Homework - Sep 21

#sagar-sir

1. Data and Functions Scattered
2. Code Duplication
3. Poor Security (Data Exposure)
4. Hard to Maintain Large Programs
5. Poor Real-World Mapping

How does OOP solve these problems????

Answer

Object-Oriented Programming (OOP) provides solutions to common issues faced in procedural programming languages like C by structuring code in a way that is more organized, secure, and easier to maintain. Here's a simple and concise explanation of how OOP addresses each of the problems you've listed:

1. Data and Functions Scattered

In procedural programming, data and the functions that operate on that data are often kept separate. This can make it difficult to understand their relationships and manage them as a program grows.

OOP Solution: Encapsulation
OOP bundles data and the functions that work on that data together into a single unit called an object. This concept, known as encapsulation, keeps related code organized and self-contained. For example, an 'Employee' object would contain both the employee's data (like name and salary) and the functions to manage that data, like `calculateBonus()` or `updateAddress()`.

2. Code Duplication

Procedural programming can lead to code duplication when similar logic is needed in different parts of a program. This makes the code harder to maintain, as a change in one place often requires changes in many others.

OOP Solution: Inheritance and Polymorphism
OOP promotes code reusability through the principle of "Don't Repeat Yourself" (DRY).

- **Inheritance** allows a new class to inherit properties and methods from an existing class. For instance, we can create a general '**Vehicle**' class and then create more specific '**Car**' and '**Truck**' classes that inherit the common attributes of a vehicle, while also having their own unique features.
- **Polymorphism** allows objects of different classes to be treated as objects of a common parent class, enabling you to write more generic and flexible code.

3. Poor Security (Data Exposure)

In procedural programming, data is often globally accessible, meaning it can be modified from anywhere in the program.This can lead to accidental data corruption and makes the program less secure.

OOP Solution: Encapsulation and Data Hiding
Encapsulation provides a protective barrier around data. In OOP, we can control access to the data within an object using access specifiers like 'private' and 'public'. By making data private, it can only be accessed and modified through the object's public methods, preventing unauthorized access and improving data integrity.

4. Hard to Maintain Large Programs

As a procedural program grows in size and complexity, it becomes increasingly difficult to manage and maintain. The lack of a clear structure can make it hard to understand the impact of changes.

OOP Solution: Modularity and Clear Structure
OOP's object-based structure provides a clear and organized framework for programs. Code is broken down into smaller, self-contained objects, making it more modular and easier to understand, debug, and maintain.

5. Poor Real-World Mapping

Procedural programming focuses on a sequence of actions or procedures, which doesn't always map intuitively to real-world problems.

OOP Solution: Real-World Modeling
OOP is designed to model real-world entities. Objects in the code can represent real-world objects, making the program's structure more logical and easier to comprehend. For example, in a banking application, we can have objects like '**Customer**,' '**Account**', and '**Transaction**' that directly correspond to their real-world counterparts. This makes the software design more intuitive and aligned with the problem it's trying to solve.

September 21, Section B

```
class Student {

    public String name;
    public String id;
    public int age;
    public String grade;

    public Student(String id, String name, int age, String grade){
        this.id = id;
        this.name = name;
        this.age = age;
        this.grade = grade;
    }

    public void displayDetails(){
```

```
        System.out.printf("Id: %s\n", this.id);
        System.out.printf("Name: %s\n", this.name);
        System.out.printf("Age: %d\n", this.age);
        System.out.printf("Grade: %s\n", this.grade);

        System.out.printf("\n");
    }
}


class Main {
    public static void main(String[] args){
        Student manoj = new Student("221026110","Manoj Shrestha", 21,"A");
        manoj.displayDetails();

        Student aditya = new Student("221026127", "Aditya Poudel", 20, "B");
        aditya.displayDetails();
    }
}
```

1. Why is it better to use a class for students instead of separate variables?

A class groups related data (id, name, age, grade, etc.) into one unit. This makes the code easier to manage, reuse, and extend compared to keeping many separate variables for each student.

2. How can we create 50 students using the same class?

We can create multiple `Student` objects using the `new` keyword, or store them in an **array** or a **list**.

```
Student[] student = new Student[50];
```

Then assign each element with a new `Student` object.

3. Can you add one more attribute (e.g., email) to the class? What changes would you need to make?

Yes. We can add a new `String email;` in the class and include it in the constructor and `displayDetails()` method. We also need to provide the email value when creating a new student.

4. If one student's age changes, how do we update it using the object?

We can directly update the attribute using the student's object.

```
manoj.age = 22;
```

5. How is this example similar to the Car class we did earlier?

Both `Student` and `Car` are classes that define **attributes** (data) and **methods** (actions). Objects are created from the class, and each object holds its own values.

6. What happens if we don’t assign values to the attributes of a Student object?

Java will assign default values depending on the type of the attribute.

```
- Numbers → `0`
- Strings (objects) → `null`
- Booleans → `false`
```

7. Can you think of more useful attributes for a Student class (e.g., subject, email, phone number)?

Yes. We can include other attributes like `phoneNumber`, `address`, `dateOfBirth`, `subjects`, `marks`, `attendance`, and `department`. These make the class more realistic and useful for managing student information.