Container Classes

Nested Collections - Exercises

COMP2603
Object Oriented Programming 1

Week 12

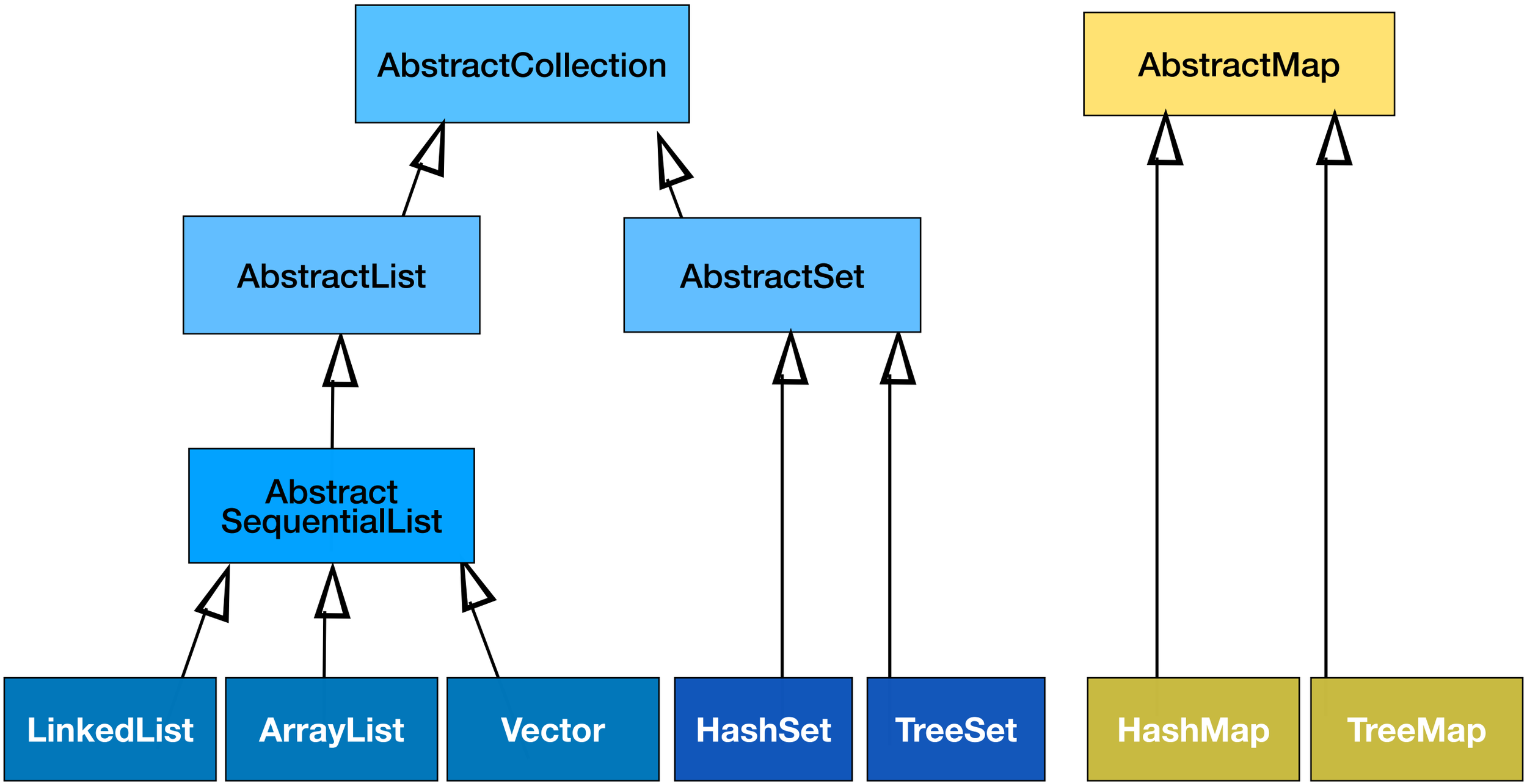
+ Box Office
+ Patron
+ Ticket

Outline

