Comparators and Iterators

Discussion 04

Announcements

Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
	2/12 Project 1B Due Weekly Survey Due			2/15 Midterm 1 (7-9pm)		
		2/20 Lab 4 Due Project 1C Due				

Content Review

Comparables

Comparables are things that **can be compared with each other**.
Any class could implement this interface.
Defines the notion of being “less than” or “greater than”.

```
public class Dog implements Comparable<Dog> {
    private String name;
    private int size;
    @Override
    public int compareTo(Dog otherDog) {
        return this.size - otherDog.size;
    }
}
```


Comparables

Can't use < and > directly on dog objects - undefined for them!

Instead, use the `compareTo` method.

~~`if (d1 < d2) {`~~

~~`} else`~~

~~`}`~~

```
if (d1.compareTo(d2) < 0) {
    // Dog 1 "less than" dog
} else {

}
```


Comparators

Comparators are things that **can be used to compare two objects.** Think of it as a "seesaw". Comparables are the things sitting on the seesaw. Not the seesaw itself!

```
public interface Comparator<T> {
    int compare(T o1, T o2);
}

public class DogComparator<Dog> implements Comparator<Dog> {
    public int compare(Dog d1, Dog d2) {
        return d1.size - d2.size;
    }
}
```


Why does compare/compareTo return an integer?

The `Comparator` interface's `compare` function takes in two objects of the same type and outputs:

- A negative integer if `o1` is "less than" `o2`
- A positive integer if `o1` is "greater than" `o2`
- Zero if `o1` is "equal to" `o2`

For `Comparable`, it is the same, except `o1` is `this`, and `o2` is the `other` object passed in.

Think of it as subtracting!

```
compare(T o1, T o2) -> o1 - o2
o1 - o2 < 0 -> o1 < o2
o1 - o2 > 0 -> o1 > o2
o1 - o2 = 0 -> o1 = o2
```

```
o1.compareTo(o2) -> o1 - o2
o1 - o2 < 0 -> o1 < o2
o1 - o2 > 0 -> o1 > o2
o1 - o2 = 0 -> o1 = o2
```


The Iterator & Iterable Interfaces

Iterators are objects that can be iterated through in Java (in some sort of loop).

```
public interface Iterator<T> {
    boolean hasNext();
    T next();
}
```

Iterables are objects that can produce an iterator.

```
public interface Iterable<T> {
    Iterator<T> iterator();
}
```


The Iterator & Iterable Interfaces

The enhanced for loop

```
for (String x : lstOfStrings) // Lists, Sets, Arrays are all Iterable!
```

is shorthand for:

```
for (Iterator<String> iter = lstOfStrings.iterator(); iter.hasNext();) {
    SomeObject x = iter.next();
}
```


Check for Understanding

1. If we were to define a class that implements the interface `Iterable<Dog>`, what method(s) would this class need to define?

2. If we were to define a class that implements the interface `Iterator<Integer>`, what method(s) would this class need to define?

3. What's one difference between `Iterator` and `Iterable`?

Check for Understanding

1. If we were to define a class that implements the interface `Iterable<Dog>`, what method(s) would this class need to define?

```
public Iterator<Dog> iterator()
```

2. If we were to define a class that implements the interface `Iterator<Integer>`, what method(s) would this class need to define?

```
public boolean hasNext()
public Integer next()
```

3. What's one difference between `Iterator` and `Iterable`?

`Iterators` are the actual object we can iterate over, i.e., think a Python generator over a list.
`Iterables` are object that can produce an iterator, i.e., an array is iterable; an iterator over the array could go through the element at every index of the array).

== vs. .equals()

- == compares if two variables point to the same object in memory.
  - null is compared with ==
- For reference types: .equals() (ex. myDog.equals(yourDog))
  - Each class can provide own implementation by overriding
  - Defaults to Object's .equals() (which is the same as ==)
  - Example: We make the Dog .equals() method return true if both Dogs have the same name
    - Dog fido = new Dog("Fido"); Dog otherFido = new Dog("Fido");
    - fido == otherFido -> false, but fido.equals(otherFido) -> true

Worksheet

1A OHQueue

```
public class OHIterator _________________ {
     OHRequest curr;

     public OHIterator(OHRequest request) {
          _________________;
     }

     public boolean isGood(String description) {
          return description.length() >= 5;
     }
```

```
@Override
_______ _______ _______ {
     while (__________________) {
          ____________________________;
     }
     ___________________________________;
}

@Override
_______ _______ _______ {
     if (_____________________________) {
          throw _____ ___________________;
     }
     ___________________________________;
     ___________________________________;
     ___________________________________;
}
}
```


