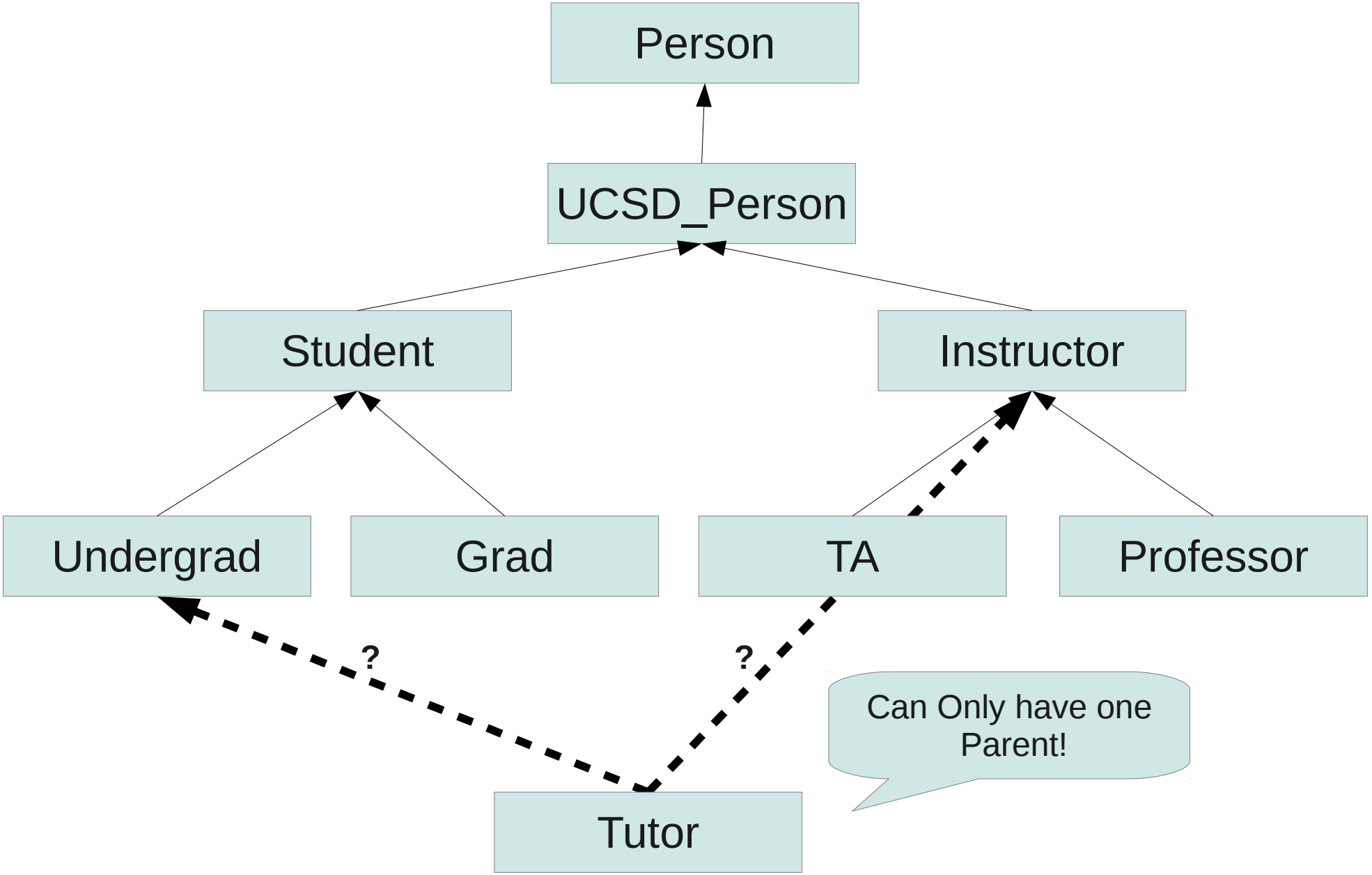
Lecture 18
CSE11 – Fall 2013
Inheritance

What is Inheritance?

- Inheritance allows a software developer to derive a new class from an existing one
  - write code once, use many times (code reuse)
- Specialization
- `extends` is the java keyword that indicates inheritance
- The existing class is called the parent class, or superclass, or base class
- The derived class is called the child class or subclass.
- As the name implies, the child inherits characteristics of the parent
- That is, the child class inherits the methods and data defined for the parent class

Inheritance Hierarchy



isA

- Defines an inheritance relationship
- Examples:
  - UCSD_Person isA person
  - Instructor isA UCSD_Person
  - Student isA UCSD_Person
  - Undergrad isA Student
  - TA isA Instructor
- Transitive: TA isA UCSD_Person
  - Undergrad isA Person

What do you get when you inherit (extend a class)

- all methods and variables
  - But, if a method/variable is `private` the subclass cannot access the method/variable
  - Private means private to the class in which the method/variable is defined
- constructor(s) of your parent
- It's recursive, you get method/variables/constructors of your parent, grandparent, great-grandparent ....