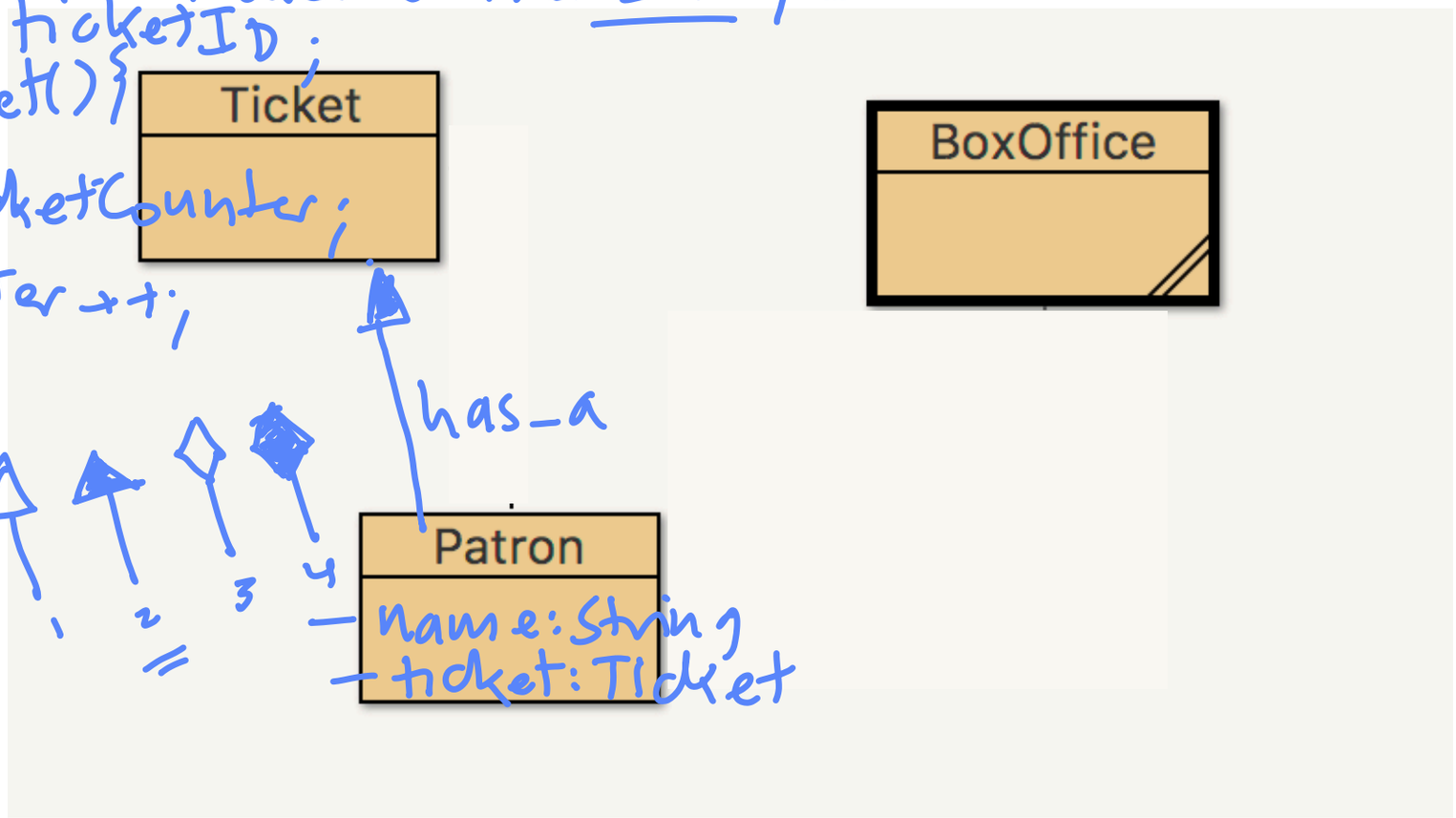
- Write code to create various collections
- Decide which collection suits the scenario
- Handle errors as requirements change for the scenarios

