SWEN 225
Software Design

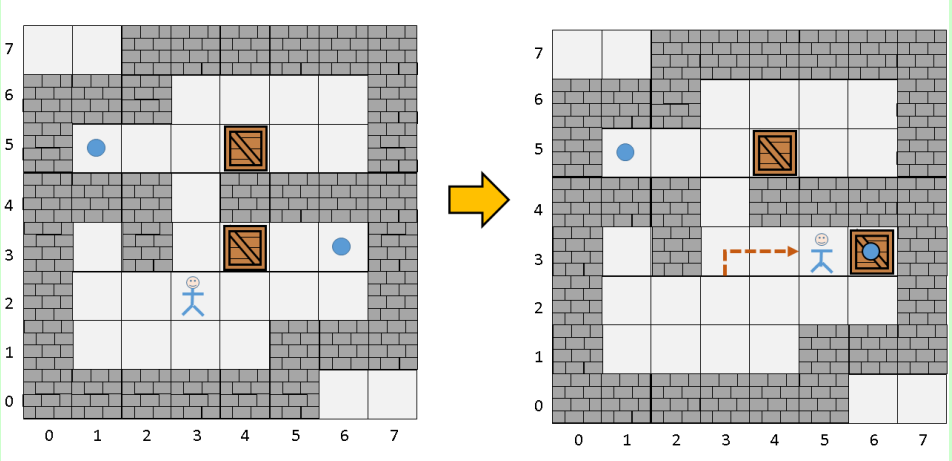
OO Design

Thomas Kühne
Victoria University of Wellington
Thomas.Kuehne@ecs.vuw.ac.nz, Ext. 5443, Room Cotton 233

Textual Analysis

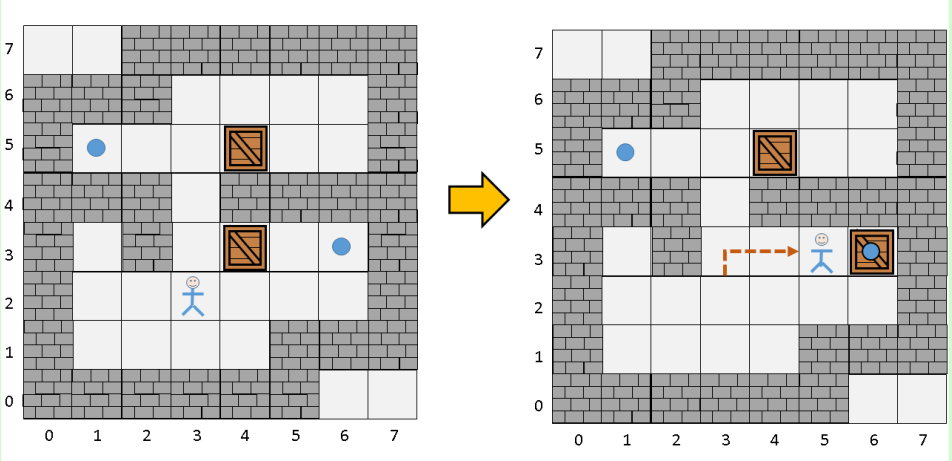
1. Read & understand documents
2. Underline **noun** and **verb phrases**
   - » convert to singular
3. Sort into three categories
   - (a) Obvious *things* and *actions*
   - (b) Unsure (part thing? Potential action?)
   - (c) Obviously *irrelevant*
4. Work through (a), (b) to produce ***candidates***
   - » Consolidate *things*, *actions*, and naming

