Making a Class Comparable

Cody Henrichsen

Java 2019

How to compare objects

- Can’t use the relational operators on Objects!!!
- Comparable interface
- String Comparison
  - Already made just for you
- Custom Implementation
  - How to do the thing
- Why???
  - Sorting things

Comparable Interface

- This is what allows for items to be sorted. AKA you need to be able to compare one instance to another instance.
- Since they are objects methods must be used, not symbols
- To standardize this across the language the Java type: interface is used so that all objects that implement this interface will function the same regardless of the implementation.
  - AKA if you know how to sort cars, you can sort bananas too
- Only one method to implement: public int compareTo(Type compared);

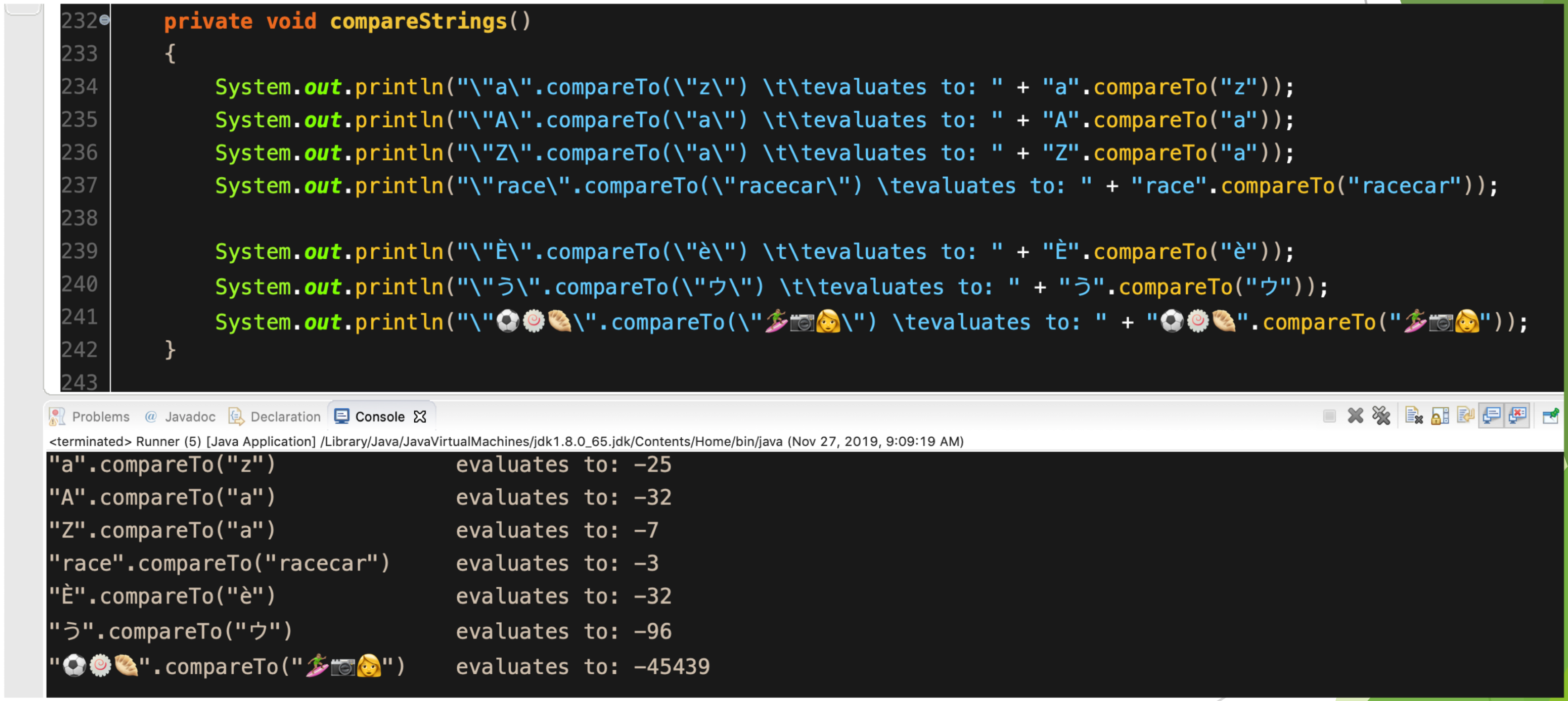
String Comparison

- The AP course description already talks about using the `.compareTo` method of a String that is provided by Java
- This is a English lexicographic comparison AKA where in the dictionary would the String on the left(calling) be found compared to the parameter (right)
- All capital letters have a lower ASCII/UNICODE value than lower case
  - All the ASCII values are earlier than everything else (Basic Latin)
  - A-Z is lower than a-z (32 apart)
  - Language dependent
    - Not all languages are contiguous
  - Emoji are not contiguous!

String compareTo Examples

- ”A”.compareTo(“a”) returns a negative value
