COM S 362
Object-Oriented Analysis & Design

GRASP

Reading

Craig Larman. Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development, Third Edition, 2004.

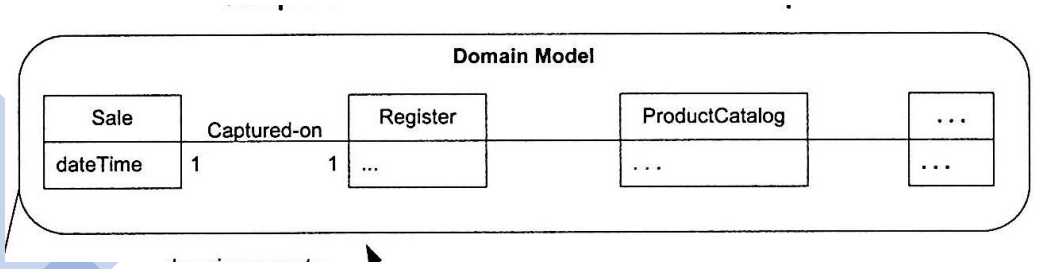
- In Ch. 17 read “A Short Example of Object Design with GRASP”, pp. 281-291

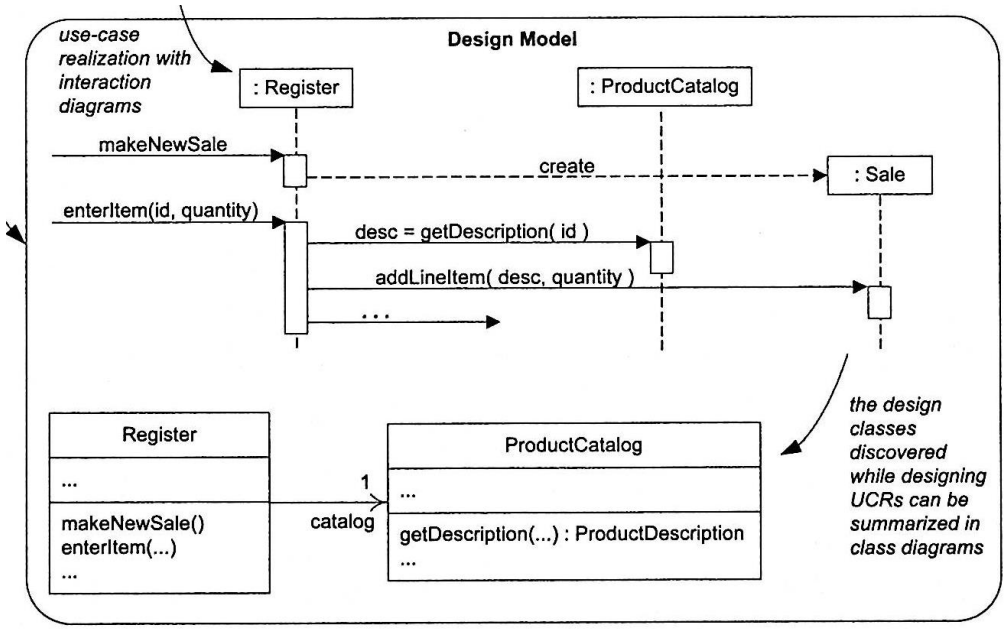
GRASP

- GRASP: General Responsibility Assignment Software Patterns
  - Patterns/principles for assigning responsibilities to software classes
  - A responsibility-driven design aid
  - Helps understand object-oriented design
  - Domain model is inspiration for design model software objects

Recap

- The object design process focuses on identifying software classes and assigning responsibilities
- The process builds on previous analysis
  - Use Case
  - Domain Model





Responsibilities

- Two kinds of responsibilities
  - Doing
  - Knowing
- Doing
  - Doing something itself, such as object creation or a calculation
  - Initiating action in other objects
  - Controlling and coordinating actives in other objects
- Knowing
  - Knowing about private encapsulated data
  - Knowing about related objects
  - Knowing about things it can derive or calculate

