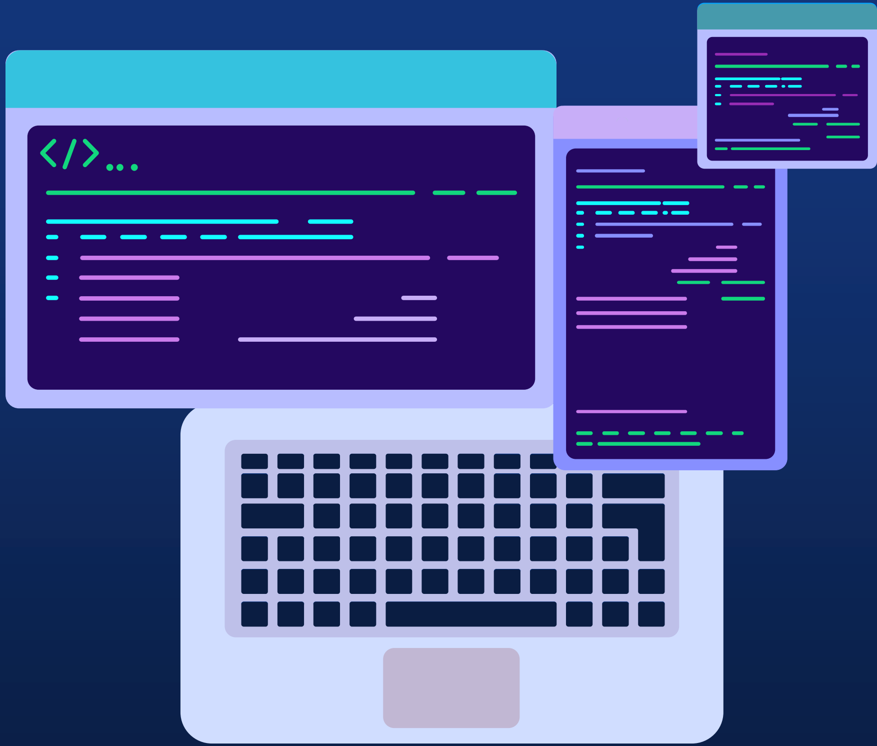
Software Design and Important concepts



Mentor: Einar Rocha

CONTENT

✓ 01
OOP Pillars

Inheritance, Polymorphism
Encapsulation, Abstraction

✓ 02
Clean Code

Meaningful Names,
Functions, Unit test
Code Smells…

03
SOLID

Single Responsiblity
Open closed
Liskov Substitution
Interface Segregation
Dependency Inversion

04
Design patterns

Singleton, Factory Method
Strategy, Observer
Builder…

03

SOLID

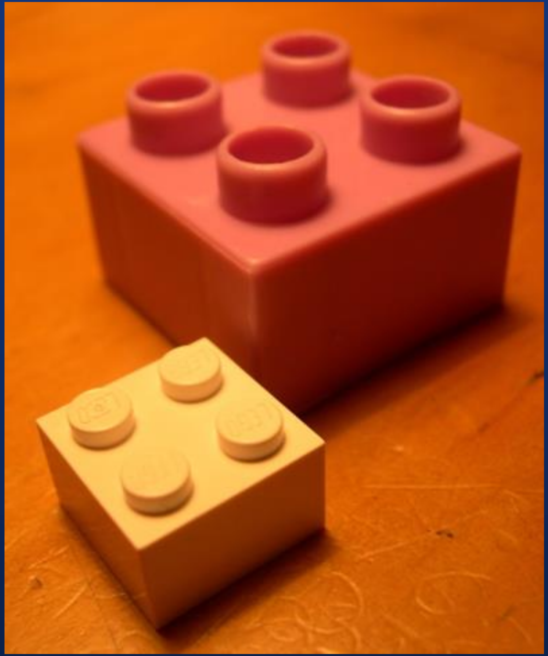
Purpose

- ✓ To create understandable, readable, and testable code that many developers can collaboratively work on.

