Object Oriented Design and Analysis
CPE 343

Lecture 7

Designing Behavior: Sequence, State and Activity Diagrams

Dr. Sally E. Goldin
Department of Computer Engineering
King Mongkut's University of Technology Thonburi
Bangkok, Thailand

Last update: 24 Feb 2020

Use Case Diagram Exercise

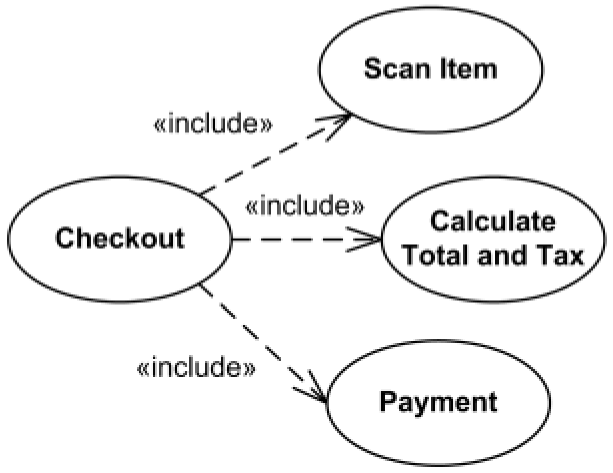
- Use cases = interactions between actors (users) and the system
  - "Server" or "system" is not an actor!
- Identify top level use cases first
  - Short name starts with verb
  - Defined by a single, clear goal
- Need lines/links to show every relationship
  - Actor participation (no text)
  - Included use case
  - Extending use case
- Use case diagrams do not show "steps" in a process
- Use cases do not usually form a hierarchy

Includes and Extends

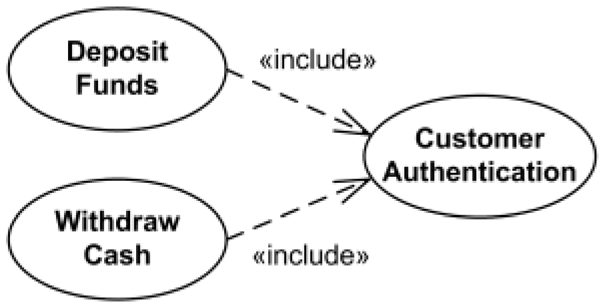
Includes

- **Sub-part of a use case, to simplify**
- **Sub-part of a use case, to share common actions**

*In both situations, included use case is a **necessary part** of including use case*



Checkout use case includes several use cases - Scan Item, Calculate Total and Tax, and Payment



***Deposit Funds** and **Withdraw Cash** use cases include **Customer Authentication** use case.*

Note direction of arrow – from primary use case to included use case

Includes and Extends (2)

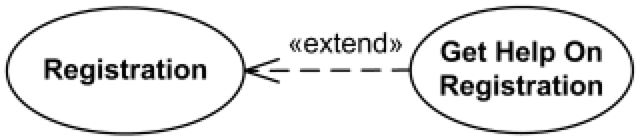
Extends

Shows an optional elaboration or addition to a primary use case

*Not a **necessary part** of extended use case*

*Has a **trigger** and **extension point***

*Will be explained in **alternative scenarios** in use case narrative*



***Registration** use case is complete and meaningful on its own.*
*It could be extended with optional **Get Help On Registration** use case.*

Note direction of arrow – from extending use case to primary use case

Software Design:
Both Structure and Behavior

Data

Components

Relationships

Outputs

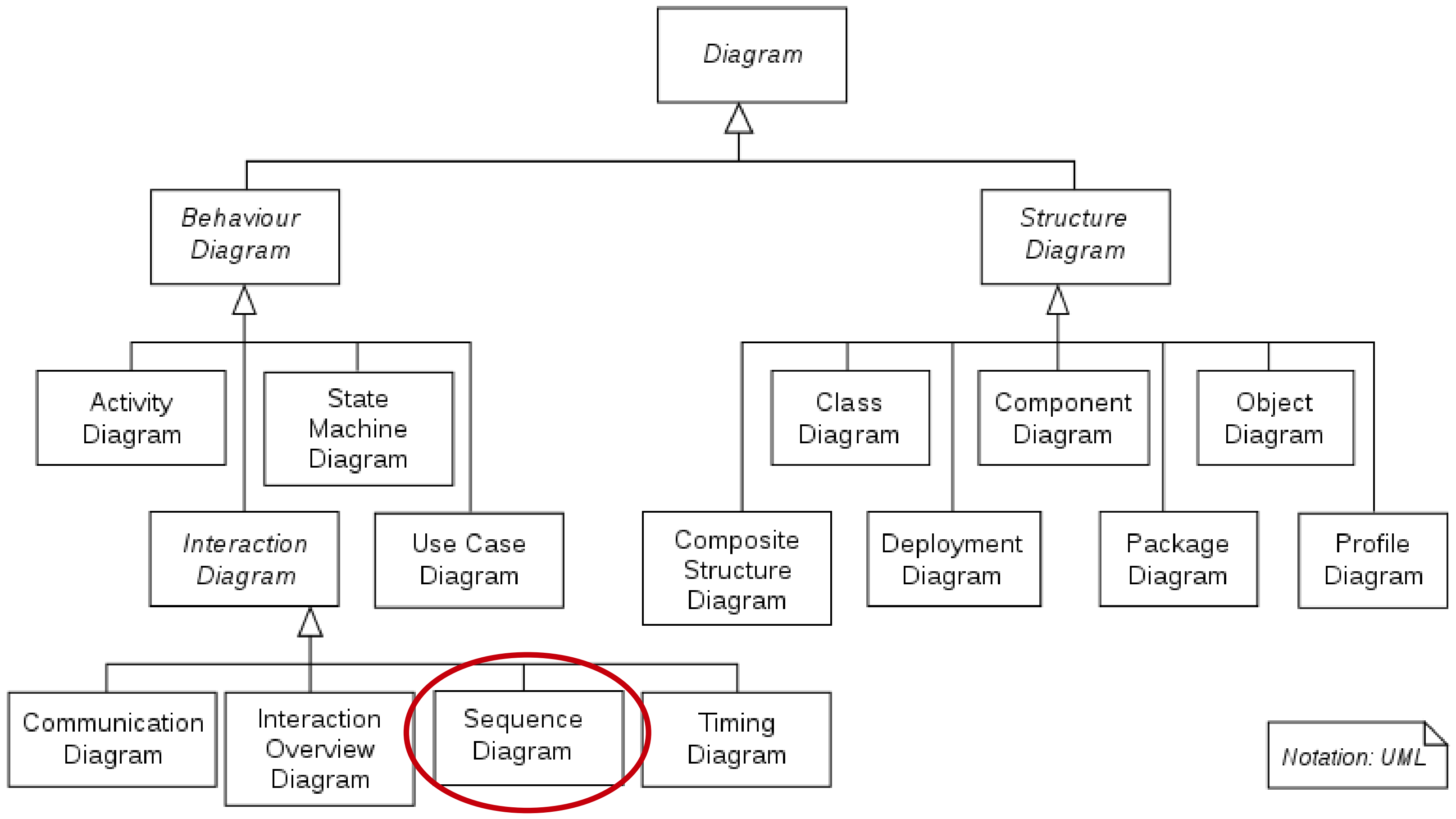
Actions

Interactions

Algorithms

State changes

UML Sequence Diagrams



What is a sequence diagram?