- ”a”.compareTo(“a”) returns 0
- ”z”.compareTo(“a”) returns a positive value

String Comparison Code



```java
    private void compareStrings()
    {
        System.out.println("\"a\".compareTo(\"z\") \t\tevaluates to: " + "a".compareTo("z"));
        System.out.println("\"A\".compareTo(\"a\") \t\tevaluates to: " + "A".compareTo("a"));
        System.out.println("\"Z\".compareTo(\"a\") \t\tevaluates to: " + "Z".compareTo("a"));
        System.out.println("\"race\".compareTo(\"racecar\") \tevaluates to: " + "race".compareTo("racecar"));

        System.out.println("\"È\".compareTo(\"è\") \t\tevaluates to: " + "È".compareTo("è"));
        System.out.println("\"う\".compareTo(\"ウ\") \t\tevaluates to: " + "う".compareTo("ウ"));
        System.out.println("\"⚽🍥🥟\".compareTo(\"🏄🏽‍♀️📷👩\") \tevaluates to: " + "⚽🍥🥟".compareTo("🏄🏽‍♀️📷👩"));
    }

```

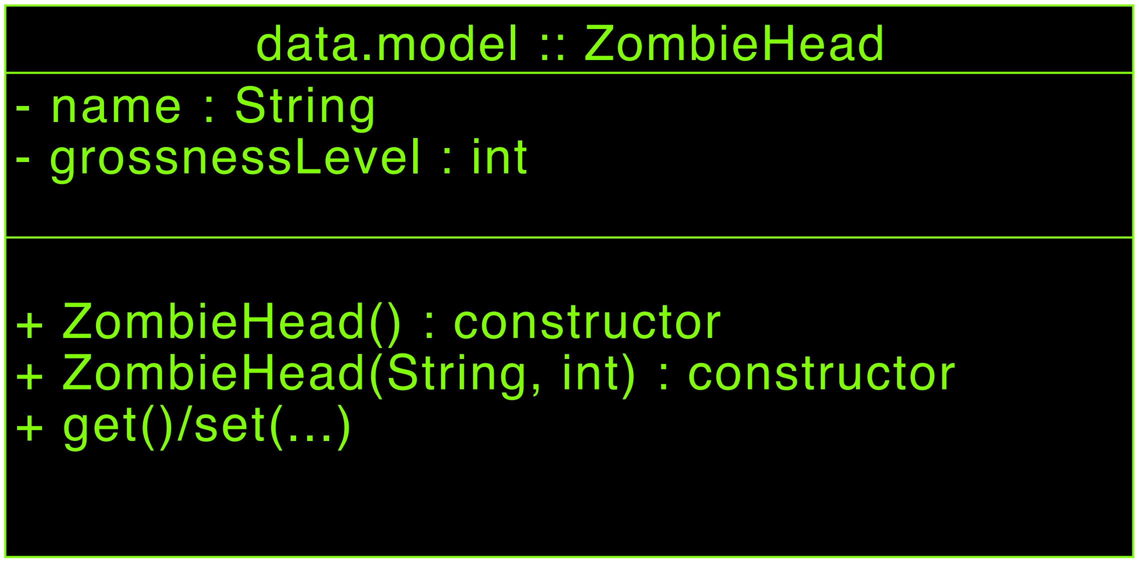
Problems @ Javadoc Declaration Console

```
<terminated> Runner (5) [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_65.jdk/Contents/Home/bin/java (Nov 27, 2019, 9:09:19 AM)
"a".compareTo("z")              evaluates to: -25
"A".compareTo("a")              evaluates to: -32
"Z".compareTo("a")              evaluates to: -7
"race".compareTo("racecar")     evaluates to: -3
"È".compareTo("è")              evaluates to: -32
"う".compareTo("ウ")             evaluates to: -96
"⚽🍥🥟".compareTo("🏄🏽‍♀️📷👩")      evaluates to: -45439
```

