Object-Oriented Software Design

Dr. Julie Greensmith
(additional material by Dr. Peer Olaf Siebers)

Overview

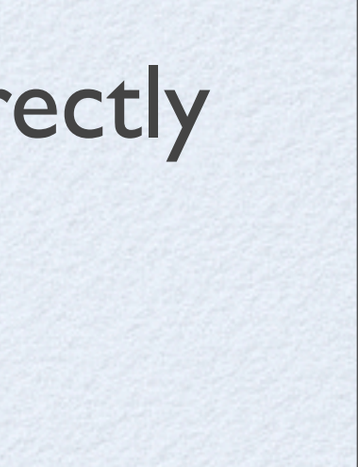
- What is object orientation
  - different from procedural and functional languages
- What are objects?
- Modelling object oriented systems
- Unified Modelling Language Taster

Todays examples

- Mostly in C++ and Python

How do we link functions and data?

- In procedural languages there is no way to directly associate functions with data structures
  - close using a `struct` in C
    - struct Ship{int xpos, int ypos}
    - can group data together
- Object orientation solves this problem



All About Objects

- *What's in the box?*

Data

Code

Message Interface

What is a Class?

- A class is a blueprint for an object
- It is a higher order data structure
- Contains both data and procedures
- Aggregated into a single structure
- Classes have attributes and methods

