Software Engineering

CS305, Autumn 2020

Week 6

Class Progress…

- Last week:
  - Logistics: feedback on PA0, PA1 posted
  - Design: Architectural Design and Styles

Class Progress…

- This week:
  - Architecture Styles (continued)
  - Detailed Design

Architectural Styles Recap..

- Popular way of making architectural design decisions to structure the system
- Result in elegant, scalable, evolvable, etc. software solutions

Architectural Styles

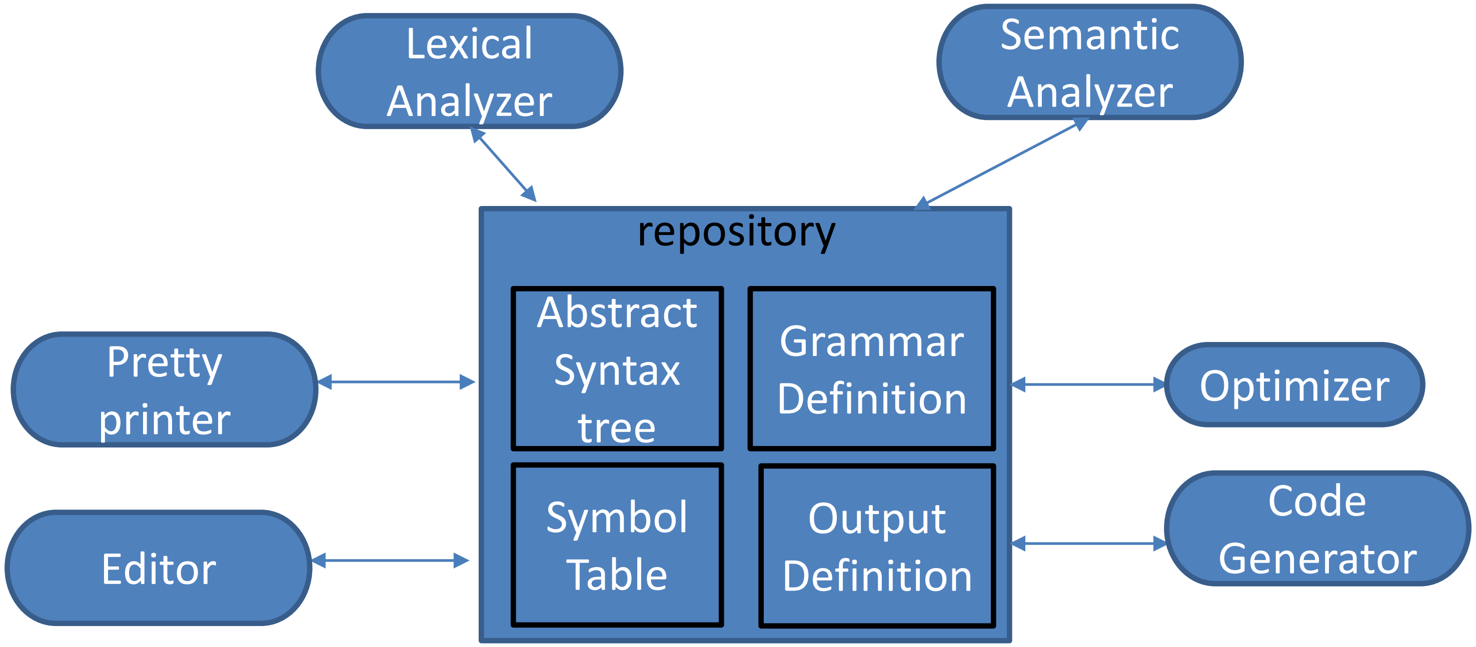
- More widely used styles of structuring a software system
  - Shared data repository Style
  - Shared services and servers style
  - Abstract machine or layered style

Repository Model

- How is data exchanged among subsystems?
  - Data is held in a central Database and all sub-modules access the Database
  - Each submodule maintains its own Database and passes data explicitly to the sub-module that needs it.
- Pros (central Database):
  - Can share large amount of data efficiently
  - Sub-modules are free from the responsibilities of data management
  - Easy integration
- Cons
  - Sub-modules must all agree on a common data model
  - Data evolution is difficult and expensive
  - Customization of data management policies is not possible

Repository Model

- E.g. Language Processing System

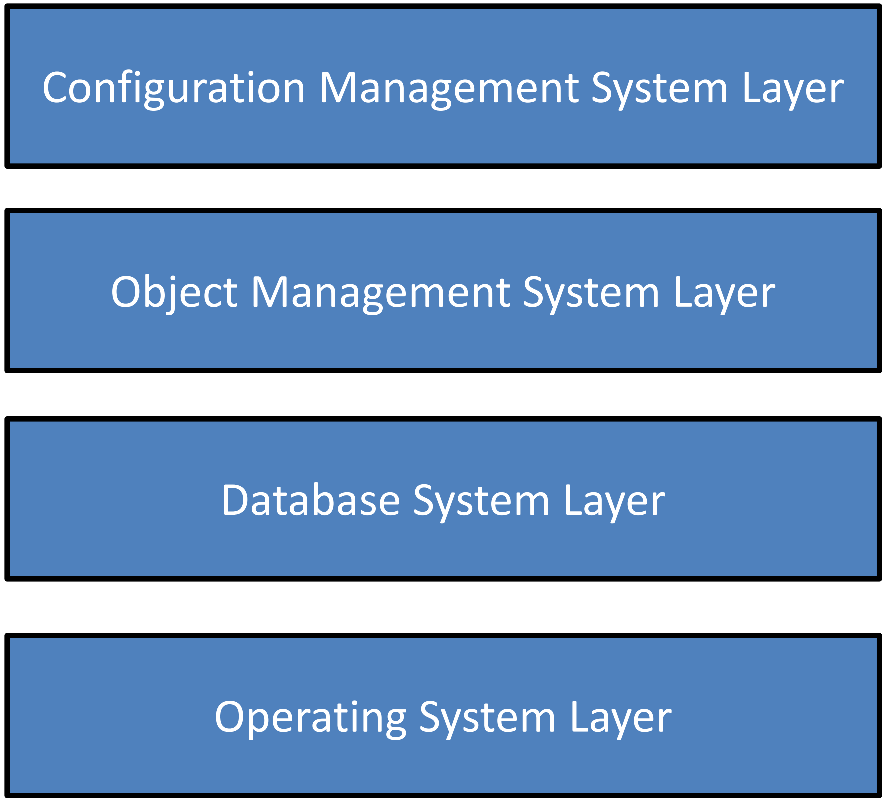


Layered Model

- How are different subsystems interfaced?
  - Each layer provides a specific set of services
- Pros:
  - Supports incremental development of subsystems in different layers.
- Cons:
  - May introduce performance overhead due to layers of abstraction

Layered Model

- E.g. Version Control System


Configuration Management System Layer
Object Management System Layer
Database System Layer
Operating System Layer

Detailed Design

- Is the process of specifying logical behavior of each component
- Typical activities:
  - Algorithm selection
  - Data structure selection
- How is it expressed?
  - Combination of pseudocode, natural language, graphical representation (e.g. behavioral, structural diagrams)

Review – Modeling and UML

Library Information System

- A software system for managing the information resources for a library
- ***Activity:*** creating a UML class diagram that models the problem
  - Include classes, their attributes and operations and the relationships among them
  - Indicate attribute types, cardinality of associations, generalization and aggregation relationships

Library Information System - Requirements

1. Each patron has one unique library card for as long as they are in the system.
2. The library needs to know at least the name, address, phone number, and library card number for each patron.
3. In addition, at any particular point in time, the library may need to know or to calculate the items a patron has checked out, when they are due, and any outstanding overdue fines.
4. Children (age 12 and under) have a special restriction–they can only check out five items at a time.
5. A patron can check out books or audio/video materials.
6. Books are checked out for three weeks, unless they are current best sellers, in which case the limit is two weeks.
7. A/V materials may be checked out for two weeks.
8. The overdue fine is ten cents per item per day, but cannot be higher than the value of the overdue item.
9. The library also has reference books and magazines, which can’t be checked out
10. A patron can request a book or A/V item that is not currently in.
11. A patron can renew an item once (and only once), unless there is an outstanding request for the item, in which case the patron must return it.

