Java Fundamentals 7-2:

Parameters and Overloading Methods Practice Activities Lesson Objectives:

- Use access modifiers
- Pass objects to methods
- Return objects from methods
- Use variable argument methods
- Overload constructors
- Overload methods
- Write a class with specified arrays, constructors, and methods

Try It/Solve It:

1. Create a segment of code that initializes a public class Fish. Let the class contain a String typeOfFish, and an integer friendliness. Do not set values to these variables yet. These are instance variables and will be set inside the class constructors.

Program:

```
package dffg;


public class Fish {


    private String typeOfFish;
    private int friendliness;


    public Fish() {
    }


    public Fish(String typeOfFish, int friendliness) {
```

```
    this.typeOfFish = typeOfFish;
    this.friendliness = friendliness;
}

public String getTypeOfFish() {
    return typeOfFish;
}

public void setTypeOfFish(String typeOfFish) {
    this.typeOfFish = typeOfFish;
}

public int getFriendliness() {
    return friendliness;
}

public void setFriendliness(int friendliness) {
    this.friendliness = friendliness;
}

public static void main(String[] args) {
    Fish fish1 = new Fish();
    fish1.setTypeOfFish("Goldfish");
    fish1.setFriendliness(8);

    Fish fish2 = new Fish("Betta", 5);
```

```
        System.out.println("Fish 1: " + fish1.getTypeOfFish() + ", Friendliness: " +
fish1.getFriendliness());
        System.out.println("Fish 2: " + fish2.getTypeOfFish() + ", Friendliness: " +
fish2.getFriendliness());
    }
}
```

Output:

```
Fish 1: Goldfish, Friendliness: 8
Fish 2: Betta, Friendliness: 5
```

2. **Create a public constructor (a method with the same name as the class) inside the class Fish. This constructor should take in no arguments. Inside the constructor, set typeOfFish to "Unknown" and friendliness to 3, which we are assuming is the generic friendliness of fish.**

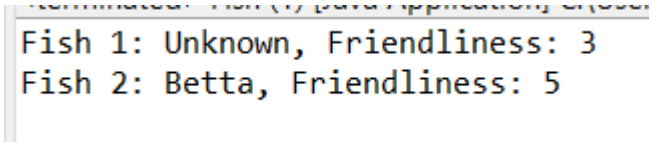
Program:

```
package nnn;


    public class Fish {
        private String typeOfFish;
        private int friendliness;


        public Fish() {
            this.typeOfFish = "Unknown";
            this.friendliness = 3;
        }


        public Fish(String typeOfFish, int friendliness) {
            this.typeOfFish = typeOfFish;
```

```
        this.friendliness = friendliness;
    }

    public String getTypeOfFish() {
        return typeOfFish;
    }

    public void setTypeOfFish(String typeOfFish) {
        this.typeOfFish = typeOfFish;
    }

    public int getFriendliness() {
        return friendliness;
    }

    public void setFriendliness(int friendliness) {
        this.friendliness = friendliness;
    }

    public static void main(String[] args) {
        Fish fish1 = new Fish();

        Fish fish2 = new Fish("Betta", 5);

        System.out.println("Fish 1: " + fish1.getTypeOfFish() + ",
Friendliness: " + fish1.getFriendliness());
```

```
            System.out.println("Fish 2: " + fish2.getTypeOfFish() + ",
Friendliness: " + fish2.getFriendliness());
        }
    }
```

Output:



3. **Create another public constructor inside the class Fish. Have this constructor take in a string t and an integer f. Let typeOfFish equal t, and friendliness equal f.**

