Classes

Andrejs Doronins
TEST AUTOMATION ENGINEER

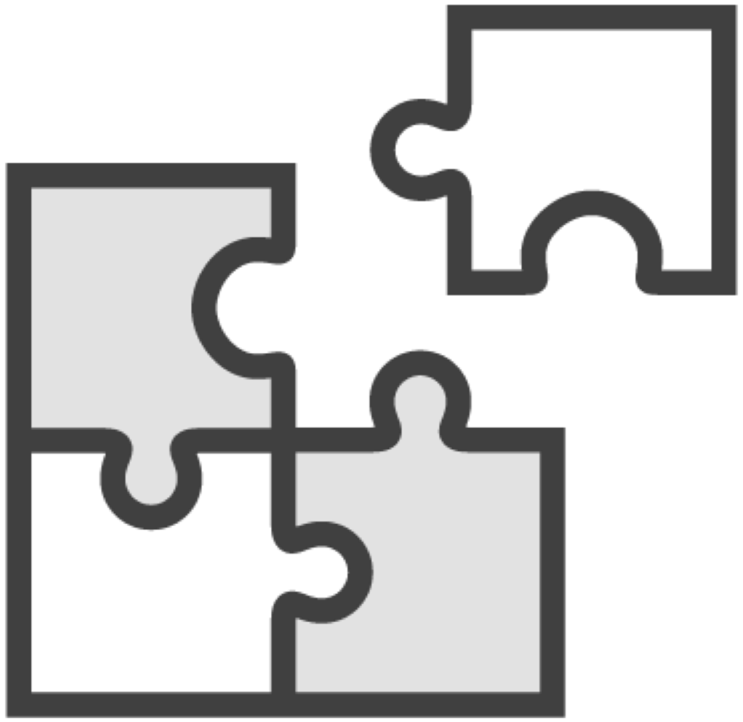
```java
class Person {

  String name;
  int age;
  PersonData data;

  Person(){
    //...
  }

  PersonData getPersonData(){
    return data;
  }

}
```



Module Overview

Revisit SRP

Cohesion & Coupling

Style Conventions

Principle of Proximity

SRP Applied to Classes

A class should do one thing ✗

A class should have only one reason to change ✓

SRP Applied

```
class SomeClass {
     String field1;
     String field2;

     void doThingA()
     void doThingB()
     void doUnrelatedThing()
}
```

```
class SomeClass {
     String field1;
     void doThingA()
     void doThingB()
}

class AnotherClass {
     String field2;
     void doUnrelatedThing()
}
```

```java
class Employee {

    String getName()
    double getSalary()
    Role getRole()

    void sendEmail()
    void calculateYearBonus()

}
```

```java
class Employee {
    // getter methods
}

class EmailService {
    void sendEmail()
}

class PayrollCalculator {
    void calculateYearBonus(Employee emp)
}
```

More on SRP

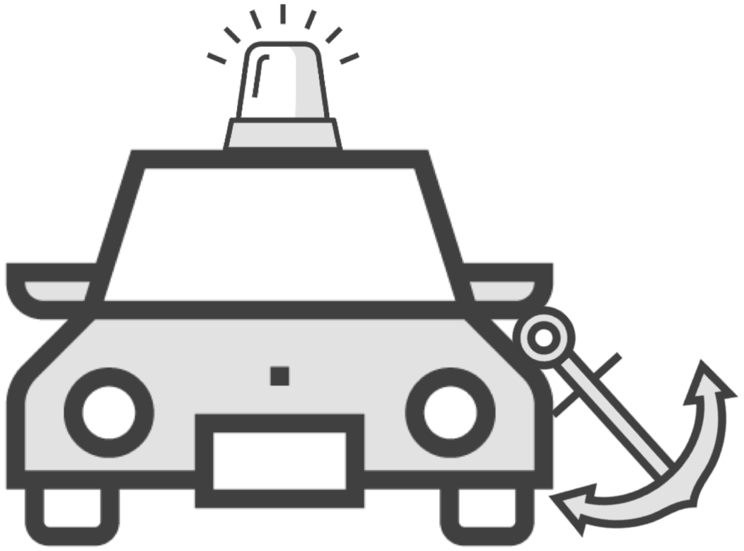
https://blog.cleancoder.com/uncle-bob/2014/05/08/SingleReponsibilityPrinciple.html

SRP leads to higher Cohesion

Cohesion

A tendency to unite

Cohesion



Cohesion (Programming)

Refers to the degree to which the elements inside a class or a module belong together.