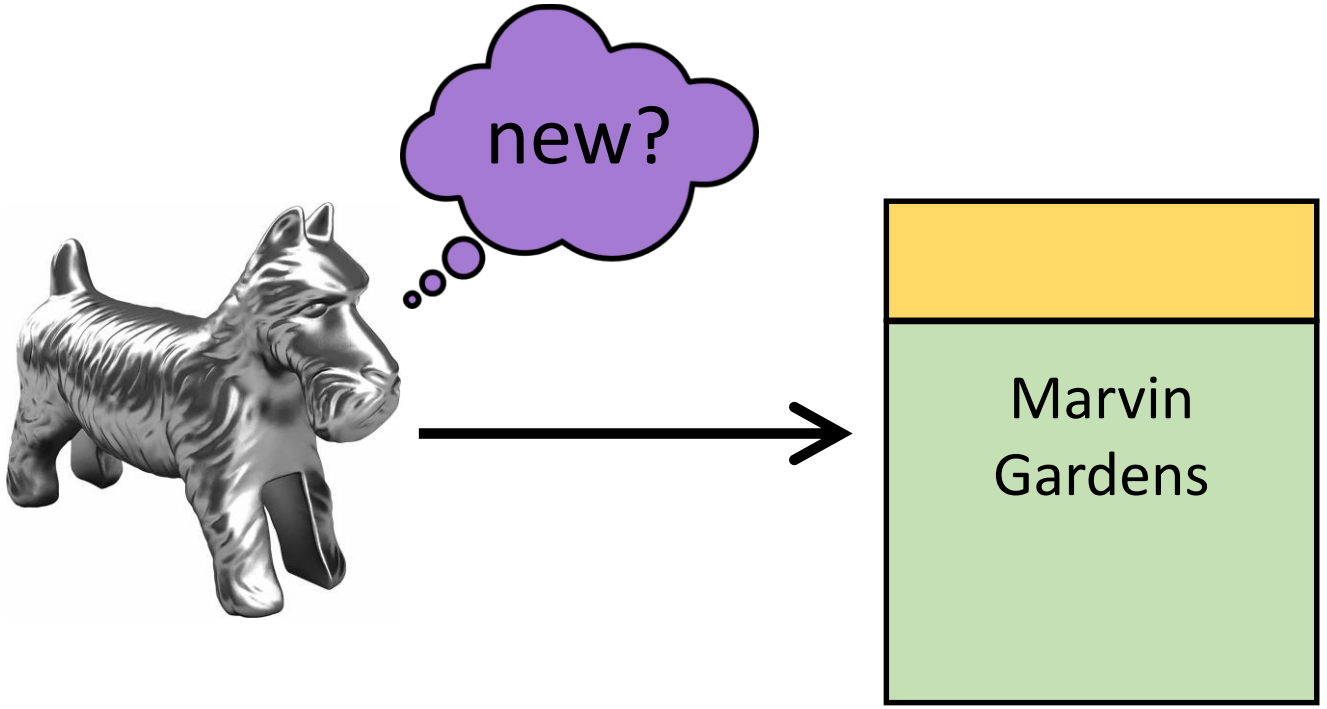
GRASP Patterns/Principles

- 5 main patterns/principles
  - Creator
  - Information Expert
  - Low Coupling
  - Controller
  - High Cohesion

Creator

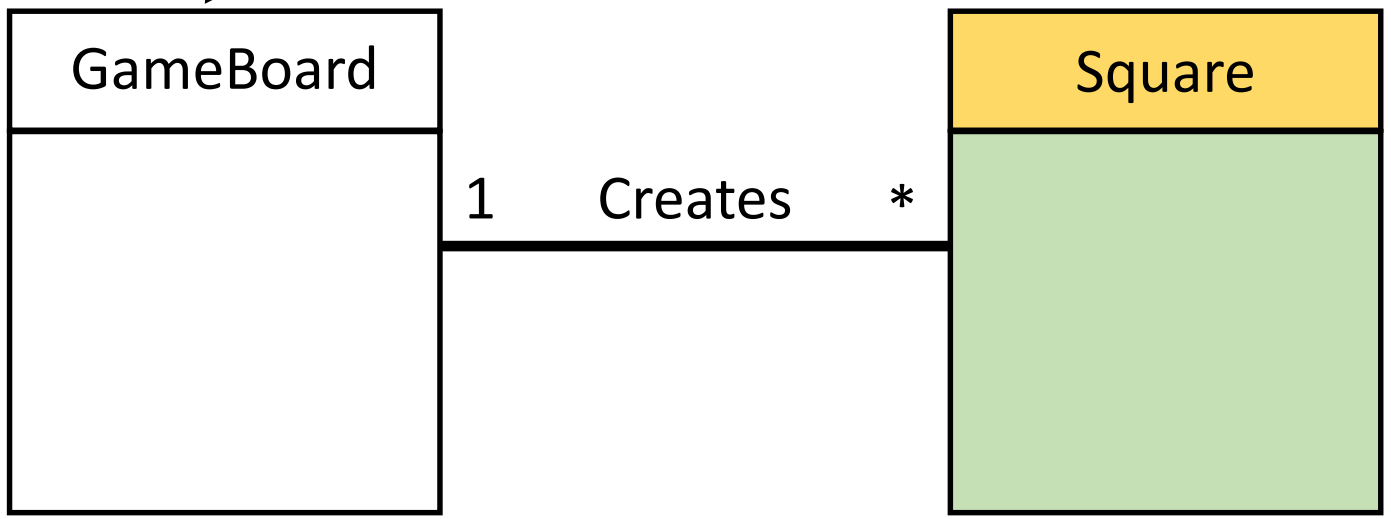
- Problem: Who creates X?
- Solution: Class B is a good candidate for the responsibility of creating A if one or more of these is true:
  - B contains or aggregates A
  - B has the initializing data for A
  - B records A
  - B closely uses A

