COMPONENTS

Java Clean code

Outline

Lesson 11

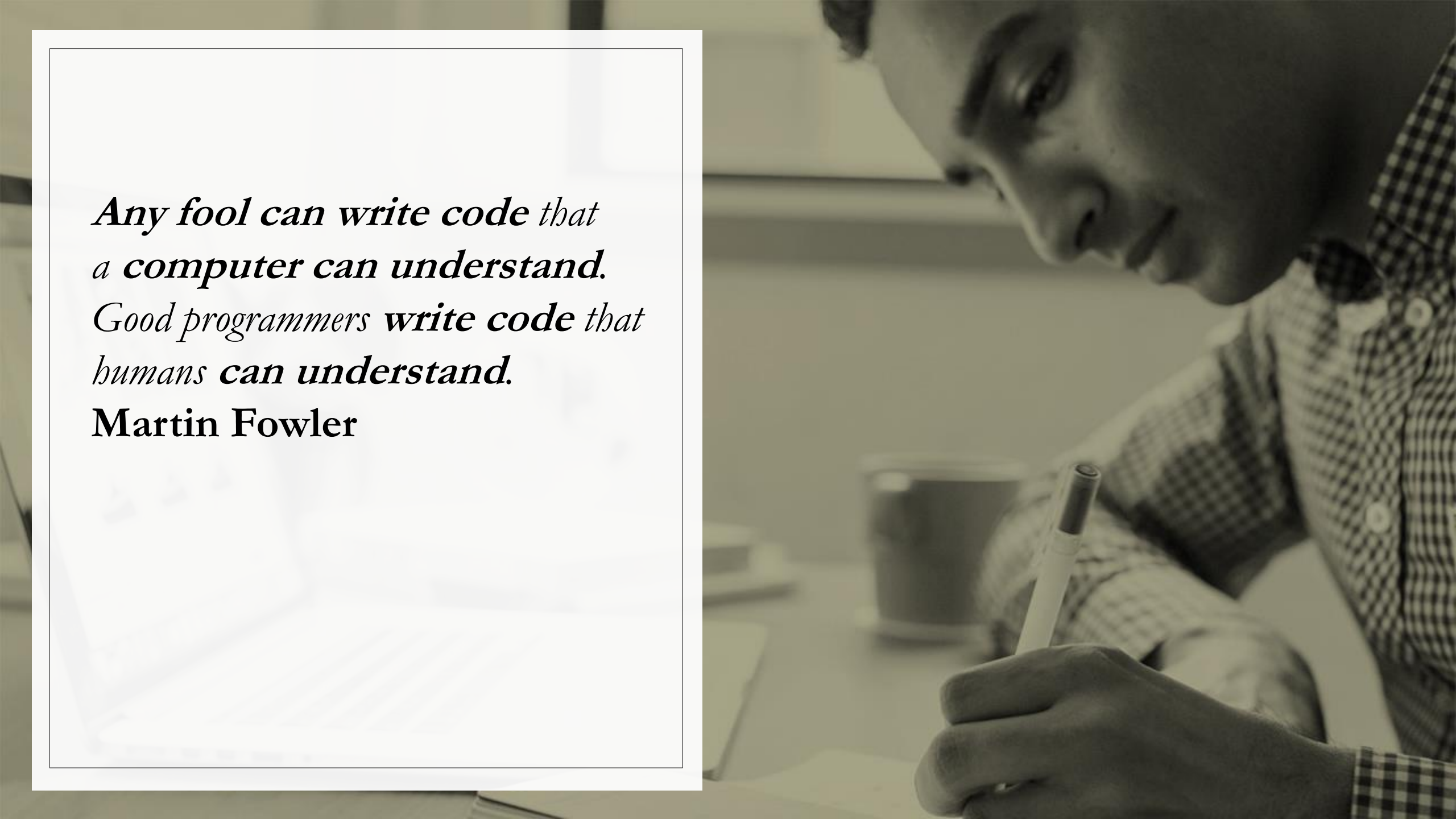
Components

Lesson 12.

Architecture

Lesson 13.