Reading: Chapter 13 - Container Classes: P. Mohan 2013

Classes in the Java Collections Framework



Scenario



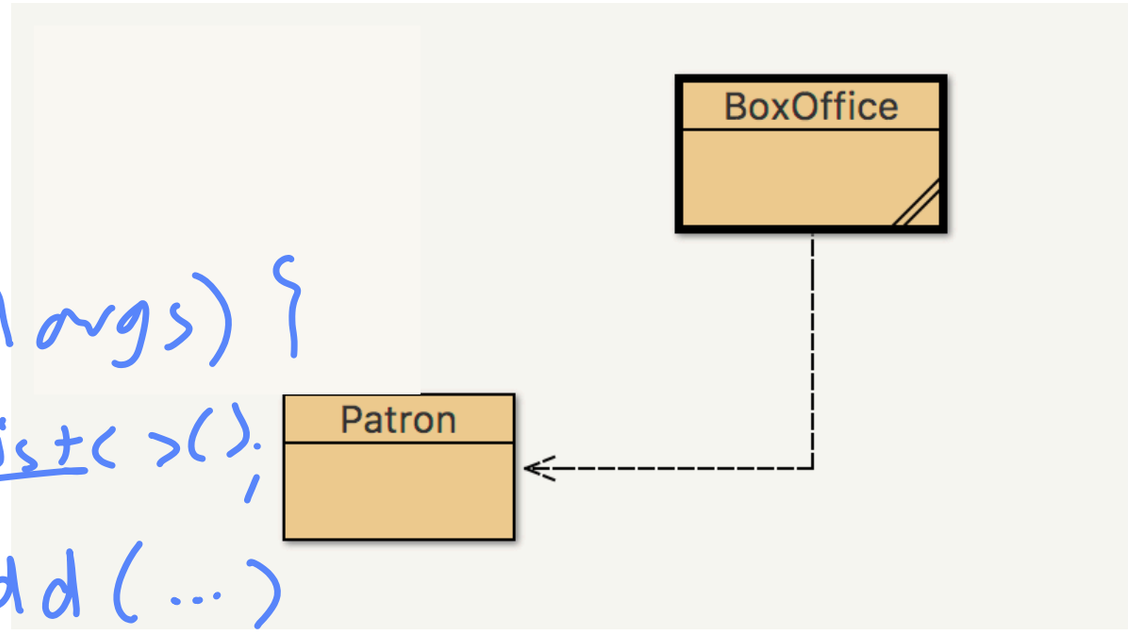
- Let's say we have a box office where patrons come to see movies. Each patron buys a ticket to a movie.
- A patron has a name.
- A ticket has a ticket number (unique; starts from 100, increments by 1)

Activity 1

- Write the code for the Patron class (Ticket could be assigned in the Patron constructor)
- Write the code for the Ticket class

Activity 2

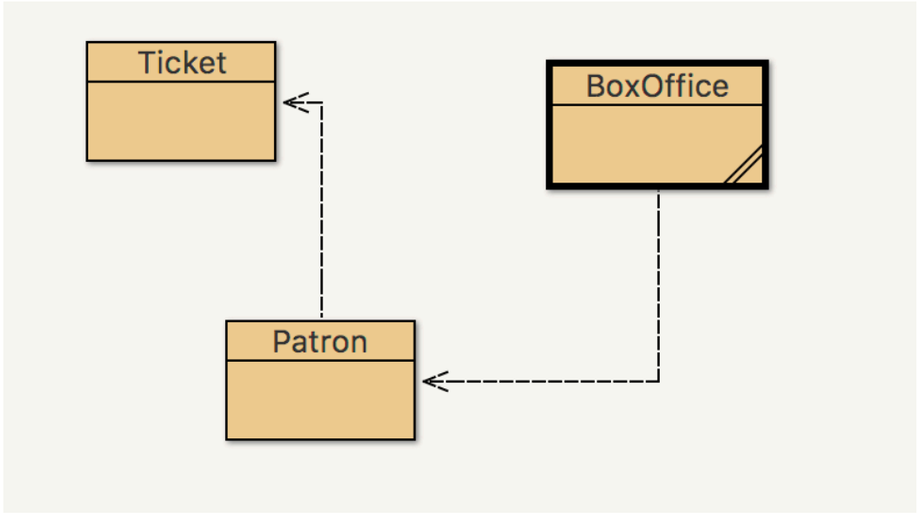
- Write the code for the BoxOffice class