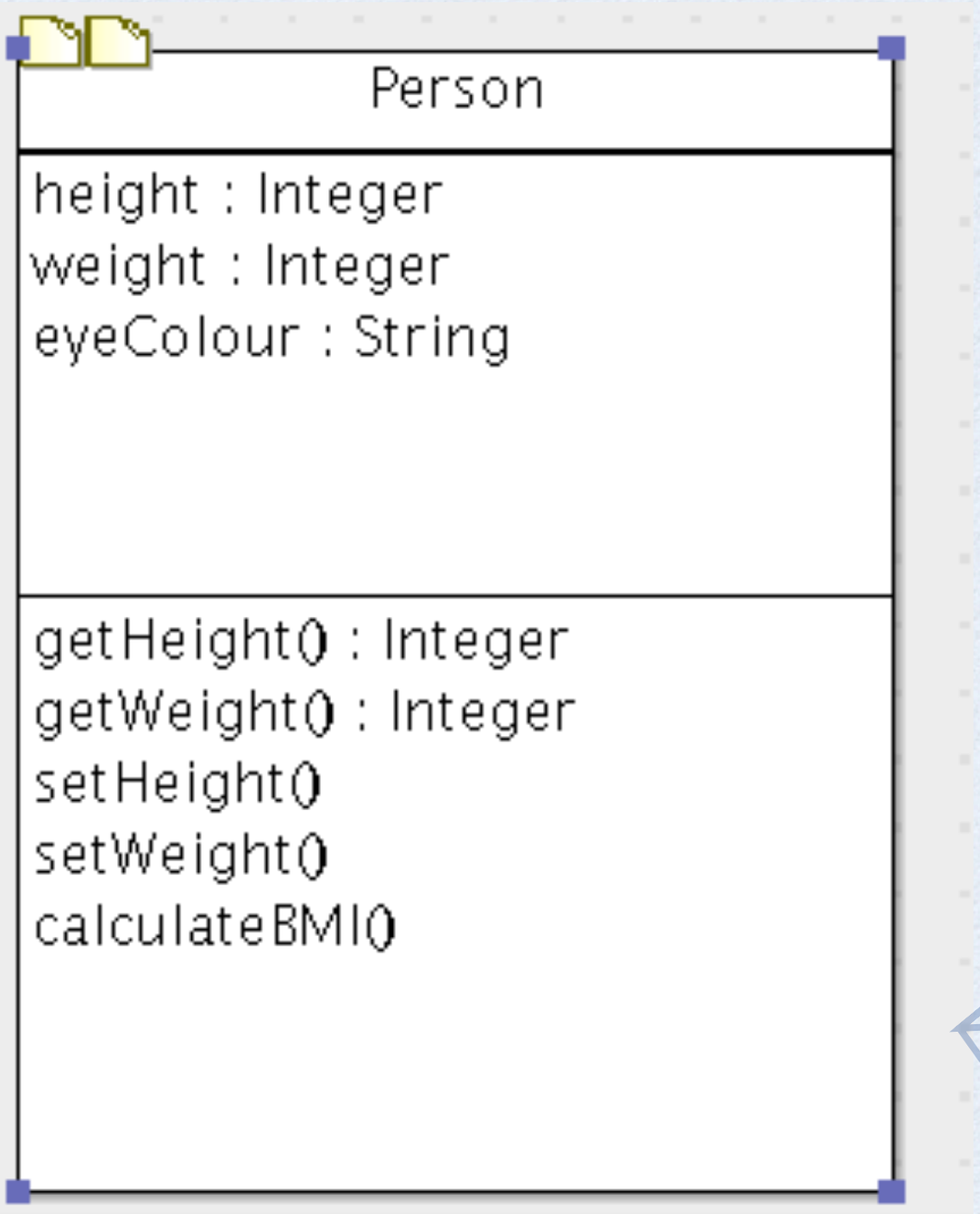
The Registrar's Data Structure

- For example the following code represents the type of program a university or college registrar might want to use to manipulate student records

```
>>> class Student:
        school ='CS'
        def calculateGrade(self):
            ...
        return finalGrade
```

An Abstract View of a Class

attributes



methods

Making an Instance- *Instantiation*

- We have the blueprint in our constructed class
- Make use of the class by defining an instance
- Use either a driver program (C++) or a driver class to then 'call' the methods of the class
- An instance is a data structure inherited from a class
- An instance can contain new or additional methods

Back to the Registrar

- We define an instance in python of two students: Jack and Jill

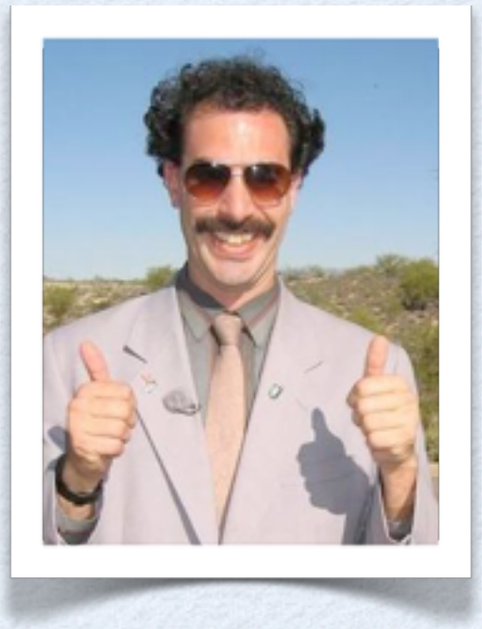
```
jill = Student()
jill.school='CS'

john = Student()
john.aveGrade()
```

We can add extra methods to act on a current object instance making the code extensible