Example



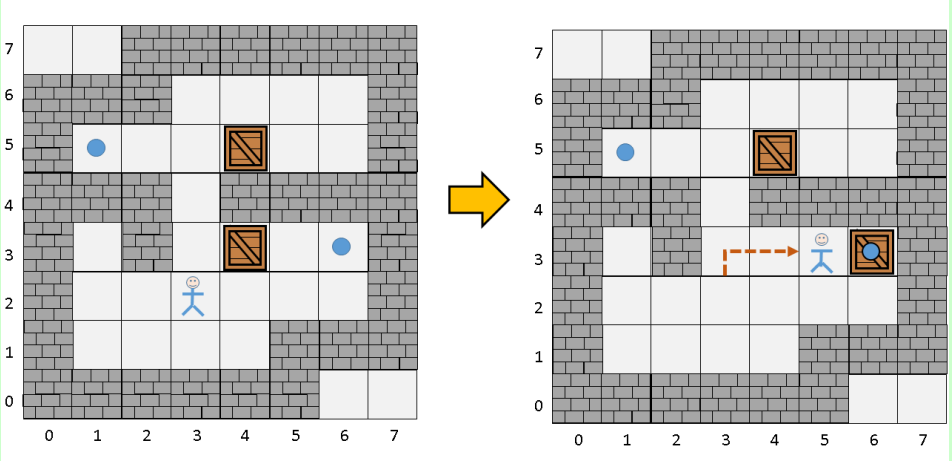
This program implements the Sokoban computer game. In the game, the worker moves around the board pushing crates into storage locations. When every crate is placed in a storage location, the game is over. The worker cannot move through crates or walls and cannot push more than one crate at a time.

Example – Nouns & Verbs



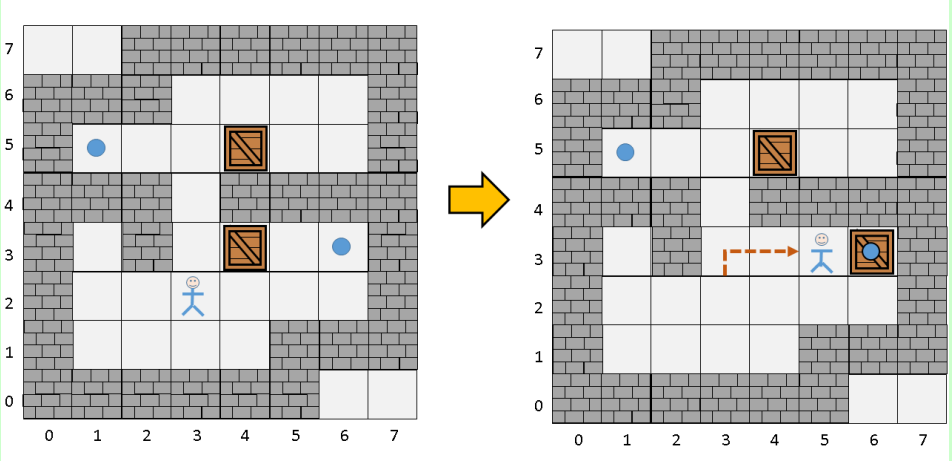
This program implements the Sokoban computer game. In the game, the worker moves around the board pushing crates into storage locations. When every crate is placed in a storage location, the game is over. The worker cannot move through crates or walls and cannot push more than one crate at a time.

Example – Focus on Essentials



This program implements the Sokoban computer game. In the game, the worker moves around the board pushing crates into storage locations. When every crate is placed in a storage location, the game is over. The worker cannot move through crates or walls and cannot push more than one crate at a time.

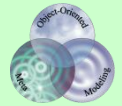
Example



This program implements the Sokoban computer game. In the game, the **worker** **moves** around the **board** **pushing** **crates** into **storage locations**. When every **crate** is **placed** in a **storage location**, the game is over. The **worker** cannot **move** through **crates** or **walls** and cannot **push** more than one **crate** at a time.

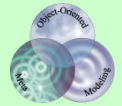
Results of Textual Analysis

- Things
  - » **Board** (comprises of locations in a grid pattern)
  - » **Wall** (location on board; obstacle)
  - » **StorageLocation** (location on board; no obstacle; stores crates)
  - » **Worker** (item on board)
  - » **Crate** (item on board; cannot walk through; can push)
- Actions
  - » **Move** (worker can change locations)
  - » **Push** (worker can push crates to different locations)
  - » **Place** (worker can place crates on a storage location)
- Discarded
  - » **Sokoban, Game, Program** (refers to program itself)



Consolidation

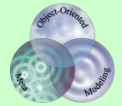
- Things
  - **Board** (comprises of locations in a grid pattern)
  - **Wall** (location on board; obstacle)
  - **StorageLocation** (location on board; no obstacle; stores crates)
  - **Worker** (item on board)
  - **Crate** (item on board; cannot walk through; can push)
- Actions
  - **Move** (worker can change locations)
  - **Push/Place** (worker can push crates to different locations)



