S.O.L.I.D Principles

Low Level Design

Single Responsibility Principle

Mayank

S.O.L.I.D Principles ?

How you write a code is equally important to what you write

You should always write any code keeping few things in mind

- Easy to maintain
- Easy to test
- Easy to scale
- Easy to understand and more

These 5 SOLID principles help you create a software more maintainable, understandable, and flexible

Single Responsibility Principle

Consider a case where you need to write a software for a Book Shop

First class that you will make is 'Book'.

```
public class Book {
      private String name;
      private String author;
      private String text;
      //constructor, getters and setters
}
```

Now this 'Book' class contains all required fields and attributes to represent a Book

Single Responsibility Principle

You can add functions in the 'Book' class to replace/modify some piece of text as well.

So you have an attribute named 'text' which contains all the text inside a Book (just an example)

Now the output of a Book's text can be streamed to 'Console' or to a file, so let's add 2 methods in our 'Book' class

- printTextToConsole()
- printTextToFile()

Single Responsibility Principle

your final 'Book' class looks like-

```
public class Book {
      private String name;
      private String author;
      private String text;
      //constructor, getters and setters

     void printTextToConsole() {
     // implementation
     }
     void printTextToFile() {
     // implementation
    }
}
```

Single Responsibility Principle

As a beginner whatever code you saw in previous slide might not seem problematic but we will see later what are the problems

Single Responsibility Principle says that always try to keep a class with just '**One Responsibility**'

or

A class should only be touched or modified if that '**One Responsibility**' is not handled or working properly

Identify how many responsibilities are handled by your 'Book' class rightnow ??

Single Responsibility Principle

```
public class Book {
     private String name;
     private String author;
     private String text;
     //constructor, getters and setters

    void printTextToConsole() {
    // implementation
    }
    void printTextToFile() {
    // implementation
   }
}
```

Info related to a book

Functions to print a book's text

Single Responsibility Principle

So we identified that currently our 'Book' class is having 2 responsibilities

1) Info about a book
2) Print texts of book to console or a file

Now let's split these 2 responsibilities in 2 separate classes

```
public class Book {
      private String name;
      private String author;
      private String text;
      //constructor, getters
     // and setters
}
```

```
public class BookPrinter {
    void printTextToConsole(String text) {
 // implementation
 }
    void printTextToFile(String text) {
 // implementation
 }
}
```

Single Responsibility Principle

Now while testing or in production if all your code works fine just that the text is not being streamed to a file properly

So creating Single Responsibility Classes helps you as -

- Since you know that **printTextToFile()** is in a separate file so it's easier for you to actually **debug** the exact class
- In case you update the code of printTextToFile(), then you just need to test the '**Book Printer**' class but if there was just a single class then once you update a function you need to **re-test** other functions which were working fine (which is not needed)

Single Responsibility Principle

'Single Responsibility Classes' have many advantages-

- **Frequency and Effects of Changes**
  If frequency of changes is high then you don't need to re-test functions & classes which are not modified

- **Easy to understand, maintain & test**
  Classes like 'Book' & 'BookPrinter' are self explanatory

there are many other advantages as well.

Mayank

Software Engineer | StoryTeller

Code Smarter

I hope you found it useful

Follow for content related to DSA | System Design