Achieving Good Object-Oriented Design

by Willem van Straten

Good design is often described in terms of design goals

Developers must learn **how** to achieve good object-oriented design

A → B

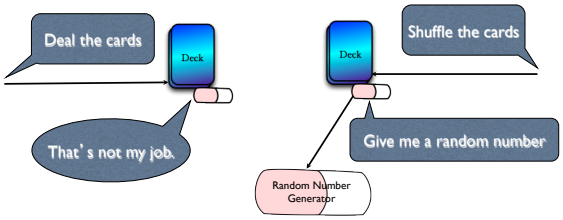
It is not enough to know the desired characteristics of the end product

Developers need principles
- or rules of thumb -
to guide design decisions

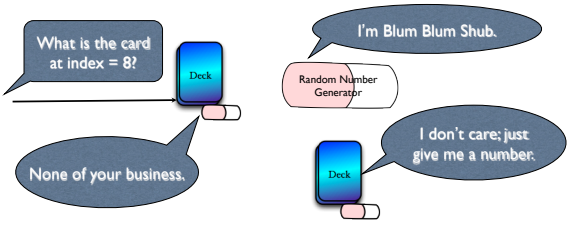
Adopt a small set of simple rules to achieve good object-oriented design

Remember these three simple rules

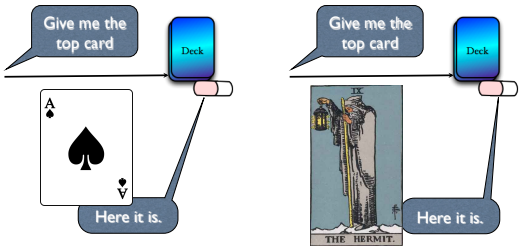
Classes should be lazy



Classes should be antisocial



Derived classes should be conformist



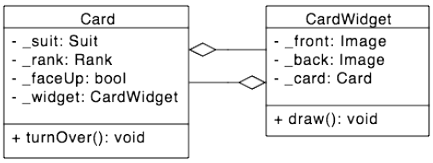
Three simple rules help to evaluate object-oriented designs

Is this class sufficiently lazy?

Card
- _suit: Suit - _rank: Rank - _faceUp: bool - _front: Image - _back: Image
+ turnOver(): void + draw(): void

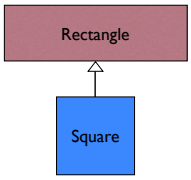
Classes should play a single role

Are these classes sufficiently antisocial?



Classes should collaborate via abstractions

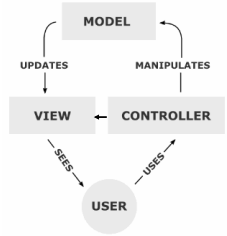
Is this derived class sufficiently conformist?



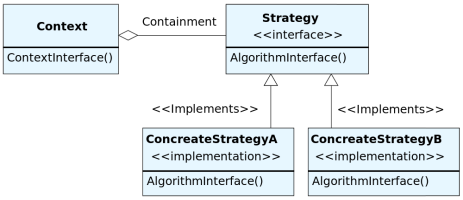
Use inheritance and polymorphism wisely

Three simple rules guide the application of OOP principles and paradigms

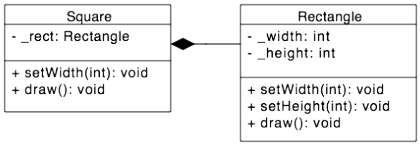
Laziness motivates Separation of Concerns



Unsociability promotes the use of Abstractions and Interfaces



Conformity guides the use of Inheritance and Polymorphism



Will you be able to apply these rules to achieve better object-oriented designs?

It is difficult to write good software without some practical guidelines

Adopt a small set of simple rules to achieve good object-oriented design

Classes should be lazy, antisocial, and conformist

Good design leads to less work

This Week's Tasks

Pass Task 14: Case Study