

**Introduction to
Object Oriented Programming 2E
Timothy A. Budd**

Chapter 7

Inheritance

Motivation for Inheritance

The basic reason for wanting to make use of inheritance is to leverage the creation of new software structures from existing software units.

- Productivity
- Quality

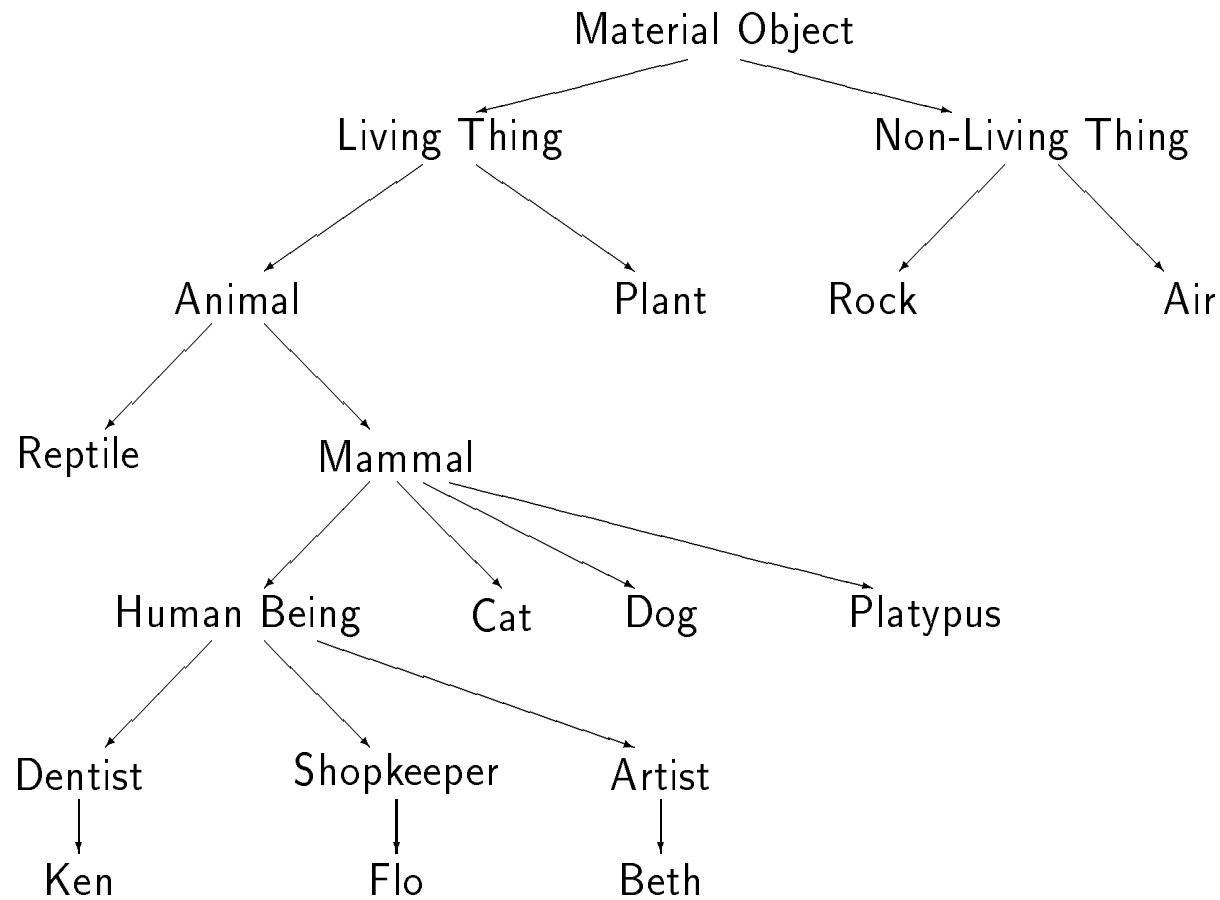
Tension between Generality and Specialization

A software system designed for a specific project must usually be very specialized.

A software system designed as a reusable tools must usually be very general.

