Object Oriented Concepts

What is an Object?

- “Object. An abstraction of something in a problem domain, reflecting the capabilities of the system to keep information about it, interact with it, or both.” (Coad and Yourdon, 1990)
- “We define an object as a concept, abstraction, or thing with crisp boundaries and meaning for the problem at hand. Objects serve two purposes: They promote understanding of the real world and provide a practical basis for computer implementation.” (Rumbaugh et al, 1991)

Classes & Objects

- “A class is the general template we use to define and create specific instances, or objects. Every object is associated with a class. An object is instantiation of a class.” (Alan Dennis, 2006)
- “Class is a concept that describes a set of objects that are specified in the same way. All objects of a given class share a common specification for their features, their semantics and the constraints upon them.” (OMG, 2004)

Classes & Objects

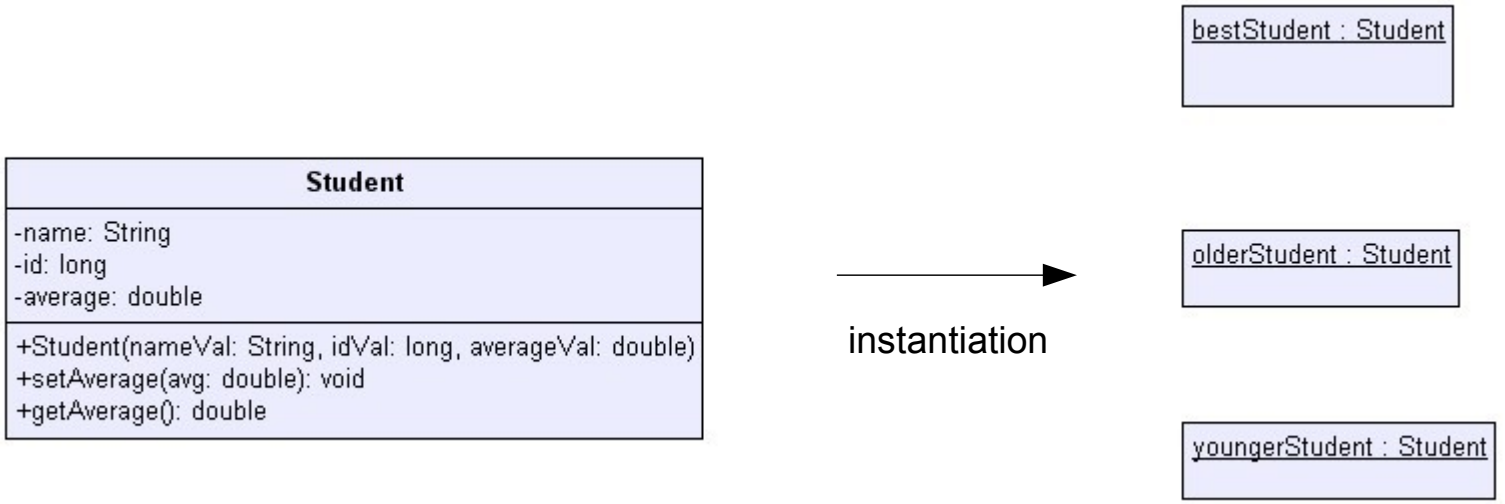
- Each attribute that was defined in our class exists in each one of objects that were instantiated from it. Each object has the same attributes with possible different values.

  The information can be anything we can think of, such as a student name, birthday, address and phone number. The attributes

- Each operation that was defined in our class can be called on each one of the objects that were instantiated from it.

  When calling a method on a specific object the method uses the values that object holds in its attributes.

Classes & Objects



Generalization

- “Generalization occurs where there is a taxonomic relationship between two classes. This means that the specification of one class is more general and applies also to the other class, while the specification of the latter is more specific and includes some details that do not apply to the former.” (OMG, 2004)