Library Information System - Nouns

1. Each patron has one unique library card for as long as they are in the system.
2. The library needs to know at least the name, address, phone number, and library card number for each patron.
3. In addition, at any particular point in time, the library may need to know or to calculate the items a patron has checked out, when they are due, and any outstanding overdue fines.
4. Children (age 12 and under) have a special restriction–they can only check out five items at a time.
5. A patron can check out books or audio/video materials.
6. Books are checked out for three weeks, unless they are current best sellers, in which case the limit is two weeks.
7. A/V materials may be checked out for two weeks.
8. The overdue fine is ten cents per item per day, but cannot be higher than the value of the overdue item.
9. The library also has reference books and magazines, which can't be checked out
10. A patron can request a book or A/V item that is not currently in.
11. A patron can renew an item once (and only once), unless there is an outstanding request for the item, in which case the patron must return it.

Nouns as candidate classes

- patron, library card, system, library, name, address, phone number, library card number, time, item, fine, child, restriction, book, A/V Material, week, best seller, limit, day, cent, value, reference book, magazine, request, age

Alternate Approach: Scenarios

- Write usage scenarios. An actor in a scenario becomes a candidate class
- E.g.
  - A patron attempts to request a book
  - A patron checks out a book and an a/v material
  - A patron attempts to renew a book
  - A patron returns a book
  - A librarian runs a report of all patrons with outstanding overdue fines
  - A child attempts to check out 2 books

Candidate Classes

```
Patron, Child, Item, Book,
LibraryCard,  AVMaterial,
ReferenceBook, Magazine,
BestSeller, Fine
```

Remaining Nouns to be Considered

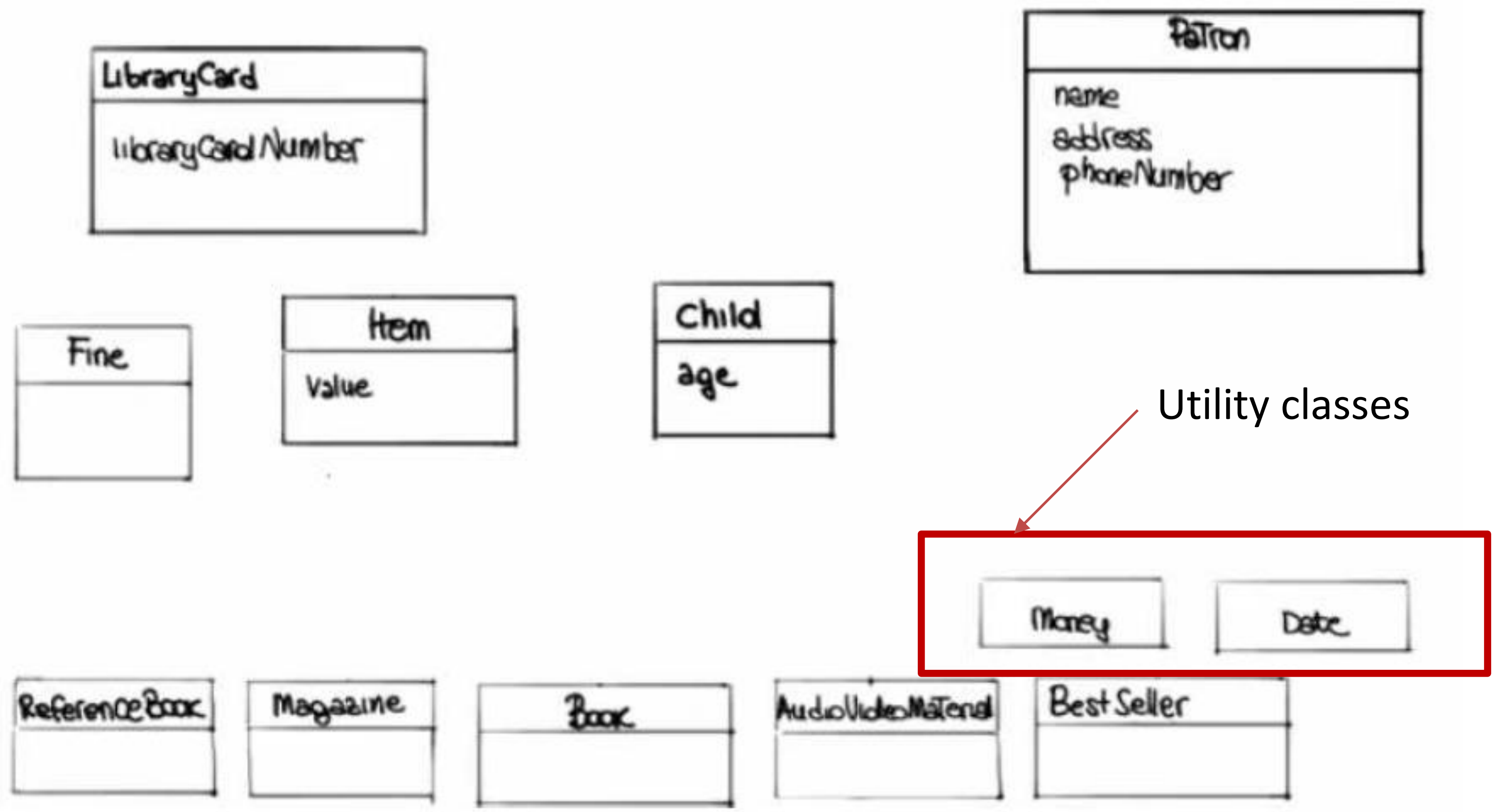
System, Library, name, address, phone number, time, fine, libraryCardNumber, restriction, week, limit, day, cent, value, age, Request

Attributes

- `name, address, phoneNumber (Patron)`
- `age (Child)`
- `libraryCardNumber (LibraryCard)`