Program:

```
package nnn;


public class Fish {

   private String typeOfFish;

   private int friendliness;


   public Fish() {

      this.typeOfFish = "Unknown";

      this.friendliness = 3;

   }


   public Fish(String typeOfFish, int friendliness) {

      this.typeOfFish = typeOfFish;

      this.friendliness = friendliness;

   }
```

```
public Fish(String t, int f) {
   this.typeOfFish = t;
   this.friendliness = f;
}

public String getTypeOfFish() {
   return typeOfFish;
}

public void setTypeOfFish(String typeOfFish) {
   this.typeOfFish = typeOfFish;
}

public int getFriendliness() {
   return friendliness;
}

public void setFriendliness(int friendliness) {
   this.friendliness = friendliness;
}

public static void main(String[] args) {
   Fish fish1 = new Fish();

   Fish fish2 = new Fish("Betta", 5);
```

```
        Fish fish3 = new Fish("Guppy", 7);


        System.out.println("Fish 1: " + fish1.getTypeOfFish() + ", Friendliness: " +
fish1.getFriendliness());
        System.out.println("Fish 2: " + fish2.getTypeOfFish() + ", Friendliness: " +
fish2.getFriendliness());
        System.out.println("Fish 3: " + fish3.getTypeOfFish() + ", Friendliness: " +
fish3.getFriendliness());
    }
}
```

Output:

```
Fish 1: Unknown, Friendliness: 3
Fish 2: Betta, Friendliness: 5
Fish 3: Guppy, Friendliness: 7
```

4. **Explain why is is possible to have more than one constructor with the same name and different arguments.**

Program:

Default Constructor:

```
public Fish() {
    this.typeOfFish = "Unknown";
    this.friendliness = 3;
}
```

Parameterized Constructor:

```
public Fish(String typeOfFish, int friendliness) {
    this.typeOfFish = typeOfFish;
    this.friendliness = friendliness;
}
```

Additional Parameterized Constructor:

```
public Fish(String t, int f) {
   this.typeOfFish = t;
   this.friendliness = f;
}
```

5.Create a method inside the class Fish called getFriendliness(), which takes in no arguments and returns the friendliness level of the fish.

Program:

```
package nnn;


public class Fish {
   private String typeOfFish;
   private int friendliness;


   public Fish() {
      this.typeOfFish = "Unknown";
      this.friendliness = 3;
   }


   public Fish(String typeOfFish, int friendliness) {
      this.typeOfFish = typeOfFish;
      this.friendliness = friendliness;
   }


   public Fish(String t, int f) {
      this.typeOfFish = t;
      this.friendliness = f;
```

```
}

public String getTypeOfFish() {
   return typeOfFish;
}

public void setTypeOfFish(String typeOfFish) {
   this.typeOfFish = typeOfFish;
}

public int getFriendliness() {
   return friendliness;
}

public void setFriendliness(int friendliness) {
   this.friendliness = friendliness;
}

public int getFriendlinessLevel() {
   return friendliness;
}

public static void main(String[] args) {
   Fish fish1 = new Fish();

   Fish fish2 = new Fish("Betta", 5);
```

```
        Fish fish3 = new Fish("Guppy", 7);

        System.out.println("Fish 1 Friendliness: " + fish1.getFriendlinessLevel());
        System.out.println("Fish 2 Friendliness: " + fish2.getFriendlinessLevel());
        System.out.println("Fish 3 Friendliness: " + fish3.getFriendlinessLevel());
    }
}
```

Output:

```
Fish 1 Friendliness: 3
Fish 2 Friendliness: 5
Fish 3 Friendliness: 7
```

6. Write a segment of code that initializes 2 new fish as defined below:

a. Fish 1: Name – Amber, Type – AngelFish, Friendliness level – 5 (very friendly)

b. Fish 2: Name – James, Type – Guppy, Friendliness level – 3 (neutral)