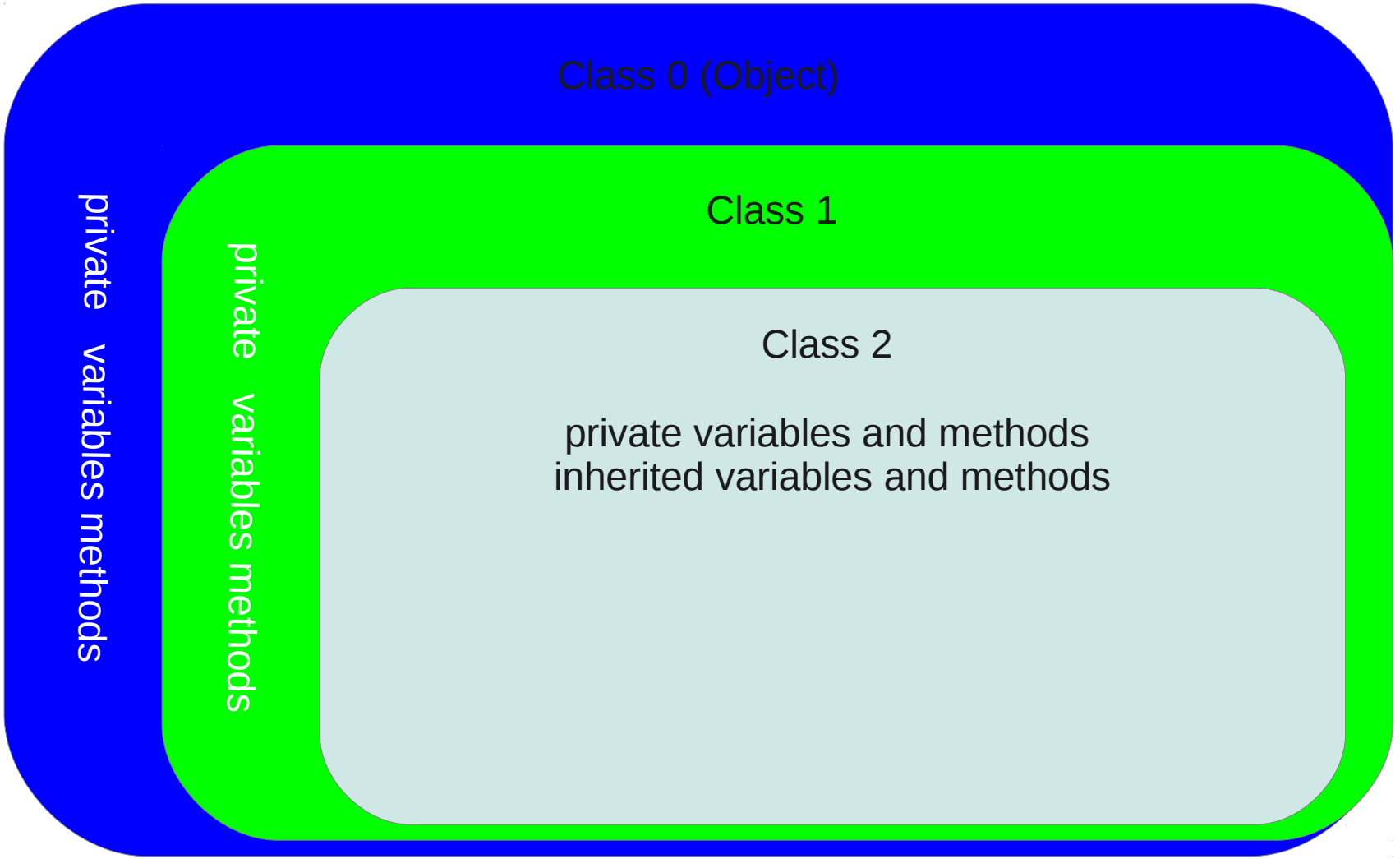
the `Object` class

- Every java class is descended from the Object class.
- Object defines a few interesting methods (and hence ALL classes have these methods)
  - getClass() - returns the runtime class
  - toString() - returns a String Representation of the class
  - equals() - method to determine if two objects are equal to each other (Note the String class defines the equals() method to be a character by character comparison of two string objects)

Constructors

- The constructor for every parent class is called whenever you do a `new`
- If your code does not supply it, **the no-parameter constructor of you parent is implicitly inserted by the compiler as the first line of your constructor**
- Your constructor can explicitly call super(. ...) as the first line of your subclass constructor. If it does. it must be the first statement.

Hierarchy Revisited



Constructor of 2 Calls Constructor of 1. Constructor 1 Calls Constructor of 0.
This is so each layer of the hierarchy can initialize all private variables

Dynamic Method Invocation

- Java always uses the method defined “closest” to the class **when the instance was created**
- Suppose ClassC extends ClassA
  - ClassC is a subclass, ClassA is the superclass
- Both classes define methodX()
- Now suppose you declare
  - `ClassC myInstance = new ClassC();`
- Which methodX() code is executed in
  - `MyInstance.methodX()` ?
- Now Declare
  - `ClassA referAsA = myInstance`;
  - Which method is invoked via `referAsA.methodX();`

`protected`

- Private variables/methods are private to the class. They CANNOT be seen by any subclasses
- Public variables/methods are available to all classes (including subclasses)
- `protected` variables/methods are seen by subclasses, but not by external classes

- Declare a method/variable as `public, private or protected`
- `canvas` is a protected variable of the WindowController Class
  - This is why you can use it without declaring it

Overriding Method Defintions

- A subclass can redefine a method with the identical signature of its superclass.
- There are times when you want to invoke the method of your super class when (re)defining in your subclass
  - use the `super.method()`
  - to invoke method() of your Superclass
-

`final`

- Have applied `final` to variables to make them into constants
- You can apply `final` to methods to indicate that they cannot be overridden by subclasses
- You can apply `final` to classes to indicate that they cannot be extended
  - e.g. `public final class Math`

abstract

- `abstract` methods are methods with no body
  - They look a lot like interfaces
  - To be useful, a subclass must provide an implementation for abstract method
  - If a class defines an abstract method, the class must be defined as abstract
- Purpose: define a hierarchy of methods/capability (outline of functionality)
- The AWT has many examples of abstract
- http://docs.oracle.com/javase/6/docs/api/java/awt/Toolkit.html