Test Driven Development



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

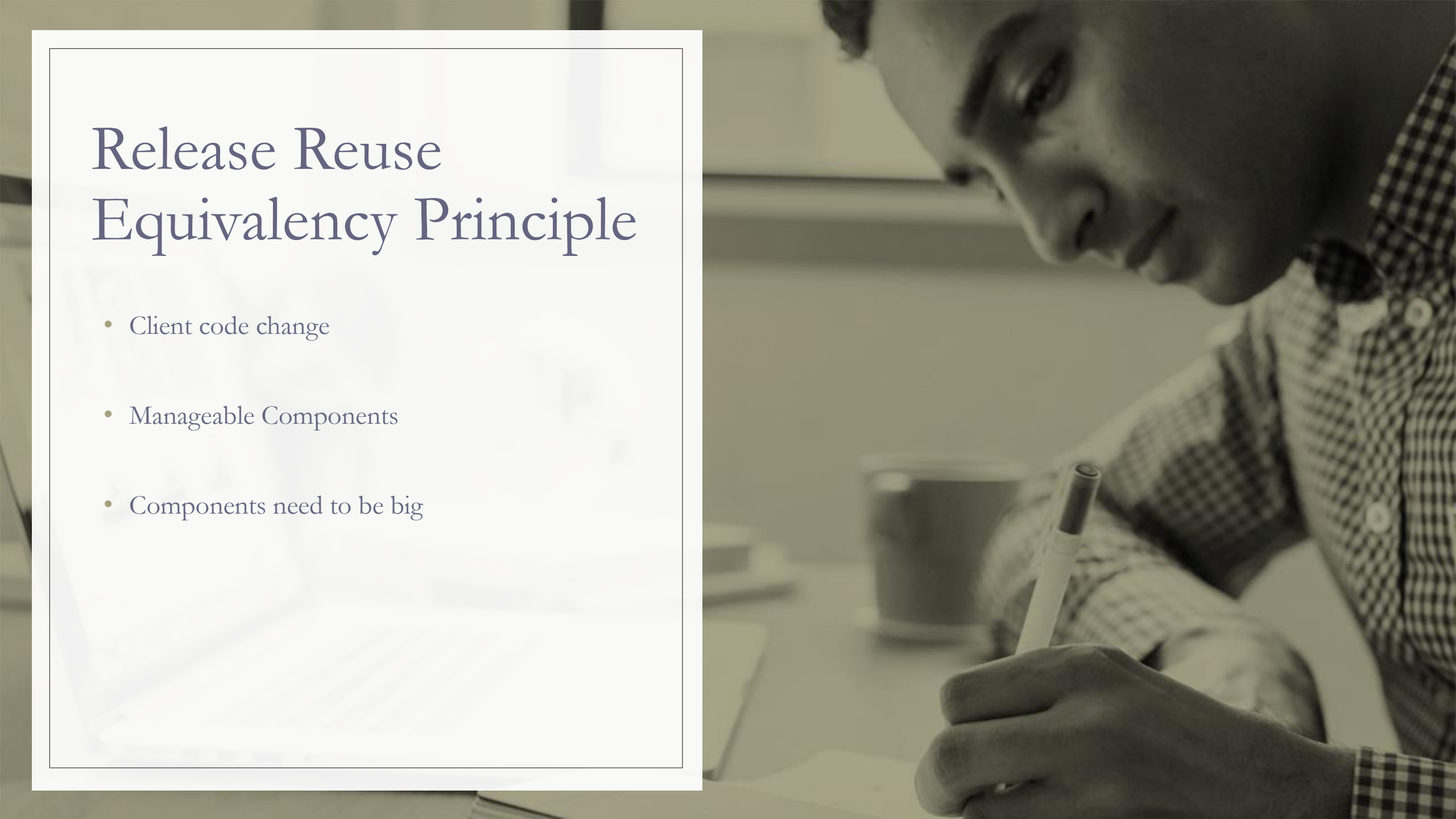
Martin Fowler

Components

- Jar, DLL: dynamic link libraries
- Loose coupling
- Consists of Classes
- Independent deployment & development

Release Reuse Equivalency Principle

- Client code change
- Manageable Components
- Components need to be big



The Common closure principle

- Same reason to change
- Isolate components
- Loose coupling

The Common reuse principle

- Similar to the Interface Segregation Principle
- Group together classes
- Separate classes.

Component coupling

- The Acyclic Dependencies Principle
- The Stable Dependencies Principle
- The Stable Abstraction Principle

The Acyclic Dependencies principle

- Downward dependencies
- It worked yesterday
- Weekly build

The Stable Dependencies principle

- Components that change should not be dependent on difficult to change components
- Components that are heavily dependent should be very stable
- Components that rely on many other components are highly volatile

The Stable Abstraction principle

- Components should be as abstract as it is stable.
- Dependencies run in the direction of abstraction
- Stable == interfaces and abstract classes, Instable == concrete

Components Summary

- Classes that change for the same reason should be in the same component.
- Classes that change for different reason should be in separate Use names that are
- The dependencies should be directed downwards not in circle
- A component should only depend on components that are more stable than he is.

Course Progress

Lesson 11	Lesson 12	Lesson 13
Components	Architecture	Test Driven Development


THANK YOU!