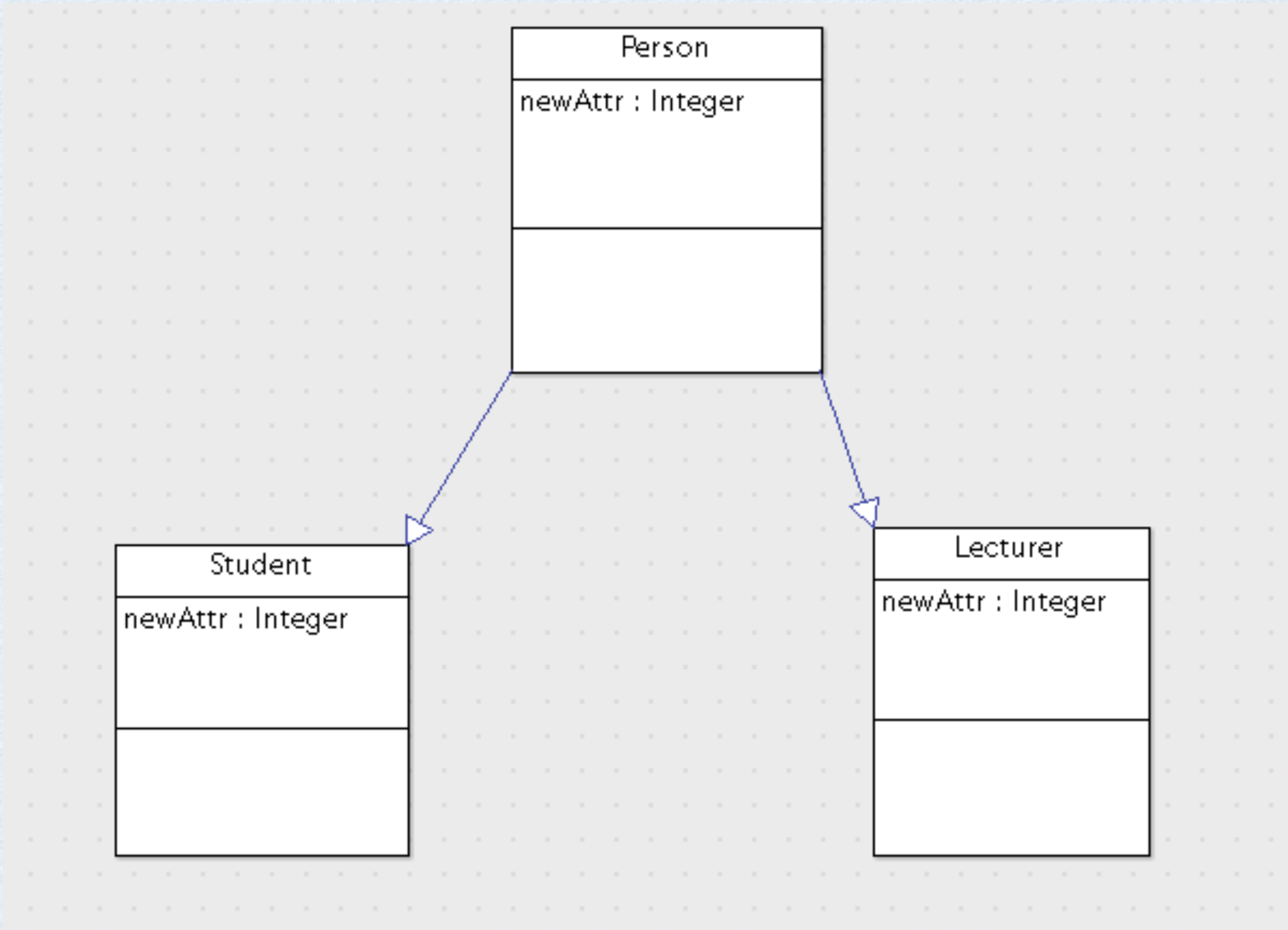
Inheritance

- By way of abstraction you can have inheritance of higher order objects
- Decompose classes into subclasses

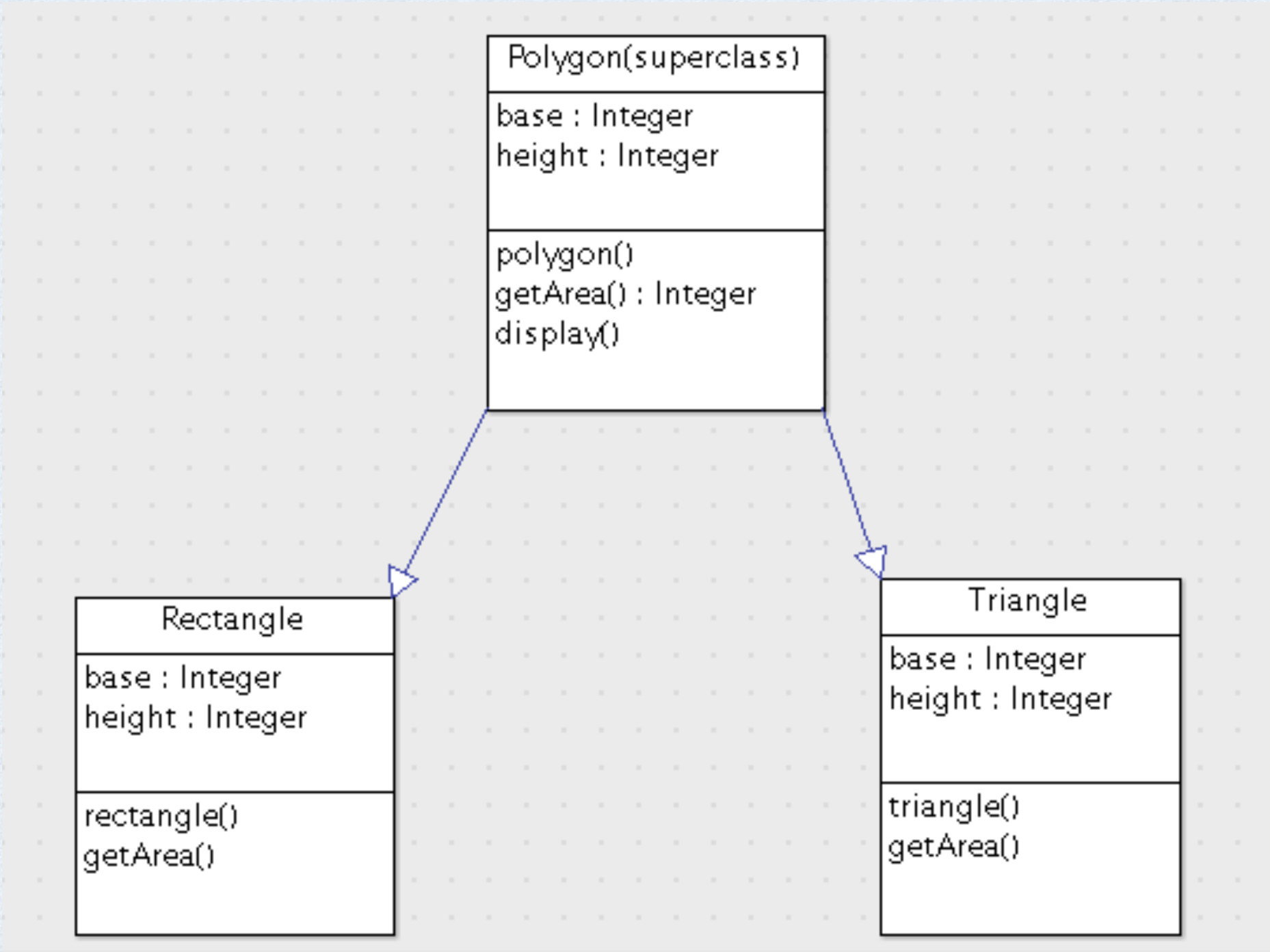
- Without repeating the methods of the ‘superclass’
- Saves time
- encourages good code reuse