Remaining Nouns to be Considered

time, fine, restriction, week, limit, day, cent, value

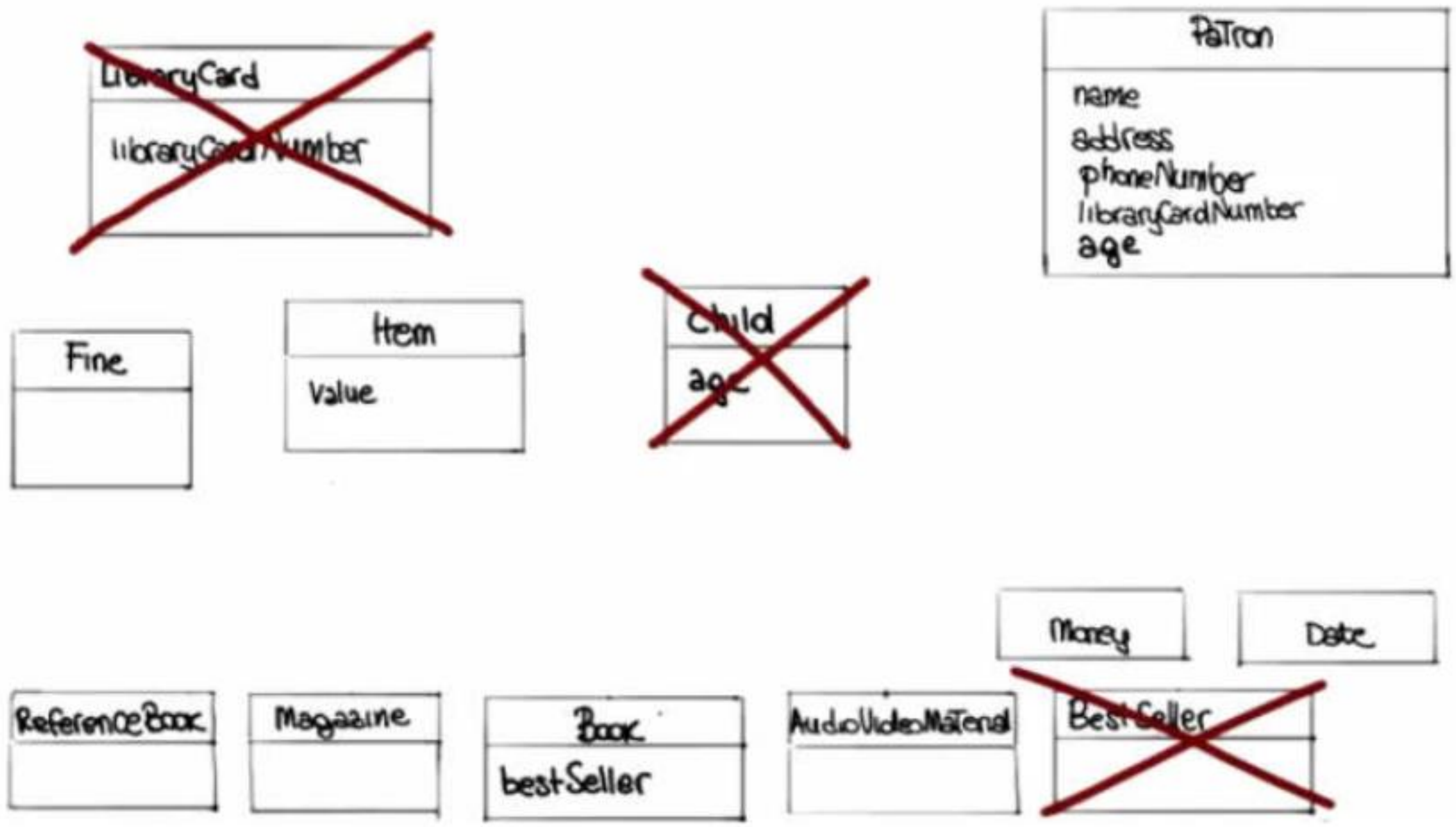

LibraryCard
libraryCardNumber
Patron
name
address
phoneNumber
Fine
Item
value
Child
age
Utility classes
Money
Date
ReferenceBook
Magazine
Book
AudioVideoMaterial
BestSeller

Utility Classes

```
Date: time, week, day, limit
Money: fine, cent, value
```

Remaining Noun to be Considered

restriction


~~LibraryCard~~
~~libraryCardNumber~~
Patron
name
address
phoneNumber
libraryCardNumber
age
Fine
Item
value
~~Child~~
~~age~~
Money
Date
ReferenceBook
Magazine
Book
bestSeller
AudioVideoMaterial
~~BestSeller~~

Inferred Attributes

- Some other attributes can be inferred from the requirements:

Noun/Data type	Parent Class	Inferred from requirement #
`dueDate / Date`	`Item`	`3, 8`
`numberOfTimesRenewed / int`	`Item`	`11`
`checkedOut / bool`	`Item`	`3-7,9-11`
`Checkoutable / bool`	`Item`	`9`

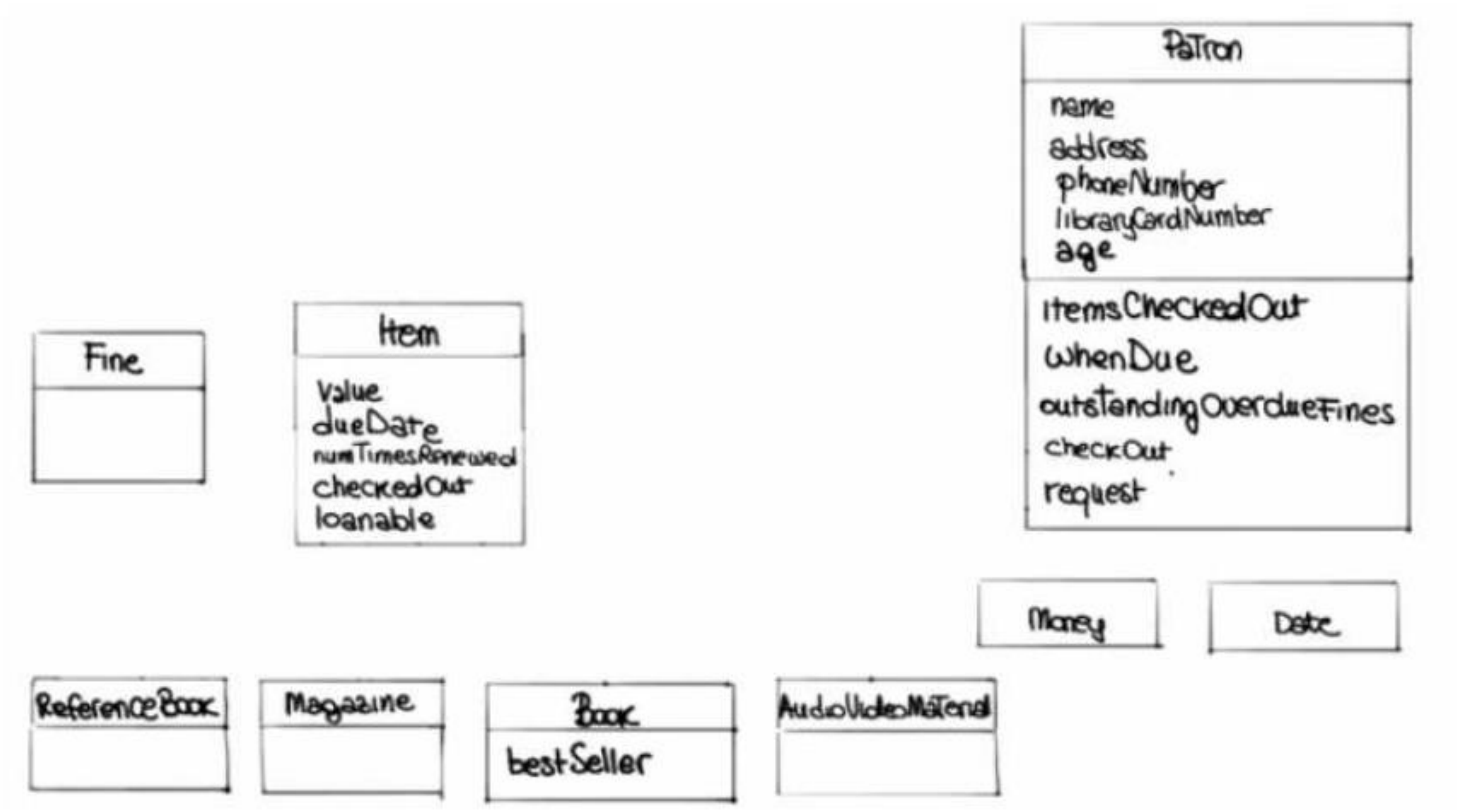
Library Information System - Verbs / Verb Phrases