1A OHQueue

```
public class OHIterator implements
Iterator<OHRequest> {
     OHRequest curr;

     public OHIterator(OHRequest request) {
          _________________;
     }

     public boolean isGood(String description) {
          return description.length() >= 5;
     }
```

```
@Override
_______ _______ _______ {
     while (__________________) {
          ______________________________;
     }
     ___________________________________;
}

@Override
_______ _______ _______ {
     if (_____________________________) {
          throw _____ ___________________;
     }
     ___________________________________;
     ___________________________________;
     ___________________________________;
}
```

```
}
```


1A OHQueue

```
public class OHIterator implements
Iterator<OHRequest> {
     OHRequest curr;

     public OHIterator(OHRequest request) {
          curr = request;
     }

     public boolean isGood(String description) {
          return description.length() >= 5;
     }
```

```
@Override
_______ _______ _______ {
     while (__________________) {
          ______________________________;
     }
     ____________________________________;
}

@Override
_______ _______ _______ {
     if (_____________________________) {
          throw _____ ___________________;
     }
     ____________________________________;
     ____________________________________;
     ____________________________________;
}
```

```
}
```


1A OHQueue

```
public class OHIterator implements
Iterator<OHRequest> {
     OHRequest curr;

     public OHIterator(OHRequest request) {
          curr = request;
     }

     public boolean isGood(String description) {
          return description.length() >= 5;
     }
```

```
     @Override
     public boolean hasNext() {




     }

     @Override
     _______ _______ _______ {
          if (_____________________________) {
               throw _____ ___________________;
          }
          ___________________________________;
          ___________________________________;
          ___________________________________;
     }
}
```

1A OHQueue

```
public class OHIterator implements
Iterator<OHRequest> {
     OHRequest curr;

     public OHIterator(OHRequest request) {
          curr = request;
     }

     public boolean isGood(String description) {
          return description.length() >= 5;
     }
```

```
@Override
public boolean hasNext() {
     while (curr != null &&
!isGood(curr.Description)) {
          curr = curr.next;
     }
     return curr != null;
}

@Override
_______ _______ _______ {
     if (_____________________________) {
          throw _____ ___________________;
     }
     ___________________________________;
     ___________________________________;
     ___________________________________;
}
}
```


1A OHQueue

```
public class OHIterator implements
Iterator<OHRequest> {
     OHRequest curr;

     public OHIterator(OHRequest request) {
          curr = request;
     }

     public boolean isGood(String description) {
          return description.length() >= 5;
     }
```

```
@Override
public boolean hasNext() {
     while (curr != null &&
!isGood(curr.Description)) {
          curr = curr.next;
     }
     return curr != null;
}

@Override
public OHRequest next() {

}
```

```
}
```


1A OHQueue

```
public class OHIterator implements
Iterator<OHRequest> {
     OHRequest curr;

     public OHIterator(OHRequest request) {
          curr = request;
     }

     public boolean isGood(String description) {
          return description.length() >= 5;
     }
```

```
@Override
public boolean hasNext() {
     while (curr != null &&
!isGood(curr.Description)) {
          curr = curr.next;
     }
     return curr != null;
}

@Override
public OHRequest next() {
     if (!hasNext()) {
          throw new
     NoSuchElementException();
     }
     OHRequest temp = curr;
     curr = curr.next;
     return temp;
}
```

```
}
```


1B OHQueue

```
public class OHQueue _____________________________________ {
    private OHRequest request;

    public OHQueue(OHRequest request) {
        ___________________________________;
    }


    @Override
    ______________ ______________ _________________ {
        ___________________________________;
    }
}
```


1B OHQueue

```
public class OHQueue implements Iterable<OHRequest>  {
    private OHRequest request;

    public OHQueue(OHRequest request) {
        _____________________________________;
    }


    @Override
    ______________ ______________ _________________ {
        _____________________________________;
    }
}
```


1B OHQueue

```
public class OHQueue implements Iterable<OHRequest>  {
    private OHRequest request;

    public OHQueue(OHRequest request) {
        this.request = request;
    }


    @Override
    ____________ ____________ _____________ {
        _________________________________;
    }
}
```