Comparable<Type>

- This is the interface that provides custom classes to be sorted
- Only one method required for implementation
  - int compareTo(Type variable)
- Design the comparison in advance!!!!!
  - Weeks of code can save hours of planning
  - Needs to be transitive
  - Needs to be clear

ZombieHead UML

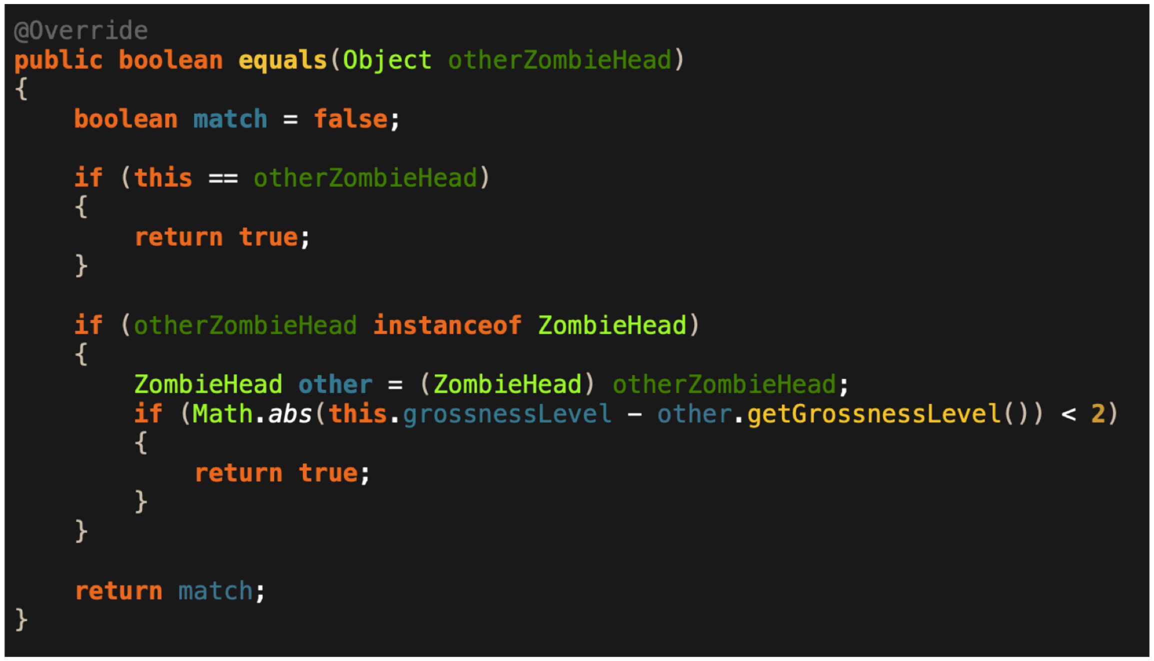


```
data.model :: ZombieHead
- name : String
- grossnessLevel : int

+ ZombieHead() : constructor
+ ZombieHead(String, int) : constructor
+ get()/set(...)
```