1. Each patron has one unique library card for as long as they are in the system.
2. The library needs to know at least the name, address, phone number, and library card number for each patron.
3. In addition, at any particular point in time, the library may need to know or to calculate the items a patron has checked out, when they are due, and any outstanding overdue fines.
4. Children (age 12 and under) have a special restriction–they can only check out five items at a time.
5. A patron can check out books or audio/video materials.
6. Books are checked out for three weeks, unless they are current best sellers, in which case the limit is two weeks.
7. A/V materials may be checked out for two weeks.
8. The overdue fine is ten cents per item per day, but cannot be higher than the value of the overdue item.
9. The library also has reference books and magazines, which can't be checked out
10. A patron can request a book or A/V item that is not currently in.
11. A patron can renew an item once (and only once), unless there is an outstanding request for the item, in which case the patron must return it.

Operations

- `query: itemsCheckedOut`
- `query: whenDue`
- `query: outstandingOverdueFines`
- `checkOut`
- `request`
- `renew`
- `return`

Operations

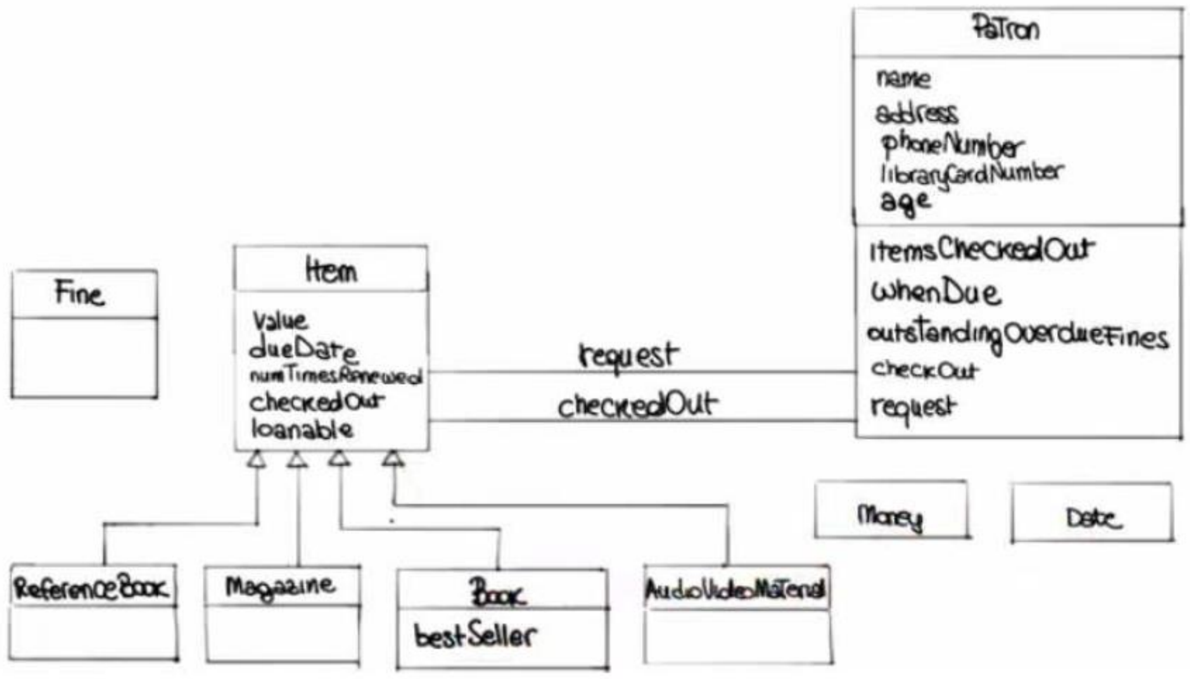

Patron
name
address
phoneNumber
libraryCardNumber
age
itemsCheckedOut
whenDue
outstandingOverdueFines
checkOut
request
Fine
Item
value
dueDate
numTimesRenewed
checkedOut
loanable
Money
Date
ReferenceBook
Magazine
Book
bestSeller
AudioVideoMaterial

Inferred Operations

- Whether or not an `Item` can be checked out
- The checkout period for an `Item`
- The per day fine for an `Item`
- Whether an `Item` has been renewed

