Comparator

As you prepare for class, please snarf the in-class code.

My Solution to the Prep-Assignment

```java
public class DumbHashSol implements IMapper {
    public boolean containsKey(String key) {
        return false;
    }
    public int get(String key) {
        return 0;
    }
    public void put(String key, int value) {
    }
    public void printAll() {
        System.out.println("This is a dumb map.");
    }
    public int size() {
        return 0;
    }
}
```

Interfaces: Types Defined by the Functions

`IMapper.java` specifies a set of functions `IMappers` must have:

```java
public interface IMapper {
    public abstract boolean containsKey(String key);
    public abstract int get(String key);
    public abstract void put(String key, int value);
}
```

In `DumbHash.java` we say `DumbHash` implements these functions:

```java
public class DumbHash implements IMapper {
}
```

Then in `HashMain.java` we can put `DumbHash` in an `IMapper` variable:

```java
IMapper hash4 = new DumbHash();
```

Interfaces: Question

Say I define a class like this:

```java
public class ComplexNumber implements Comparable<ComplexNumber> {
    // ...
}
```

Are these lines legal:

```java
Comparable<ComplexNumber> c = new ComplexNumber(2, 3);
int result = c.compareTo(new ComplexNumber(0, .5));
```
Interfaces: Question

Say I define a class like this:

```java
public class ComplexNumber implements Comparable<ComplexNumber> { 

// Are these lines legal:
Comparable<ComplexNumber> c = new ComplexNumber(.2, .7);
int result = c.compareTo(new ComplexNumber(0,0));
```

Say ComplexNumber had a `add(ComplexNumber c)` method. Why would this line generate a syntax error in Java?

```java
Comparable<ComplexNumber> c = new ComplexNumber(.2, .7);
c.add(new ComplexNumber(0,0));
```

Things to Remember about Interfaces

1. An interface defines a list of functions
2. If a class implements all the functions in an interface, it can declare that it `implements` that interface
3. If a function implements an interface, that object can be used where there are variables of that interface
4. If you have a variable of a particular interface, you can only call methods on the variable from the interface’s list of functions

A new interface: Comparator

- Similar to `compareTo`, except its `compareTo` method takes two parameters
- You use it to compare objects that you can’t change (e.g. String...you can’t change the way it’s `compareTo` works). Or objects that don’t have a “natural” ordering.

Questions about the Zzzzz class

What would the `compareTo` method return if you passed it “abcz” and “bbbbbbzbb”?
A. Something less than 0
B. Something greater than 0
C. 0

Why does “zzzzninja” end up before “ninjaz”?
A. “z”s occurs at an earlier index in “zzzzninja” than it does in “ninjaz”
B. “zzzzninja” is longer than “ninjaz”
C. “zzzzninja” has more zs than “ninjaz”
D. “zzzzninja” occurs first in the original list
Imagine representing integers with strings of U and D. Each U (“up”) adds one to the represented integer. Each D “down” subtract ones from the represented integer. So the string “UDUDDDD” corresponds to -3.

Write some code that sorts strings based on their numerical value. Use the Zzzzzz class to help you.

When you’re finished, submit the code via Ambient.

---

Things to Remember about Interfaces

1. An interface defines a list of functions
2. If a class implements all the functions in an interface, it can declare that it implements that interface
3. If a function implements an interface, that object can be used where there are variables of that interface
4. If you have a variable of a particular interface, you can only call methods on the variable from the interface’s list of functions

---

A new interface: Comparator

- Similar to compareTo, except its compareTo method takes two parameters
- You use it to compare objects that you can’t change (e.g. String...you can’t change the way it’s compareTo works). Or objects that don’t have a “natural” ordering.