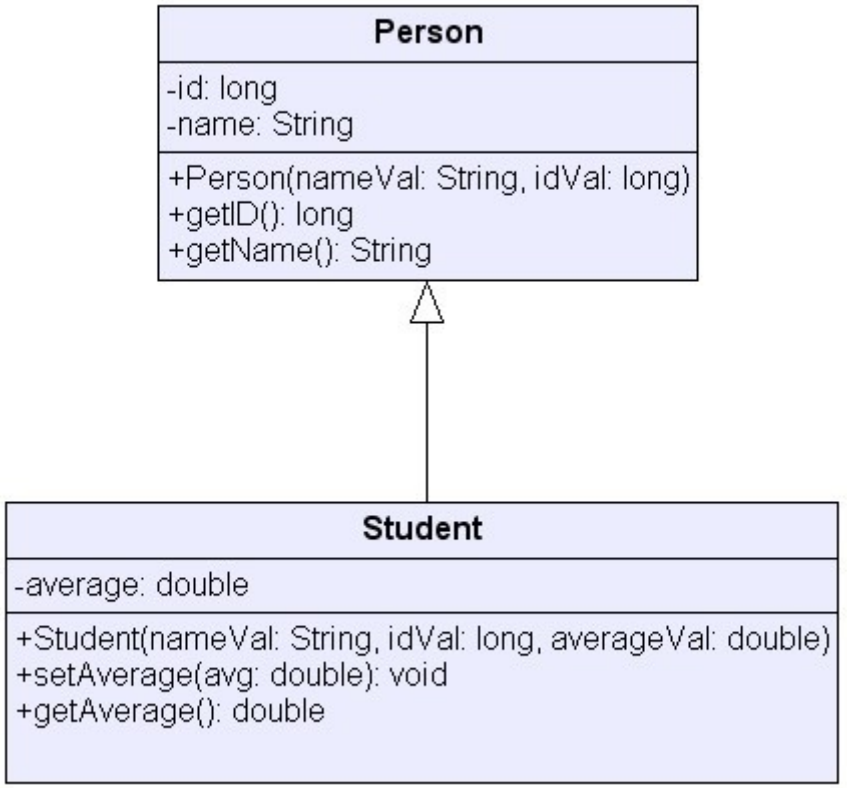
  “In other words, any instance of the more specific class is also indirectly an instance of the more general class. The more specific class inherits all features of the more general one, while also adding some features that are uniquely its own.” (OMG, 2004)

Inheritance

- “Inheritance is the mechanism for implementing generalization and specialization in an object oriented programming language.” (Simon Bennett, 2006)

  When two classes are related by the mechanism of inheritance, the more general class is called a superclass in relation to the other, and the more specialized one is called its subclass.

