How to implement compareTo

What is compareTo?

`compareTo` is a method in the `Comparabale` interface that many other Java classes rely on to know how to compare objects against each other. For example, the String class utilizes compareTo in order to compare two Strings together and see if they are less than, equal to, or greater than each other.

The actual implementation of `compareTo` is up to you. That is, you can decide which attributes of your class you want to compare to decide if the objects or less than, equal to, or greater than each other.

The only rule is you must implement `Comparable` interface, and you must follow the standard formart of:

- If object is determined to be less than, return a negative number (-1)
- If object is determined to be equal to, return 0
- If object is determined to be greater than, return a positive value (1)

Step 1: Implement Comparable

On the class you want to add `compareTo to`, add `implements Comparable<Class>` to the class declaration, where `Class` is the name of your class. For example:

```
public class Bike implements Comparable<Bike> {
    ....
}
```

If you are implementing other interfaces, just seperate them with a comma. For example:

```
public class Bike implements Serializable, Comparable<Bike> {
    ....
}
```

Step 2: Implement compareTo method

As soon as you add the `Comparable` interface to your class, you'll see a red squiggle line in IntelliJ. That's because the `Comparable` interface requires you to implement the `compareTo` method according to it's specifications.

Do this by writing the following method signature, making sure to include the `@Override` annotation.

```
@Override
    public int compareTo(Bike bike) {

        return 0;
    }
```

The parameter you accept into the method will be the class you are writing this method in. You are comparing another object of the same type to the this instance of the object.

Inside the method you implement your custom logic to determine if the object passed in is less than, equal to, or greater than the current instance.

The below example compares bike objects based on `wheelSize`. You can compare on any attribute, and/or multiple attributes!

```
@Override
public int compareTo(Bike bike) {
    if(this.wheelSize < bike.getWheelSize()) {
        return -1;
    } else if (this.wheelSize > bike.getWheelSize()) {
        return 1;
    }
    // objects wheelSize are the same, return 0
    return 0;
}
```

Step 3: Using this compareTo to sort an ArrayList

One of the common uses of `compareTo` is to sort a List. There are many ways to sort, but an easy built-in method is in the `Collections` class.

```
Collections.sort(bikeArrayList);
```

This only works because I implemented the `Comparable` interface on the `Bike` class. `Collections.sort()` expects that the List you pass is comprised of a class that implements `Comparable`. If it did not, you would see a red squiggle error in IntelliJ, and it would let you know it needs to implement `Comparable`.

Behind the scenes, `Collections.sort()` calls the `compareTo` method you created inside your class to know how to sort the objects inside the list. By default, it is in ascending order.