1. The box office keeps track of patrons using a collection called patrons.
   - Let's use an ArrayList for now
2. Patrons are added to the box office using an addPatron(..) method.
3. Create an accessor for the patrons collection.

Activity 3

- Create a few Patron objects
- Add them to the box office's patron collection.
- Print the names of the patrons in the collection.



Tip: You can put a main method inside a class for quick testing rather than a separate runner class. Simply remove the main method if you are deploying the class in an application.

Activity 4

- Let's try to retrieve a patron from the patrons collection in the BoxOffice class
  - Write a method called findPatron(..) in the BoxOffice class to find a patron with a given name
  - If found, print out "Patron found <insert name>" or "Patron not found <insert name>" otherwise
  - Show how this method can be invoked using the name "Johannes Borg"

Google ArrayList API

Activity 5

private HashMap<String, Patrons> siteface = new Hashmap<>();
patrons

- Let's use a HashMap instead of the ArrayList for the patrons collection so that given a name, the patron object can be found efficiently. k → v
  - Modify the BoxOffice class as necessary.
- What changes were necessary when HashMap is used instead of the ArrayList?
- Which method doesn't work now?

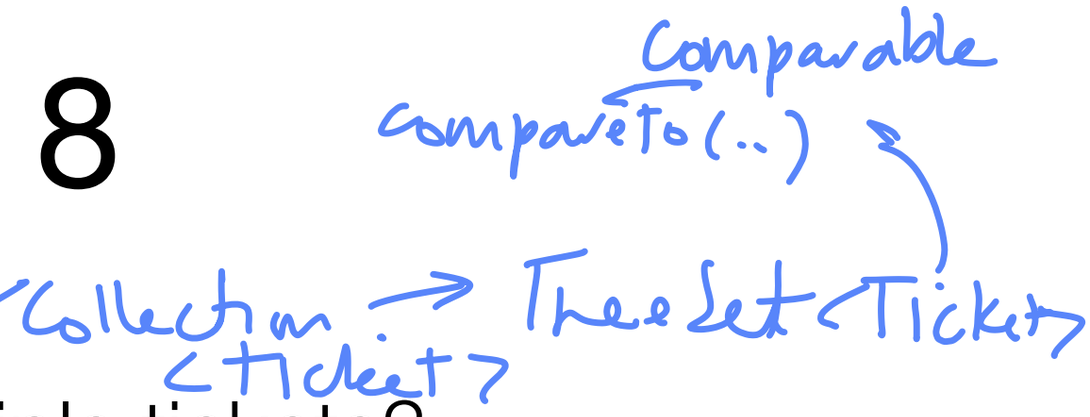
Activity 6

- Try to fix the broken code so that we can find a Patron with a given name in the HashMap.
  - Amend the code.

Activity 7

- What happens if we have two patrons with the same name? The latter replaces the former.
- We want to be able to store Patrons with the same name, but we need a unique key.
- What would make them unique? The Ticket (unique number).
  - Use that in conjunction with the patron name to generate a unique hash code.

Activity 8



- What about if a Patron has multiple tickets?
- Let's modify the Patron class so that a patron can have multiple tickets. What must we do?
  - Patron class: <brainstorm changes>
  - BoxOffice class: <brainstorm changes>