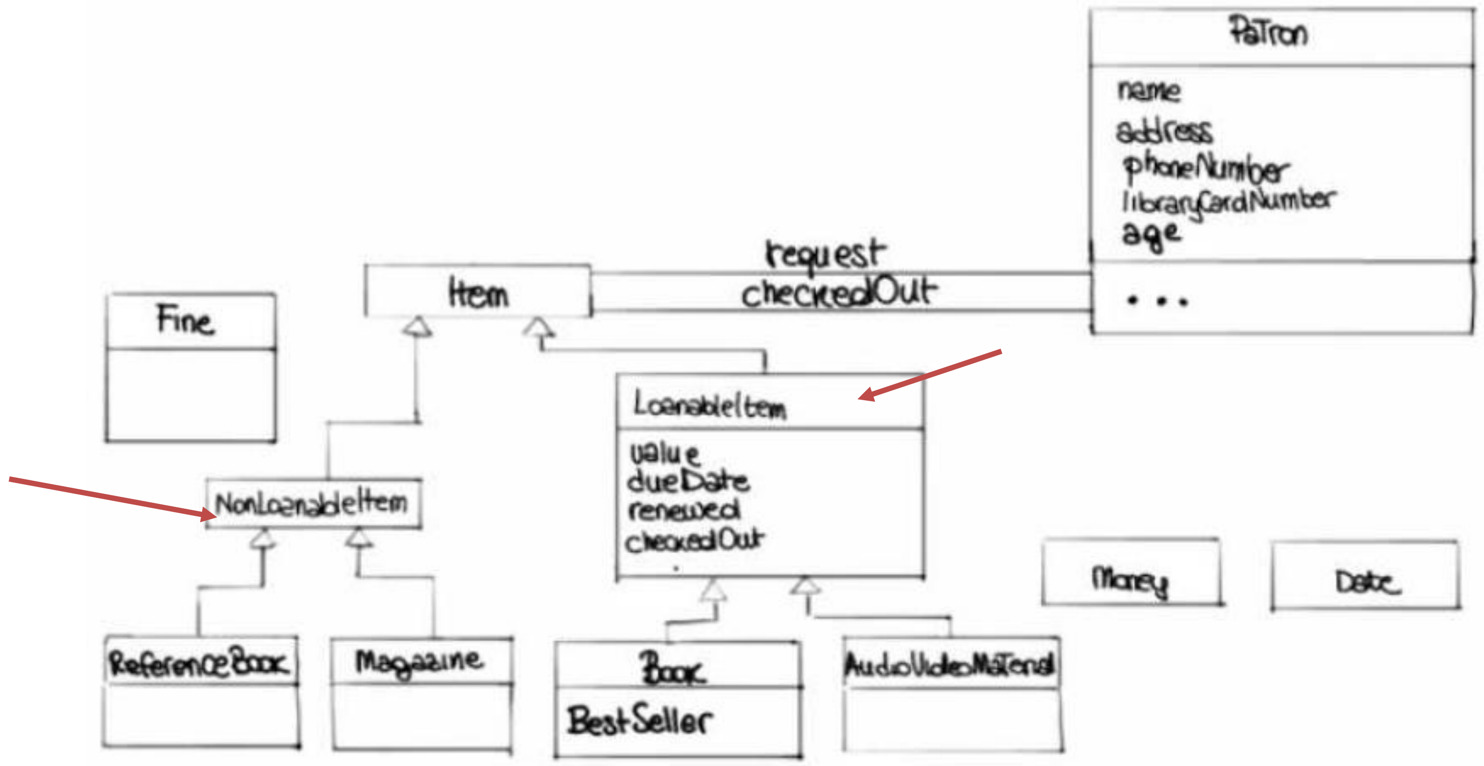
Associations

- Checking out an item is associated with `Patron` and an `Item`.
- Likewise for `Request`



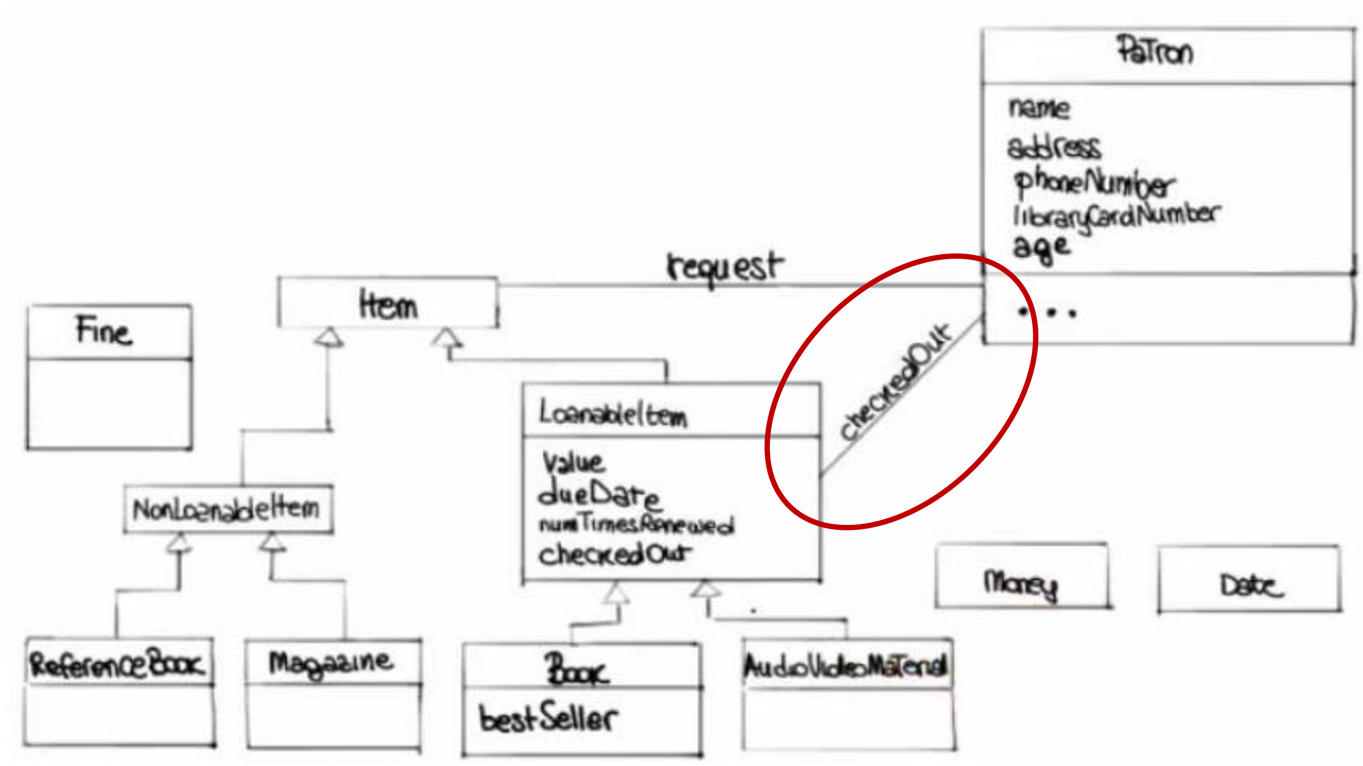
Associations

- Items can be checkoutable or non checkoutable



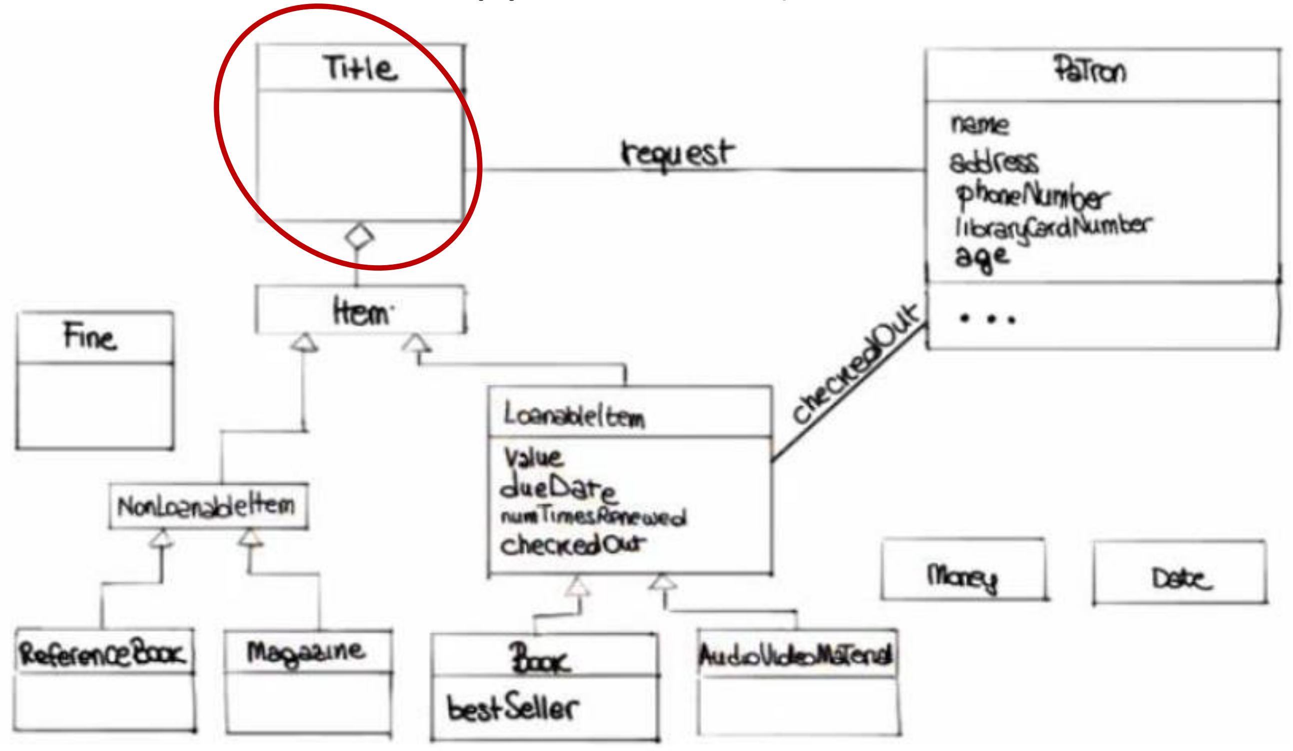
Refining Associations

- Not all Items can be checkoutable!