Program:

```
package nnn;

public class Fish {
    private String typeOfFish;
    private int friendliness;

    public Fish() {
        this.typeOfFish = "Unknown";
        this.friendliness = 3;
```

```
}

public Fish(String typeOfFish, int friendliness) {
   this.typeOfFish = typeOfFish;
   this.friendliness = friendliness;
}

public Fish(String t, int f) {
   this.typeOfFish = t;
   this.friendliness = f;
}

public String getTypeOfFish() {
   return typeOfFish;
}

public void setTypeOfFish(String typeOfFish) {
   this.typeOfFish = typeOfFish;
}

public int getFriendliness() {
   return friendliness;
}

public void setFriendliness(int friendliness) {
   this.friendliness = friendliness;
```

```
    }

    public int getFriendlinessLevel() {
        return friendliness;
    }

    public static void main(String[] args) {
        Fish fish1 = new Fish("AngelFish", 5);

        Fish fish2 = new Fish("Guppy", 3);

        System.out.println("Fish 1: Type – " + fish1.getTypeOfFish() + ",
Friendliness level – " + fish1.getFriendlinessLevel());
        System.out.println("Fish 2: Type – " + fish2.getTypeOfFish() + ",
Friendliness level – " + fish2.getFriendlinessLevel());
    }
}
```

Output:

```
Fish 1: Type – AngelFish, Friendliness level – 5
Fish 2: Type – Guppy, Friendliness level – 3
```

7.Create a method nicestFish that takes in two fish as parameters, compares the friendliness level of two fish, and returns the fish with the higher friendliness. Test this method with the fish defined in problem 6. (Friendliness scale: 1 mean, 2 not friendly, 3 neutral, 4 friendly, 5 very friendly) Hint: fishName.getFriendliness() gives you the integer number of the friendliness of fishName. You have already created getFriendliness() in problem 5.

Program:

```
package nnn;
```

```
public class Fish {
    private String typeOfFish;
    private int friendliness;

    public Fish() {
        this.typeOfFish = "Unknown";
        this.friendliness = 3;
    }

    public Fish(String typeOfFish, int friendliness) {
        this.typeOfFish = typeOfFish;
        this.friendliness = friendliness;
    }

    public Fish(String t, int f) {
        this.typeOfFish = t;
        this.friendliness = f;
    }

    public String getTypeOfFish() {
        return typeOfFish;
    }

    public void setTypeOfFish(String typeOfFish) {
        this.typeOfFish = typeOfFish;
```

```
}

public int getFriendliness() {
   return friendliness;
}

public void setFriendliness(int friendliness) {
   this.friendliness = friendliness;
}

public int getFriendlinessLevel() {
   return friendliness;
}

public static Fish nicestFish(Fish fish1, Fish fish2) {
   if (fish1.getFriendlinessLevel() > fish2.getFriendlinessLevel()) {
      return fish1;
   } else {
      return fish2;
   }
}

public static void main(String[] args) {
   Fish fish1 = new Fish("AngelFish", 5);

   Fish fish2 = new Fish("Guppy", 3);
```

```
        Fish nicerFish = Fish.nicestFish(fish1, fish2);


        System.out.println("The nicest fish is of type – " +
nicerFish.getTypeOfFish() + " with a friendliness level of – " +
nicerFish.getFriendlinessLevel());
    }
}
```

Output:

```
The nicest fish is of type – AngelFish with a friendliness level of – 5
```

8. Modify the method nicestFish() to take be a variable argument method that takes in a variable number of fish and returns the nicest fish out of the fish it is given. Hint: Inside of the method, create a new fish called temp. Set temp equal to the first fish passed into the method. Use a for loop to go through all the fish passed into the method and if you discover a fish that is more friendly than temp, set temp equal to that fish. After the for loop is complete, temp should be the friendliest fish. Return temp.

