Design Defects and Restructuring

LECTURE 03

SAT, SEP 19, 2020

Program Structure

Don’ts for Structured Programming

- Goto
- Break (unless under switch statement)
- Continue
- Return (other than last line in a function)

Side effects to avoid

- Global Variables
- Global Objects (unless they are read only)
- Static Variables and Objects (unless they are initialized once)

Don’ts for better programming practices

- Getters and Setters (unless they have purposeful names)
- Mutation

Object Oriented Analysis

Object Types

- Person (Role)
- Product
- Event – Result of Happening, Action or Association (Time?)
- Spatial – Location – Organization – Group
- Temporal – Time

Generalization / Classification – Example

A basic leave management system is required

Zero or more leaves can be associated to each employee

A leave can be associated to an employee based on his/her location, designation and gender. Each leave has its own rules or conditions

Some of the rules or conditions are

- An employee cannot gain more leaves than a threshold value in a year
- An employee cannot request more leaves than a threshold value in a year, or in a month, or during the employment
- Balance of a leave cannot be more than a threshold value based on employee's designation
- A leave that is exclusive to a gender
- A leave that is exclusive to a location

Objects and Classes

Object Recognition

Class Creation

Top Down Approach

Bottom Up Approach

Abstraction

Dependency

Object Dependency

There are number of ways an object can be used

It depends on

- which class is creating an object
- when the object is being created, and
- where it is being created

We need to understand 3 factors

- Scope
- Creator class
- Location and event of creation

Object Dependency

Scope

- The object is defined at the class level
- The object is defined at the method level

Creation

- The class is creating the object where it is defined
- Another class is creating the object and the reference is provided to the class where the object is defined

Location of creation

- Constructor
- Within the method

Object Dependency Examples

```
class ClassA
{
    private ClassB classB = new ClassB();
}
```

Object Dependency Examples

```
class ClassA
{
    private ClassB classB;

    public ClassA()
    {
        classB = new ClassB();
    }
}
```

Object Dependency Examples

```
class ClassA
{
    private ClassB classB;

    public ClassA()
    {
    }

    public void processClassB()
    {
        classB = new ClassB();
    }
}
```

Object Dependency Examples

```
class ClassA
{
    private ClassB classB;

    public ClassA()
    {
        classB = Factory.GetClassB();
    }
}
```

Object Dependency Examples

```
class ClassA
{
    private ClassB classB;

    public ClassA()
    {
    }

    public void processClassB()
    {
        classB = Factory.GetClassB();
    }
}
```

Object Dependency Examples

```
class ClassA
{
    private ClassB classB;

    public ClassA(ClassB objClassB)
    {
        classB = objClassB;
    }
}
```

Object Dependency Examples

```
class ClassA
{
    private ClassB classB;

    public ClassA()
    {
    }

    public void processClassB(ClassB objClassB)
    {
        classB = objClassB;
    }
}
```

Object Dependency Examples

```
class ClassA
{
    public ClassA()
    {
    }

    public void processClassB()
    {
        ClassB classB;

        classB = new ClassB();
    }
}
```

Object Dependency Examples

```
class ClassA
{
    public ClassA()
    {
    }

    public void processClassB()
    {
        ClassB classB;

        classB = Factory.GetClassB();
    }
}
```

Object Dependency Examples

```
class ClassA
{
    public ClassA()
    {
    }

    public void processClassB(ClassB objClassB)
    {
        ClassB classB;

        classB = objClassB;
    }
}
```

Software Design