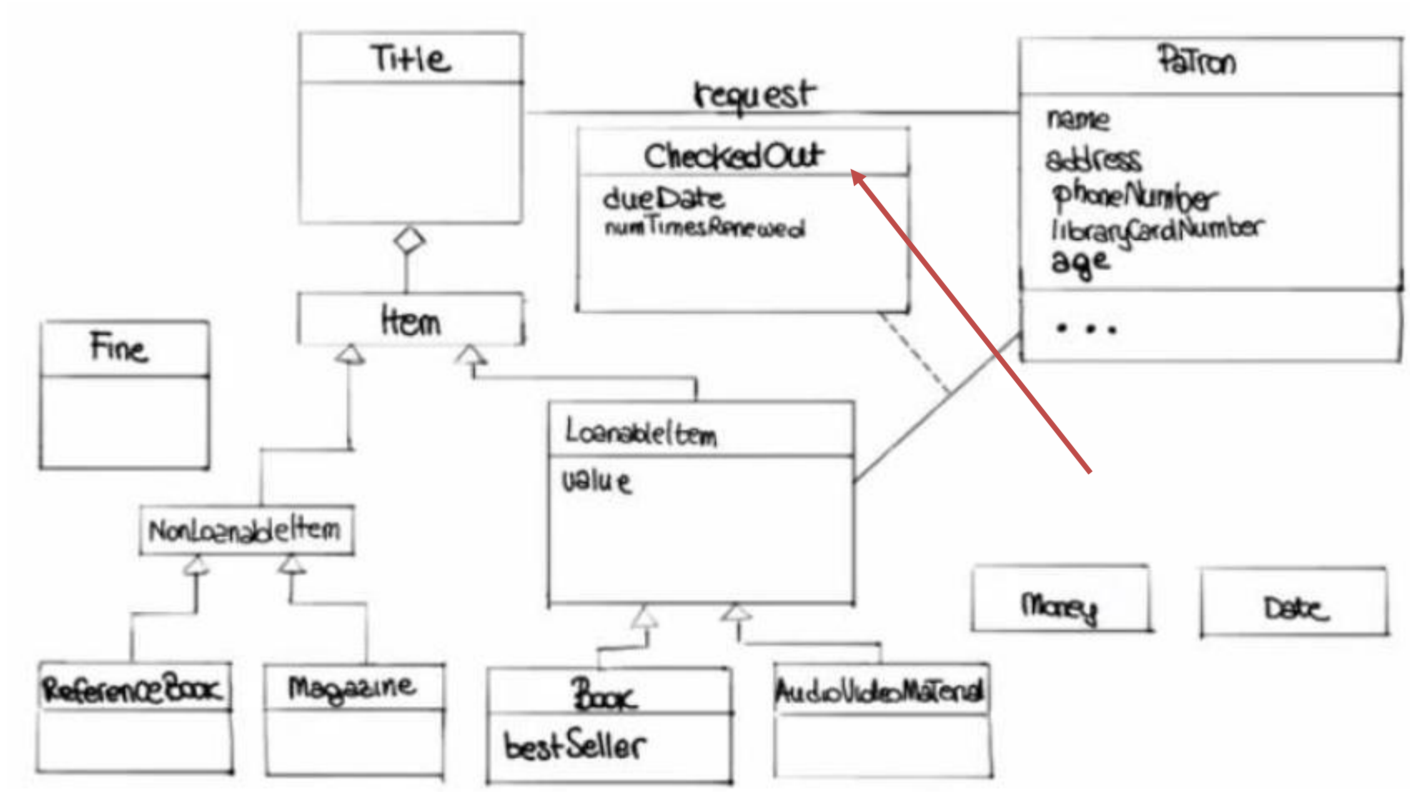
Refining Class Diagram

- A Patron checks out a Title! And not really an item (there can be more than one copy of the item).