1B OHQueue

```
public class OHQueue implements Iterable<OHRequest>  {
    private OHRequest request;

    public OHQueue(OHRequest request) {
        this.request = request;
    }


    @Override
    public Iterator<OHRequest> iterator() {
        return new OHIterator(request);
    }
}
```


1C OHQueue

```
public class TYIterator _________________________ {

    public TYIterator(OHRequest queue) {
        ________________________;
    }

    @Override
    _____________ ___________ _____________ {
        OHRequest result = ____________________;
        if (_________________________________________) {
            ____________________;
        }
        return ____________________;
    }

}
```


1C OHQueue

```
public class TYIterator extends OHIterator {

    public TYIterator(OHRequest queue) {
        _________________________;
    }

    @Override
    ______________ ____________ _____________ {
        OHRequest result = ______________________;
        if (_________________________________________) {
            ______________________;
        }
        return ______________________;
    }

}
```


1C OHQueue

```
public class TYIterator extends OHIterator {

    public TYIterator(OHRequest queue) {
        super(queue);
    }

    @Override
    ______________ ____________ _______________ {
        OHRequest result = ______________________;
        if (_________________________________________) {
            ______________________;
        }
        return ______________________;
    }

}
```


1C OHQueue

```
public class TYIterator extends OHIterator {

    public TYIterator(OHRequest queue) {
        super(queue);
    }

    @Override
    public OHRequest next() {
        OHRequest result = ______________________;
        if (_________________________________________) {
            ______________________;
        }
        return ______________________;
    }

}
```


1C OHQueue

```
public class TYIterator extends OHIterator {

    public TYIterator(OHRequest queue) {
        super(queue);
    }

    @Override
    public OHRequest next() {
        OHRequest result = super.next();
        if (_____________________________________) {
            ____________________;
        }
        return ____________________;
    }

}
```


1C OHQueue

```
public class TYIterator extends OHIterator {

    public TYIterator(OHRequest queue) {
        super(queue);
    }

    @Override
    public OHRequest next() {
        OHRequest result = super.next();
        if (result.description.contains(“thank u”)) {
            ______________________;
        }
        return ______________________;
    }

}
```


1C OHQueue

```
public class TYIterator extends OHIterator {

    public TYIterator(OHRequest queue) {
        super(queue);
    }

    @Override
    public OHRequest next() {
        OHRequest result = super.next();
        if (result.description.contains(“thank u”)) {
            super.next();
        }
        return ______________________;
    }

}
```


1C OHQueue

```
public class TYIterator extends OHIterator {

    public TYIterator(OHRequest queue) {
        super(queue);
    }

    @Override
    public OHRequest next() {
        OHRequest result = super.next();
        if (result.description.contains(“thank u”)) {
            super.next();
        }
        return result;
    }

}
```


1D OHQueue

```
public static void main(String[] args) {
    OHRequest s5 = new OHRequest("I deleted all of my files, thank u", "Elana",
    true, null);
    OHRequest s4 = new OHRequest("conceptual: what is Java", "Stella", false, s5);
    OHRequest s3 = new OHRequest("git: I never did lab 1", "Omar", true, s4);
    OHRequest s2 = new OHRequest("help", "Angel", false, s3);
    OHRequest s1 = new OHRequest("no I haven't tried stepping through", "Ashley",
    false, s2);

    ____________________________;
    for (____________________________) {
        ___________________________________;
    }
}
```


1D OHQueue

```
public static void main(String[] args) {
    OHRequest s5 = new OHRequest("I deleted all of my files, thank u", "Elana",
    true, null);
    OHRequest s4 = new OHRequest("conceptual: what is Java", "Stella", false, s5);
    OHRequest s3 = new OHRequest("git: I never did lab 1", "Omar", true, s4);
    OHRequest s2 = new OHRequest("help", "Angel", false, s3);
    OHRequest s1 = new OHRequest("no I haven't tried stepping through", "Ashley",
    false, s2);

    OHQueue q = new OHQueue(s1);
    for (_____________________________) {
        ___________________________________;
    }
}
```


1D OHQueue

```
public static void main(String[] args) {
    OHRequest s5 = new OHRequest("I deleted all of my files, thank u", "Elana",
    true, null);
    OHRequest s4 = new OHRequest("conceptual: what is Java", "Stella", false, s5);
    OHRequest s3 = new OHRequest("git: I never did lab 1", "Omar", true, s4);
    OHRequest s2 = new OHRequest("help", "Angel", false, s3);
    OHRequest s1 = new OHRequest("no I haven't tried stepping through", "Ashley",
    false, s2);

    OHQueue q = new OHQueue(s1);
    for (OHRequest r : q) {
        ____________________________________;
    }
}
```