ZombieHead comparison

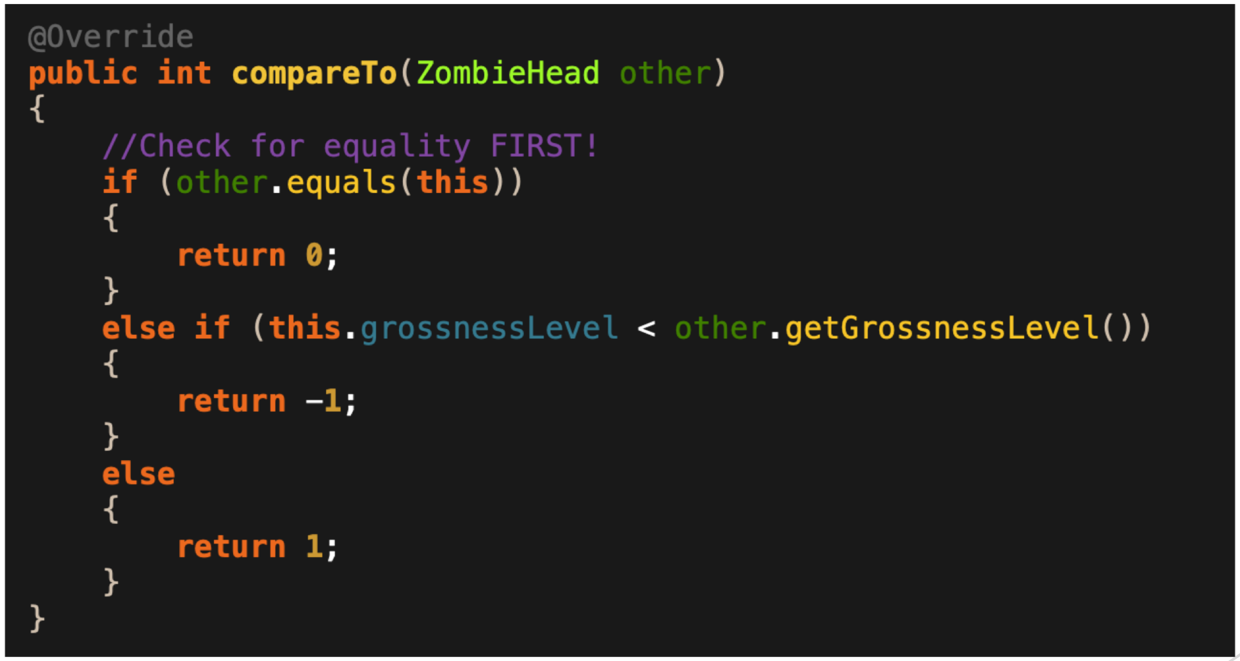
- We started with the .equals method and it was a bit squishy but that only allows for binary comparison AKA is this the same as the other
- We need to make the compareTo method look at what makes one ZombieHead greater or lesser than another
- The data members are where we see the differences
- Name can be VERY distinct and unlikely to ever have a match
- GrossnessLevel is discrete and easily measurable and so is the source for the comparison
- This showed the squishy method of evaluating equality from the previous video was too squishy
  - Updated the absolute difference from less than 20 to be less than 2