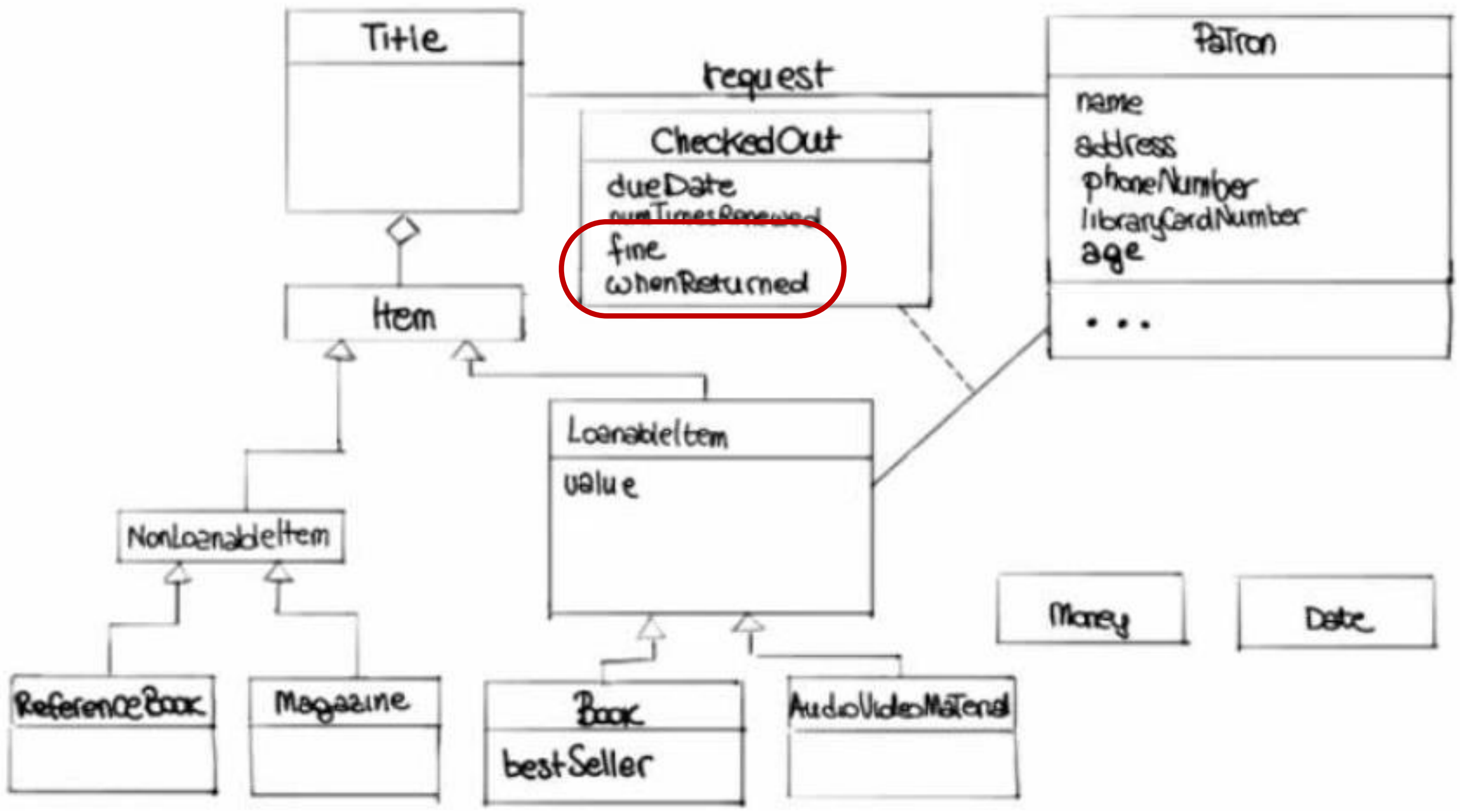


Refining Class Diagram – association class

- `dueDate` is not really an attribute of `LoanableItem`

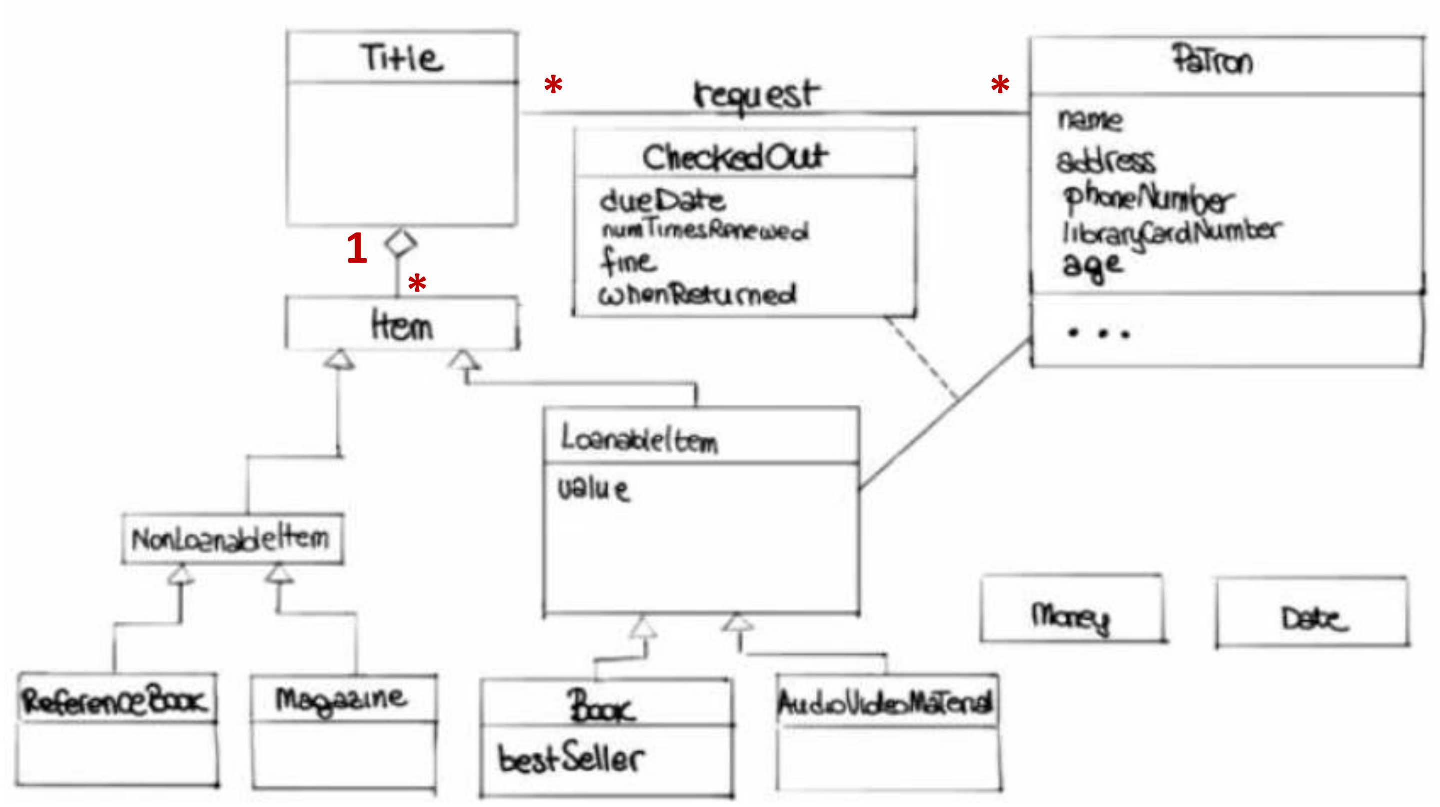


Refining Class Diagram



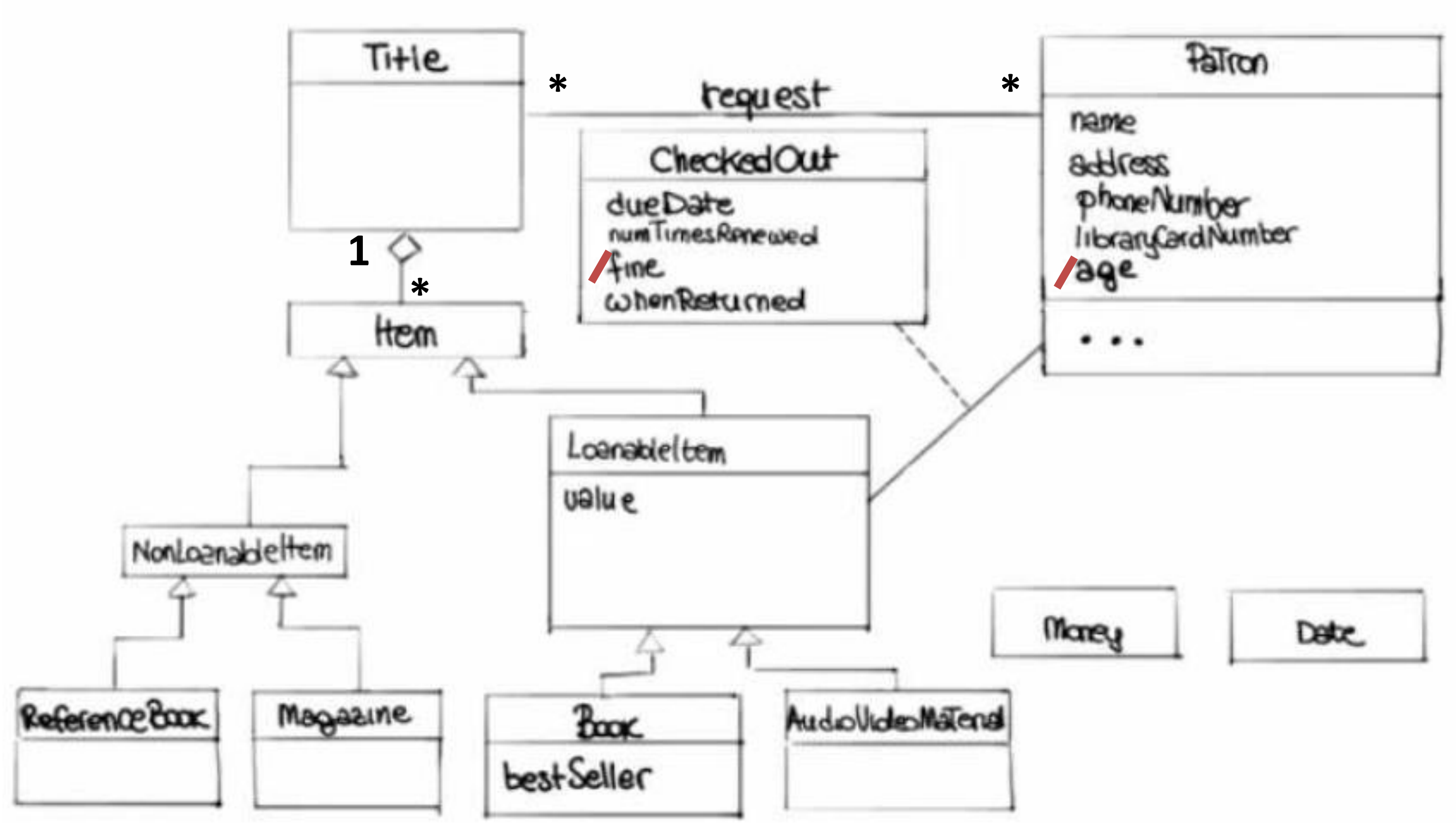
Refining Class Diagram - cardinality

- How many entities are involved in a relationship
  - E.g. for each item there is only one title. For a title, there can be several items (denoted by *).



Refining Class Diagram – Derived Attributes

- `age` can be derived from `DOB`, similarly `fine` can be derived from `whenReturned`



Missing Requirements

- The system should provide operations for the `Patron` to make or cancel a request
- The system should provide an operation for a `Patron` to renew an Item
- The system should allow a `Patron` to pay a fine

Design Patterns

- General, reusable solutions for commonly occurring design problems
  - Reuse design solutions that have worked in the past and build upon them
- Goal: effective design for software systems
  - Avoid “reinventing the wheel”
- Span a broad spectrum of abstraction and application:
  - Architectural patterns
  - Component patterns (also referred to as design patterns)
  - Interface design patterns
  - Webapp patterns
  - Mobileapp patterns

Design Patterns - History

- 1977: Christopher Alexander introduces the idea of patterns: “A Pattern Language”
- 1987: Cunningham and Beck use Alexander’s ideas and create 5-pattern language for Smalltalk programmers
- 1987: Erich Gamma’s thesis on importance of patterns and how to capture them
- 1992: Jim Coplien’s book “Advanced C++ Programming Styles and Idioms”
- 1994: Gamma, Helm, Johnson, and Vlissides publish “Design Patterns: Elements of Reusable Object-Oriented Software”
  - Gang of Four (GOF) book
  - A catalogue of patterns

