Why Inheritance?

1. Organization (during analysis):
   - Inheritance helps us with the construction of taxonomies to deal with the application domain
     - when talking the customer and application domain experts we usually find already existing taxonomies
2. Reuse (during object design):
   - Inheritance helps us to reuse models and code to deal with the solution domain
     - when talking to developers

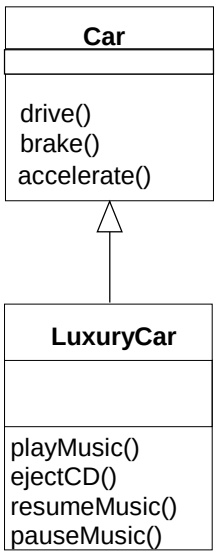
The use of Inheritance

- Inheritance is used to achieve two different goals
  - Description of Taxonomies
  - Interface Specification
- Description of Taxonomies
  - Used during *requirements analysis*
  - Activity: identify application domain objects that are hierarchically related
  - Goal: make the analysis model more understandable
- Interface Specification
  - Used during *object design*
  - Activity: identify the signatures of all identified objects
  - Goal: increase reusability, enhance modifiability and extensibility

Inheritance can be used during Modeling as well as during Implementation

- Starting Point is always the requirements analysis phase:
  - We start with use cases
  - We identify existing objects ("class identification")
  - We investigate the relationship between these objects; "Identification of associations":
    - general associations
    - aggregations
    - inheritance associations.

Example of Inheritance



Superclass:

```
public class Car {
    public void drive() {…}
    public void brake() {…}
    public void accelerate() {…}
}
```

Subclass:

```
public class LuxuryCar extends Car
{
    public void playMusic() {…}
    public void ejectCD() {…}
    public void resumeMusic() {…}
    public void pauseMusic() {…}
}
```

Inheritance comes in many Flavors

Inheritance is used in four ways:

- Specialization
- Generalization
- Specification Inheritance
- Implementation Inheritance.

Discovering Inheritance

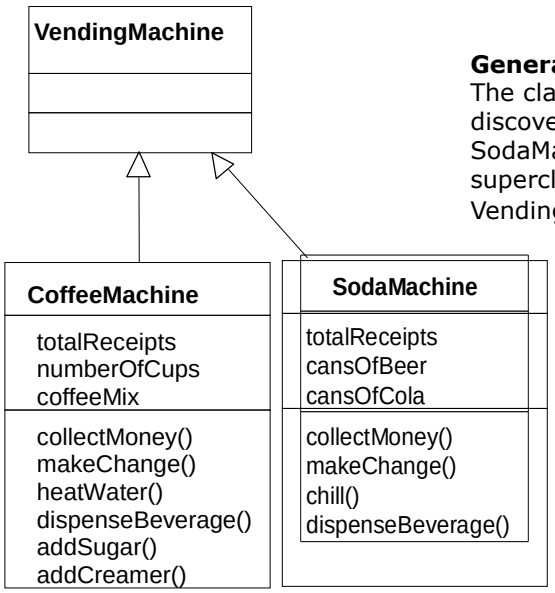
- To "discover" inheritance associations, we can proceed in two ways, which we call specialization and generalization

- Generalization: the discovery of an inheritance relationship between two classes, where the sub class is discovered first.
- Specialization: the discovery of an inheritance relationship between two classes, where the super class is discovered first.

Generalization

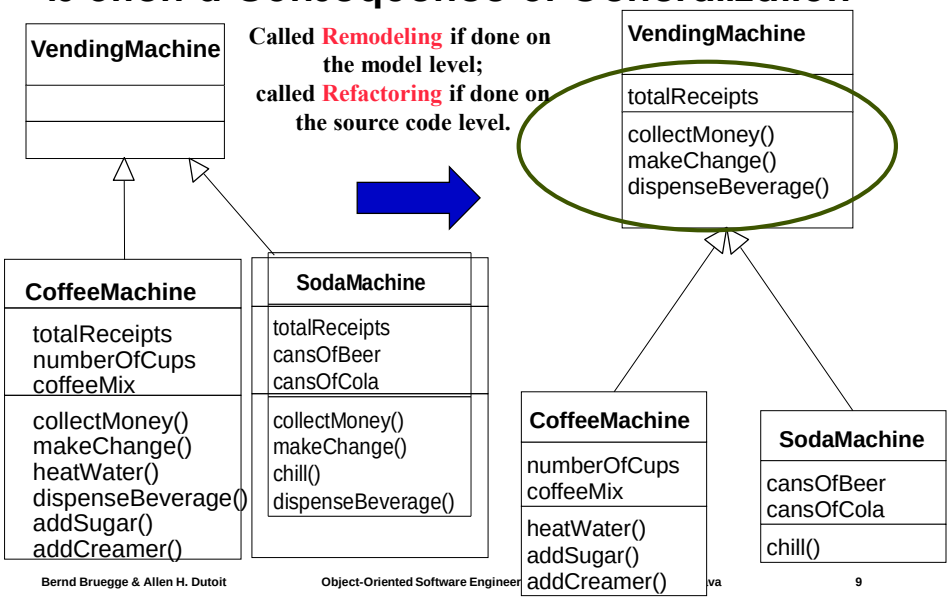
- First we find the subclass, then the super class
- This type of discovery occurs often in science and engineering:
  - **Biology**: First we find individual animals (Elefant, Lion, Tiger), then we discover that these animals have common properties (mammals).
  - **Engineering:** What are the common properties of cars and airplanes?