Inheritance allows one to take a very general software component and specialize it for use in a specific project.

Abstract idea of Inheritance



Practical Meaning of Inheritance

- Data areas and behavior defined in a parent class are accessible from within a child class.
- Data areas and behavior of a child class are an extension (strictly larger) than the data areas and behavior of parent classes.

A child class is in a certain sense both an extension and a restriction of the parent class.

Idealized Image of Inheritance

Consider the following argument:

- Instances of the subclass must possess all data areas associated with the parent class.
- Instances of the subclass must implement, through inheritance at least (if not explicitly overridden) all functionality defined for the parent class. (They can also define new functionality, but that is unimportant for the present argument).
- Thus, an instance of a child class can mimic the behavior of the parent class and should be *indistinguishable* from an instance of the parent class if substituted in a similar situation.

Subclass, Subtype and Substitutability

A subtype is a class that satisfies the principle of substitutability.

A subclass is something constructed using inheritance, whether or not it satisfies the principle of substitutability.

The two concepts are independent. Not all subclasses are subtypes, and (at least in some languages) you can construct subtypes that are not subclasses.

Forms of Inheritance

Inheritance is use for a variety of different purposes and in a variety of different ways.

Heuristics for Subclassing – Specialization

By far the most common form of inheritance is for specialization.

Heuristics for Subclassing – Specification

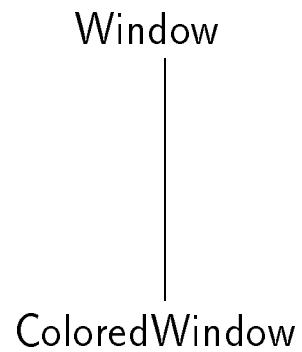
The next most common form of inheritance involves specification. The parent class specifies some behavior, but does not implement the behavior.

Heuristics for Subclassing – Construction

The parent class is used only for its behavior, the child class has no *is-a* relationship to the parent.

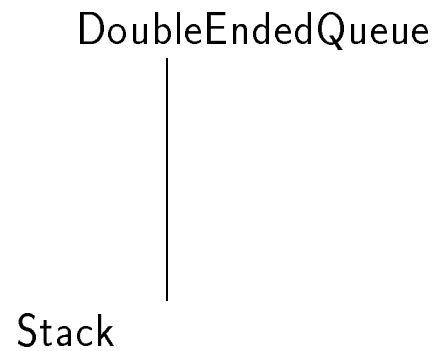
Heuristics for Subclassing – Generalization or Extension

The child class generalizes or extends the parent class by providing more functionality.



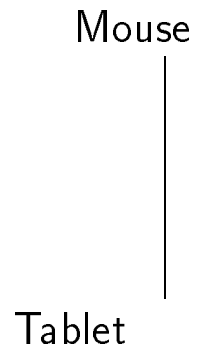
Heuristics for Subclassing – Limitation

The child class limits some of the behavior of the parent class.



Heuristics for Subclassing – Variance

The child class is simply a variation of the parent class.



Summary of Forms of Inheritance

- **Specialization.** The child class is a special case of the parent class; in other words, the child class is a subtype of the parent class.
- **Specification.** The parent class defines behavior that is implemented in the child class but not in the parent class.
- **Construction.** The child class makes use of the behavior provided by the parent class, but is not a subtype of the parent class.
- **Generalization.** The child class modifies or overrides some of the methods of the parent class.
- **Extension.** The child class adds new functionality to the parent class, but does not change any inherited behavior.
- **Limitation.** The child class restricts the use of some of the behavior inherited from

the parent class.

