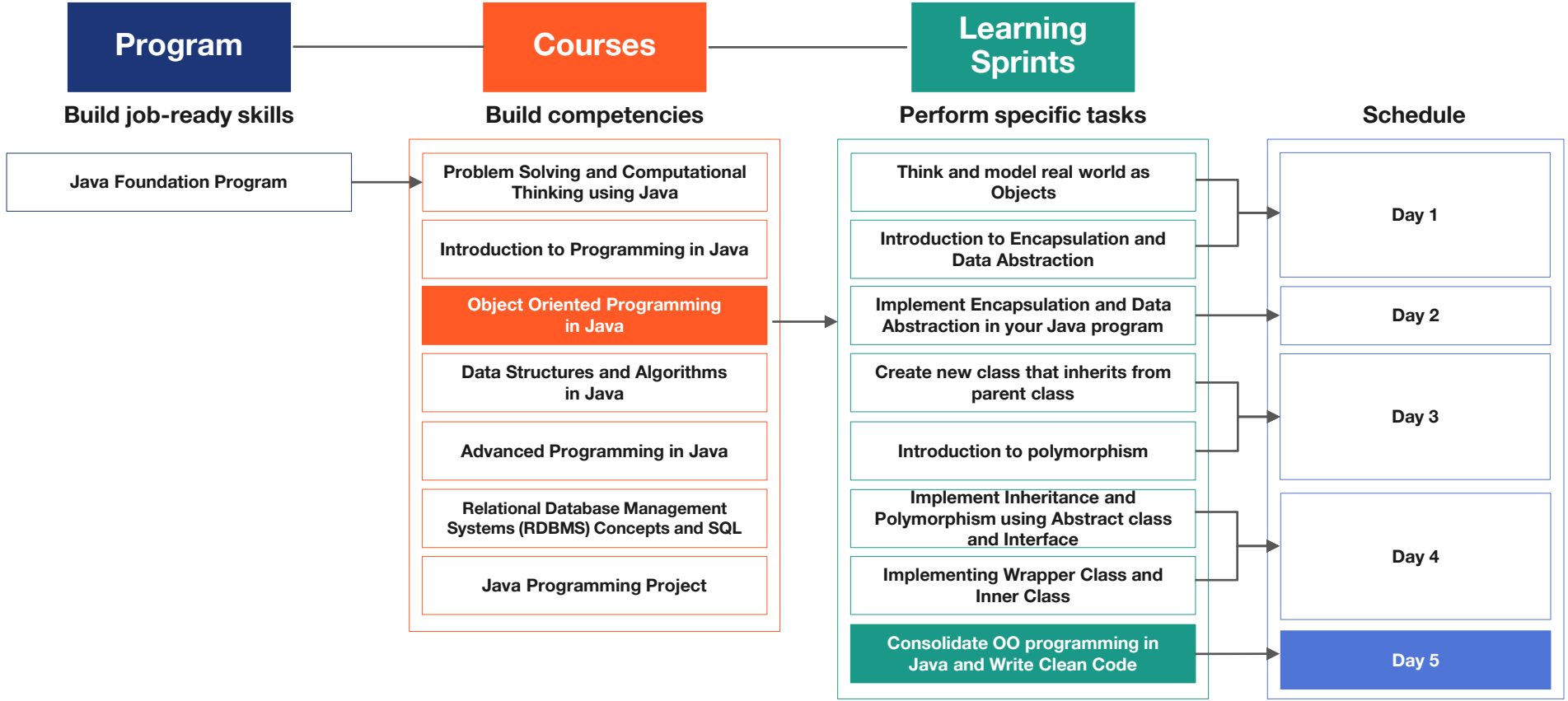
Java Program: Course 3: Plan



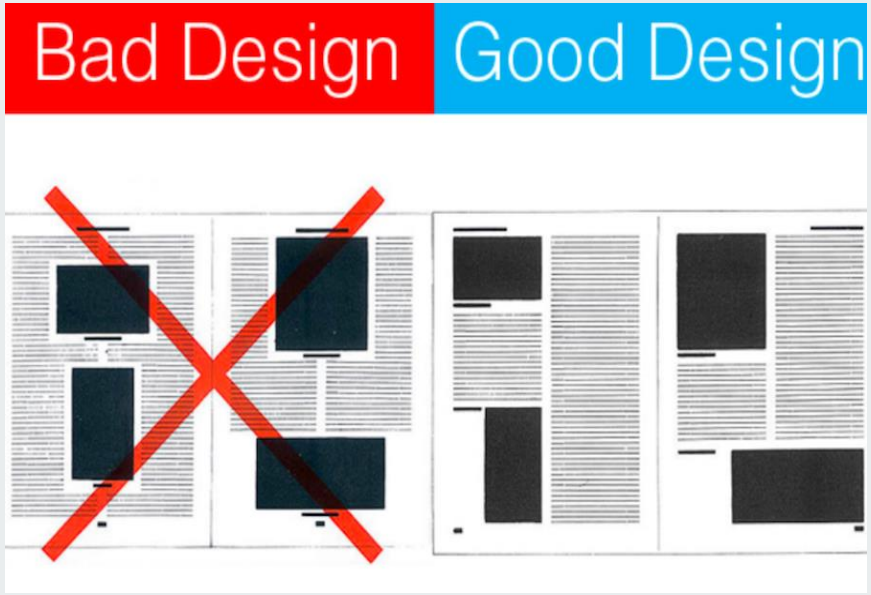
Think and Tell

What is software design?

Why is it important to have a good design?

https://mozaicworks.com/blog/question-software-design-part-5-big-projects/

Software Design



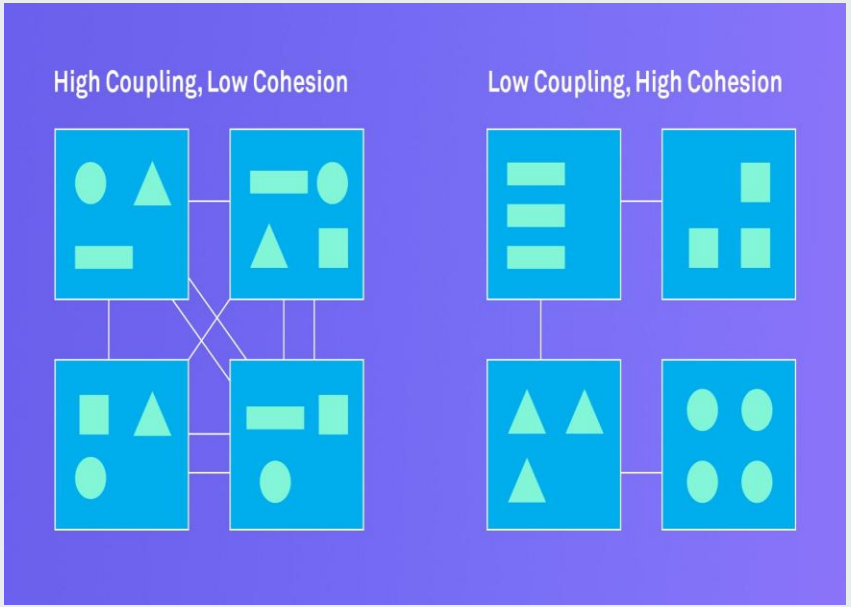
https://medium.com/mockplus/6-bad-ui-design-examples-common-errors-of-ui-designers-550d90768252

Look at the two adjacent images.

Why is the design of the second image considered better?

What are the criteria for good design?

Software Design



Can you list some of the characteristics of good design in software programming?

Writing a Clean Code

- How can we write clean code in Java?
- How can we ensure that the software is easy to test, change and add features?

