COGNIZANT DIGITAL NURTURE – 4.0

WEEK – 1 (6373793 – GHAMANA S)

Design Patterns and Principles

Exercise 1: Implementing the Singleton Pattern

Program Debug Debug Result Java (11)

```java


import java.util.*;
class Singleton{
    private Singleton(){}

    private static Singleton instance;

    public static Singleton getInstance(){
        if(instance==null){
            instance=new Singleton();
        }
        return instance;
    }
}
public class Main{
    public static void main(String args[]){
        Singleton obj=Singleton.getInstance();
        System.out.println("Singleton instance obtained!");
    }
}
```

Compiler Message

```
Compilation successful
```

Custom Testcase

Output

```
Singleton instance obtained!
```

Notes:

6373793 - Superset ID

Cognizant Digital Nurture - 4.0

MODULE - 1 WEEK - 1

Design Patterns and Principles.

1) The SOLID principles of object-oriented design.

Reason for SOLID principles:

SOLID principles introduced by Robert C. Martin in 2000

These concepts were later built by Michael Feathers. - introduced SOLID acronym.

In last 20 years, these five principles have revolutionized the world of object-oriented programming changing the way that we write software.

What is SOLID and how does it help us write better code?

Martin and Feathers' design principles encourages us to create more * maintainable * understandable and * flexible software.

To reduce and save ourselves a lot of headaches further the road! as our application grow in size, we can reduce their complexity.

Five concepts of SOLID principles:

- ✓ Single Responsibility
- ✓ Open/closed
- ✓ Liskov Substitution
- ✓ Interface segregation
- ✓ Dependency Inversion

Single Responsibility Principle :

This principle states that a class should only have one responsibility.

Furthermore, it should only have one reason to change.

How does this principle help us to build better software?

Benefits : ✓ Testing - A class wid 1 respo. will have few bugs
✓ Lower coupling - less functionality in a single class will have fewer dependencies
✓ Organization :- Smaller, well-organised classes are easier to search the monolithic ones

Open for Extension, closed for Modification.

O- SOLID, known as the open-closed principle. (i.e) classes should be open for extension but closed for modification.

This will help ourselves from modifying the existing code and causing the potential new bugs.

Liskov Substitution :

The most complex of the five principles. (i.e) If class A is a subtype of class B we should be able to replace B with A without disrupting the behaviour of our program

Example :

By throwing the car without an engine into the mix, we are inherently changing the behaviour of our program.

This is the blatant violation of Liskov Substitution and is a bit harder to fix than our previous two principles.

Interface Segregation :

I - SOLID stands for interface segregation, and it simply means that larger interfaces should be split into smaller ones.

By doing so, we can ensure that implementing classes only need to be concerned about the methods that are of interest to them

Dependency Inversion :

The principle of dependency inversion refers to the decoupling of software modules. This way, instead of high-level modules depending on low-level modules, both will depend on abstractions.

Conclusion:

Deep-dive into the SOLID principles of object-oriented design.

Robert C. Martin in his 2000 paper introduced

" Design Principles and Design Patterns ".

Something which is not static, belongs to an object

In simple words:

Non-static member inside a static is not allowed

↓ belong to instance

↓ does not belong to instance

System.out.println();

↓ class ↓ var. ↓ Method → Java people provided.

Singleton class:

It is a class in which we can create only one object.

Only one object is allowed.

So the constructor can be in private.

(i.e): private student() { }

↓

So No one can access and only one object creation is allowed.

Properties of the OOPs: ⇒ Pillar of Java.

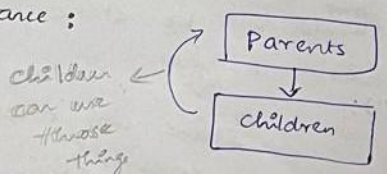
i) Inheritance

ii) Polymorphism

iii) Encapsulation

iv) Abstraction.

Inheritance:



[money, human traits, assets]

In OOPs:

Child

class

}

Types

1)

2)

den

live session notes:

LIVE MEETING SESSION NOTES - WEEK-1

Design pattern, Data structures and Algorithms.

Understanding SOLID Principles in OOPS - Writing clean, Scalable and Maintainable code. → Recipe [Ex]

↓

Cafe → Crowd increases → No changes → but satisfied the customers

Types of Design Patterns.

SOLID - S - Single Responsibility Principle.

→ class should have only one reason to change.

Creating an Application:

O - Open/closed Principle.

→ Software entities should open for the extension and close for the modification.

L - Liskov Substitution.

I - Interface Segregation.

D -

Three Types of

- Creational Patterns - Obj creation mechanism.
- Structural Patterns -
- Behavioral Patterns - Deal with obj interaction & Responsibility

→ Singleton
Factory Method
Abstract ..