1D OHQueue

```
public static void main(String[] args) {
    OHRequest s5 = new OHRequest("I deleted all of my files, thank u", "Elana",
    true, null);
    OHRequest s4 = new OHRequest("conceptual: what is Java", "Stella", false, s5);
    OHRequest s3 = new OHRequest("git: I never did lab 1", "Omar", true, s4);
    OHRequest s2 = new OHRequest("help", "Angel", false, s3);
    OHRequest s1 = new OHRequest("no I haven't tried stepping through", "Ashley",
    false, s2);

    OHQueue q = new OHQueue(s1);
    for (OHRequest r : q) {
        System.out.println(r.name);
    }
}
```


1D OHQueue

```
public static void main(String[] args) {
    OHRequest s5 = new OHRequest("I deleted all of my files, thank u", "Elana",
    true, null);
    OHRequest s4 = new OHRequest("conceptual: what is Java", "Stella", false, s5);
    OHRequest s3 = new OHRequest("git: I never did lab 1", "Omar", true, s4);
    OHRequest s2 = new OHRequest("help", "Angel", false, s3);
    OHRequest s1 = new OHRequest("no I haven't tried stepping through", "Ashley",
    false, s2);

    OHQueue q = new OHQueue(s1);
    for (OHRequest r : q) {
        System.out.println(r.name);
    }
}
```

```
In the OHQueue class:
@Override
public Iterator<OHRequest> iterator() {
    return new OHIterator(queue);
    return new TYIterator(queue);
}
```


1E OHQueue

```
public class OHRequestComparator implements Comparator<_____________________________> {
    @Override
    public int compare(_____________________________ o1, _________________________ o2) {
        // use as many lines as you need









    }
}
```


1E OHQueue

```
public class OHRequestComparator implements Comparator<OHRequest> {
    @Override
    public int compare(OHRequest o1, OHRequest o2) {
        if (o1.isSetup && !o2.isSetup) {
            return -1;
        }









    }
}
```


1E OHQueue

```
public class OHRequestComparator implements Comparator<OHRequest> {
    @Override
    public int compare(OHRequest o1, OHRequest o2) {
        if (o1.isSetup && !o2.isSetup) {
            return -1;
        } else if (!o1.isSetup && o2.isSetup) {
            return 1;
        }









    }
}
```


1E OHQueue

```
public class OHRequestComparator implements Comparator<OHRequest> {
     @Override
     public int compare(OHRequest o1, OHRequest o2) {
          if (o1.isSetup && !o2.isSetup) {
               return -1;
          } else if (!o1.isSetup && o2.isSetup) {
               return 1;
          } else if (o1.description.equals("setup") && !o2.description.equals(“setup”)) {
               return -1;
          }




     }
}
```


1E OHQueue

```
public class OHRequestComparator implements Comparator<OHRequest> {
    @Override
    public int compare(OHRequest o1, OHRequest o2) {
        if (o1.isSetup && !o2.isSetup) {
            return -1;
        } else if (!o1.isSetup && o2.isSetup) {
            return 1;
        } else if (o1.description.equals("setup") && !o2.description.equals(“setup”)) {
            return -1;
        } else if (!o1.description.equals("setup") && o2.description.equals(“setup”)) {
            return 1;
        }


    }
}
```


1E OHQueue

```
public class OHRequestComparator implements Comparator<OHRequest> {
     @Override
     public int compare(OHRequest o1, OHRequest o2) {
          if (o1.isSetup && !o2.isSetup) {
               return -1;
          } else if (!o1.isSetup && o2.isSetup) {
               return 1;
          } else if (o1.description.equals("setup") && !o2.description.equals(“setup”)) {
               return -1;
          } else if (!o1.description.equals("setup") && o2.description.equals(“setup”)) {
               return 1;
          }
          return 0;

     }
}
```


1E OHQueue (alternate solution)

```
public class OHRequestComparator implements Comparator<OHRequest> {
     @Override
     public int compare(OHRequest o1, OHRequest o2) {
          if (o1.isSetup == o2.isSetup) {
               return Boolean.compare(o1.description.equals("setup"),
                                      o2.description.equals("setup"));
          }
          return Boolean.compare(o1.isSetup, o2.isSetup)
     }
}
```