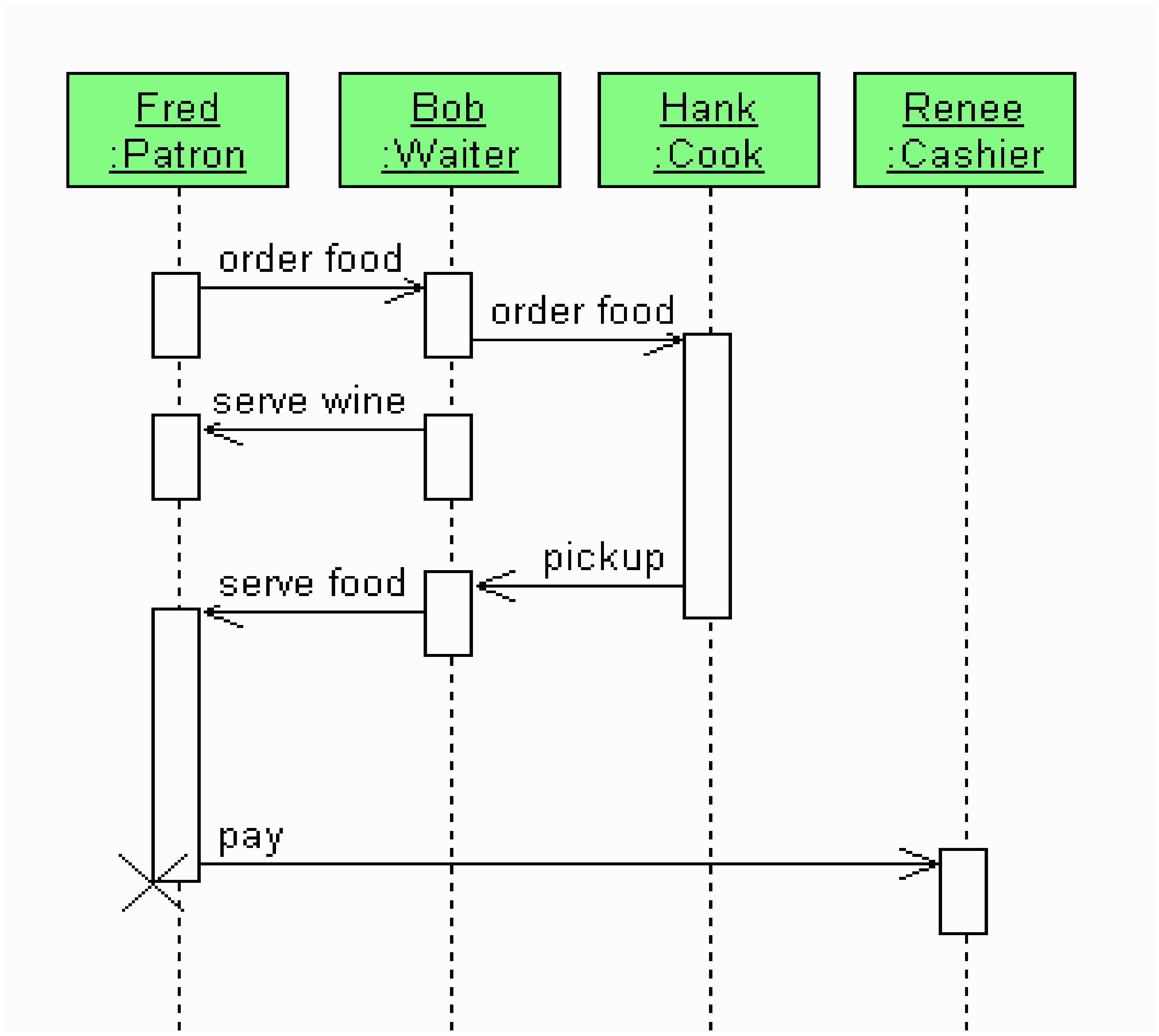
“**Interaction diagrams** describe how groups of objects collaborate in some behavior. The UML defines several forms of interaction diagrams, of which the most common is sequence diagrams.

Typically, a sequence diagram captures the behavior of a single scenario. The diagram shows a number of example objects and the messages that are passed between these objects within the use case.” Martin Fowler, *UML Distilled 3rd Edition (2004)*

Important words:

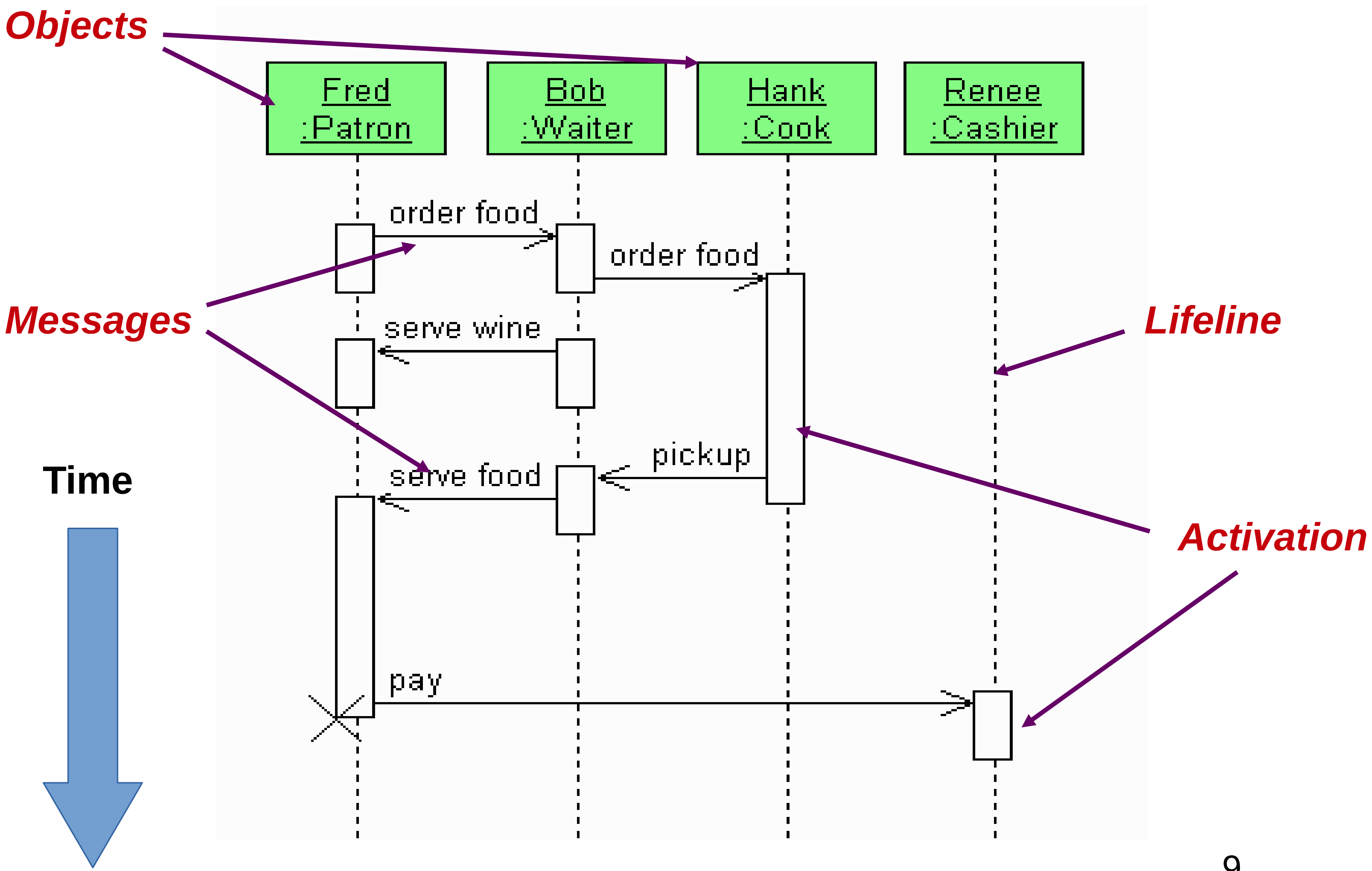
- ***Objects*** *(class instances)*
- ***Messages*** *(method calls)*
- ***Collaborate*** *(work together)*

