Today’s topic:

- Interfaces
- Vocab from chapter 8
What Is a Framework?

A goal of OO is code reuse

- One way to promote code reuse is standardization. (The other calls this plug and play).

- A common framework makes it easier to learn various applications within the framework.

- It also makes a developer’s life easier by promoting maximum code reuse

Code Reuse Revisited

- Inheritance and composition allow for reuse for basically one class.

- Frameworks focus on reusing whole or partial systems.
What Is a Contract?

Weisfeld defines *contract* as:

“Any *mechanism* that requires a developer to *comply* with the *specifications* of an API.”

*(In this way an API is a framework).*

The Term Contract

The term *contract* is widely used in many aspects of business, including software development.

- Do not confuse the concept presented here with other possible software design concepts called contracts.

- Enforcement is *vital* because it is always possible (perhaps even easy) for a developer to *break* a contract.
Previously...

- We talked about *abstract* methods, this is one way to create a *contract* in the API.

- We used them in situations where we wanted to define a class and its interface, without providing a *complete* implementation of each an *every* method in the class.

---

Interfaces

- In some situations we want to *separate* the *interface* completely from the *implementation* details.

- In these cases we can define an *interface* that contains a group of methods *without any* implementation at all.
What actions can you take with a deck of cards?

• Shuffle
• Sort
• Draw top card
• Deal out a number of cards
• Cut the deck
• Pick card at random

We have defined an interface for a deck of cards. These are actions we would expect to take with any deck of cards.

We create an interface with requirements for other programmers.

They can choose to implement the interface however they wish, but it must conform to the interface we define.

This ensures that the objects can be used interchangeably together.
Contracts are “plug-in points” into your code.

- Anyplace where you want to make parts of a system abstract, you can use a contract.

- Instead of using objects of specific classes, you can use any object that implements the contract.

- For example, you may use Lego with MegaBlocks, and many other generic building block sets.

Using Interfaces

An interface specifies certain behavior, but not the implementation.

- When you implement an interface in a class, you are honoring the contract to provide concrete behaviors by implementing the abstract methods.

- How you implement these methods is up to you, but by contract, you have provide the concrete methods.
You cannot create instances an interface.

An interface cannot contain instance variables.

The interface may only contain variables that are defined as static and final.

An interface cannot contain constructors.

By definition all of the methods in an interface are abstract and do not include implementations.

An interface can extend multiple interfaces

A class may implement multiple interfaces.

Both abstract classes and interfaces provide abstract methods.

However, abstract classes require a strict inheritance relationship and therefore a defined hierarchy of classes exists.

Abstract classes also provide some implementation that are shared by with the subclasses.

Interfaces can be used for classes that are not related and do not provide implementation.
public interface InterfaceName{
    //Any static final variables.
    public static final int MY_CONSTANT = 0;

    //Interface Methods
    public void methodOne();
    public int methodTwo();
    public double methodThree(int myNum);
}

public class Classname implements InterfaceName{
    // Must implement all methods defined in the
    // InterfaceName.
}

An example: Iterator
- [http://docs.oracle.com/javase/1.7.0/docs/api/java/util/Iterator.html](http://docs.oracle.com/javase/1.7.0/docs/api/java/util/Iterator.html)

If you wanted to create a class that implements the Iterator interface, what methods are you required to implement?

- hasNext()
- next()
- remove()

We have already used a class that implements the Iterator interface: the Scanner.

A way to multiple inheritance?

- In Java you cannot extend multiple classes.

- You can, however, implement multiple interfaces.

- Interfaces can be used as a way to utilize polymorphism to use an object in different situations, and a way around the multiple inheritance limitation.
Why use interfaces?

- They allow programmers to define an agreed way that the software will interface and work together.
- The implementation details are left to each individual situation, but the interface is predefined.
- In many situations there are industry standards that define the requirements for a system.
- For example, there are standard interfaces for TCP-IP, SFTP, etc. You can write your own implementation that meet these standards.

For example

- Suppose there is an Interface:

  ```java
  public interface Product{
      public String getSKU();
      public float getPrice();
      public String getDescription();
  }
  ```

  This requires any class that implements this interface to include implementations for these methods.
Many classes could implement this interface, and all be treated as products using a common interface.

- We could implement any number of different classes as products within our system.
  - DVDs, Chips, Tires, Gum, Shirts, etc...

Example

Within my Catalog class I can then have a method to add any Product to my catalog:

```java
public void addItem(Product aProduct)
    items.add(aProduct);
}
```

Now I can add any objects that implement the Product interface to my catalog.