What might the Student and Staff classes inherit from the Person data structure?



Another Inheritance Example



Abstraction

- You have defined your necessary components to achieve desired functionality, but there may be more methods you need to program this reliably
- eg. data of customerID and method add( );
- Maybe we want additional methods to validate the input to the method add( ), so we implement the auxilliary methods
- CALL the methods from the add function
- All the methods are not necessary for the abstract class model

Encapsulation

- Implement a method to “hide” other methods
- e.g. lots of set/get invocations lumped into one initialise function
- Only the object's own methods can directly inspect or manipulate its own fields
- WHY? Simplify the programmers message interface with the class

Polymorphism

- Overriding the methods in the class for a particular instantiation or sub class

- Triangle and Rectangle both use getArea( )
  - both use this function differently
- In Java this is done using the “super” syntax

A Quick Recap

- OBJECTS are black boxes which contain data and functions called attributes and methods
- A CLASS is a blueprint for an object
- Make an instance of the class using a driver
- Classes can inherit properties from other classes via INHERITANCE
- Includes abstraction, encapsulation and polymorphism

Modelling with UML

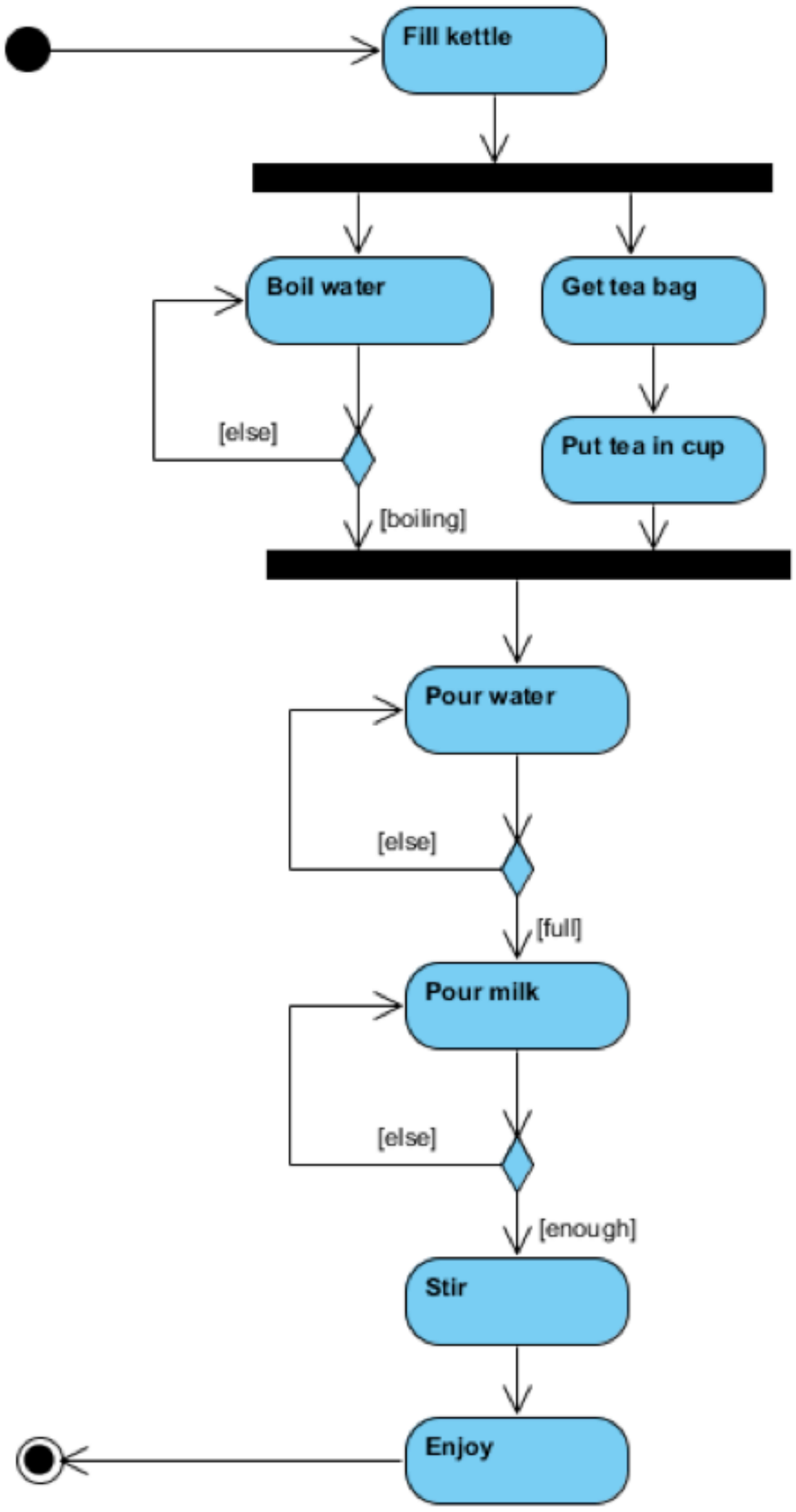
- There are many ways of modelling systems
- A model is an abstract representation of a process
- Use models to derive designs
- Use of a modelling framework so engineers can work together to understand and implement the models