Agenda

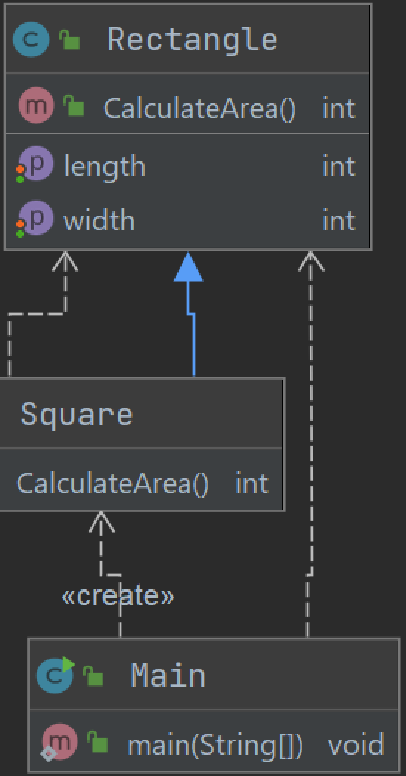
- Liskov Substitution Principle — Design by contract…
- Interface Segregation Principle

Liskov Substitution Principle

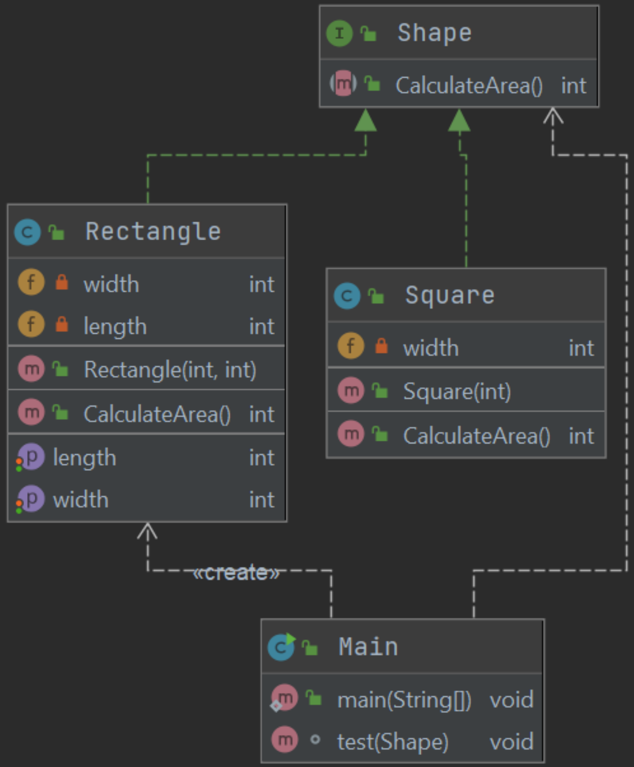


Subtypes must be substitutable for their base types.

Refactor


Rectangle
CalculateArea() int
length int
width int
Square
CalculateArea() int
«create»
Main
main(String[]) void

Refactor



Shape
CalculateArea() int
Rectangle
width int
length int
Rectangle(int, int)
CalculateArea() int
length int
width int
Square
width int
Square(int)
CalculateArea() int
«create»
Main
main(String[]) void
test(Shape) void

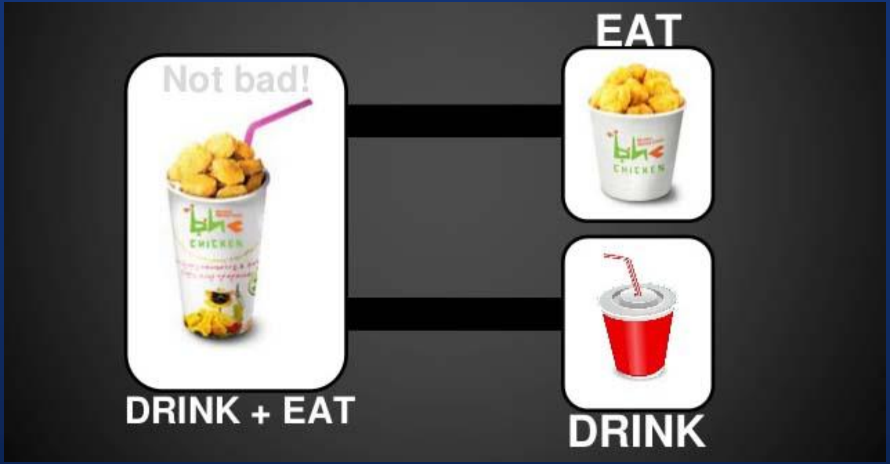
Heuristics - A degenerate function in a derivative

```
class Base {
    public void f() {/*some code*/}
}

class Derived extends Base {

    @Override
    public void f() {
    }
}
```