A Simple Sequence Diagram

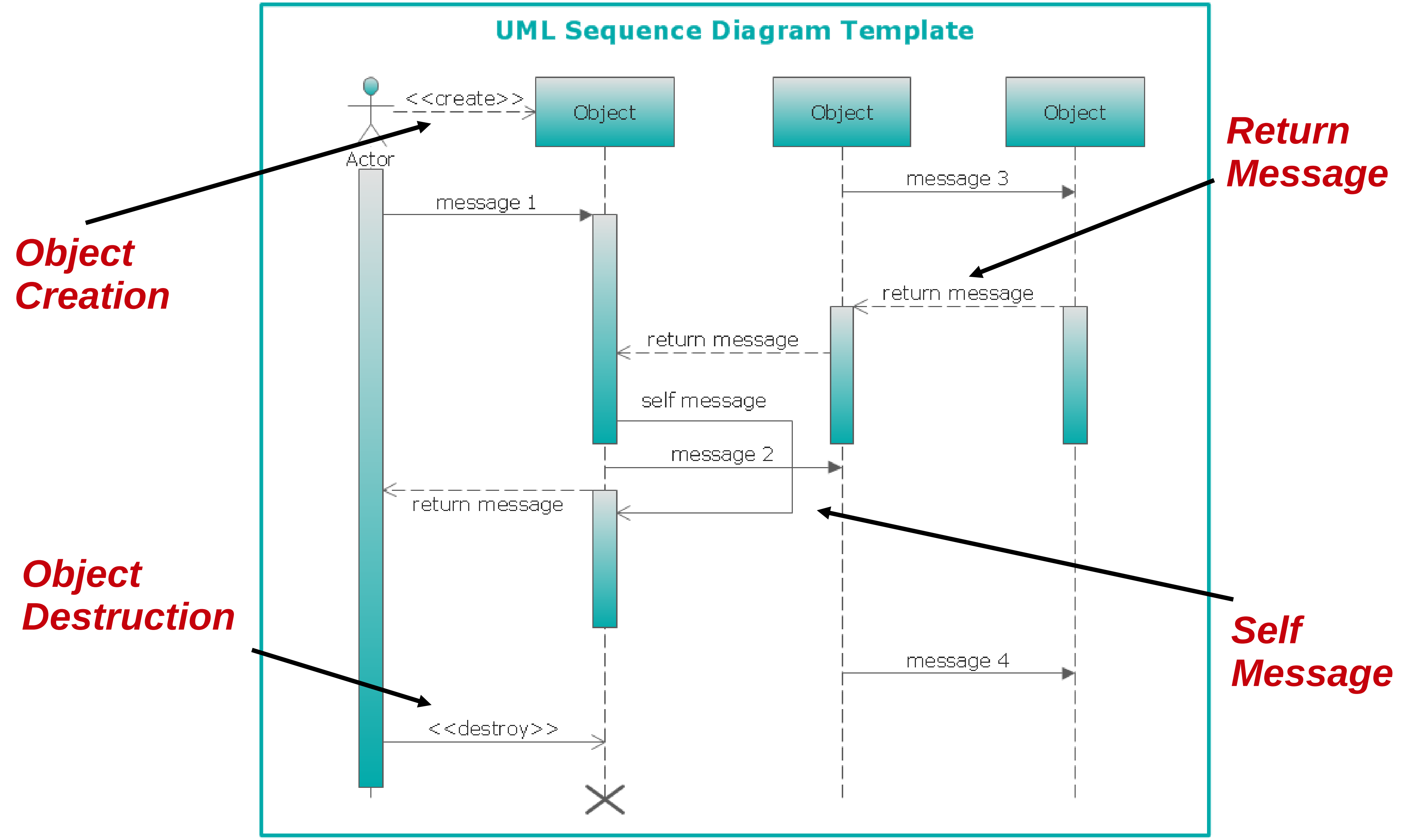


https://commons.wikimedia.org/wiki/File:Restaurant-UML-SEQ.gif

Some Sequence Diagram Components



More Components



http://www.conceptdraw.com/How-To-Guide/picture/UML-sequence-diagram-template.png

Why create sequence diagrams?

Use cases => interaction between actors and the “system”

Sequence diagrams => interaction between objects *within* the system

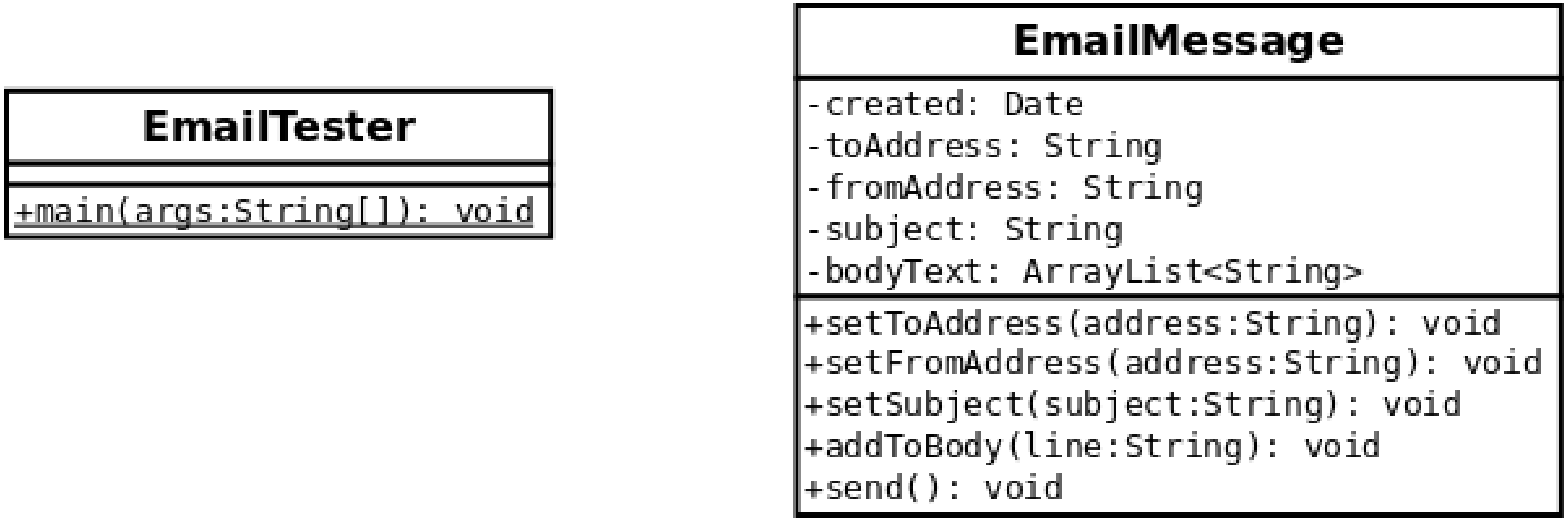
Sequence diagrams expand behavioral description to the next levels of detail

Interaction reveals structure

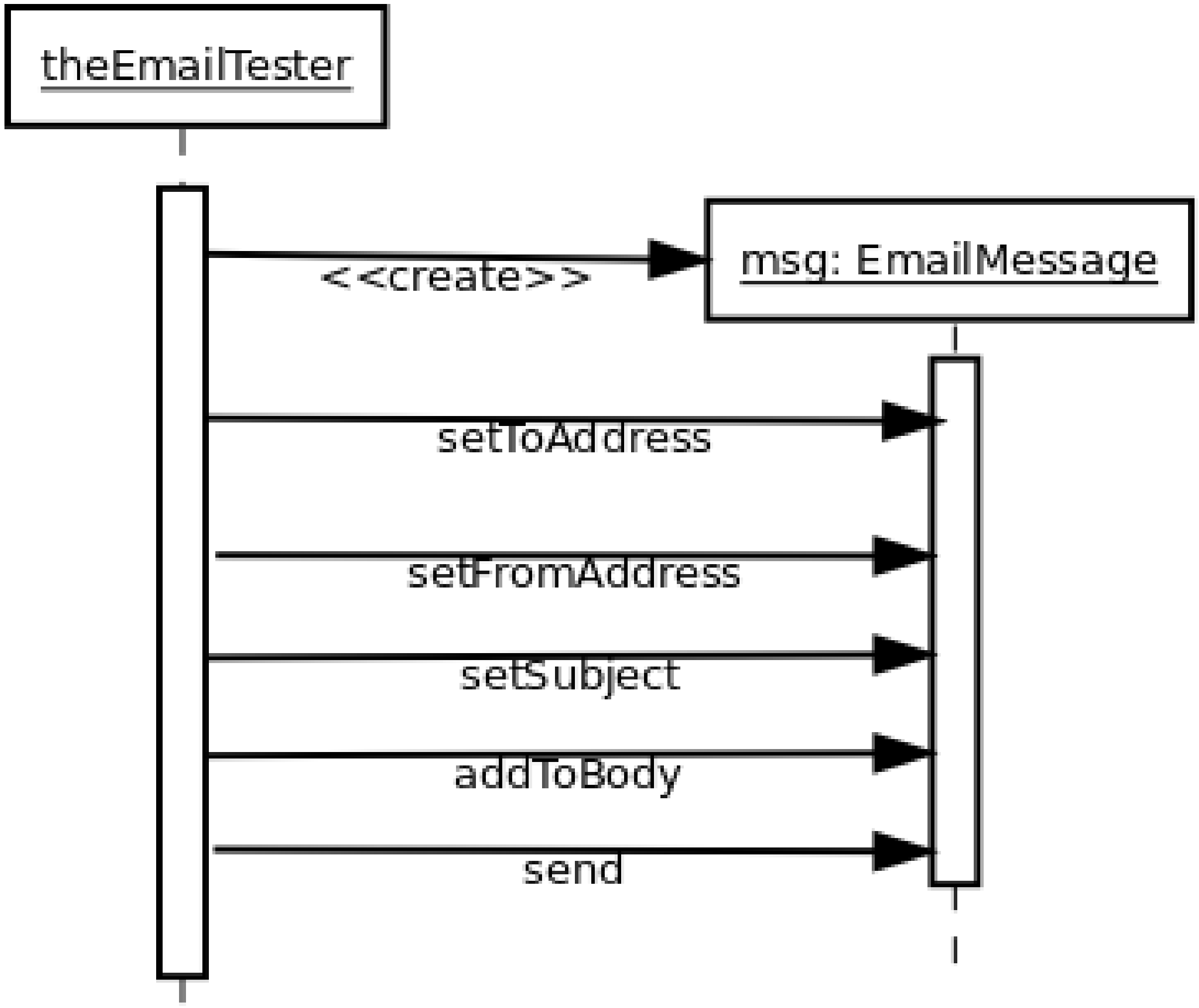
Creating sequence diagrams can help you to:

- ***Discover new classes***
- ***Identify new methods***
- ***Refine your ideas about relationships***