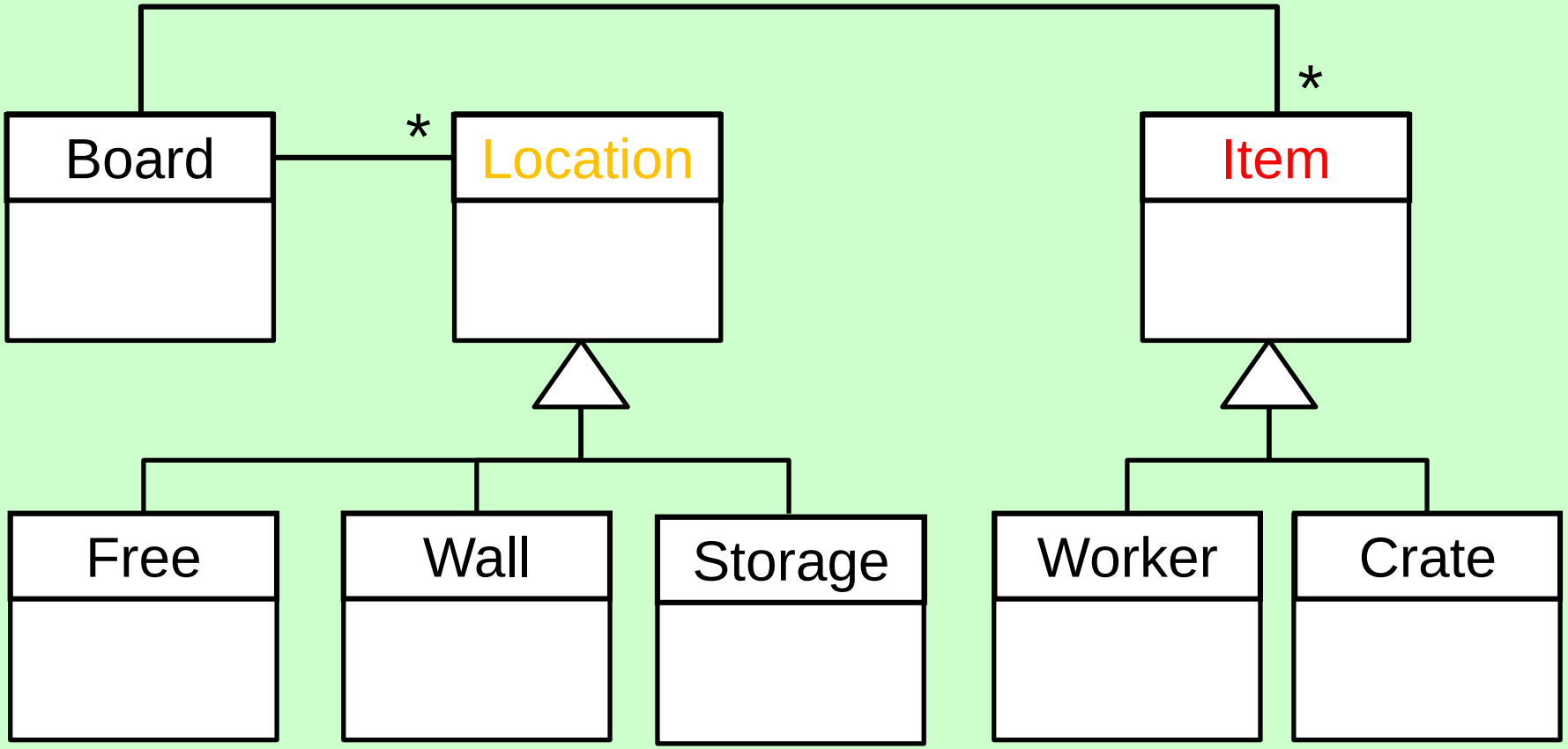
Consolidation

- Things
  - » **Board** (comprises of locations in a grid pattern)
  - » **Location** (location on board: free, wall, storage location)
  - » **Item** (worker or crate)

- Actions
  - » **Move/Push/Place** (worker moves or pushes or places)

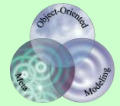


Specialisation (a.k.a. inheritance)