Cohesive Class

Fields and methods are co-dependent

Methods that use multiple class fields indicate higher cohesion

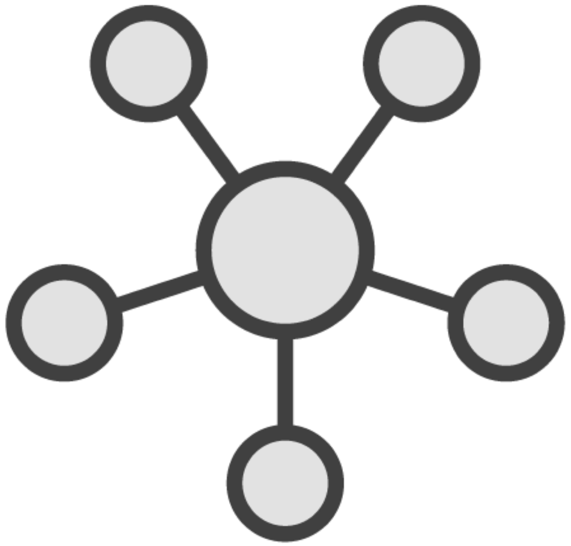
Methods use other methods inside the same class

SRP != Cohesion

Cohesive but Not SRP

```
User user;
void saveChanges(){
    dbContext.save();
    logger.log("User table updated with: " + whatever);
     raiseEvent(new EmailNotification(user));
}
void raiseEvent(Event event){
  // ...
};
```

Cohesion at Different Levels



Class

Package

Module

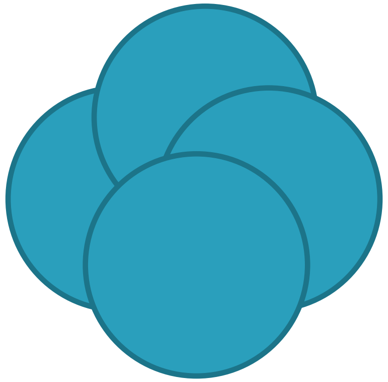
Systems

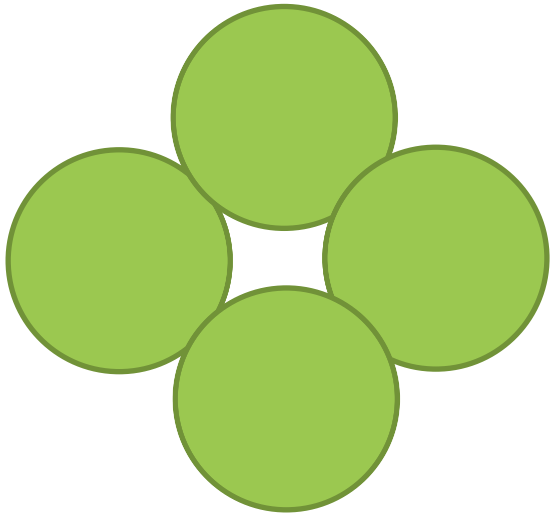
Coupling

The degree of interdependence between software modules or classes, a measure of how interconnected they are

Coupling

```
public class A {
    private B b = new B(); // A coupled with B
}
```





Tight Coupling

Classes are so tied, that you cannot change one without changing the other.

Loose Coupling

Change in one class requires no or minimum changes in other classes

Tight coupling is maintenance hell

To Reduce Coupling

Adhere to SRP

Increase Cohesion

Program to an Interface

Maintain strong Encapsulation

Use Dependency Injection

Stronger Encapsulation

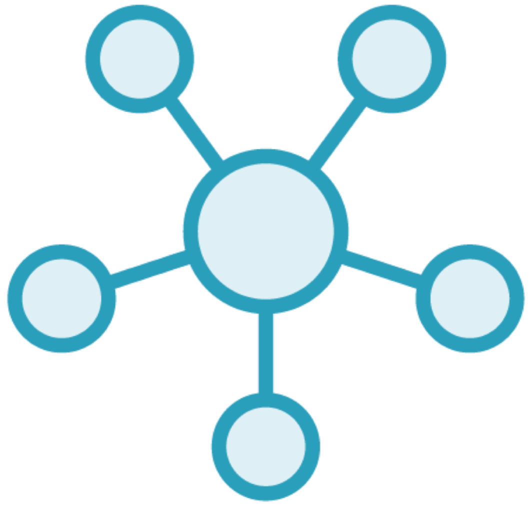
```
public class A {
    private String a; // for internal use only
    private String b;
    public void doSomething(){
        doAnotherThing();
    }
    private
    ~~public~~ void doAnotherThing(){ //...   }

    ~~public String getA() { return a;}~~
}
```

Don’t make methods and fields public “just because”