Class Diagram for Exercise 1

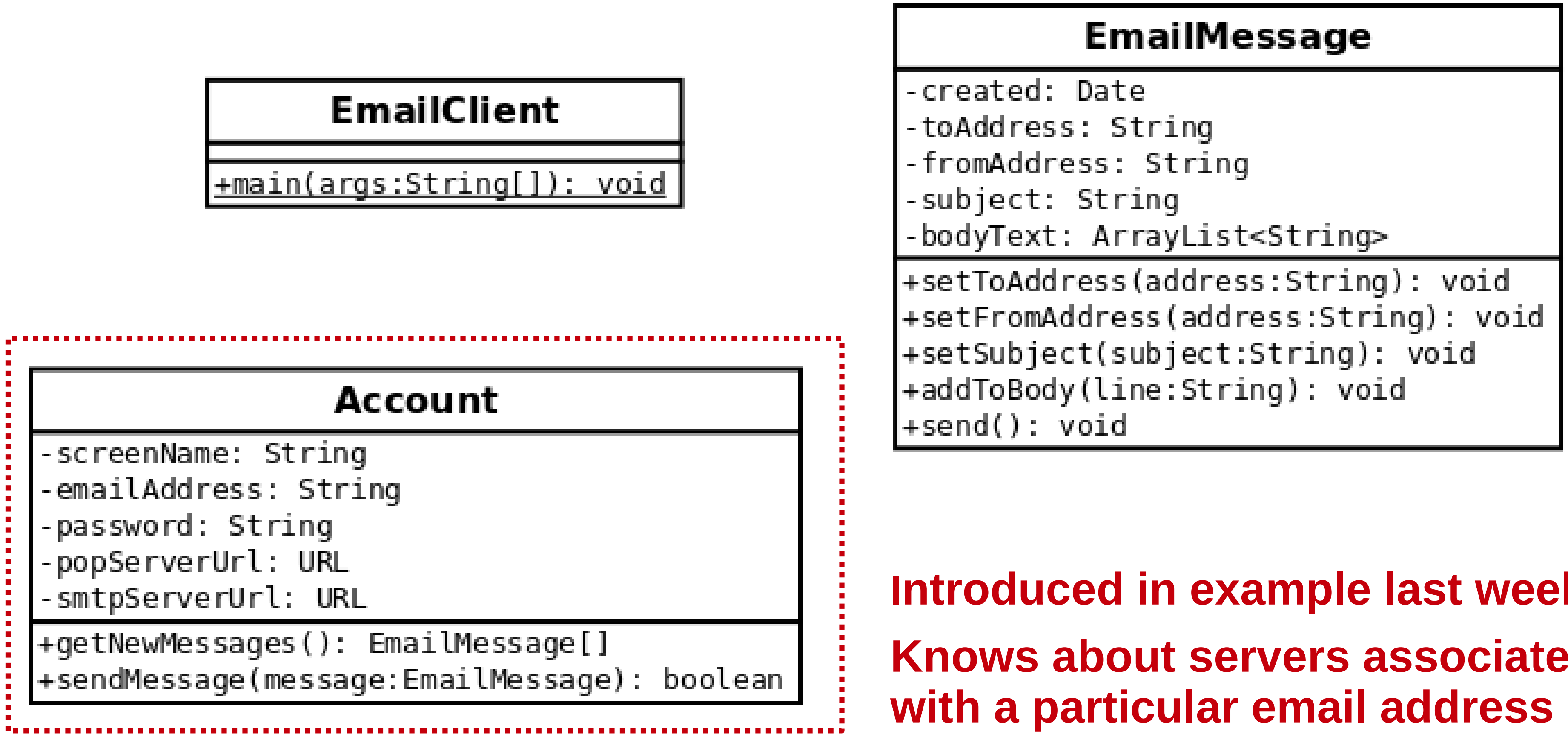

EmailTester
+main(args:String[]): void
EmailMessage
-created: Date
-toAddress: String
-fromAddress: String
-subject: String
-bodyText: ArrayList<String>
+setToAddress(address:String): void
+setFromAddress(address:String): void
+setSubject(subject:String): void
+addToBody(line:String): void
+send(): void

Sequence Diagram 1



This was fine for our simple Java exercise, but what if we want to start building an actual email application?

Add Account Class

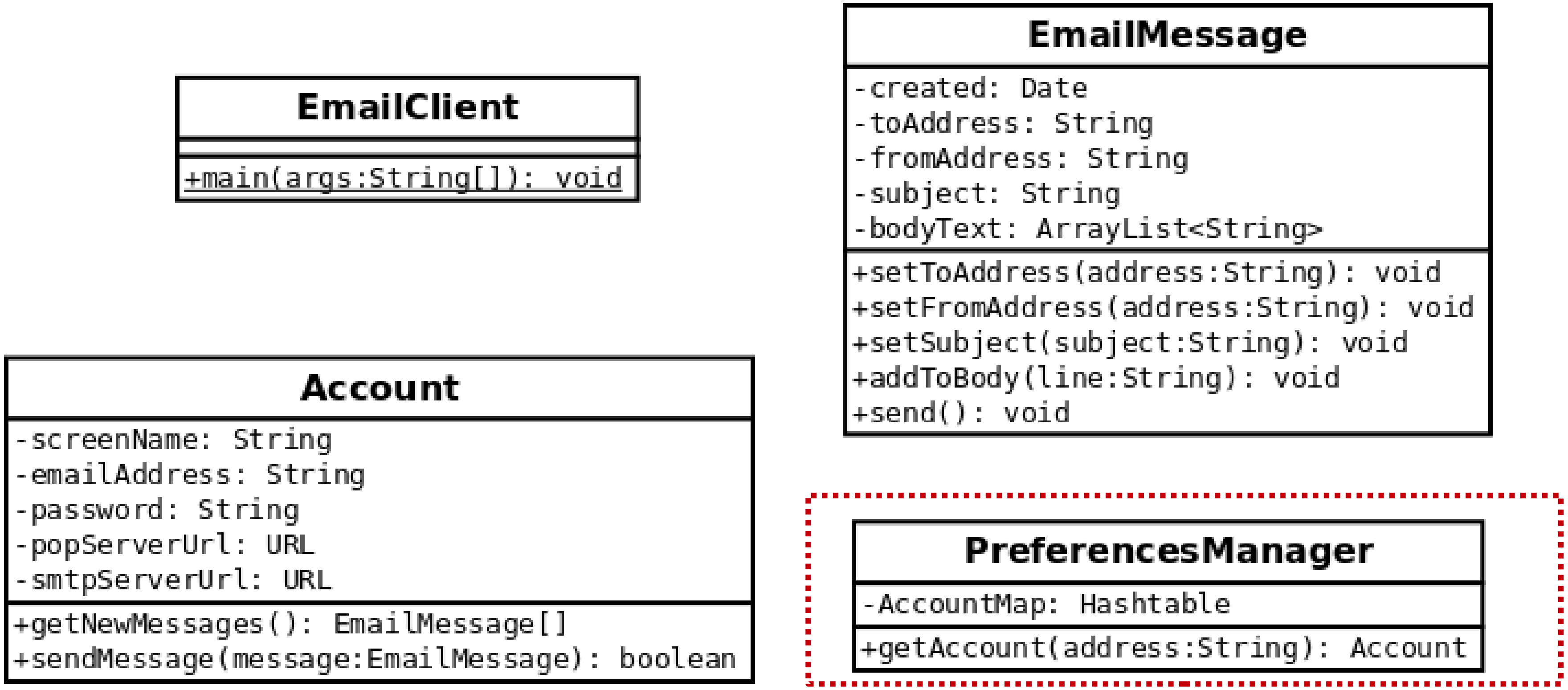

EmailClient
+main(args:String[]): void
EmailMessage
-created: Date
-toAddress: String
-fromAddress: String
-subject: String
-bodyText: ArrayList<String>
+setToAddress(address:String): void
+setFromAddress(address:String): void
+setSubject(subject:String): void
+addToBody(line:String): void
+send(): void
Account
-screenName: String
-emailAddress: String
-password: String
-popServerUrl: URL
-smtpServerUrl: URL
+getNewMessages(): EmailMessage[]
+sendMessage(message:EmailMessage): boolean