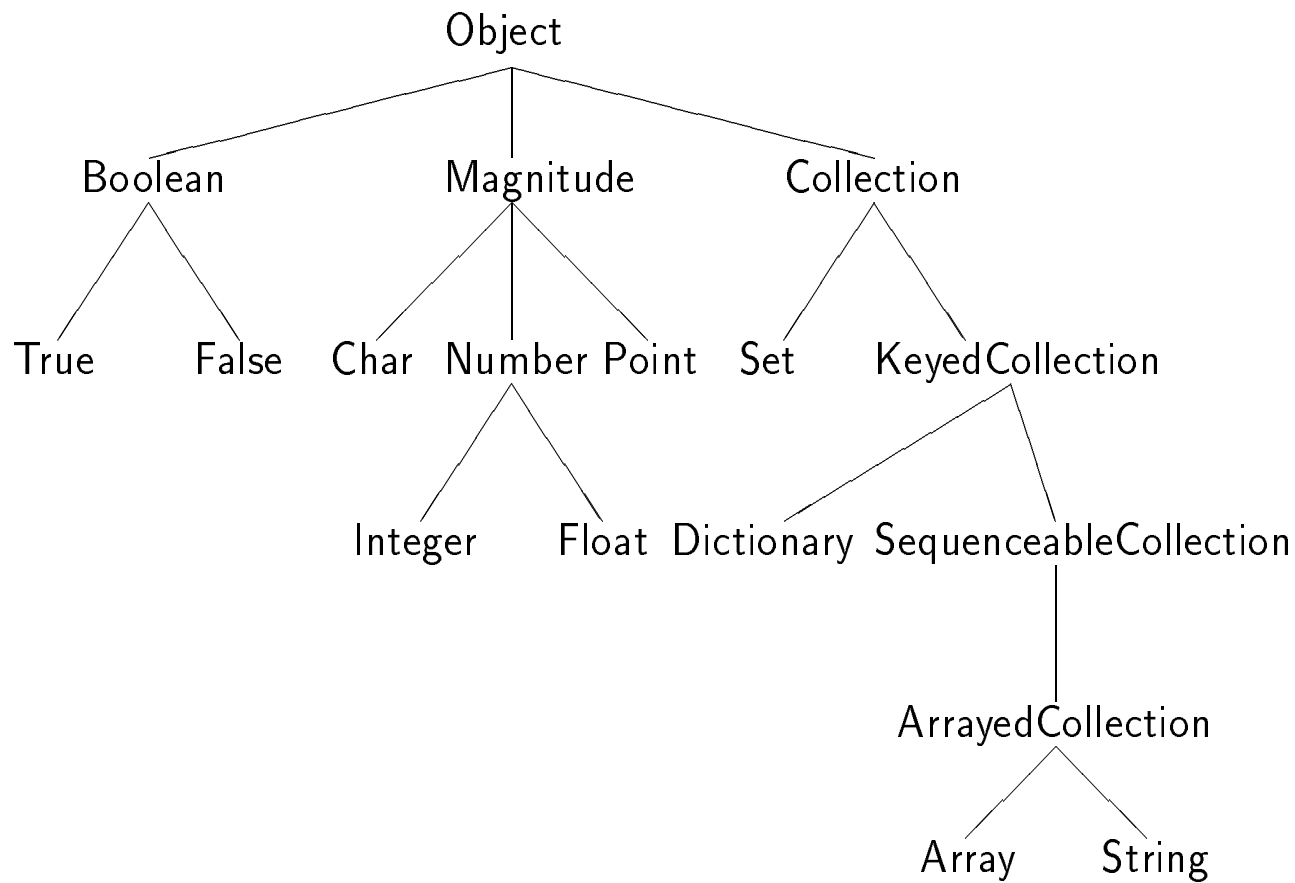
- **Variance.** The child class and parent class are variants of each other, and the class-subclass relationship is arbitrary.
- **Combination.** The child class inherits features from more than one parent class. This is multiple inheritance and will be the subject of a later chapter.

Trees versus Forests

There are two common views of class hierarchies:

- All classes are part of a single large class hierarchy.
- Classes are only placed in hierarchies if they have a relationship – results in many small hierarchies.

A Portion of the Little Smalltalk Hierarchy



Benefits of Inheritance

- Software Reuse
- Code Sharing
- Improved Reliability
- Consistency of Interface
- Rapid Prototyping
- Polymorphism
- Information Hiding

The Costs of Inheritance

- Execution speed
- Program size
- Message Passing Overhead
- Program Complexity