Lets Make a Cuppa

What steps do we want to go through?

Steps

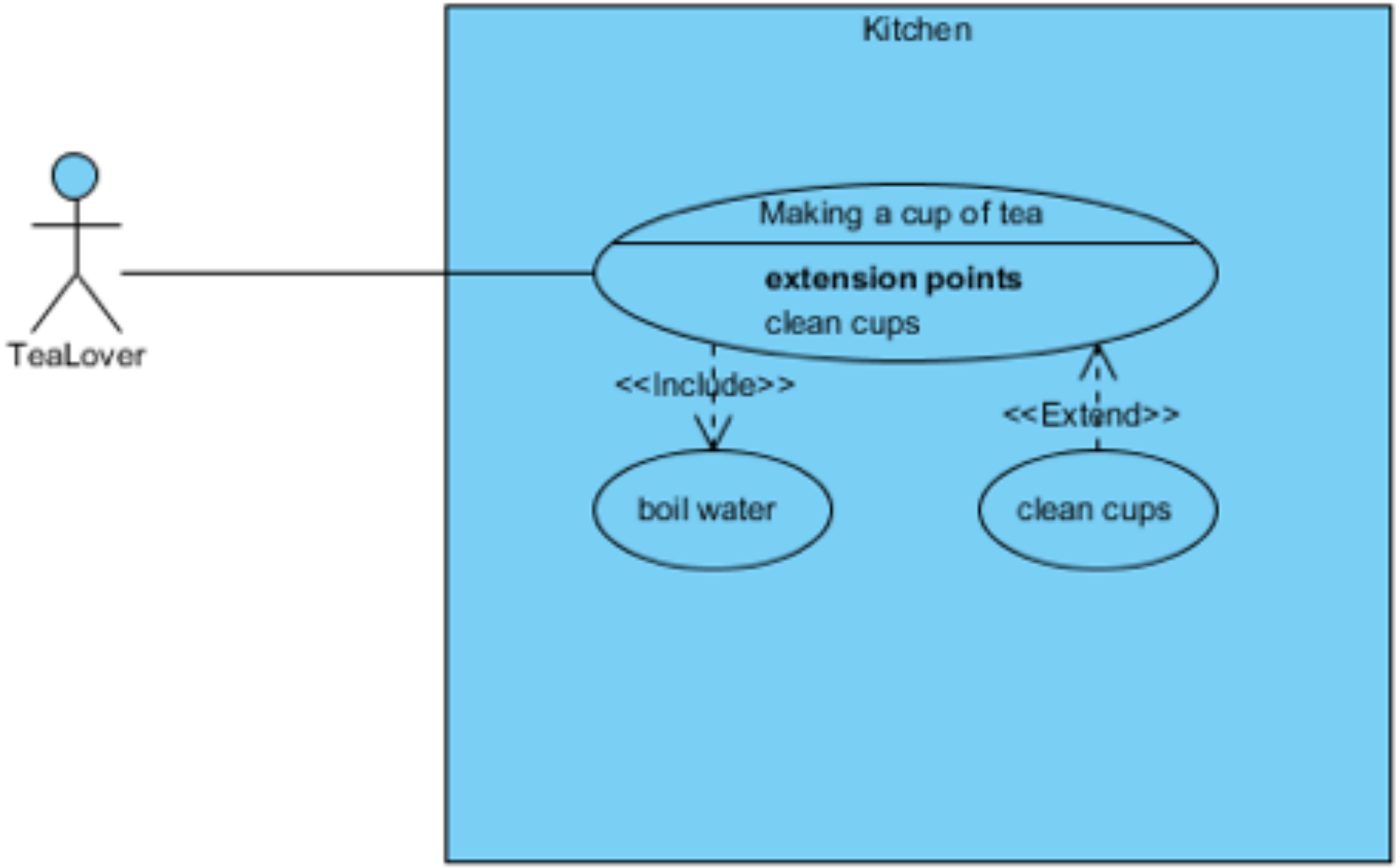
1. Get cup
2. Get kettle
3. Get teabag
4. Get milk
5. Get spoon
6. (Get water)
7. Fill kettle (water)
8. Boil kettle
9. Put teabag into cup
10. Pour water until full
11. Add milk
12. Stir with spoon