Introduced in example last week

Knows about servers associated with a particular email address

Knows how to send and receive emails via those servers

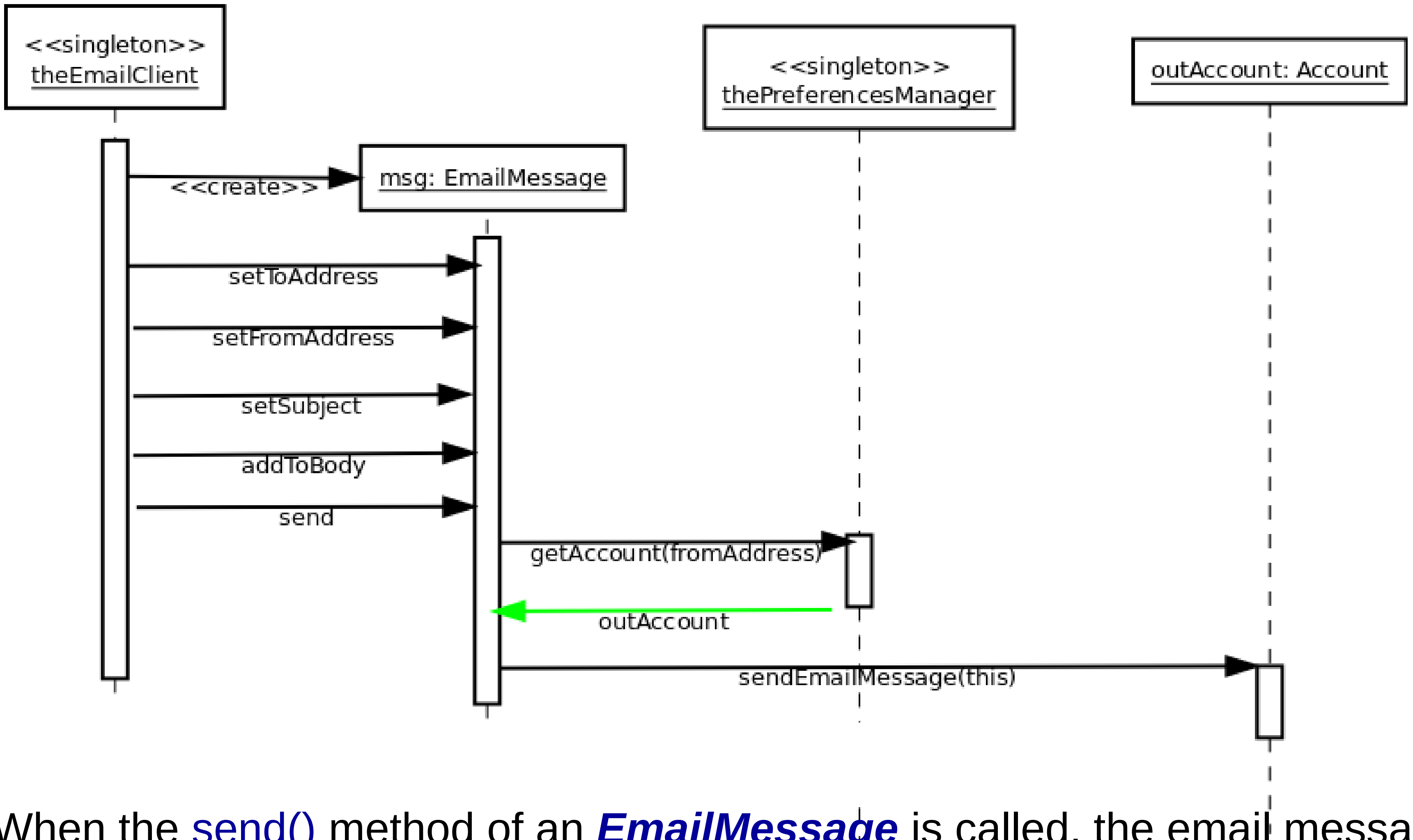
Who knows which account to use?


EmailClient
+main(args:String[]): void
EmailMessage
-created: Date
-toAddress: String
-fromAddress: String
-subject: String
-bodyText: ArrayList<String>
+setToAddress(address:String): void
+setFromAddress(address:String): void
+setSubject(subject:String): void
+addToBody(line:String): void
+send(): void
Account
-screenName: String
-emailAddress: String
-password: String
-popServerUrl: URL
-smtpServerUrl: URL
+getNewMessages(): EmailMessage[]
+sendMessage(message:EmailMessage): boolean
PreferencesManager
-AccountMap: Hashtable
+getAccount(address:String): Account

To implement the send() method on ***EmailMessage***, need the ***Account***

We introduce a new class, ***PreferencesManager***, to associate email addresses with accounts

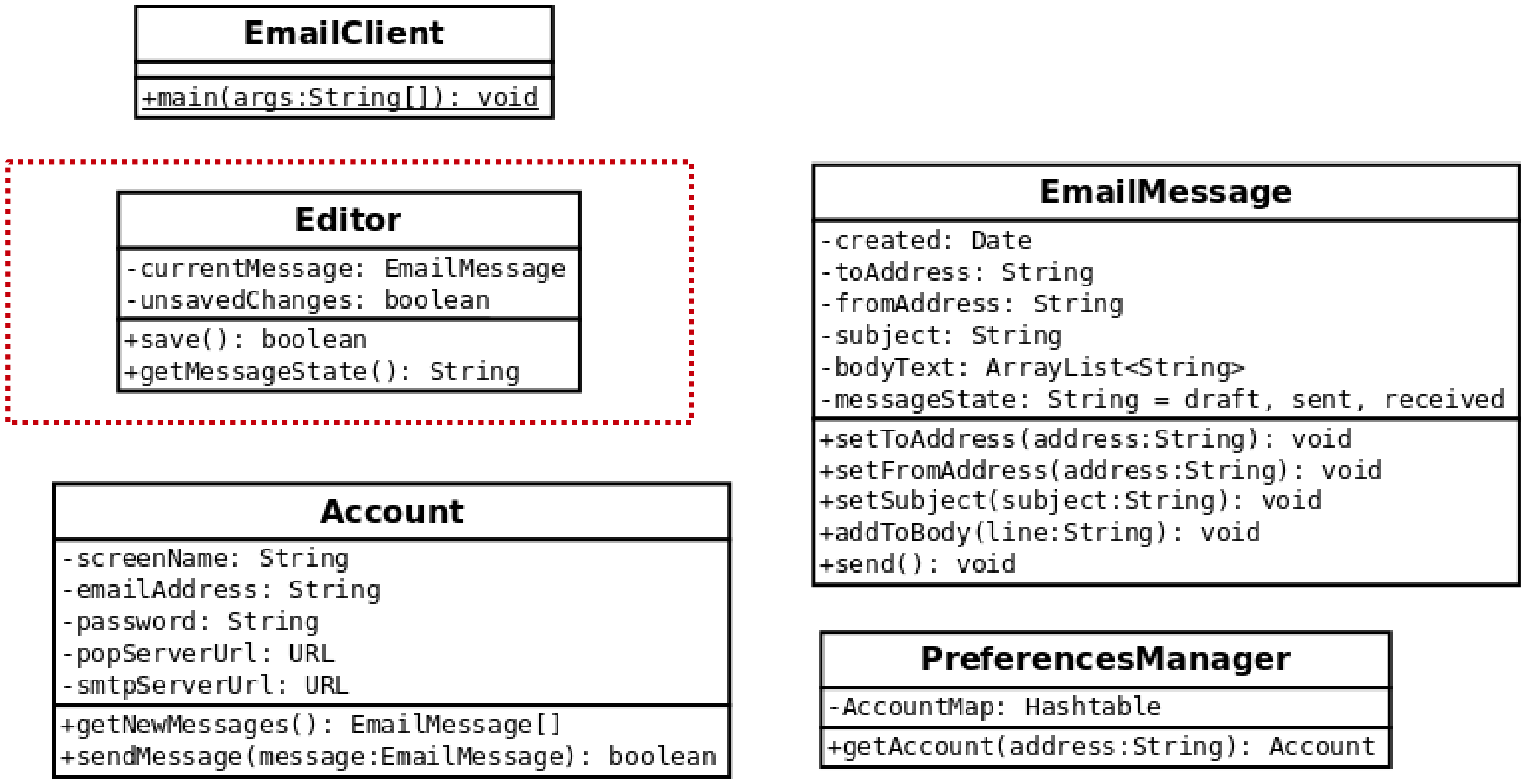
Sequence Diagram 2



When the send() method of an ***EmailMessage*** is called, the email message:

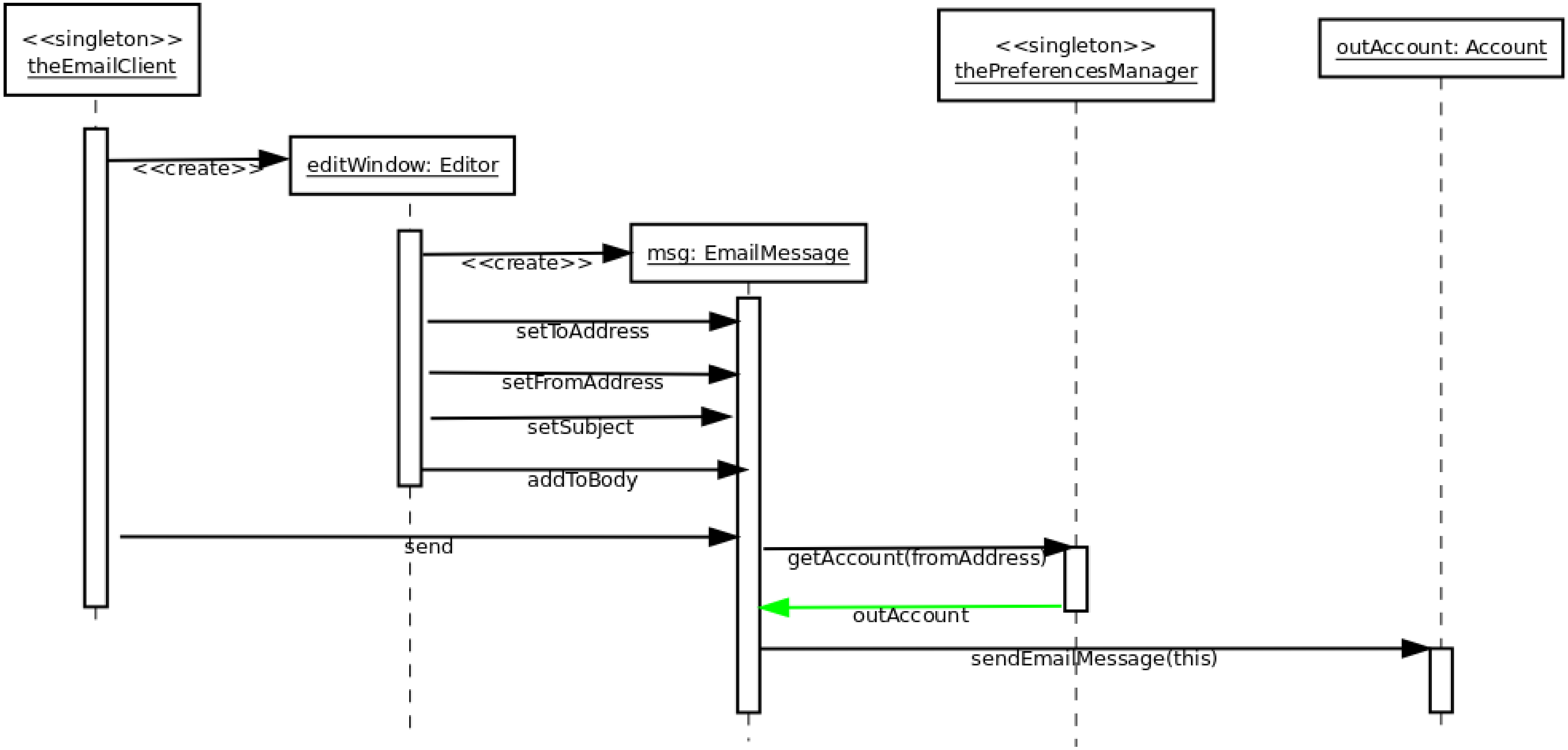
1. Calls getAccount() method on the ***PreferencesManager***
2. Gets the ***Account*** object back as a return value
3. Calls the sendEmailMessage() method on the account object