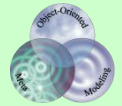
Class/Responsibilities/Collaborators

- **OO design based on**
  - **C**lasses
  - **R**esponsibilities (cf. SWEN 225 Reading List)
  - **C**ollaborators
- **Responsibilities**
  - high-level (≠ single operations)
  - what to know; what to do
  - short phrases containing an **action verb**
- **Collaborators**
  - other classes that are needed to perform a task or to get information from



CRC Cards

Class Name	
responsibility	Collaborator

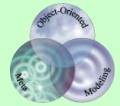


CRC Cards

Board	
aggregates locations	Location
knows item locations	Item
observes game status	Storage

What Responsibilities?

- *What responsibilities for Sokoban?*
  - *managing locations*
  - *knowing locations of items*
  - *knowing if a storage location is full*
  - *deciding when game is over*
  - *knowing which directions the worker can move towards*
  - *knowing whether the worker moves, pushes, or places*
  - *…*



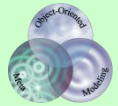
Some CRC Cards for Sokoban

Board	
aggregates locations	Location
knows item locations	Item
knows game status	Storage

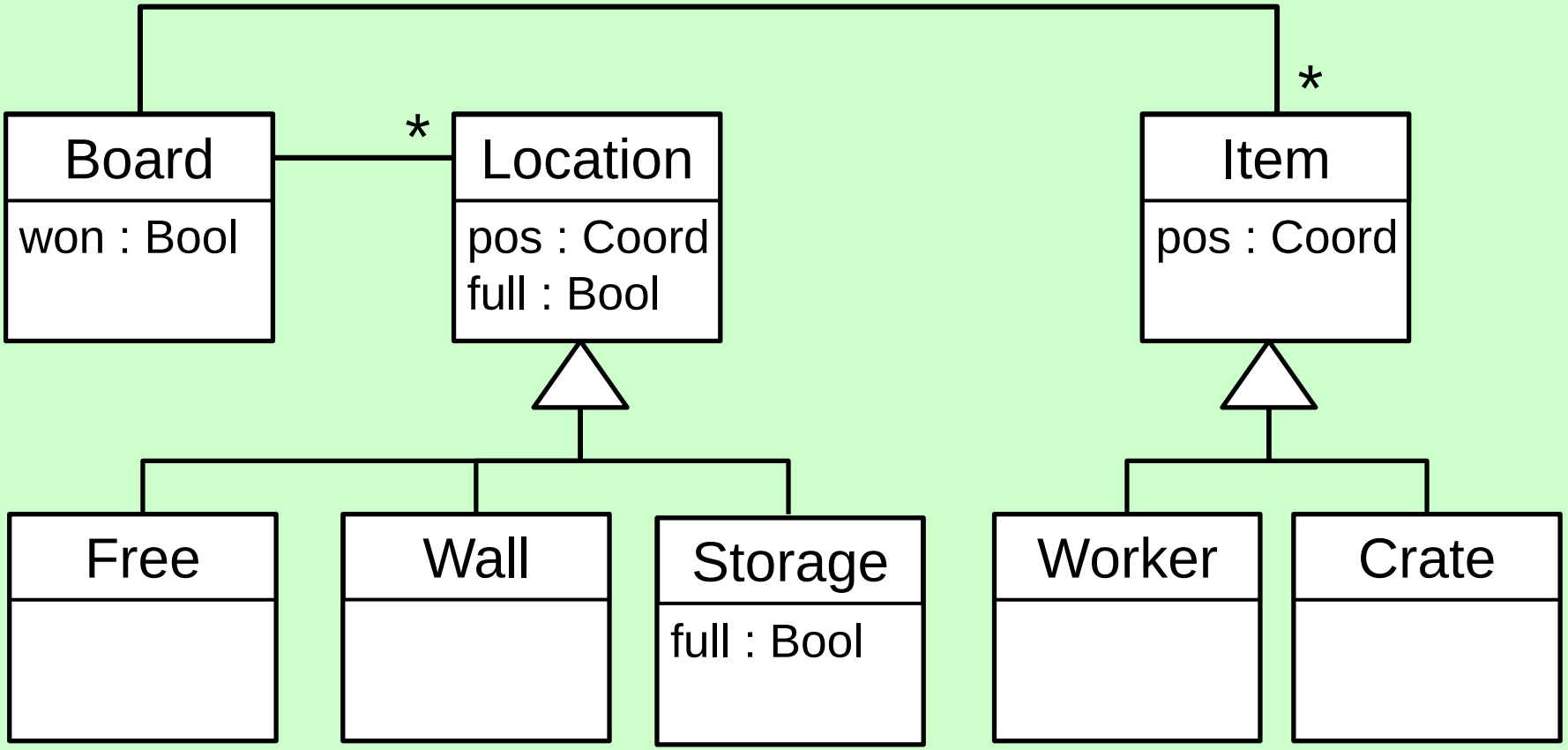
Worker	
knows position	Location
moves,	Location,
pushes,	Crate
places	Storage