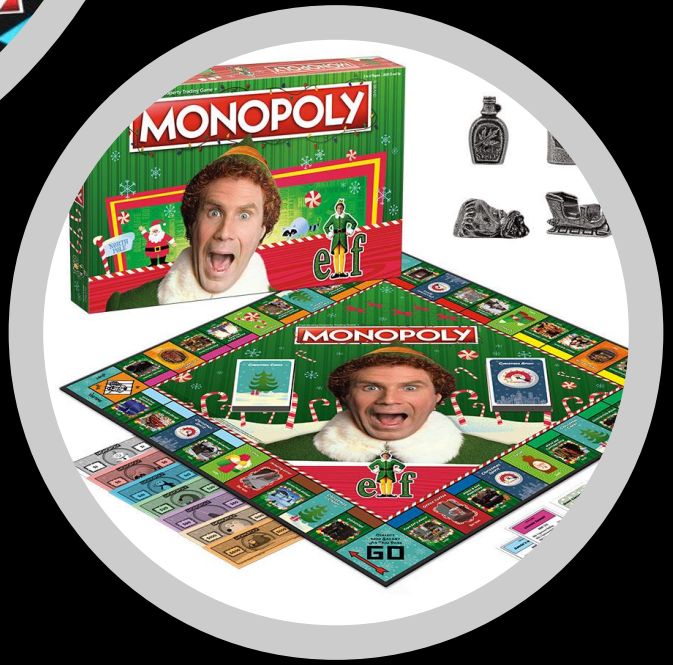
Who calls new?



Who calls new?