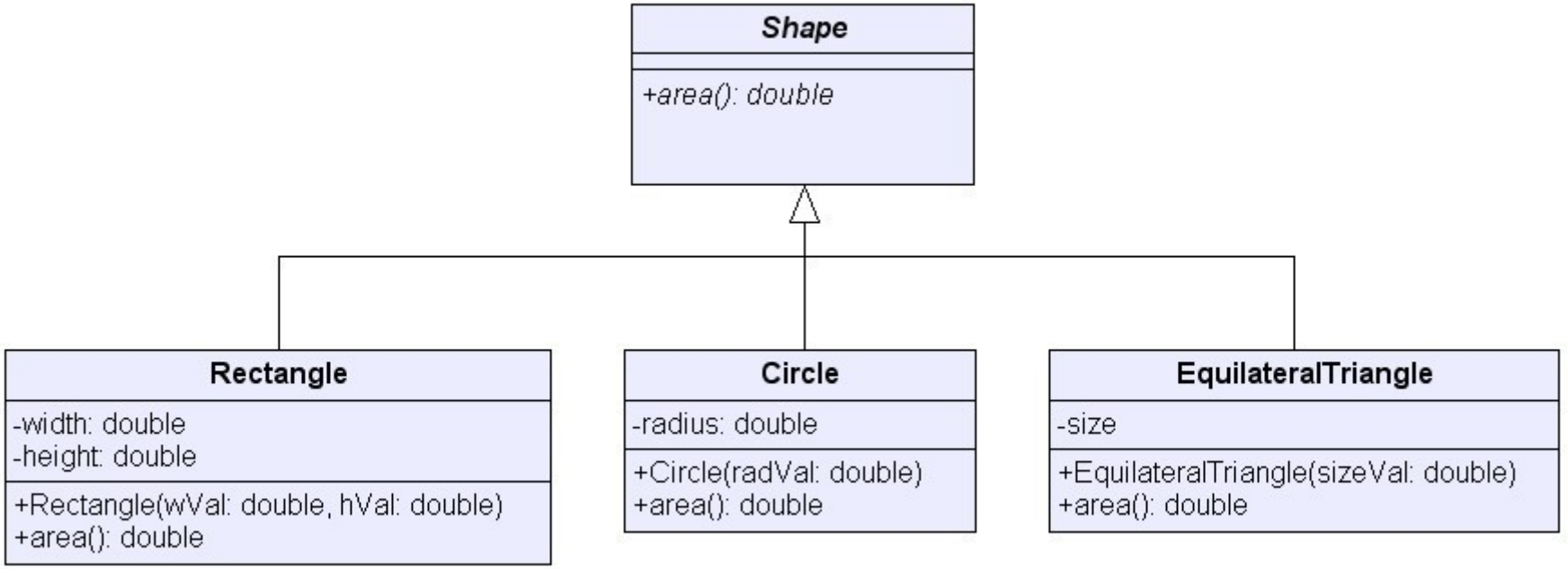
Inheritance



Polymorphism

- “Polymorphism literally means 'an ability to appear as many forms' and it refers to the possibility of identical messages being sent to objects of different classes, each of which responds to the message in a different, yet still appropriate, way. Each receiving object is responsible for knowing how to respond to each message it receives.” (Simon Bennett, 2006)

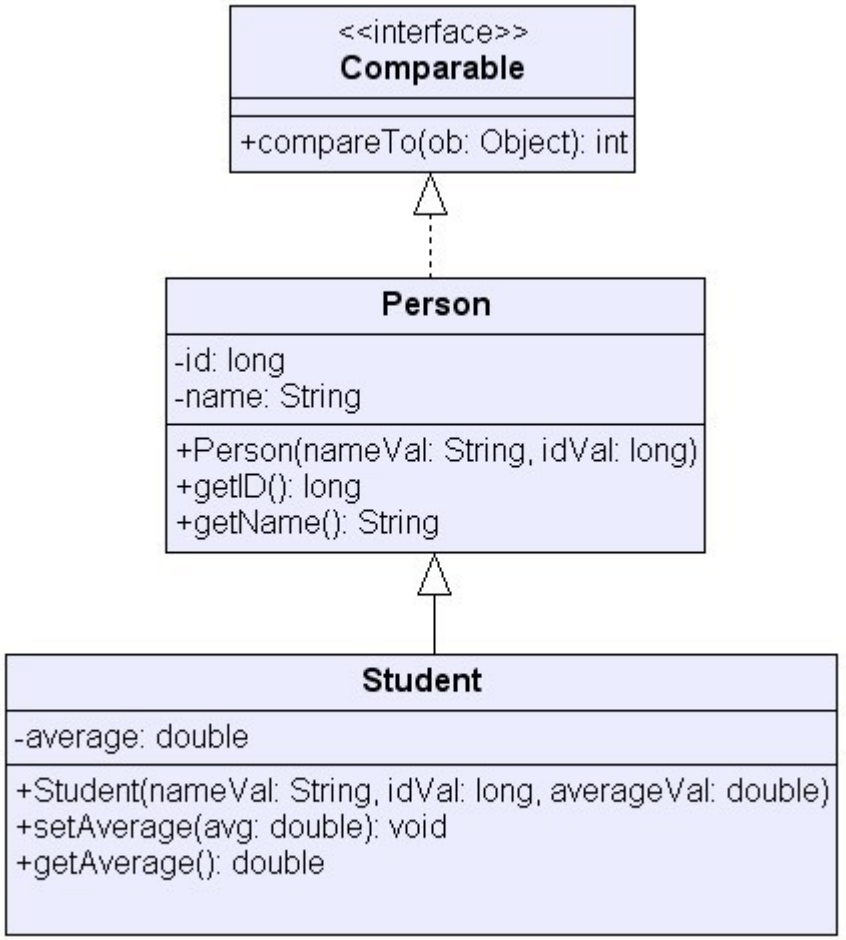
Polymorphism



Interfaces

- Interface is that part of the boundary between two interacting systems across which they communicate. Interface is the set of all signatures for the public operations of a class, component, application or a whole system.

