Class Relationships

- Consider the following three classes:
  - ProgrammableThermostat.
  - AnalogThermostat.
  - Thermostat.
- A “programmable thermostat” and an “analog thermostat” are thermostats.
  - Anything a thermostat can do, a “programmable thermostat” and an “analog thermostat” can do as well.
  - You should be able to replace a thermostat with either a “programmable thermostat” or an “analog thermostat”.

Inheritance

- The term, inheritance, is used in many object oriented (OO) programming languages to describe the generalization relationship.
- Inheritance is a relationship where one class shares the structure or behavior defined in one class (single inheritance) or more (multiple inheritance).

Terminology

- When the state and behavior of one class is a subset of the state and behavior of another, more general class, the classes are said to be related by inheritance.
- The more general class is referred to as the superclass, or parent.
- The second class is called a subclass of the superclass and is said to inherit the state and behavior of the superclass.
Super and Sub Classes

Is-A and Has-A

- The “Is A” relationship holds between two classes when one class is a specialized instance of the second.
  - a programmable thermostat is a thermostat.
  - a sports car is a car and a minivan is a car.
  - a train is certainly not a car, and a car is not a train.
- The “Has A” relationship holds when one class is a component of the second.
  - a car has a steering wheel, but certainly a car is not a steering wheel.

Inheritance

- Classes can be organized in a hierarchical structure based on the concept of inheritance.
- Inheritance
  - The property that instances of a sub-class can access both data and behavior associated with a superclass.
- In programming languages, inheritance means that the behavior and data associated with subclasses are always an extension of the properties associated with the parent class.
  - A child is a more specialized form of the parent.
Transitivity

- Inheritance is always transitive
  - A class can inherit features from superclasses many levels away
  - If class Dog is a subclass of class Mammal, and class Mammal is a subclass of Animal, then Dog will inherit attributes from both Mammal and Animal
- Multiple inheritance occurs when a subclass has more than one superclass
- Java does not support multiple inheritance, but other OO languages do (C++, Eiffel, …)

Substitutability

- Since
  - Instances of a subclass contain all the state and behavior associated with the superclass
- This means
  - An instance of a subclass can mimic the behavior of the superclass and should be indistinguishable from an instance of the superclass
- So
  - It is possible to substitute instances of the subclass for the superclass in any situation with no observable effect

Benefits of Inheritance

- One view of inheritance is that it provides a way to specify some properties/behaviors that all subclasses must exhibit
- Inheritance can be used to re-use code
- Inheritance also provides the ability to generalize
  - A method can be written to work with the super-class but subclasses can be passed as arguments
### Types of Inheritance

<table>
<thead>
<tr>
<th>Form of Inheritance</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specification</td>
<td>The superclass defines behavior that is implemented in the subclass but not in the superclass. Provides a way to guarantee that subclass implements the same behavior.</td>
</tr>
<tr>
<td>Specialization</td>
<td>The subclass is a specialized form of the superclass but satisfies the specification of the parent class in all relevant aspects.</td>
</tr>
<tr>
<td>Extension</td>
<td>The subclass adds new functionality to the parent class, but does not change any inherited behavior.</td>
</tr>
<tr>
<td>Limitation</td>
<td>The subclass restricts the use of some of the behavior inherited from the superclass.</td>
</tr>
<tr>
<td>Combination</td>
<td>The subclass inherits features from more than one superclass (i.e., multiple inheritance).</td>
</tr>
</tbody>
</table>

### Specification

- The **specification** form of inheritance specifies the required set of behaviors that any of its subclasses must provide.
  - The subclass inherits the specification of the behavior of its superclass, and it is up to the subclass to provide the implementation for all of these behaviors.
- The specification form of inheritance specifies what the subclass must do, not how they will do it.

### Clock Class

```
Clock

Behavior

Set Current Time
Get Current Time
```
Specialization

- The **specialization** form of inheritance is most easily defined in terms of the “is a” relationship.
  - The subclass is a more specialized form of the superclass.
- Specialization differs from specification in that the subclass inherits both the specification of a behavior and the implementation by the superclass.
  - Now the parent states not only what you must do, but for at least some of the behaviors how you will do it.

---

Clocks

- **Clock**
  - Current Time
  - Display Time
- **AnalogClock**
  - Current Time
  - Display Time
  - Specification only (NOT implemented)
- **DigitalClock**
  - Current Time
  - Display Time
  - Implementation provided by superclass inherited by subclasses
  - This specialization provided by subclass
  - Specification of behavior inherited from parent class

---

Extension

- This is the **extension** form of inheritance is used to add totally new capabilities to the subclass.
  - The subclasses have the same behavioral properties as the superclass except that in each subclass these behaviors may be implemented quite differently.
- The subclass will have behaviors that are not present in the superclass.
  - Note that substitutability still holds.
- Consider the class **AtomicClock**.
Limitation

• There is probably no need to provide a way to manually set the time on an atomic clock.
• This is an example of the **limitation** form of inheritance.
• The strict definition of the limitation form of inheritance says that you remove a behavior that is inherited from a superclass from the subclass.
  – What is wrong with this definition in terms of the general principle of inheritance?

The Answer

• It violates the “is a” relationship.
  – If we simply remove the ability to set the time on an atomic clock it is no longer a clock because it does not exhibit all of the behaviors of the superclass.
• Therefore, when using the limitation form of inheritance a behavior is not actually removed, but simply implemented as a “no operation”.
  – An atomic clock would still provide a set time method, but invoking that method will have no effect on the state of the clock.

Multiple Inheritance (Combination)

• In **single inheritance** a subclass has at most one direct superclass from which it inherits behavior.
• **Combination** is a form of inheritance in which a subclass inherits directly from two or more superclasses.
• The term **multiple inheritance** is commonly used to describe the combination form of inheritance in programming languages.
• Not all object-oriented programming languages allow multiple inheritance.
Multiple Inheritance

Why Is It a Problem?

- Earlier we said that not all languages support multiple inheritance.
- Why?
  - As a hint think about the diagram on the previous page.
    - The radio class has a TurnOn behavior.
    - The clock class has a TurnOn behavior.
    - ...

Inheritance and UML