GameBoard aggregates (is composed of) Square objects, so it is a natural creator



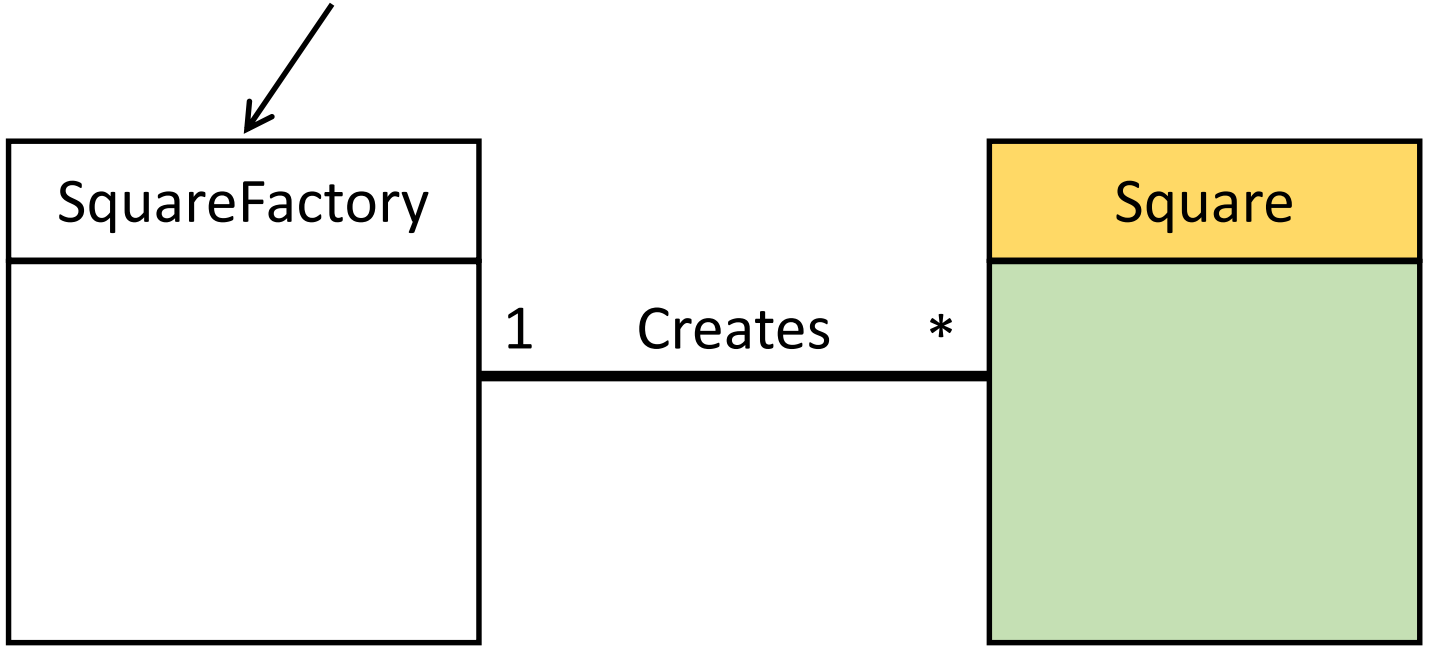


New Requirement

- Need multiple versions of games, each with slightly different rules and theme

Who calls new?

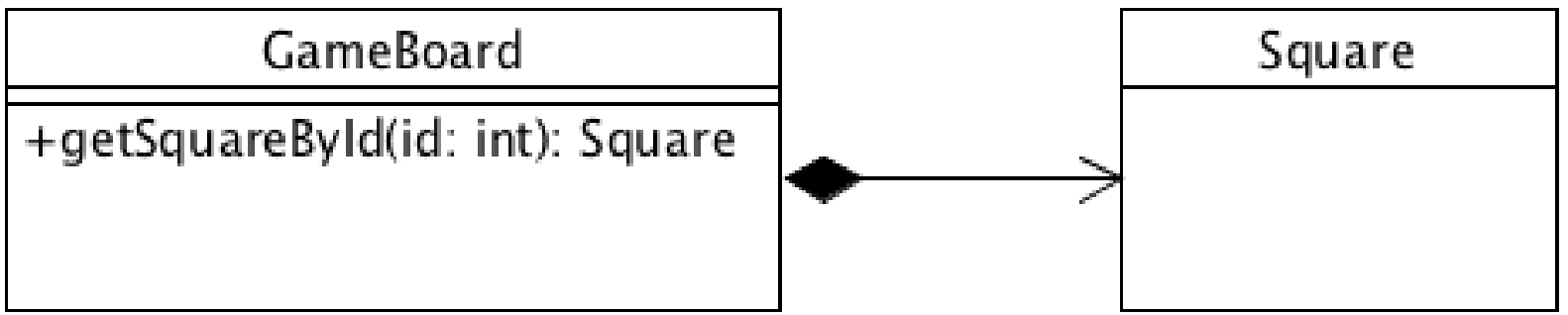
We will explore alternative creators that decouple the responsibility of creation.



Information Expert

- Problem: What is a basic principle by which to assign responsibilities to object?
- Solution: Assign a responsibility to the class that has the information needed to fulfill it.

GameBoard aggregates (is composed of) Square objects, so it is a natural location for lookup by id responsibility



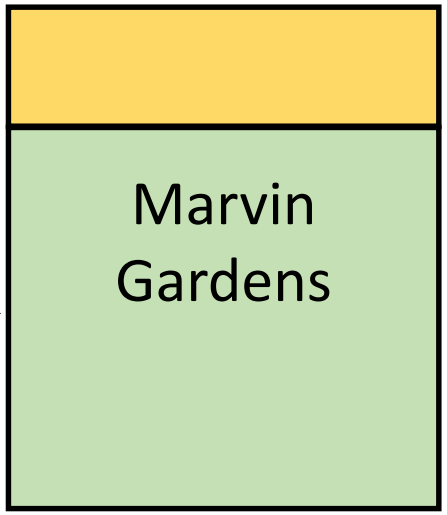
Controller

- Question: What is the first object beyond the UI layer that receives and coordinates a system event?
- System event: event generated by external actor (e.g., actor selects “end sale”)
- Maintaining independence (decoupling) between UI and application control is a basic design pattern (Model-View-Controller).
- Assign the controller responsibility to either:
  - A class representing the overall system, device, or subsystem (called a *façade controller*)
  - A class that represents a use case scenario within which the system event occurs (called a *use case controller*)

Low Coupling

- Problem: How to reduce the impact of change?
- Solution: Assign responsibilities that keep (unnecessary) coupling low
- Coupling is how strongly one element is connected to, has knowledge of, or depends on other elements

Don’t make Dog know about PropertySquare, if we do, Dog is coupled, a change to PropertySquare can impact Dog.



High Cohesion

- Problem: How to keep objects focused, understandable and manageable?
- Solution: Assign responsibilities so that cohesion remains high.
- Cohesion measures how functionally related the operation of a software element are to each other
- Bad cohesion and bad coupling often go hand in hand