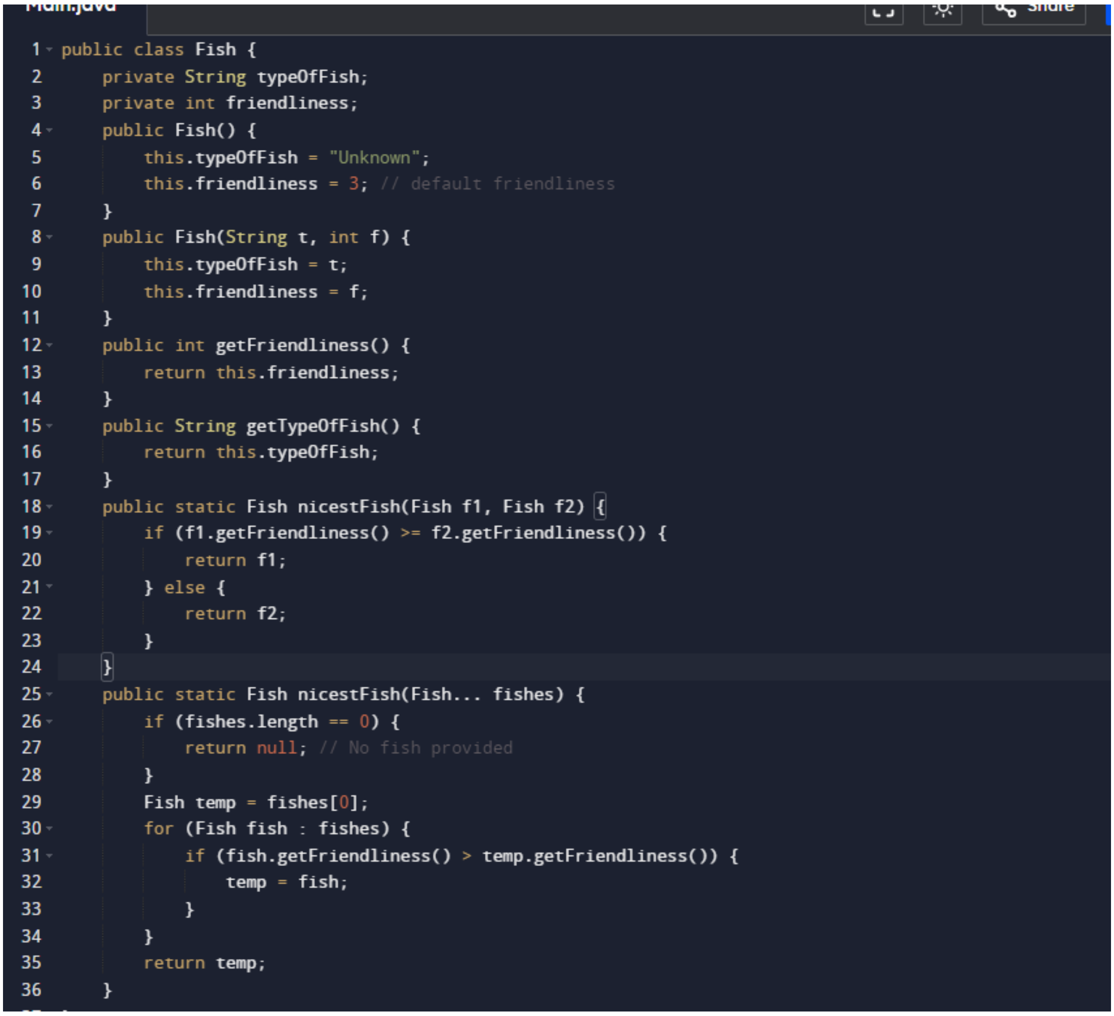
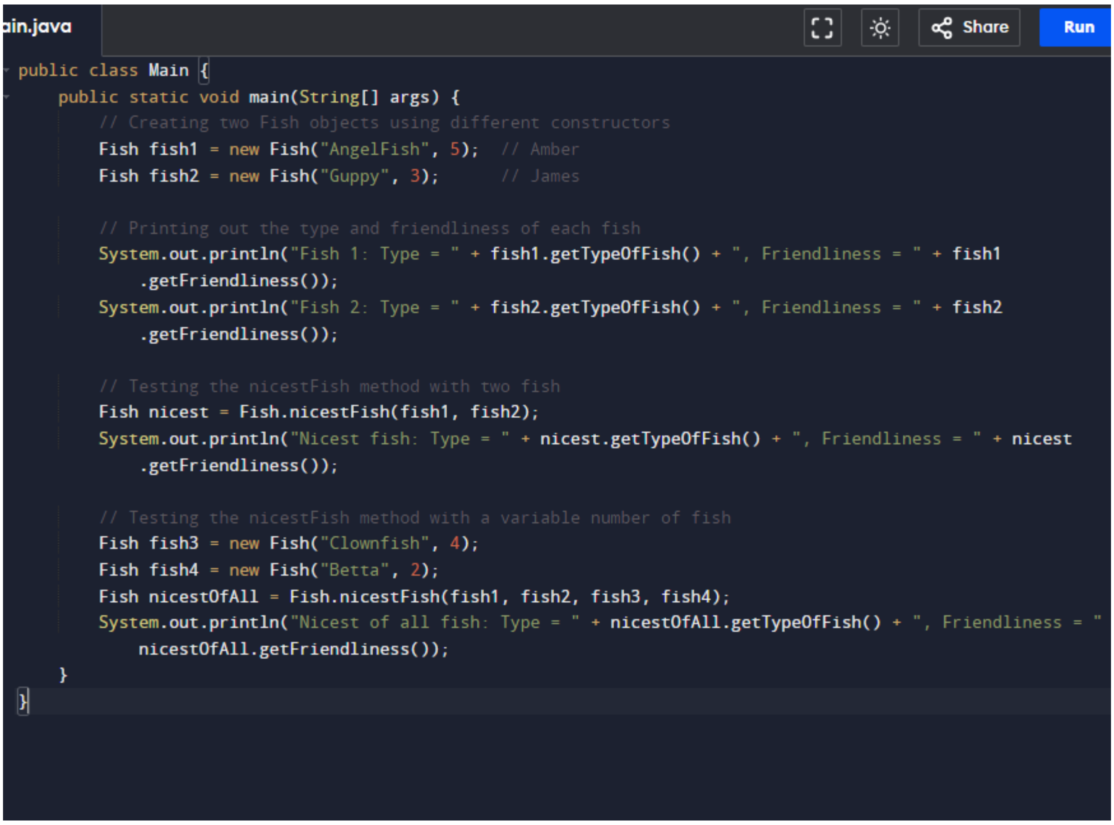
7.2_java fundamentals

