Look closer at Animal class

- We can instantiate an Animal object:
  Animal a = new Animal();
- However, what does such an animal object looks like and behaves?
We need Animal class, but not generic animal object

• We still need such a super class called Animal. However, we don’t want to instantiate an animal object.
• We want to instantiate objects of Animal’s sub class, like dog, cat, sheep, etc.
• We don’t need a generic animal object.
• How do we enforce this?
Abstract class

• We can declare a class abstract to make sure there is no object of this class type can be instantiated.

abstract public class Animal
{
}
}
Abstract class

• An abstract class has no use, no value, unless it is extended.
• Instances of a subclass of the abstract class are doing the real work at run time.
Abstract methods

• An abstract method has no body,
  public abstract void behave();
• An abstract method can only exists in an abstract class.
• In sub class, implementing an abstract method is just like overriding a method.
Demo

- Abstract class MyAnimal
- Sub classes: MyCat.java, MySheep.java
- Sub class needs to implement all abstract method from MyAnimal class.
What if a class needs more than one super class

- A cat IS A kind of Animal.
- A cat IS A kind of Pet.
- However, Java language doesn’t allow a class extends two super classes. This language feature is to avoid “deadly diamond of death” problem.
- How can we do it, then?
Interface

• An interface has only abstract methods.
• A class that implements an interface must implement all its abstract methods.
• To define an interface,
  public interface Pet {}
• To implement an interface,
  public class Cat extends Animal implements Pet {}
A class can implement multiple interfaces

- Define Pet as an interface
- A Cat class can extends Animal1 abstract class and implements Pet interface.
Demo

- Abstract class: MyAnimal
- Interface: Pet
- Sub class: MyCat1, MySheep1
- Sub class needs to implement abstract methods from both abstract class, Animal, and interface Pet.
package

• Package is a collection of classes of related purpose.
• These classes are declared as of same package.
• Theses classes are stored within the same directory tree.
• Theses classes are accessed through their package name.
Package definition

• A package declaration statement:
  package doctors;

• Assume doctors package has three classes, Doctor, FamilyPractitioner, and Surgeon.

• All three java file should be stored under directory name doctors. The same is true for all class files.

• Usage: doctors.Surgeon, doctors.Doctor.
Demo package doctors
protected

- Access modifier: public, private, protected
- protected int average(double d1, double d2)
  {return (d1+d2)/2;}
- A “protected” modifier means the method, or variable is visible only to its sub classes, or classes within the same package.
import statement

• Import statement brings one or more members of a package into view. This allows you to use those members directly, without explicit package qualification.

• Without import statement:
  doctors.Surgeon1 s = new doctors.Surgeon1();

• With import statement: import doctors.*;
  Surgeon1 s = new Surgeon1();
Demo TestDoctor1.java