# OOP Assignment 1

| | |
|---|---|
| Name: | Syed Muhammad Raza Ali |
| Enrolment: | 02-134231-028 |
| Course: | Object Oriented Programming |
| Faculty: | Miss Sameena Javed |

# ASSIGNMENT 01

Marks: 05

---

**QUESTION (Do as Directed)** **(2.5+2.5 = 5 Marks)**
**(CLO1, PLO1, C2)**

1. **Infer** the four pillars of object-oriented programming (OOP) - encapsulation, inheritance, polymorphism, and abstraction - contribute to building robust and maintainable software systems? Provide examples for each pillar to illustrate their importance.

2. **Summarize** the concept of the Constructor, and keep the following statement in consideration to elaborate your answer. Also, write a small program to copy a constructor.

```
Polynomial p = new Polynomial(new Term(2.0, 2), new Term(3.0, 1), new Term(-1.0, 0));
```

The Polynomial class has a constructor that takes a variable number of Term arguments. It uses the Arrays.asList method to convert the array of terms into a list. It also has a method evaluate that takes a variable x and returns the value of the expression evaluated at x. The evaluate method iterates through the list of terms and computes the value of each term using the Math.pow method and adds it to the result.

- Also derive the polynomial equation which the class Polynomial is handling.

****************************

## 1. Four Pillars of the Object Oreinted Programming:

The four pillars of object-oriented programming are:

- Abstraction
- Encapsulation
- Inheritance
- Polymorphism

## Abstraction

Quite simply, abstraction is the concept of wrapping up complex actions in simple verbs. Describe each thing you've abstracted clearly, and hide the complexity. Someone can always pop the hood if they need to.

To use a cooking metaphor: Puff pastry is incredibly hard to make. Even professionals don't make their own puff pastry. I don't need to know how puff pastry is made in order to buy it and use it.

*Why tho?*

We can't really work until we can hold the problem in our minds. I takes time and effort to build up a mental picture of your code, and the larger that mental picture is, the more likely you are to make a mistake.

*Where have you seen this?*

The `append` and `insert` methods of your data structures. Instead of calling keep_going_until_you_find_a node_that_is_nil_and_create_a_new_node_and_fill_it_with_this_data, we can call `append` or `insert`

*Importance for Robustness and Maintainability:*

**Reduced Complexity:** Abstraction helps in managing complexity by providing a simplified view of an object. This allows developers to work with higher-level concepts and reduces the cognitive load.

### Isolation of Changes:

By providing a clear interface, abstraction isolates the external code from the internal implementation. This means that changes to the internal details can be made without affecting the code that uses the object.

## Encapsulation

As much as you can, keep state and logic internal. This can mean have as few attr readers as possible, or as few instance vars as possible. The less you have to keep track off at any given time, the better.

To use a cooking metaphor: Have you ever seen a recipe where they split out some of the incredients into other recipes. Maybe you're baking a cake, and instead of the butter and powdered sugar being a part of the cake recipe, the cake simply asks for "frosting", and links to another recipe to make the frosting.

*Why tho?*

Code organization. Where does it make sense to keep things together, and where are responsibilities seperate

*Where have you seen this?*

In battleship, I asked you to think about what responsibilities belonged where. There wasn't a right answer to this question, but everyone created multiple classes, and grouped things that belonged together. Then you can just call a few methods to access a wealth of complexity.

*Relating Abstraction and Polymorphism:*

Abstraction and encapsulation support each other. If you don't group like things together, then you're going to have a harder time abstracting them. They'll have to be passing data back and forth.

*Importance for Robustness and Maintainability:*

### Data Protection:

Encapsulation helps in preventing direct access to an object's data. This means that internal details can be hidden, which prevents unintended modification and ensures data integrity.

### Code Flexibility:

The internal implementation of an object can be changed without affecting the code that uses it, as long as the external interface remains the same. This reduces the impact of changes and makes maintenance easier.

# Inheritance

Classes can have parent classes. Child classes will inherit all of the behavior and attributes of the parent class. Child classes can then choose to overwrite some of those as necessary.

To use a cooking metaphor: Every omelette follows basically the same process. Break some eggs, cook those eggs, add some ingredients, serve. All omelette But a Denver omelette is not the same as an omelette du fromage.

*Why tho?*

Code organization. Where does it make sense to keep things together, and where are responsibilities separate

- As our applications become more complex, we can find that we often have classes that are very similar. And there is often a hierarchy to them.
- DRY. If I change something in the parent class, all the children also get that change.