Location	
knows position	Coordinate

Storage	
knows location	Location
knows state	Crate
(empty or full)	



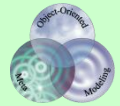
Design using Generalisation



Movement Responsibility

Movement When the user presses a key to move, the worker should move in the corresponding direction

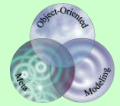
- Method **move(Direction)**
  - » *Where should we place this method?*
  - » *What are the implications of our choice?*
  - » *What does this method actually do?*



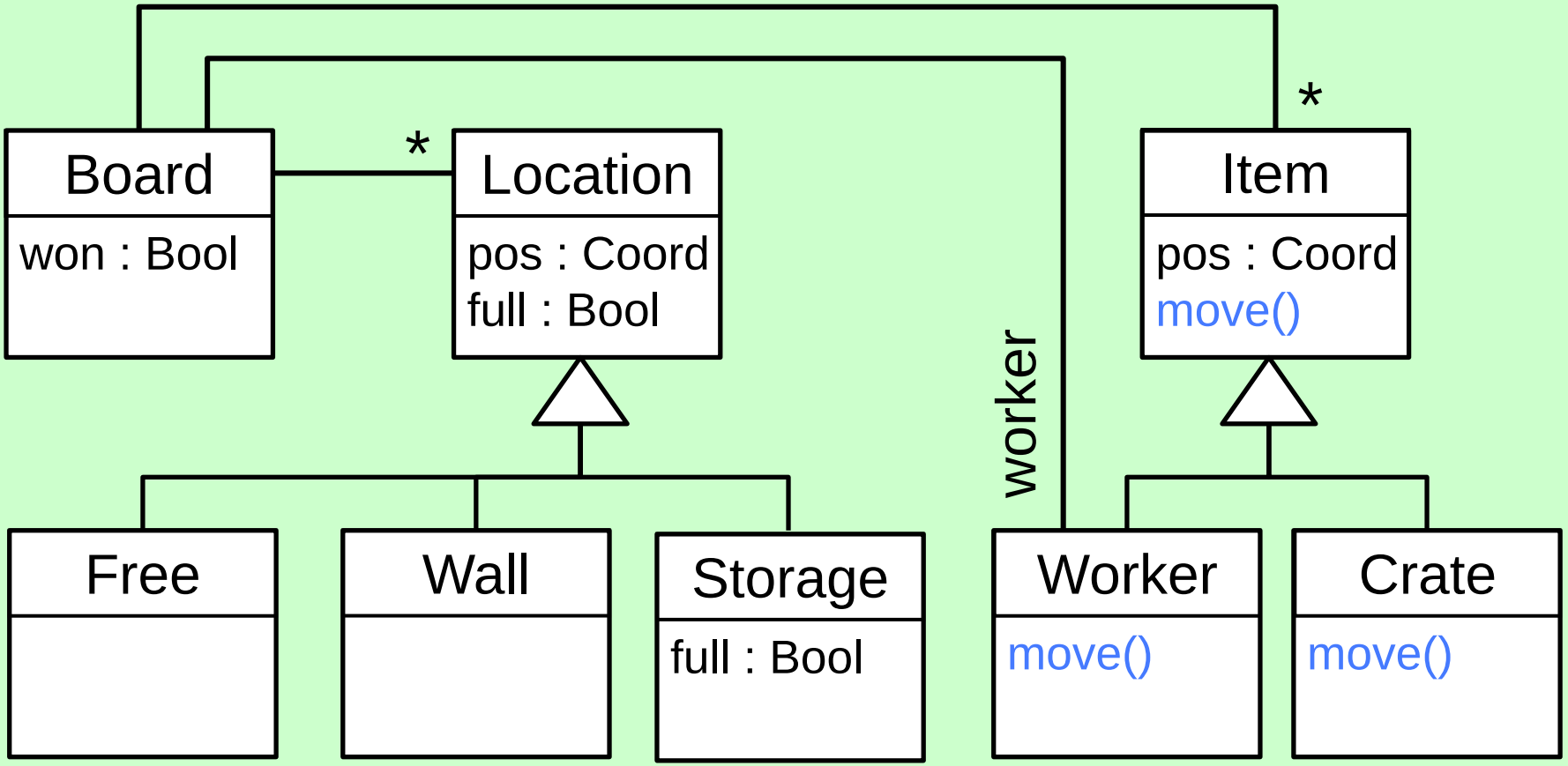
Movement Responsibility

What does moving the worker involve?

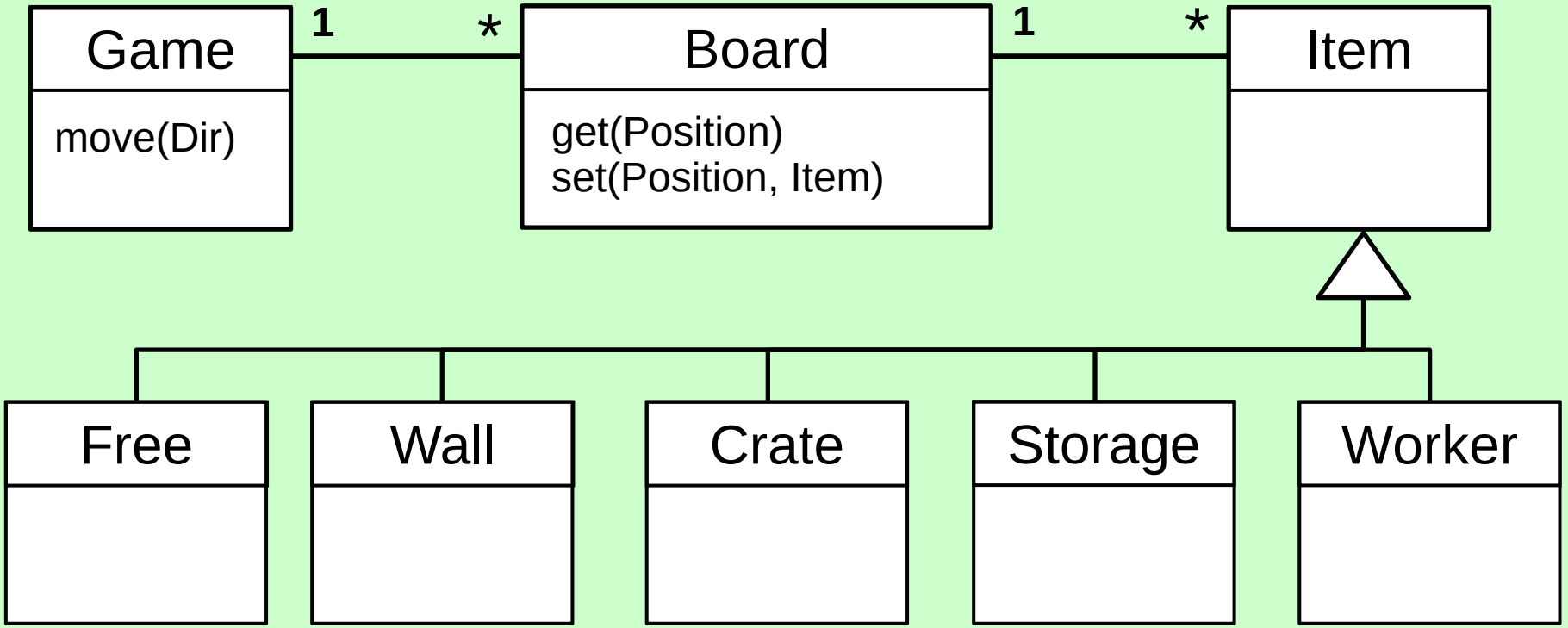
- determine location of worker
- calculate destination position
- check destination position (wall / item)
- remove worker from current position
- put worker in destination position
- … (deal with special scenarios)