ZombieHead equals implementation



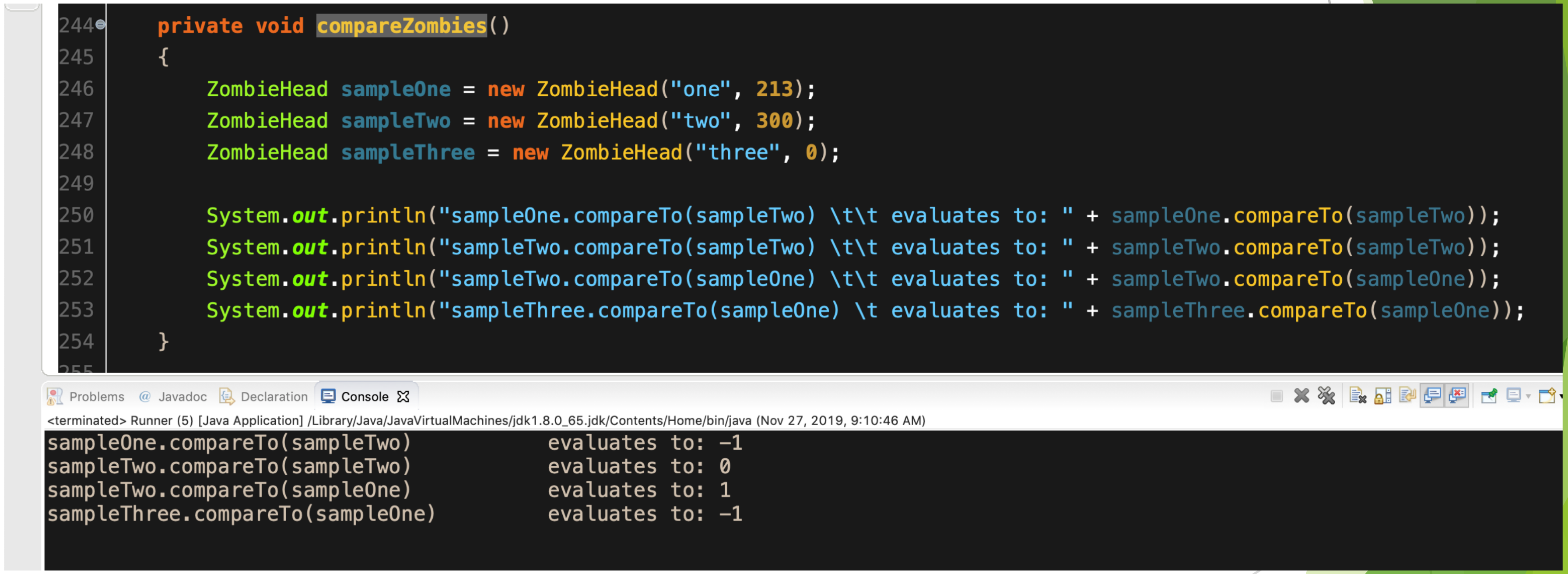
```
@Override
public boolean equals(Object otherZombieHead)
{
    boolean match = false;

    if (this == otherZombieHead)
    {
        return true;
    }

    if (otherZombieHead instanceof ZombieHead)
    {
        ZombieHead other = (ZombieHead) otherZombieHead;
        if (Math.abs(this.grossnessLevel - other.getGrossnessLevel()) < 2)
        {
            return true;
        }
    }

    return match;
}
```

ZombieHead compareTo implementation



```
@Override
public int compareTo(ZombieHead other)
{
    //Check for equality FIRST!
    if (other.equals(this))
    {
        return 0;
    }
    else if (this.grossnessLevel < other.getGrossnessLevel())
    {
        return -1;
    }
    else
    {
        return 1;
    }
}
```