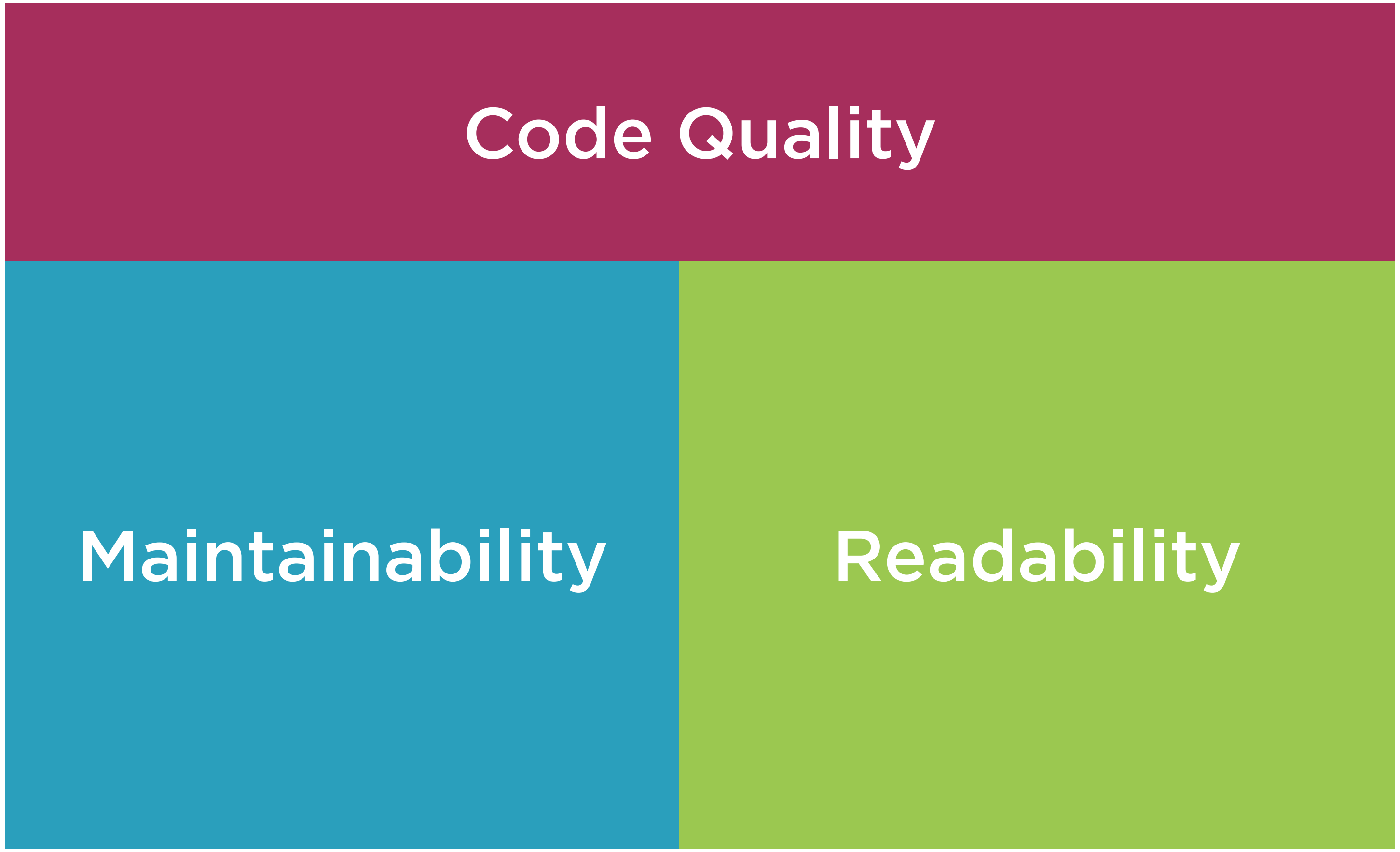
SRP

One reason to change



High Cohesion is good

Low Coupling is good


Code Quality
Maintainability
Readability

Code style matters

Without Punctuation

scrooge never painted out old marley's name there it stood years afterwards above the warehouse door scrooge and marley the firm was known as scrooge and marley sometimes people new to the business called scrooge scrooge and sometimes marley but he answered to both names it was all the same to him

With Punctuation

Scrooge never painted out Old Marley's name.

There it stood, years afterwards, above the warehouse door: Scrooge and Marley. The firm was known as Scrooge and Marley. Sometimes people new to the business called Scrooge Scrooge, and sometimes Marley, but he answered to both names.

It was all the same to him.

"[Twitter Java style guide] is the distillation of many combined man-years of software engineering and Java development experience"

Twitter

Java Style Conventions

https://google.github.io/styleguide/javaguide.html

https://github.com/twitter/commons/blob/master/src/java/com/twitter/common/styleguide.md

https://www.oracle.com/technetwork/java/codeconvtoc-136057.html

Principle of Proximity

SRP
Open Closed Principle
Liskov Substitution Principle
Interface Segregation Principle
Dependency Inversion

Summary

✓ SRP, Cohesion and Coupling

✓ Style conventions

✓ Principle of Proximity