Should the *EmailClient* directly create *EmailMessage* objects?


EmailClient
+main(args:String[]): void
Editor
-currentMessage: EmailMessage
-unsavedChanges: boolean
+save(): boolean
+getMessageState(): String
Account
-screenName: String
-emailAddress: String
-password: String
-popServerUrl: URL
-smtpServerUrl: URL
+getNewMessages(): EmailMessage[]
+sendMessage(message:EmailMessage): boolean
EmailMessage
-created: Date
-toAddress: String
-fromAddress: String
-subject: String
-bodyText: ArrayList<String>
-messageState: String = draft, sent, received
+setToAddress(address:String): void
+setFromAddress(address:String): void
+setSubject(subject:String): void
+addToBody(line:String): void
+send(): void
PreferencesManager
-AccountMap: Hashtable
+getAccount(address:String): Account

Add an ***Editor*** class to enter and modify email text.

Sequence Diagram 3



The ***Editor*** controls the content of the ***EmailMessage***

However, the top level client is still responsible for the send() command

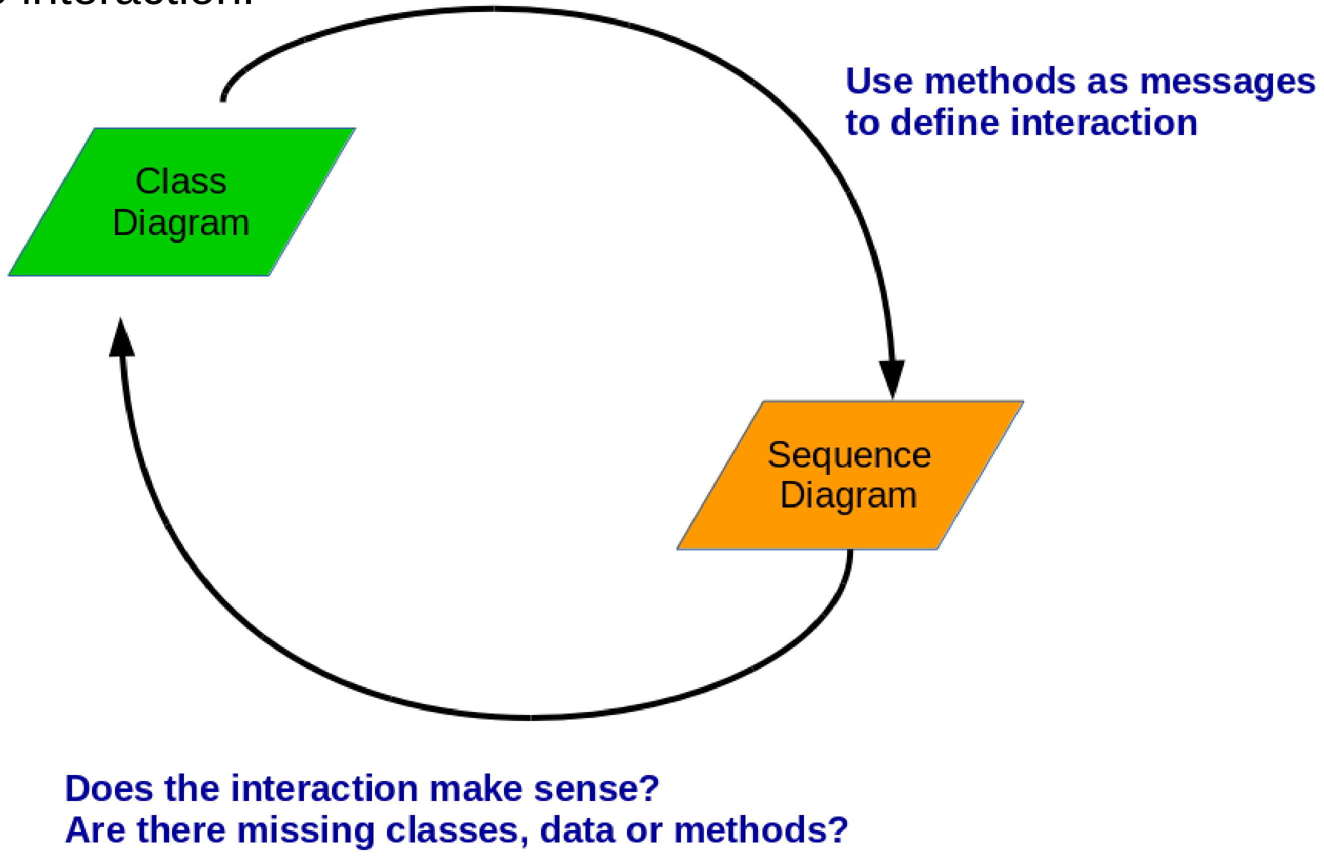
Iterating between Structure and Interaction

This example shows the true benefit of using UML.

The diagrams focus the designer's attention on specific aspects of the system design.

Studying interaction suggests changes to structure.

Modified structure changes the interaction.


Class Diagram
Use methods as messages to define interaction
Sequence Diagram
Does the interaction make sense?
Are there missing classes, data or methods?

More sequence diagram notation

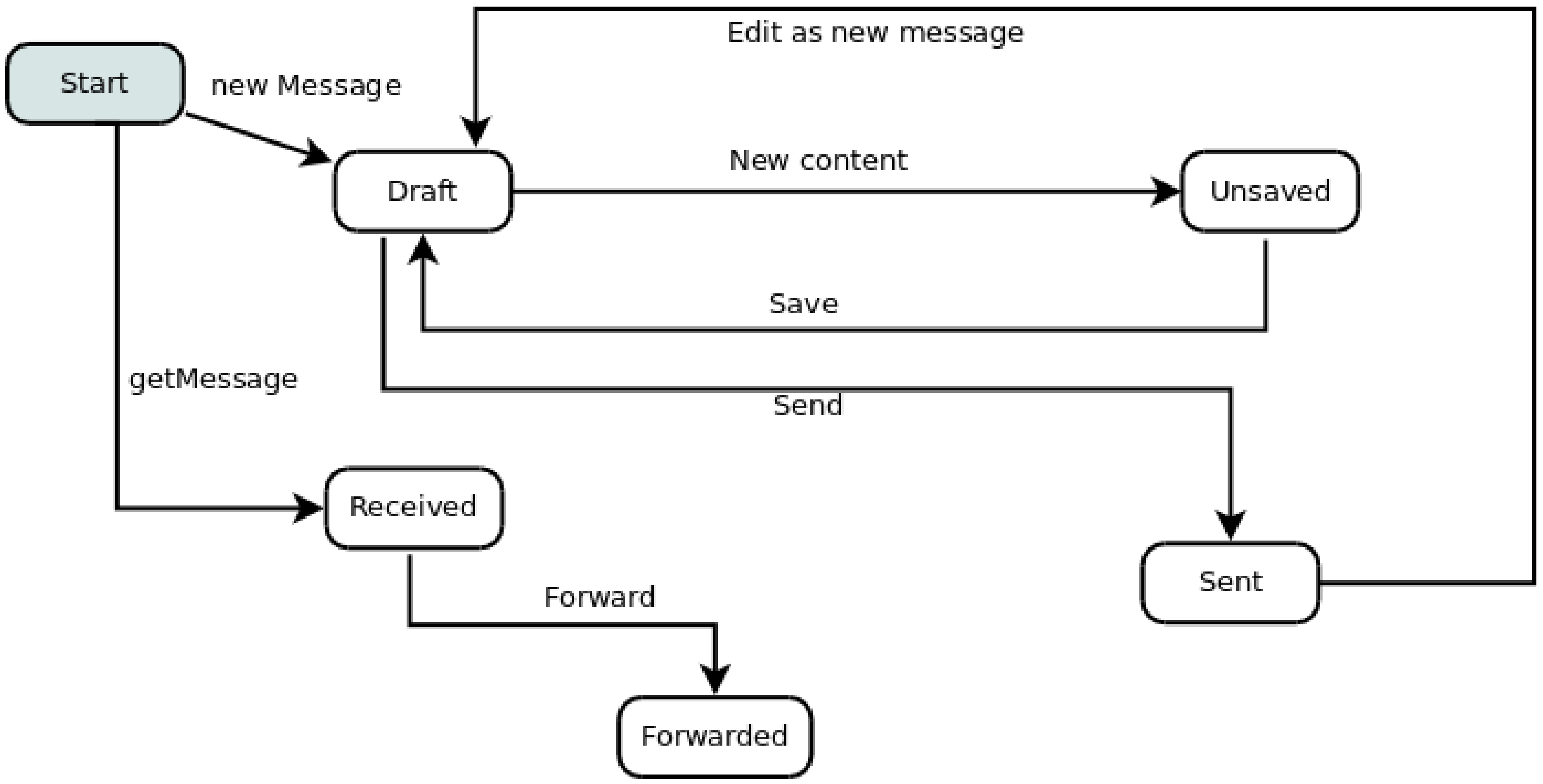
- Self-messages – when an object calls one of its own methods
- Iteration and conditional notations
  - different between UML 1 and UML 2
- Synchronous versus asynchronous messages
- Notations for showing passed and returned information

