

A grayscale photograph of a person with dark hair wearing large headphones, looking intently at a laptop screen. Their hand is resting on their chin in a thoughtful pose. The background is blurred, showing what appears to be a desk or office environment. A white rectangular border is superimposed over the image, framing the text.

# THE SINGLE RESPONSIBILITY PRINCIPLE

Java Clean code

# Outline

## **Lesson 6.**

The Single Responsibility  
Principle

## **Lesson 7.**

The Open Closed Principle

## **Lesson 8.**

The Liskov Substitution Principle

## **Lesson 9.**

The Interface Segregation  
Principle

## **Lesson 10.**

The Dependency Inversion  
Principle

*Any fool can write code that  
a computer can understand.  
Good programmers write code that  
humans can understand.*

**Martin Fowler**



# SOLID

- Single Responsibility Principle
- Open Closed Principle
- Liskov Substitute Principle
- Interface Segregation Principle
- The Dependency Inversion Principle



# The Single Responsibility Principle

Class should have one and only one reason to change

- Loose coupling
- Code is easier to maintain
- Easier to add new functionality
- Easier to debug







# Spring Data Repository

```
public interface DriverRepository  
extends CrudRepository<Driver,  
  
Long> {  
  
    List findByLastName(String  
lastName);
```

```

FileManager.java  Employer.java
1  package com.kirilanastasov.srp;
2
3  import java.io.File;
13
14  public class FileManager {
15
16      public void downloadFile(String location) throws IOException {
17          URL website = new URL("http://www.google.com/");
18          ReadableByteChannel rbc = Channels.newChannel(website.openStream());
19          FileOutputStream fos = new FileOutputStream("google.html");
20          fos.getChannel().transferFrom(rbc, 0, Long.MAX_VALUE);
21      }
22
23      public void parseTheFile(File file) throws FileNotFoundException {
24          Scanner input = new Scanner(file);
25          while (input.hasNext()) {
26              String nextToken = input.next();
27              // or to process line by line
28              String nextLine = input.nextLine();
29          }
30          input.close();
31      }
32
33      public void persistTheData(List<String> data) {
34          Employer entity = new Employer(100, "name");
35          entity.random = 1234;
36          Object gson;
37      }
38  }
39

```

SRP

```
FileManager.java  Employer.java  FileDownloader.java  FileParser.java  DataPe

1 package com.kirilanastsov.srp;
2
3 import java.io.FileOutputStream;
4 import java.io.IOException;
5 import java.net.URL;
6 import java.nio.channels.Channels;
7 import java.nio.channels.ReadableByteChannel;
8
9 public class FileDownloader {
10
11     public void downloadFile(String location) throws IOException {
12         URL website = new URL("http://www.google.com/");
13         ReadableByteChannel rbc = Channels.newChannel(website.openStream());
14         FileOutputStream fos = new FileOutputStream("google.html");
15         fos.getChannel().transferFrom(rbc, 0, Long.MAX_VALUE);
16     }
17 }
18
```

SRP



```
FileManager.java  Employer.java  FileDownloader.java  FileParser.java  DataPersist.jav

1 package com.kirilanastasov.srp;
2
3 import java.io.File;
4 import java.io.FileNotFoundException;
5 import java.util.Scanner;
6
7 public class FileParser {
8     public void parseTheFile(File file) throws FileNotFoundException {
9         Scanner input = new Scanner(file);
10        while (input.hasNext()) {
11            String nextToken = input.next();
12            String nextLine = input.nextLine();
13        }
14        input.close();
15    }
16 }
17
```

SRP

```
FileManager.java  Employer.java  FileDownloader.java  FileParser.java  DataPersist.java  ⌵
1  package com.kirilanastsov.srp;
2
3  import java.util.List;
4
5  public class DataPersist {
6      public void persistTheData(List<String> data) {
7          Employer entity = new Employer(100, "name");
8          entity.random = 1234;
9          Object gson;
10     }
11
12 }
13
```

SRP

A grayscale photograph of three students in a classroom. Two young men are on the left, one sitting on a stool and leaning over a desk, the other standing behind him. A young woman stands on the right, looking at a laptop on the desk. The background is a wall with a grid of small holes.

# Single Responsibility Lesson Summary

- A Class should have one and only one reason to change.
- SRP is one of the most commonly used principles in software development.
- It is easy to violate the Single Responsibility Principle.

# Course Progress

## Lesson 6

The Single  
Responsibility  
Principle

## Lesson 7

The Open  
Closed Principle

## Lesson 8

The Liskov  
Substitution  
Principle

## Lesson 9

The Interface  
Segregation  
Principle

## Lesson 10

The Dependency  
Inversion  
Principle





# The Single Responsibility Principle

- <https://blog.cleancoder.com/uncle-bob/2014/05/08/SingleResponsibilityPrinciple.html>



A sepia-toned photograph of a person clapping their hands. In the foreground, a wooden desk holds an open notebook with a smartphone resting on it. A laptop is partially visible in the background. A dark grey rectangular box with a thin white border is overlaid on the left side of the image, containing the text 'THANK YOU!' in a white serif font.

THANK YOU!