Generalization Example: Modeling a Coffee Machine

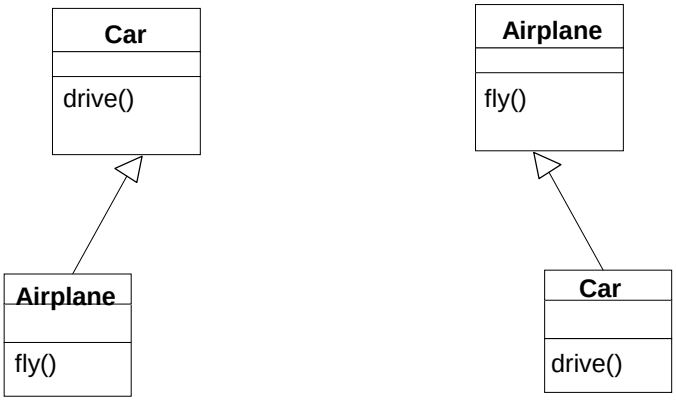


Generalization:
The class CoffeeMachine is discovered first, then the class SodaMachine, then the superclass VendingMachine

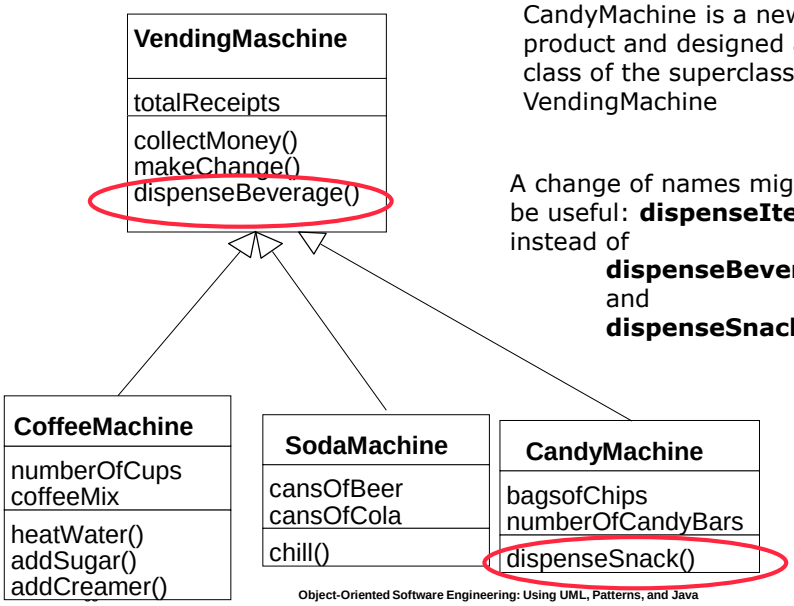
Restructuring of Attributes and Operations is often a Consequence of Generalization



Which Taxonomy is correct for the Example in the previous Slide?