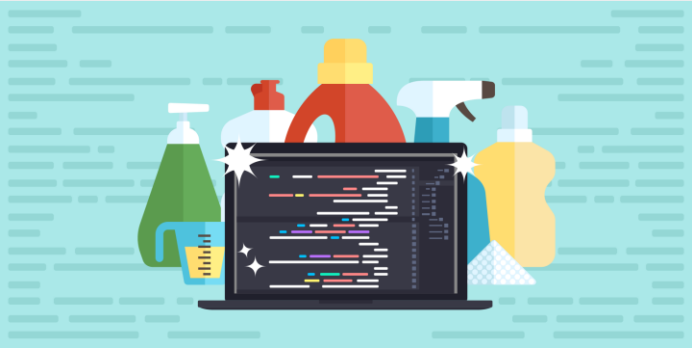
Writing Clean Code

Learning Objectives

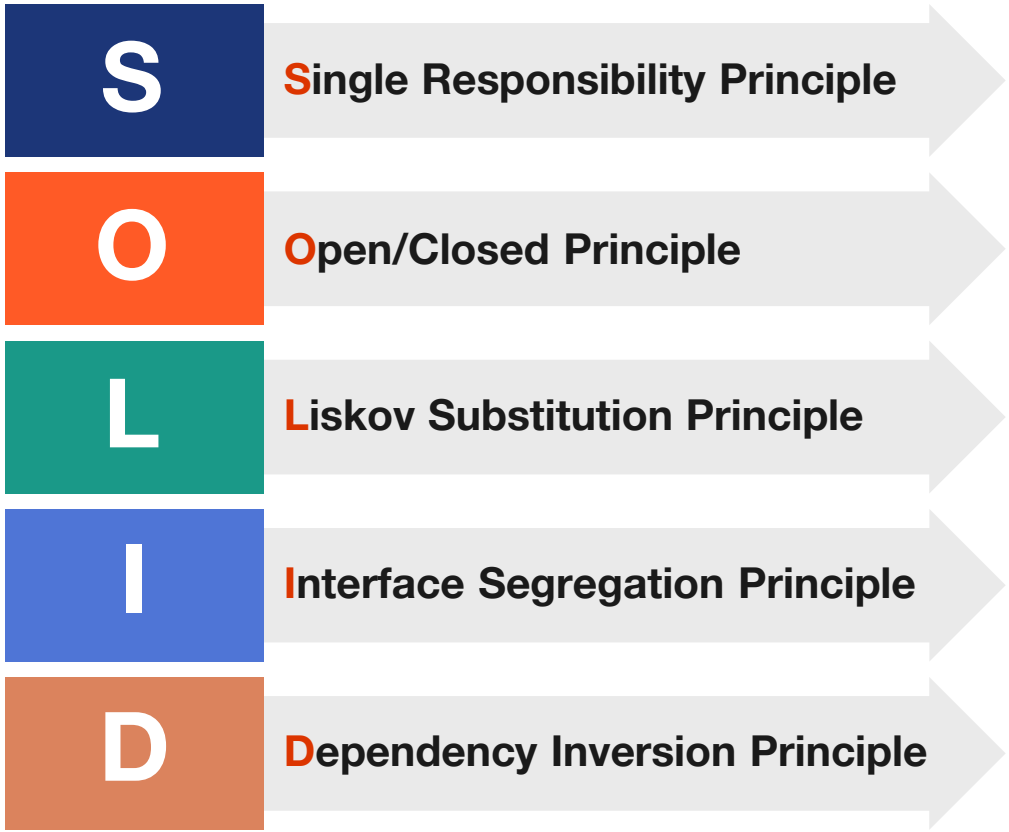
- Define SOLID
- List the five principles of SOLID
- Explain the purpose of SOLID principles
- Demonstrate the use of the five principles

What is SOLID?

What is SOLID in object-oriented computer programming?



The SOLID Principles



Single Responsibility Principle

https://devscopeninjas.azurewebsites.net/2017/04/28/solid-principles/

“A class should have one and only one reason to change.”

Interactive Demo

Demonstrate the Single Responsibility Principle.

The sample code mentioned below is placed at the repo url. You can either use this or any other code of your choice for the demo:

https://myrepos.stackroute.niit.com/core-java-contents-for-mentors/course-demo/solid-principle-demo-code.git

Open/Closed Principle

https://devscopeninjas.azurewebsites.net/2017/04/28/solid-principles/

“Software entities should be open for extension, but closed for modification.”

Interactive Demo

Demonstrate the Open/Closed Principle.

The sample code mentioned below is placed at the repo url. You can either use this or any other code of your choice for the demo:

https://myrepos.stackroute.niit.com/core-java-contents-for-mentors/course-demo/solid-principle-demo-code.git

Liskov Substitution Principle

Let q(x) be a property provable about object x of type T.

Then q(y) should be provable for object y of type S where S is a subtype of T.