Heuristics – Specific type on declaration side

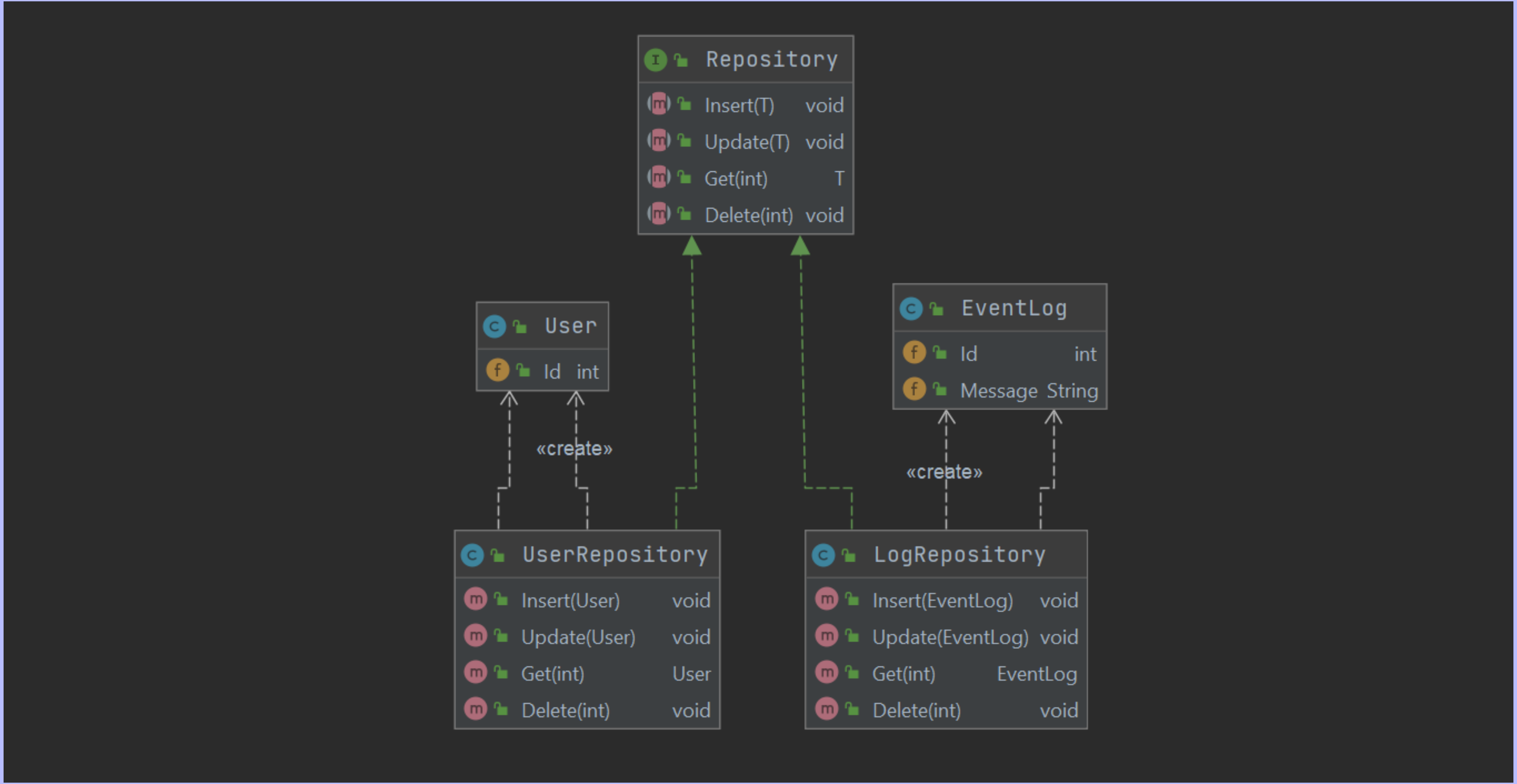
```
public class Square extends Rectangle {

    @Override
    public int CalculateArea() {
        return getWidth() * getWidth();
    }
}
…………………………



public static void main(String[] args) {
    Square square = new Square();
}
```