ZombieHead Comparison Code

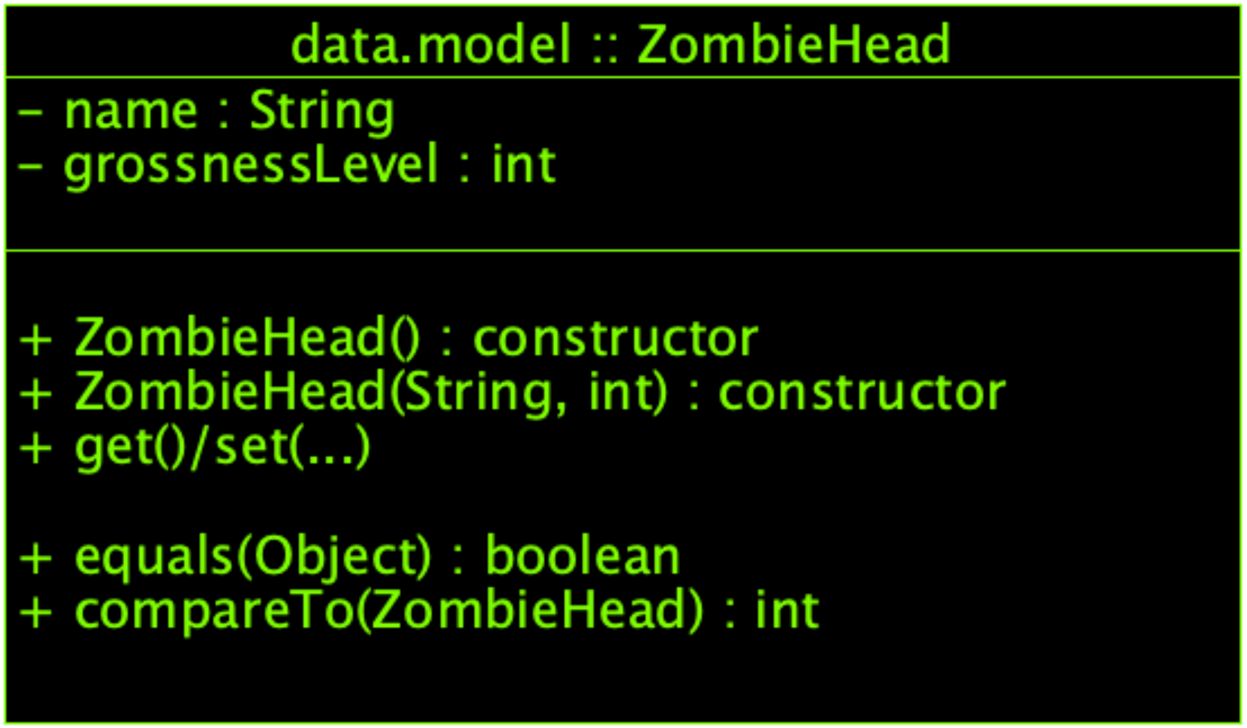


```java
    private void compareZombies()
    {
        ZombieHead sampleOne = new ZombieHead("one", 213);
        ZombieHead sampleTwo = new ZombieHead("two", 300);
        ZombieHead sampleThree = new ZombieHead("three", 0);

        System.out.println("sampleOne.compareTo(sampleTwo) \t\t evaluates to: " + sampleOne.compareTo(sampleTwo));
        System.out.println("sampleTwo.compareTo(sampleTwo) \t\t evaluates to: " + sampleTwo.compareTo(sampleTwo));
        System.out.println("sampleTwo.compareTo(sampleOne) \t\t evaluates to: " + sampleTwo.compareTo(sampleOne));
        System.out.println("sampleThree.compareTo(sampleOne) \t evaluates to: " + sampleThree.compareTo(sampleOne));
    }
```

<terminated> Runner (5) [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_65.jdk/Contents/Home/bin/java (Nov 27, 2019, 9:10:46 AM)

```
sampleOne.compareTo(sampleTwo)          evaluates to: -1
sampleTwo.compareTo(sampleTwo)          evaluates to: 0
sampleTwo.compareTo(sampleOne)          evaluates to: 1
sampleThree.compareTo(sampleOne)        evaluates to: -1
```

Updated ZombieHead UML



```
data.model :: ZombieHead
- name : String
- grossnessLevel : int

+ ZombieHead() : constructor
+ ZombieHead(String, int) : constructor
+ get()/set(...)

+ equals(Object) : boolean
+ compareTo(ZombieHead) : int
```