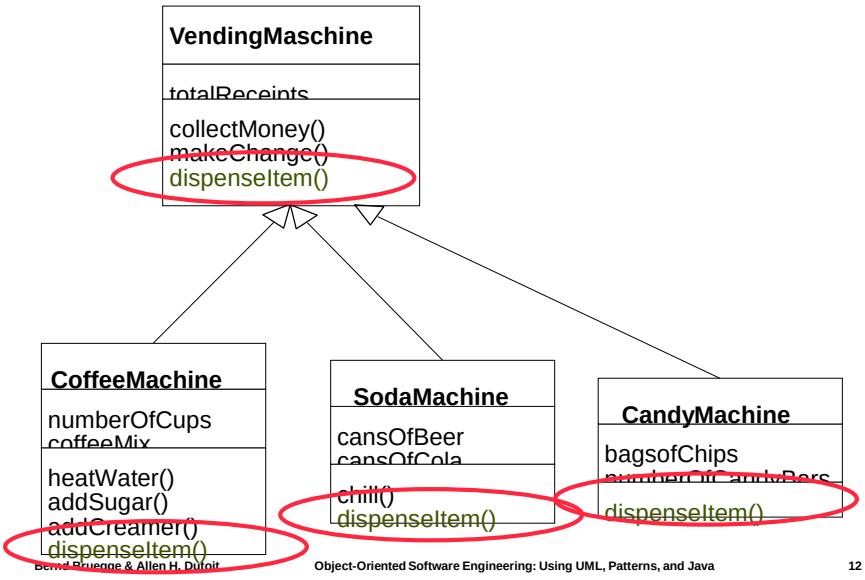
Another Example of a Specialization



CandyMachine is a new product and designed as a sub class of the superclass VendingMachine

A change of names might now be useful: **dispenseItem()** instead of **dispenseBeverage()** and **dispenseSnack()**

Example of a Specialization (2)



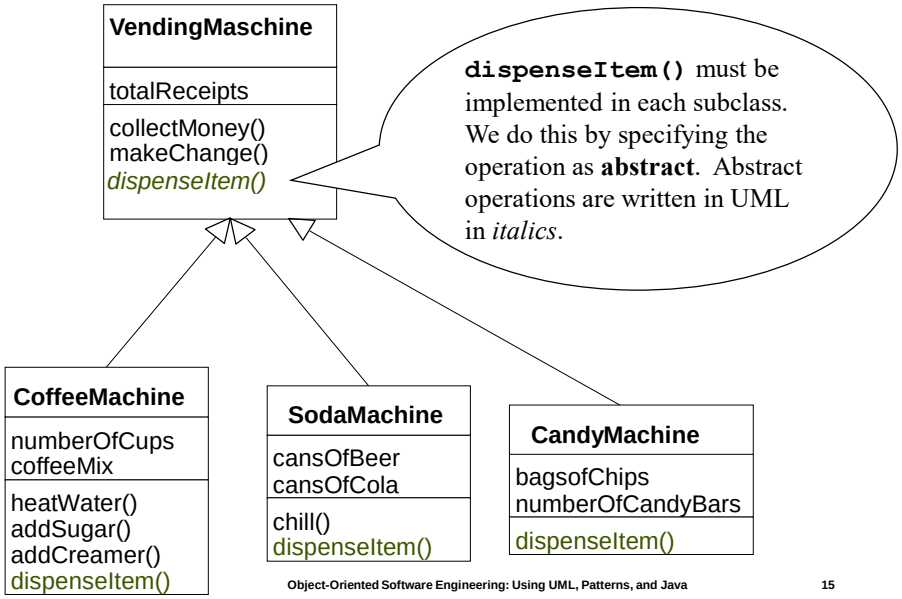
Implementation Inheritance vs. Specification Inheritance

- Implementation Inheritance: The combination of inheritance and implementation
  - The Interface of the superclass is completely inherited
  - Implementations of methods in the superclass ("Reference implementations") are inherited by any subclass
- Specification Inheritance: The combination of inheritance and specification
  - The Interface of the superclass is completely inherited
  - Implementations of the superclass (if there are any) are not inherited.

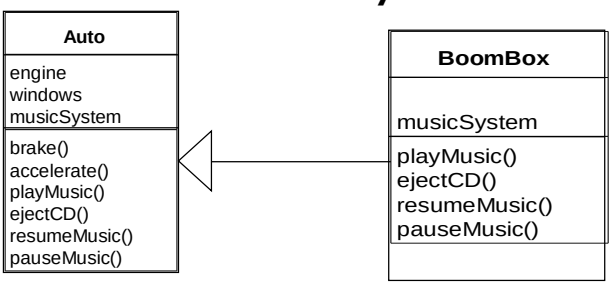
Abstract Methods and Abstract Classes

- Abstract method:
  - A method with a signature but without an implementation (also called abstract operation)
- Abstract class:
  - A class which contains at least one abstract method is called abstract class
- Interface: An abstract class which has only abstract methods
    - An interface is primarily used for the specification of a system or subsystem. The implementation is provided by a subclass or by other mechanisms.

Example of an Abstract Method



What we do to save money and time



Existing Class:

```
public class Auto {
  public void drive() {…}
  public void brake() {…}
  public void accelerate() {…}
  public void playMusic() {…}
  public void ejectCD() {…}
  public void resumeMusic() {…}
  public void pauseMusic() {…}
}
```

Boombox:

```
public class Boombox
extends Auto {
  public void drive() {};
  public void brake() {};
  public void accelerate()
{};
}
```