A subclass should behave in such a way that it will not cause problems when used instead of the superclass.

Liskov Substitution Principle (contd.)

Source: https://devscopeninjas.azurewebsites.net/2017/04/28/solid-principles/

Interactive Demo

Demonstrate the Liskov Substitution Principle.

The sample code mentioned below is placed at the repo url. You can either use this or any other code of your choice for the demo:

https://myrepos.stackroute.niit.com/core-java-contents-for-mentors/course-demo/solid-principle-demo-code.git

Interface Segregation Principle

Source: https://devscopeninjas.azurewebsites.net/2017/04/28/solid-principles/

"Clients should not be forced to depend upon interfaces that they don't use."

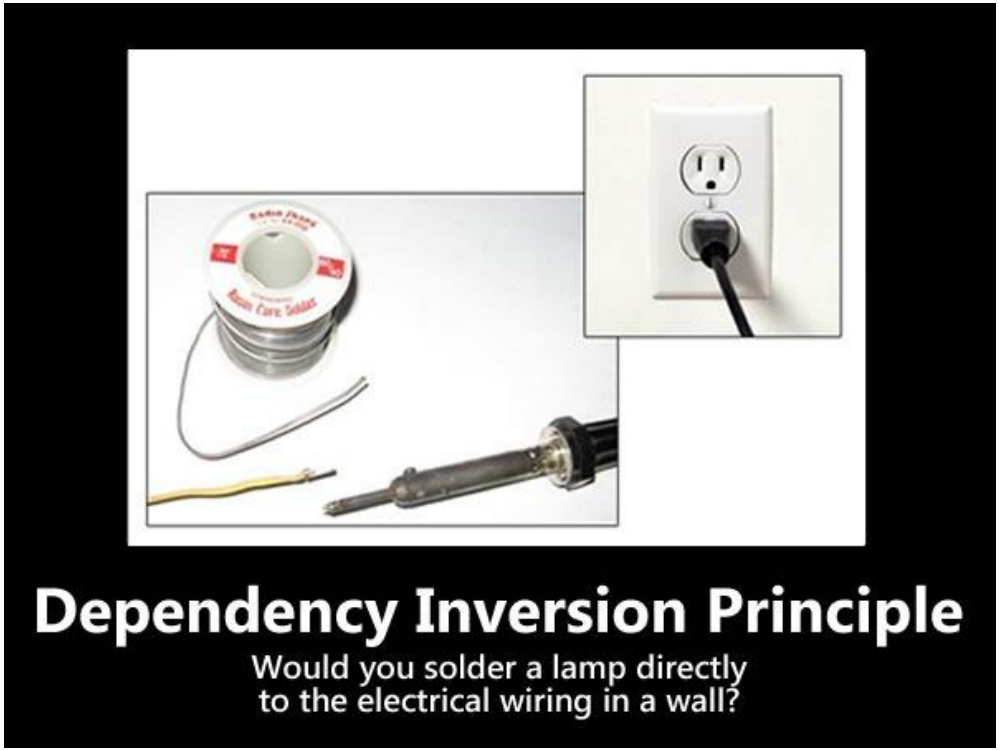
Interactive Demo

Demonstrate the Interface Segregation Principle.

The sample code mentioned below is placed at the repo url. You can either use this or any other code of your choice for the demo:

https://myrepos.stackroute.niit.com/core-java-contents-for-mentors/course-demo/solid-principle-demo-code.git

Dependency Inverse Principle



Source: https://devscopeninjas.azurewebsites.net/2017/04/28/solid-principles/

Dependency Inverse Principle (contd.)

“High-level modules should not depend on low-level modules.

Both should depend on abstractions.”

“Abstractions should not depend upon details. Details should depend

upon abstractions.”

Interactive Demo

Demonstrate the Dependency Inverse Principle.

The sample code mentioned below is placed at the repo url. You can either use this or any other code of your choice for the demo:

https://myrepos.stackroute.niit.com/core-java-contents-for-mentors/course-demo/solid-principle-demo-code.git

Key Takeaways

- Single Responsibility Principle “S”
- Open/Closed Principle “O”
- Liskov Substitution Principle “L”
- Interface Segregation Principle “I”
- Dependency Inversion Principle “D”