Use what you think you need

Sequence diagrams in general are not good for capturing detailed logic

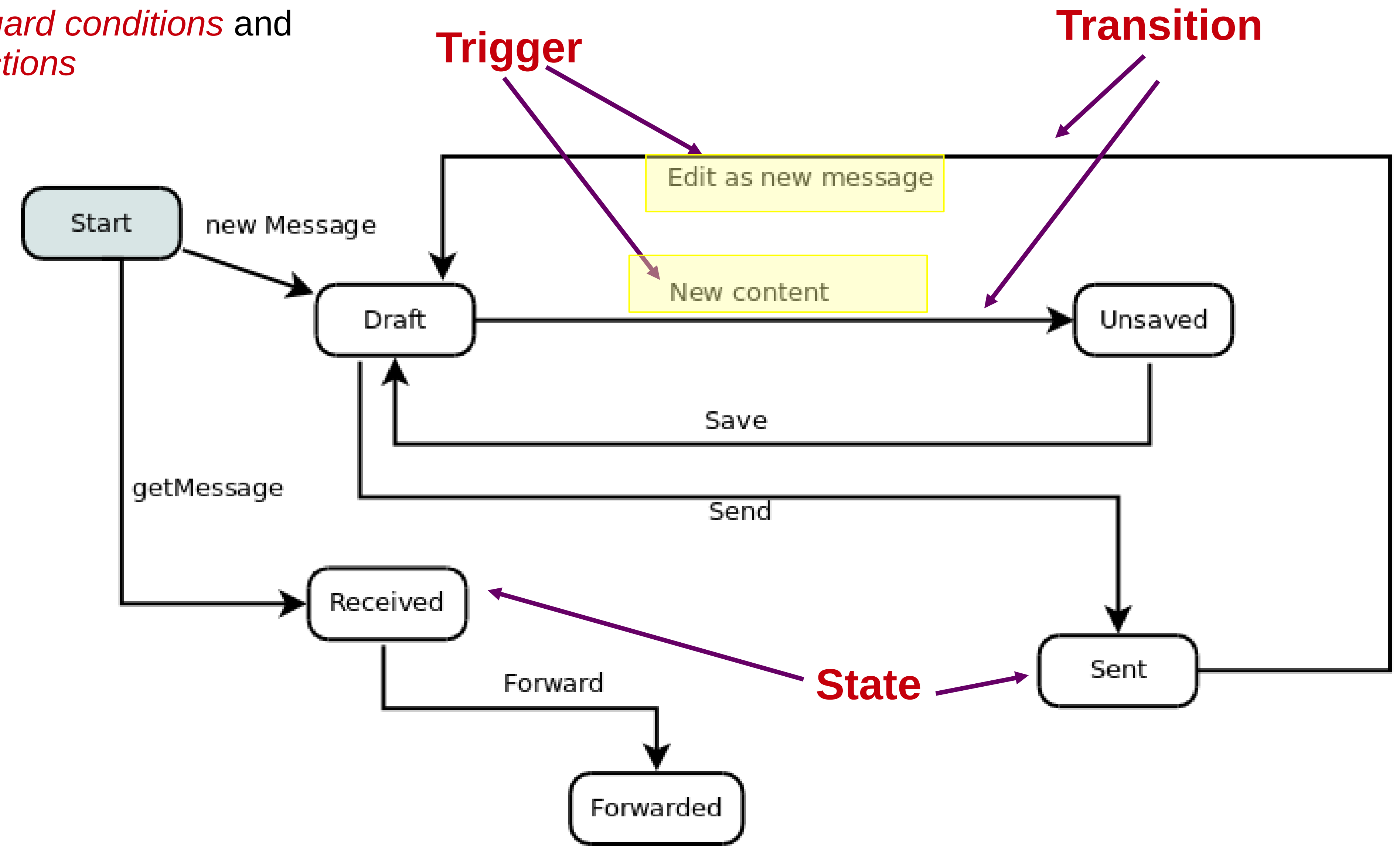
State Diagrams

State diagram for ***EmailMessage***

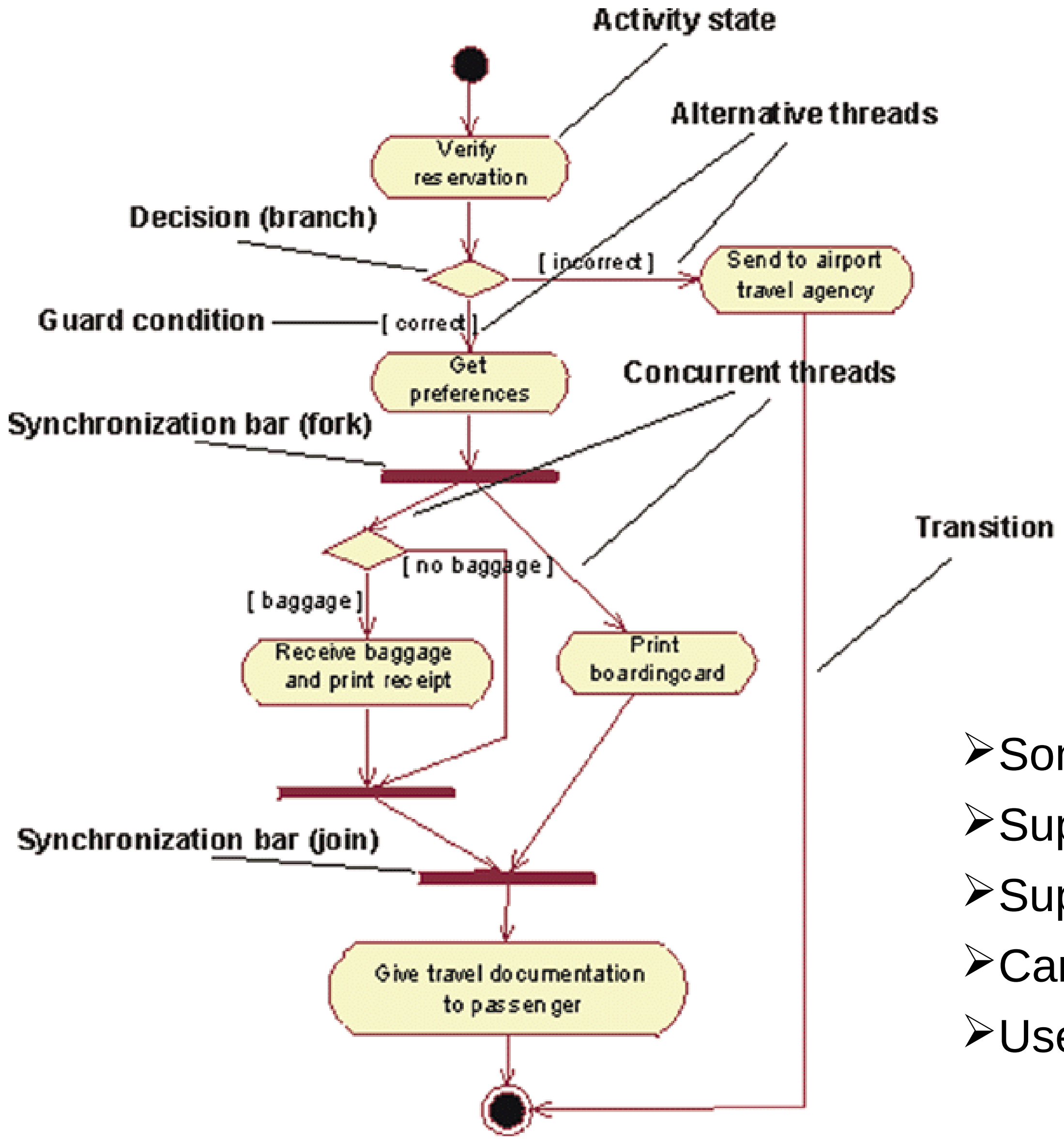


State Diagrams

Transitions can also have *guard conditions* and *actions*



Activity Diagrams



Airport Check-In

- Somewhat similar to flow charts
- Support concurrent activities
- Support external “signals”
- Can get complicated
- Used in many different ways

https://www.ibm.com/developerworks/rational/library/2802.html

Next Few Weeks

No class next week – midterm exam period (no exam in this class)