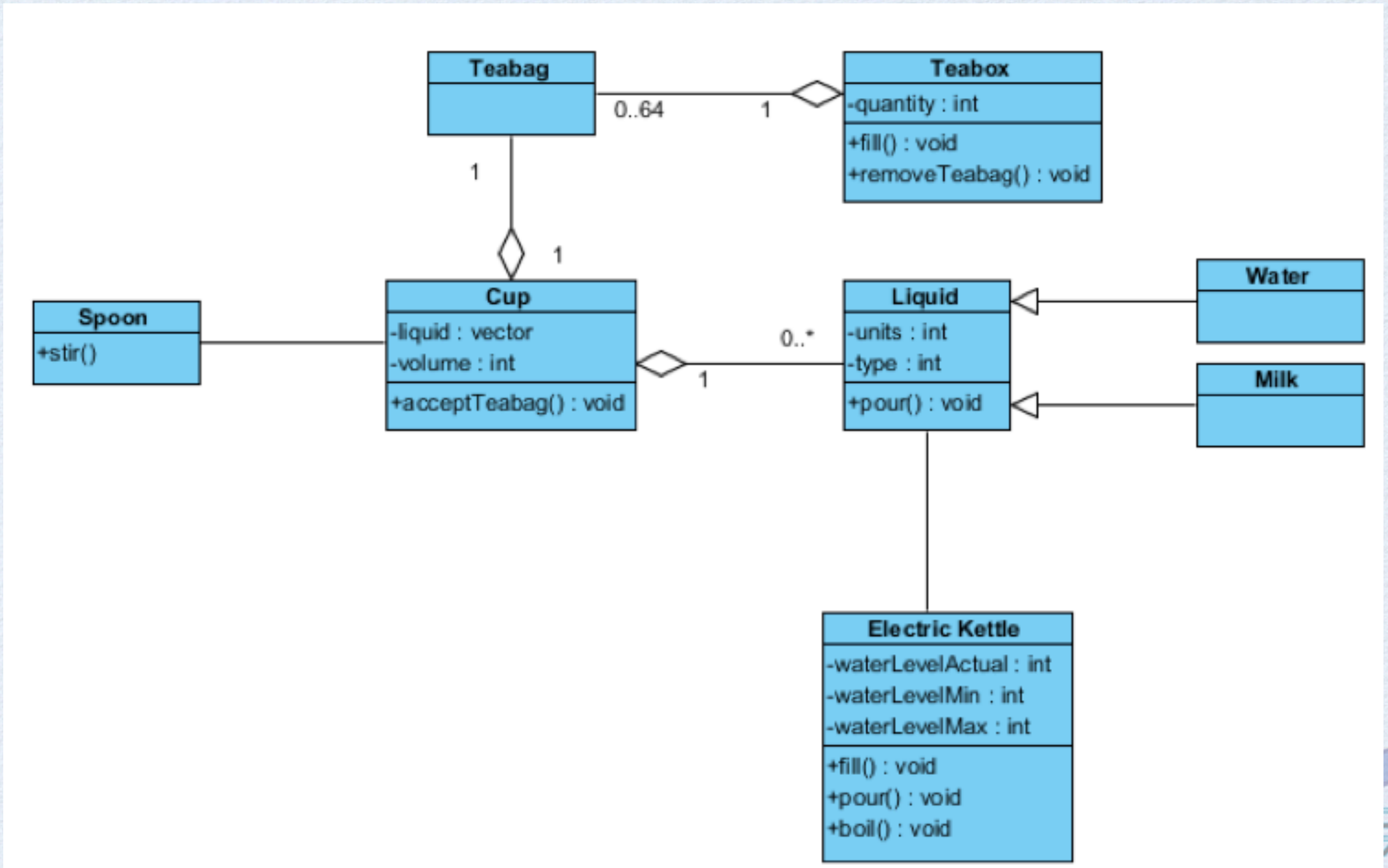
How can we model this process?