Program:

```
package nnn;


public class Fish {
    private String typeOfFish;
    private int friendliness;


    public Fish() {
        this.typeOfFish = "Unknown";
        this.friendliness = 3;
    }
```

```
public Fish(String typeOfFish, int friendliness) {
   this.typeOfFish = typeOfFish;
   this.friendliness = friendliness;
}

public Fish(String t, int f) {
   this.typeOfFish = t;
   this.friendliness = f;
}

public String getTypeOfFish() {
   return typeOfFish;
}

public void setTypeOfFish(String typeOfFish) {
   this.typeOfFish = typeOfFish;
}

public int getFriendliness() {
   return friendliness;
}

public void setFriendliness(int friendliness) {
   this.friendliness = friendliness;
}
```

```
public int getFriendlinessLevel() {
    return friendliness;
}

public static Fish nicestFish(Fish... fishArray) {
    if (fishArray == null || fishArray.length == 0) {
        throw new IllegalArgumentException("No fish provided");
    }

    Fish temp = fishArray[0];

    for (Fish fish : fishArray) {
        if (fish.getFriendlinessLevel() > temp.getFriendlinessLevel()) {
            temp = fish;
        }
    }

    return temp;
}

public static void main(String[] args) {
    Fish fish1 = new Fish("AngelFish", 5);
    Fish fish2 = new Fish("Guppy", 3);
    Fish fish3 = new Fish("Betta", 4);

    Fish nicest = Fish.nicestFish(fish1, fish2, fish3);
```

```
        System.out.println("The nicest fish is of type – " + nicest.getTypeOfFish() +
" with a friendliness level of – " + nicest.getFriendlinessLevel());
    }
}
```

Output:

```
The nicest fish is of type – AngelFish with a friendliness level of – 5
```

9. Test your method nicestFish() with the fish described in problem 6. Which fish is returned?

Program:

```
package nnn;


public class Fish {
    private String typeOfFish;
    private int friendliness;


    public Fish() {
        this.typeOfFish = "Unknown";
        this.friendliness = 3;
    }


    public Fish(String typeOfFish, int friendliness) {
        this.typeOfFish = typeOfFish;
        this.friendliness = friendliness;
    }
```

```
public Fish(String t, int f) {
   this.typeOfFish = t;
   this.friendliness = f;
}

public String getTypeOfFish() {
   return typeOfFish;
}

public void setTypeOfFish(String typeOfFish) {
   this.typeOfFish = typeOfFish;
}

public int getFriendliness() {
   return friendliness;
}

public void setFriendliness(int friendliness) {
   this.friendliness = friendliness;
}

public int getFriendlinessLevel() {
   return friendliness;
}

public static Fish nicestFish(Fish... fishArray) {
```

```
        if (fishArray == null || fishArray.length == 0) {
            throw new IllegalArgumentException("No fish provided");
        }

        Fish temp = fishArray[0];

        for (Fish fish : fishArray) {
            if (fish.getFriendlinessLevel() > temp.getFriendlinessLevel()) {
                temp = fish;
            }
        }

        return temp;
    }

    public static void main(String[] args) {
        Fish fish1 = new Fish("AngelFish", 6);
        Fish fish2 = new Fish("Guppy", 3);

        Fish nicest = Fish.nicestFish(fish1, fish2);

        System.out.println("The nicest fish is of type – " + nicest.getTypeOfFish() +
" with a friendliness level of – " + nicest.getFriendlinessLevel());
    }
}
```

Output:

```
The nicest fish is of type - AngelFish with a friendliness level of - 6
```

10. Determine the best access modifier for each of the following situations:
a. A class Employee records the name, address, salary, and phone number.

Program:

```
public class Employee {
    private String name;
    private String address;
    private double salary;
    private String phoneNumber;

    public Employee(String name, String address, double salary, String phoneNumber) {
        this.name = name;
        this.address = address;
        this.salary = salary;
        this.phoneNumber = phoneNumber;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
```

```
    public String getAddress() {
        return address;
    }

    public void setAddress(String address) {
        this.address = address;
    }

    public double getSalary() {
        return salary;
    }

    public void setSalary(double salary) {
        this.salary = salary;
    }

    public String getPhoneNumber() {
        return phoneNumber;
    }

    public void setPhoneNumber(String phoneNumber) {
        this.phoneNumber = phoneNumber;
    }
}
```

b. An adding method inside of a class BasicMath that is also used in the Algebra class.

Program:

```
public class BasicMath {
    public static int add(int a, int b) {
        return a + b;
    }
}
```