GoF Book – A Patterns Catalogue

- Focus on patterns that are relevant to object oriented design
- Classified based on purpose:
  - Fundamental Patterns – basic ones
  - Creational Patterns – support object creation
  - Structural Patterns – support putting objects together / composing
  - Behavioral Patterns – focus on realizing interaction of objects
  - Concurrency Patterns – support concurrency

Patterns Catalogue

Creational Patterns

Abstract Factory
Factory Method
Singleton
Lazy Initialization

Fundamental Patterns

Delegation Pattern
Interface Pattern
Proxy Pattern

Structural Patterns

Bridge Pattern
Adapter Pattern
Decorator Pattern

Behavioral Patterns

Chain of Responsibility
Iterator
Observer
State
Strategy
Visitor

Concurrency Patterns

Active Object
Monitor Object
Thread Pool

Patterns - Format

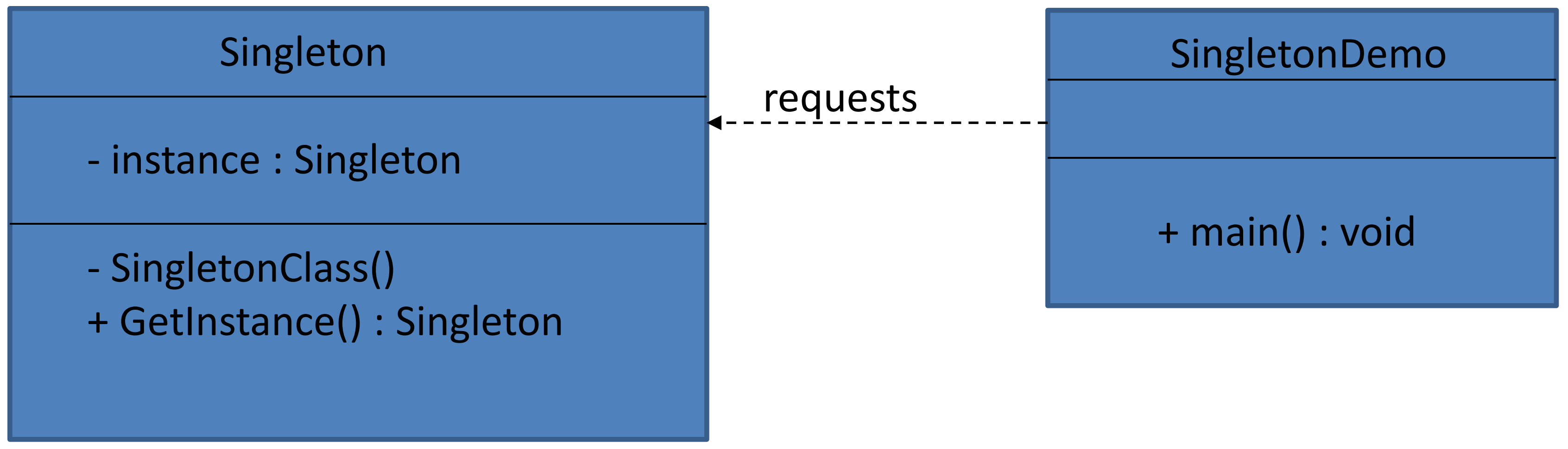
- Name
- Intent:
  design issue or problem
- Also known as:
  other names for
- Motivating scenario
- Applicability/context
- Structure:
  static object model
- Participants:
  description of classes
- Collaborations:
  event traces
- Consequences:
  tradeoffs
- Implementation strategy:
  in various languages
- Sample code
- Known uses:
  in existing systems
- Related patterns

Patterns - Format

- Name
- Intent:
  design issue or problem
- Also known as:
  other names for
- Motivating scenario
- Applicability/context
- Structure:
  static object model
- Participants:
  description of classes
- Collaborations:
  event traces
- Consequences:
  tradeoffs
- Implementation strategy:
  in various languages
- Sample code
- Known uses:
  in existing systems
- Related patterns

Singleton Pattern

- Ensure *only one instance* of a class is created


Singleton
- instance : Singleton
- SingletonClass()
+ GetInstance() : Singleton
requests
SingletonDemo
+ main() : void