Abstract Classes

Abstract Classes

- classes which can be extended, but cannot be initialized

```
public abstract class Mammal { }

public class Dog extends Mammal { }

Dog dog = new Dog();
```

OK

```
Mammal mammal = new Mammal();
```

NOK !!

- class `Mammal` is **abstract class**, and class `Dog` is **concrete class**

Abstract Methods

- marked with `abstract` keyword
- don't have a body (!!)
- the implementation (body) is done in classes which extend an abstract class

```
public abstract class Mammal {
  public abstract void speak();
}

public class Dog extends Mammal {
  public void speak() { System.out.println("Woof!"); }
}

public class Cat extends Mammal {
  public void speak() { System.out.println("Meow"); }
}
```

Rules for Using Abstract Methods

1. Only instance methods can be marked `abstract`
   - not variables, constructors, static methods, etc.
2. Abstract method can only be declared in an `abstract` class.
3. Non-abstract class which extends abstract class must implement **all** inherited methods.
4. All other rules with overriding methods apply.

```
abstract class Animal {
    public abstract void speak();
}
abstract class Mammal extends Animal {
    public abstract void walks();
}
public class Dog extends Mammal {
    @Override
    public void speak() { System.out.println("Woof!"); }

    @Override
    public void walks() { System.out.println("Dog walks."); }

    public static void main(String[] args) {
        Dog dog = new Dog();
        dog.speak();
        dog.walks();
    }
}
```

Dog extends Mammal which extends Animal

=> Dog is a first concrete class, which inhertits abstract methods from both Mammal and Animal

```
Woof!

Dog walks.
```

Keep in mind...

- abstract classes can have constructors
  - but they can be called only using `super()` from the child class
- abstract class or method **cannot** be marked `final`

  (wouldn't make any sense)
- abstract method **cannot** be marked `private`

  (obviously)
- static method **cannot** be overriden
  - therefore, `abstract static` is not allowed