**Video Name**: Interfaces - Custom

**Topics**:  
- defining an interface  
- using an interface  
- interface requirements

**Java Class(es)**: interfaces package, Student, BankAccount, DataSet, Measurable

Example: DataSet class stores the sum and the max for a collection of objects. Objects must have some type of value. We create an interface named Measurable to enforce this constraint.

**KEY POINT**: We want DataSet to be used with *lots* of different types of objects, which are otherwise not related (e.g., BankAccount, Student). The only thing these classes must have in common is that they have some value which can be returned.

Interface definition specifies *abstract* methods.

```java
public interface Measurable {
    public double getMeasure();
}
```

**Implementing this interface:**

```java
public class Student implements Comparable<Student>, Measurable {

    @Override
    public double getMeasure() {
        return gpa;
    }
}
```

Interface creates a new data type. So can use the interface as a data type name:

```java
    private Measurable max;
```

We can specify a function parameter using the interface name, which means that *any* class that implements that interface can be passed into the function. We can then call the methods that are part of that interface.

```java
    public void add(Measurable measurable) {
        sum += measurable.getMeasure();
    }
```
Interface Requirements:

- Most methods are abstract (Java used to require all methods to be abstract. No longer required, but still best practice)
- All methods must be public (makes sense, since methods bodies aren’t included in the interface, no need for private methods)
- Interfaces may not have instance variables (also makes sense, because we don’t’ instantiate interfaces directly, only classes that implement the interface... if the interface defined an instance variable, we’d be adding data to implementing classes, which would not be a good idea).
- Interface may have constants (constants typically shared among objects, so we don’t have the same issues as with instance variables).
- A class can only extend one other class, but can implement as many interfaces as desired.