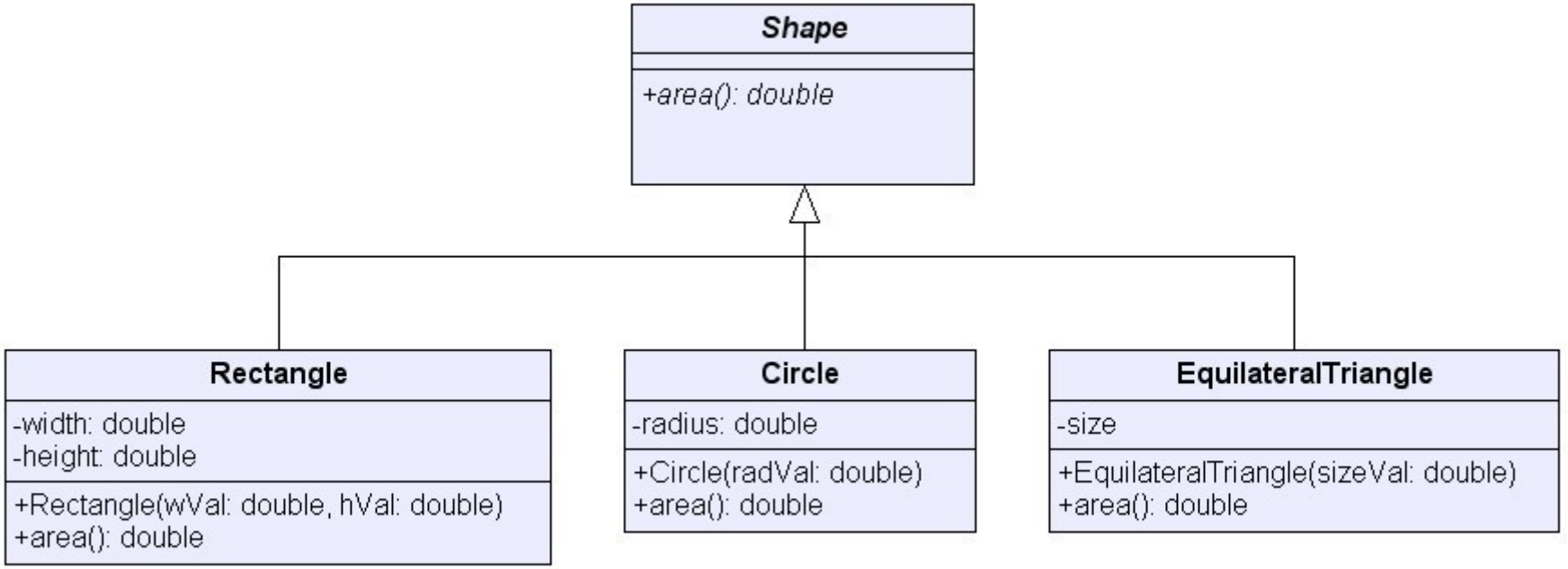
Interfaces



Abstract Class

- An abstract class is a class that we cannot instantiate. It is a super-class that acts only as a generalized template for its instantiated subclasses.

Abstract Class



Design Patterns

- “Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice” (Christopher Alexander, 1977)

Software Design Patterns

- “Software patterns are reusable solutions to recurring problems that we encounter during software development.” (Mark Grand, 2002)

Software Design Patterns History

- The idea of using patterns evolves from the architecture discipline.

  Christopher Alexander refer to patterns in the architecture discipline in his book “A Pattern Language”, that was published via Oxford University Press in 1977.

- The first GUI software patterns were set in 1987.

  Those patterns were set by Ward Cunningham and Kent Beck via their “Using Pattern Languages for Object-Oriented Programs” book.

Software Design Patterns History

- The Gang of Four (AKA GOF) publish their “Design Patterns” book in 1994.

  Erich Gamma, Richard Helm, John Vlissides, and Ralph Johnson work together and publish the “*Design Patterns” book via* Addison Wesley publishing house in 1994. Their books sets the fundamental classic design patterns we all know.

- Concurrently with Java EE development, new design patterns are set.

  A detailed catalog of these patterns together with code samples and detailed tutorials can be found at http://java.sun.com/blueprints/patterns/index.html.