March 12 – Introduction to “Design Patterns”

March 19 – Evaluating OO Designs

March 26, April 2 - In-class critique sessions

First draft of project design document due **Tuesday March 17 by noon** (electronic submission – will be a link on the course home page)

Project Design Document

PDF document with the following content (in this order):

1. Title page – topic, team name, team members
2. Abstract – one to two paragraph description of your system
3. Use case diagram
4. Use case narratives (text) for each use case in diagram
5. Class diagram(s)
6. Sequence diagrams for main success scenario of each use case
7. A list of unresolved issues about your design – things you are not sure about

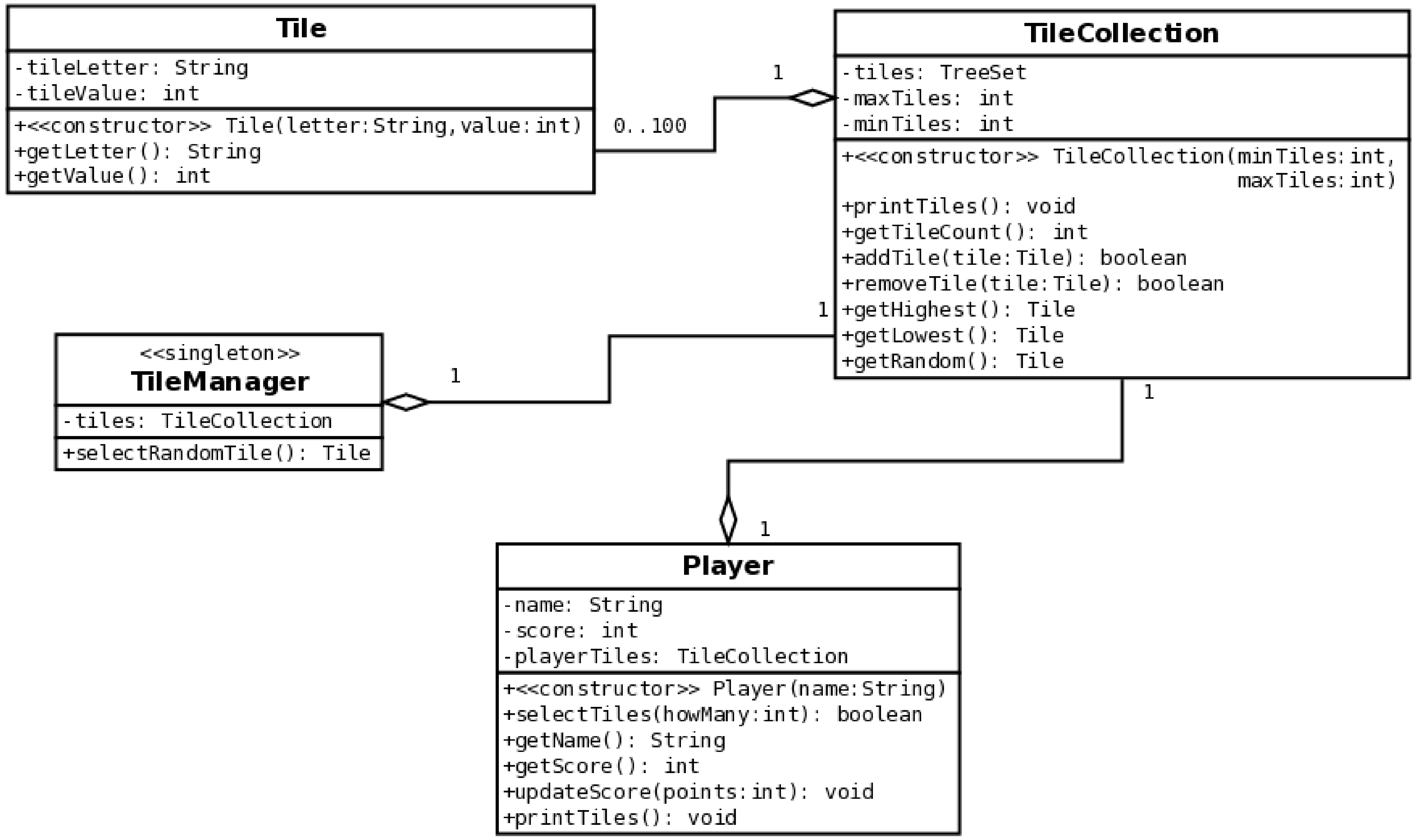
- ➢ **Please put titles on all diagrams. Please be sure all are readable.**
- ➢ **Use UML notes to explain things as necessary.**
- ➢ **Break up class diagrams into multiple sub-diagrams if necessary.**
- ➢ **No hand-drawn diagrams accepted.**
- ➢ **Remember your classmates will be seeing and evaluating your design!**

Assignments

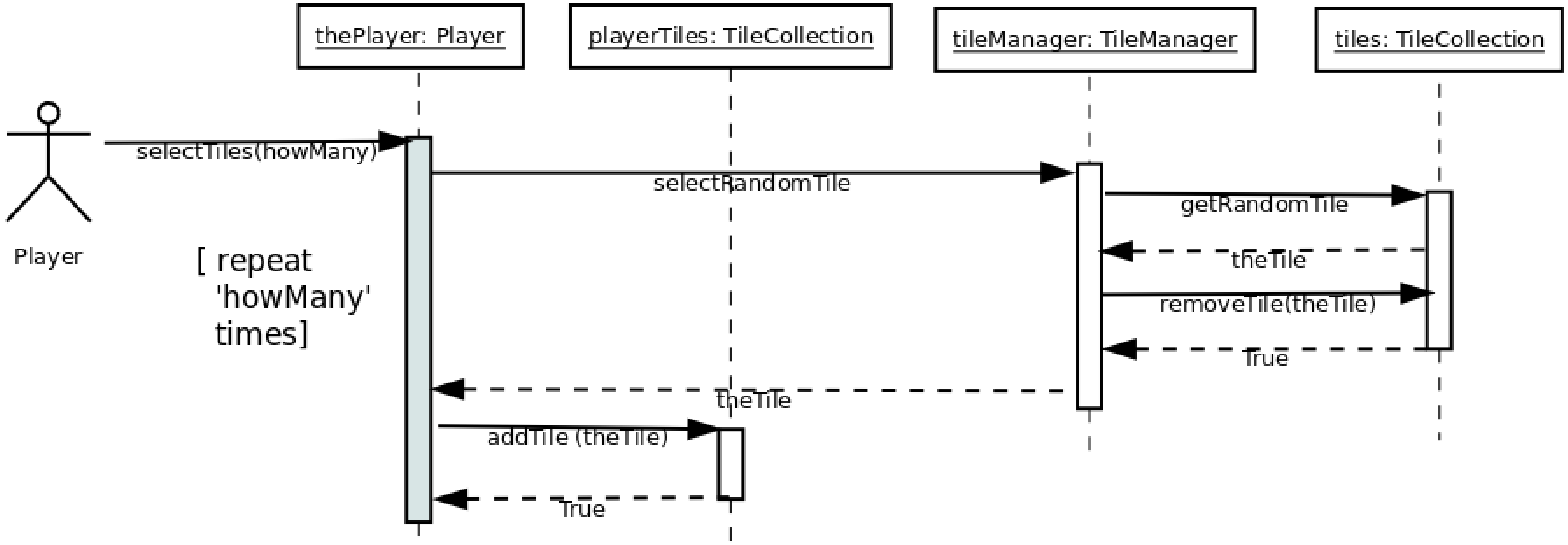
- **Read chapter 4 from *UML Distilled* (on web page)**
- **Given the class diagram and the sequence diagram on the following pages, *implement and test* these classes and methods in Java**

Due on Tuesday, March 12th by 17:00. Upload your Java files in a zip or tar (not rar) file, using the usual link. This is an individual, not a team assignment.

Class Diagram for Exercise



Sequence Diagram for Exercise



Notes on UML Diagram Content

- Methods on *TileCollection* and *Player* that return boolean values will return true, unless the operation violates the limits (maxTiles, minTiles) of the *TileCollection*
- Create a main() method in the *Player* class to test your code. This method should call the Player() constructor to create an instance of a player, and then call the selectTiles() method on that instance. Then call printTiles() to show the tiles that were selected.
- Pass a value of 7 for the howMany argument to selectTiles(). This should also be the maximum limit on the playerTiles tile collection.