Interface Segregation Principle

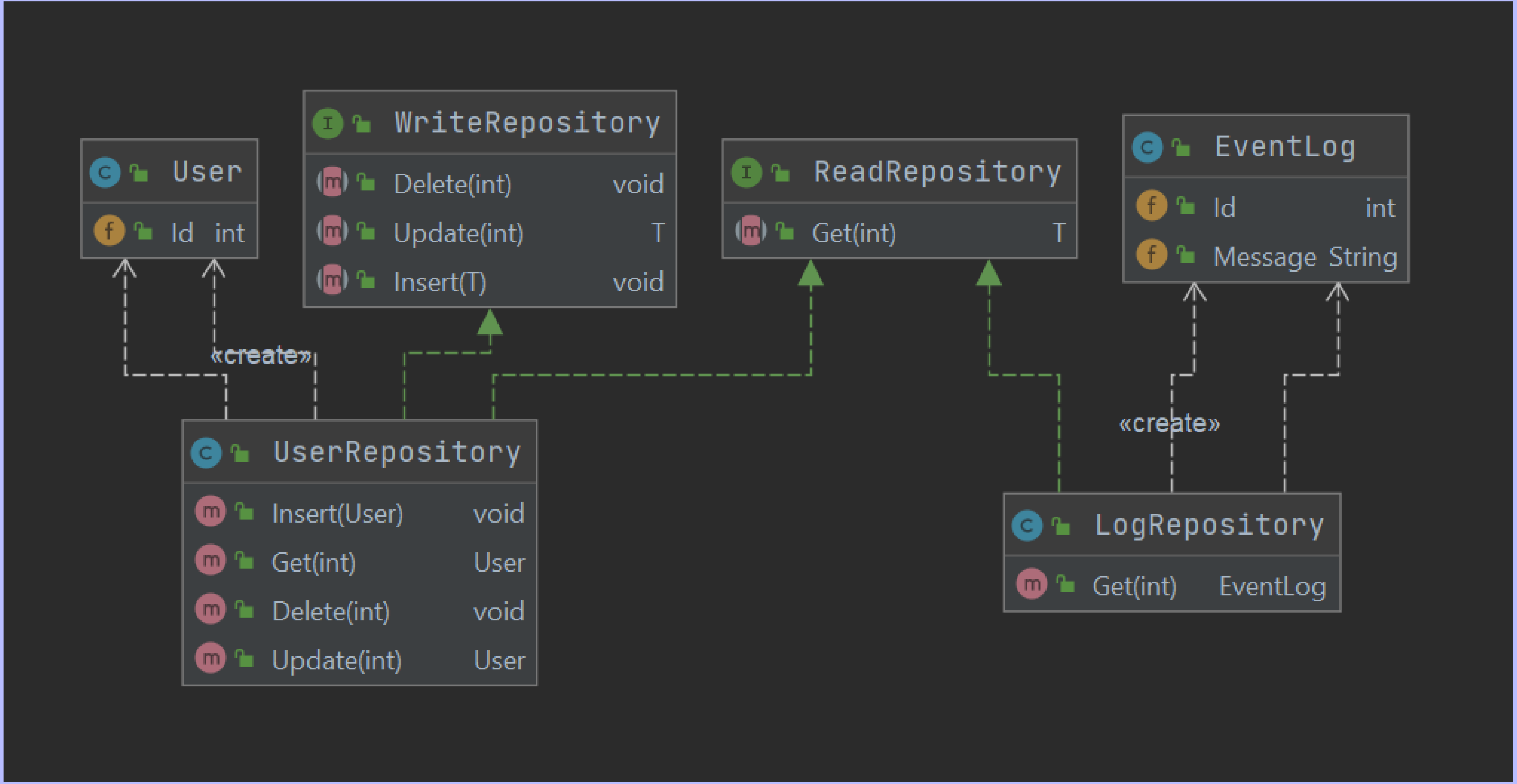


Clients should not be forced to depend on methods they do not use.

Refactor


Repository
Insert(T) void
Update(T) void
Get(int) T
Delete(int) void
User
Id int
EventLog
Id int
Message String
«create»
«create»
UserRepository
Insert(User) void
Update(User) void
Get(int) User
Delete(int) void
LogRepository
Insert(EventLog) void
Update(EventLog) void
Get(int) EventLog
Delete(int) void

Refactor


User
Id int
WriteRepository
Delete(int) void
Update(int) T
Insert(T) void
ReadRepository
Get(int) T
EventLog
Id int
Message String
«create»
UserRepository
Insert(User) void
Get(int) User
Delete(int) void
Update(int) User
«create»
LogRepository
Get(int) EventLog