Use a sequence diagram
Showing parallelism

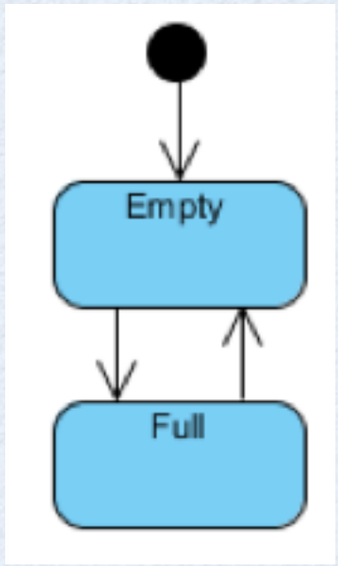
What about the stakeholder?



Class Diagrams



State Charts for Modelling State Machines

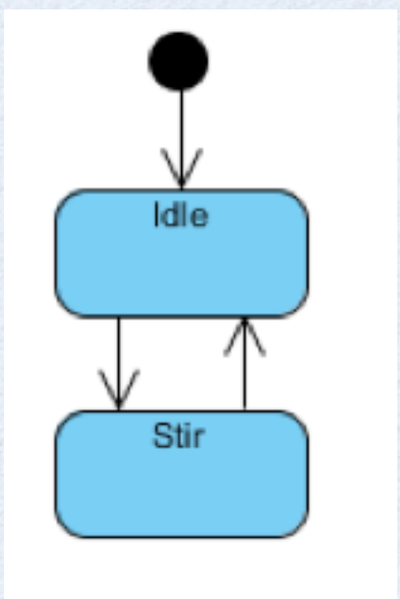


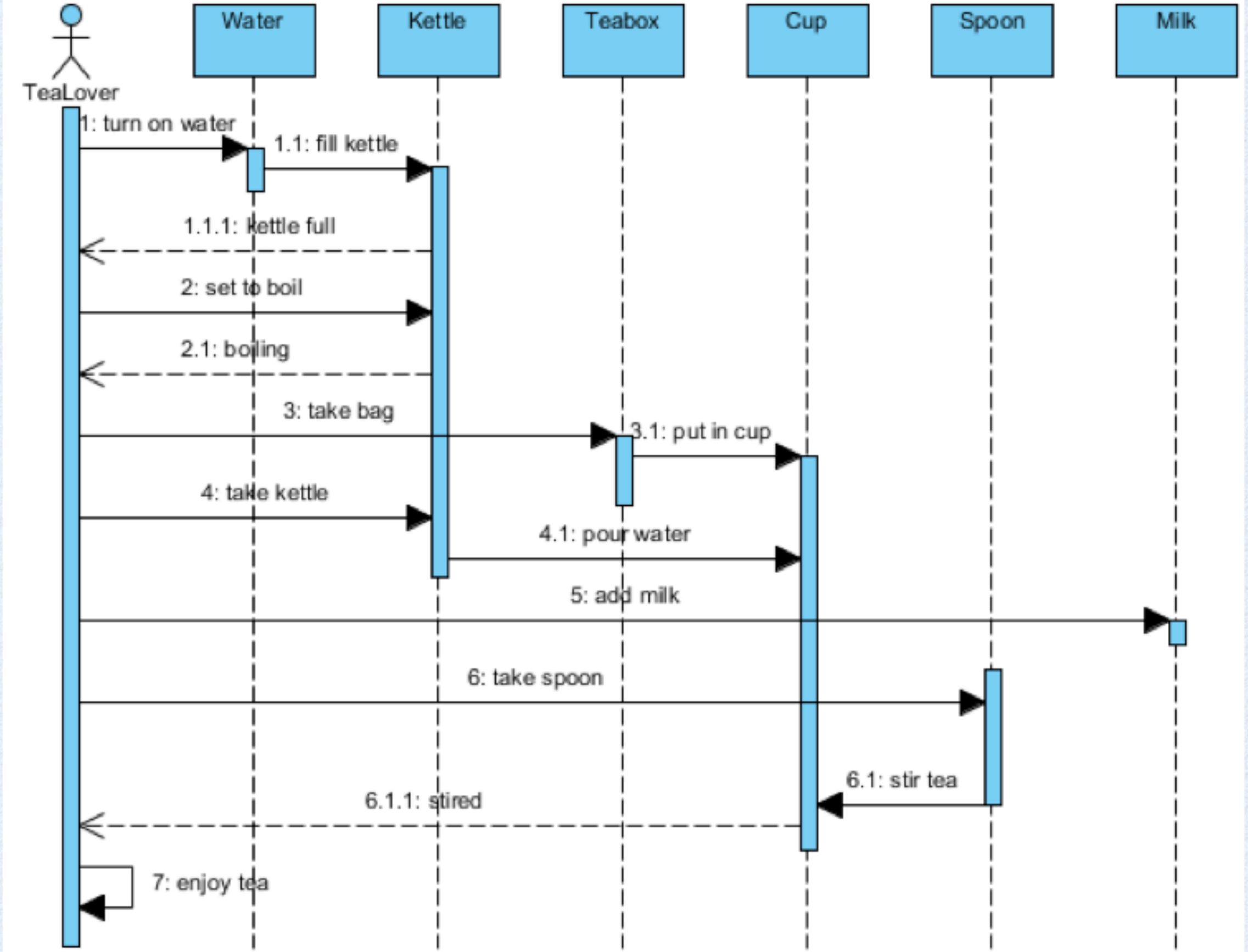
Kettle

Cup

Teabox?

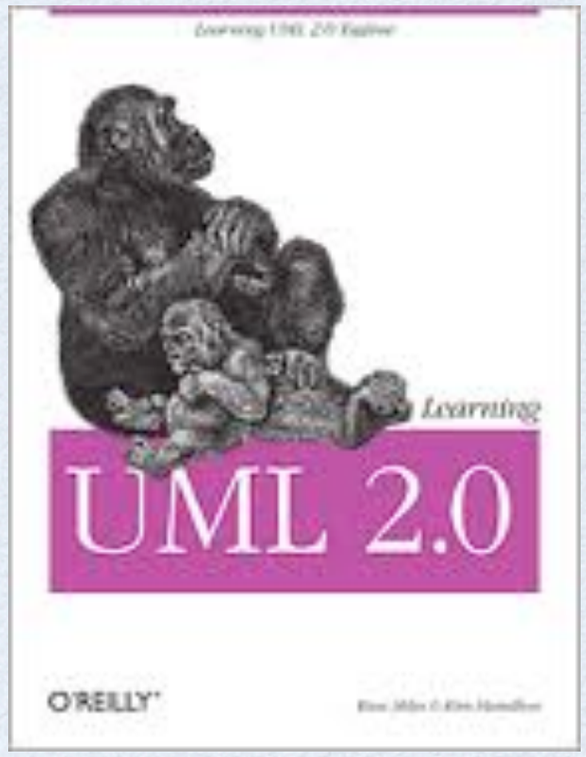
Spoon




TeaLover
Water
Kettle
Teabox
Cup
Spoon
Milk
1: turn on water
1.1: fill kettle
1.1.1: kettle full
2: set to boil
2.1: boiling
3: take bag
3.1: put in cup
4: take kettle
4.1: pour water
5: add milk
6: take spoon
6.1: stir tea
6.1.1: stired
7: enjoy tea

Many different UML Diagrams

- Class Diagrams
- Use Case Diagrams
- State Charts
- Activity Diagrams
- Deployment Diagrams



Automatically Generating Code

- Some UML editing packages auto generate code from your UML diagrams
- Can be extremely useful and time saving
- Still need to fill in the functionality of the methods
- Also need to construct driver class/program

From the top

- Identify the nouns and verbs to construct your objects
- Identify your classes, superclasses and subclasses
- Reuse your code using inheritance
- Explore different perspectives using a combination of class diagrams, activity diagrams and state charts
- !!Caution, make sure you are using the right kind of arrow

Other types of modelling

- Often based on functional programming paradigms
- Allow you to verify that parts of a component are actually going to do computationally what you want them to do
- E.g. Coq which is a model prover (G52MC2) Can be quite complicated!

- Formal specification languages including 'Z'
- Modelling the system using states, state variables and operations

STATE ::= patients | fields | setup | ready | beam_on

EVENT ::= select_patient | select_field | enter | start | stop | ok | intlk

FSM == (STATE EVENT) STATE

Learn as much as you can!

- Object orientation will be a big part of your life if you chose to continue in Computing
- The more you read, the more you can apply
- In prep for this lecture I watched at least 4 different online lectures
  - MIT Open Courseware Introduction to Object Oriented Programming is a great place to start
- Moving on to using objects in python in the Tuesday lab
- Try modelling something trivial so you can get the hang of it!
  - what other examples can you think of?