public - A type of access modifier that permits access from anywhere.
constructor - Used to assign initial values to instance variables of a class.
varargs - A way to call a method with a variable number of arguments.
overloading - Having more than one constructor or method with the same name but different arguments.
private - A type of access modifier that permits access only from inside the same class.
default constructor - A constructor that does not have any parameters.
access modifier - Used to specify accessibility for variables, methods, and classes.



```java
public class Fish {
    private String typeOfFish;
    private int friendliness;
    public Fish() {
        this.typeOfFish = "Unknown";
        this.friendliness = 3; // default friendliness
    }
    public Fish(String t, int f) {
        this.typeOfFish = t;
        this.friendliness = f;
    }
    public int getFriendliness() {
        return this.friendliness;
    }
    public String getTypeOfFish() {
        return this.typeOfFish;
    }
    public static Fish nicestFish(Fish f1, Fish f2) {
        if (f1.getFriendliness() >= f2.getFriendliness()) {
            return f1;
        } else {
            return f2;
        }
    }
    public static Fish nicestFish(Fish... fishes) {
        if (fishes.length == 0) {
            return null; // No fish provided
        }
        Fish temp = fishes[0];
        for (Fish fish : fishes) {
            if (fish.getFriendliness() > temp.getFriendliness()) {
                temp = fish;
            }
        }
        return temp;
    }
```



```java
public class Main {
    public static void main(String[] args) {
        // Creating two Fish objects using different constructors
        Fish fish1 = new Fish("AngelFish", 5);  // Amber
        Fish fish2 = new Fish("Guppy", 3);      // James

        // Printing out the type and friendliness of each fish
        System.out.println("Fish 1: Type = " + fish1.getTypeOfFish() + ", Friendliness = " + fish1
            .getFriendliness());
        System.out.println("Fish 2: Type = " + fish2.getTypeOfFish() + ", Friendliness = " + fish2
            .getFriendliness());

        // Testing the nicestFish method with two fish
        Fish nicest = Fish.nicestFish(fish1, fish2);
        System.out.println("Nicest fish: Type = " + nicest.getTypeOfFish() + ", Friendliness = " + nicest
            .getFriendliness());

        // Testing the nicestFish method with a variable number of fish
        Fish fish3 = new Fish("Clownfish", 4);
        Fish fish4 = new Fish("Betta", 2);
        Fish nicestOfAll = Fish.nicestFish(fish1, fish2, fish3, fish4);
        System.out.println("Nicest of all fish: Type = " + nicestOfAll.getTypeOfFish() + ", Friendliness = "
            nicestOfAll.getFriendliness());
    }
}
```

3. Explanation of Constructor Overloading
Constructor Overloading allows a class to have more than one constructor with different parameter lists. This is possible because the Java compiler differentiates constructors based on the number and type of arguments. This is useful for creating objects in various ways while reusing the same class.

4. Access Modifiers
Access Modifiers in Java specify the accessibility of classes, methods, and variables:

Public: Accessible from anywhere.
Private: Accessible only within the same class.
Protected: Accessible within the same package and subclasses.
Default (no modifier): Accessible only within the same package.
Example Access Modifiers:
a. For a class Employee with sensitive details, such as salary, you might use private for the instance variables and provide public getters and setters to access them.

b. For an adding method used across classes, you would use public to ensure that it is accessible from other classes like Algebra.

Summary
The provided code covers:

Class and instance variable setup.
Constructors: No-argument and parameterized.
Methods: For returning friendliness, comparing fish, and handling variable arguments.
Access Modifiers: Explanation and practical use.