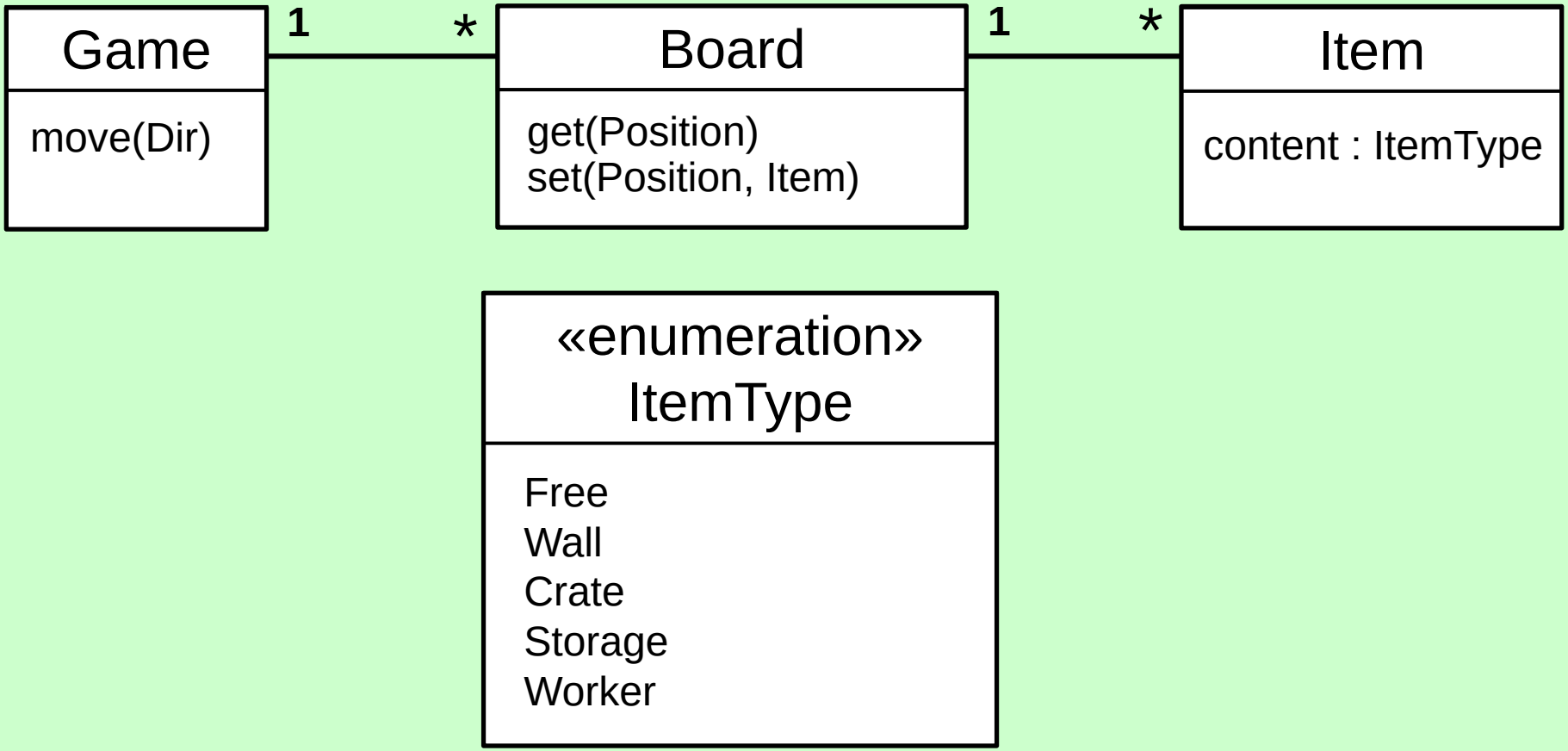
Design with "Move" Operation



Alternative Design



Alternative Design



Design Aspects

Forces influencing design decisions

- Efficiency
  - » space
  - » time

} often at odds with each other

- Maintainability
  - » information hiding
- Reuse

} often at odds with each other