*Where have you seen this?*

Minitest. Your tests are classes. Every test file you create is a new class. You're using methods like `assert` and `refute`, and when you create a `setup` method, you don't have to call it. The parent class calls it. There are all kinds of behavior that your tests have because they inherit from Minitest::Test

*Importance for Robustness and Maintainability:*

### Code Reusability:

Inheritance promotes reuse of code. Common functionality can be defined in a base class, and specialized behavior can be added in derived classes. This reduces redundancy and leads to more maintainable code.

### Hierarchy and Organization:

It allows for the creation of class hierarchies, which models real-world relationships and helps in organizing and understanding complex systems.

## Polymorphism

Defined as "the condition of occurring in several different forms". Basically, it just means that we can call the same method on different objects.

*Why tho?*

Naming is hard. If there's a verb/method name that works for similar processes, just use that same name. Even if it doesn't actually refer to the same thing happening

It also allows you to send different types of objects to the same method. If both `Student` and `Teacher` have an `email` attribute, then they can both be sent as a parameter to `send_email`.

To use a cooking metaphor: You don't need a new verb to `slice` each food. You can slice an apple, and a carrot and a watermelon. Your process might differ for each food, and it might not, but you can use the same verb.

*Where have you seen this?*

- Strings and arrays both have length. They both return integers, and they both mean roughly the same thing, but the internal operation differs between the two classes.
- In the inheritance lesson, you saw that all employee types had a `total_compensation` method, but the behavior of that method differs.

*Relating Inheritance and Polymorphism*

Inheritance and Polymorphism kind of go together. At least in the sense that all child classes will have the methods and attributes that were defined on the parent class.

*Importance for Robustness and Maintainability:*

## Flexibility:

Polymorphism allows for flexibility in code design. It enables the use of a single method to perform different operations depending on the type of object it is called on. This reduces the need for complex conditional logic.

## Plug-and-Play Components:

Different objects that share a common interface can be used interchangeably. This allows for the creation of modular, interchangeable components that can be replaced or extended without affecting the overall system.

## 1.Constructor:

A constructor in a class is a special method used to initialize an object when it is created. Its purpose is to set initial values for the attributes of the object so that it is in a valid state to perform operations.

In the given statement:

```
Polynomial p = new Polynomial(new Term(2.0, 2), new Term(3.0, 1), new Term(-1.0, 0));
```

Here, a new object of the class Polynomial is being created using its constructor. The constructor takes a variable number of Term arguments, which are used to define the terms of the polynomial. The Arrays.asList method is used to convert the array of terms into a list, which can then be processed within the class.

The polynomial equation being handled by the Polynomial class can be derived from the given terms:

$2.0x^2+3.0x^2-1.0x^0$

This polynomial represents a quadratic equation with coefficients:

Where:

$a^2=2.0$

$a^1=3.0$

$a^0=-1.0$

And

$a_n, a_{n-1}, \ldots, a_1, a_0$ are coefficients.

x is the variable.

The evaluate method in the Polynomial class likely computes the value of this polynomial expression for a given value of x by iterating through the list of terms, using the Math.pow method to raise x to the appropriate power, and adding it to the result.

Here's a small Java program to demonstrate the concept and usage of a constructor:

```
import java.util.Arrays;
import java.util.List;

class Term {
    private double coefficient;
    private int exponent;

    public Term(double coefficient, int exponent) {
        this.coefficient = coefficient;
        this.exponent = exponent;
    }

    public double getCoefficient() {
        return coefficient;
    }

    public int getExponent() {
        return exponent;
    }
}

class Polynomial {
    private List<Term> terms;

    public Polynomial(Term... terms) {
        this.terms = Arrays.asList(terms);
    }

    public double evaluate(double x) {
        double result = 0.0;
        for (Term term : terms) {
            result += term.getCoefficient() * Math.pow(x, term.getExponent());
        }
        return result;
    }
}

public class Main {
    public static void main(String[] args) {
        Polynomial p = new Polynomial(new Term(2.0, 2), new Term(3.0, 1), new Term(-1.0, 0));

        double x = 2.0;
        double result = p.evaluate(x);

        System.out.println("Value of the polynomial for x = " + x + " is: " + result);
    }
}
```

In this program, we have Term and Polynomial classes. The Term class represents a term in a polynomial with a coefficient and an exponent. The Polynomial class has a constructor that takes a variable number of Term arguments and an evaluate method to calculate the value of the polynomial for a given x. The main method creates a Polynomial object and evaluates it for a specific x.