Anonymous classes

Implementing an interface on-the-fly, inline

Implementing an interface on-the-fly

- You have a simple interface (perhaps only one abstract method)
- You need an instance of an implementing class
- You don’t think you will need to publish (create public) an implementing class, you’d rather need an instance here and now™
- You want to keep the implementation close to where it is being used

Enter anonymous classes!

An anonymous class - extending Object

- Doesn’t have a name - just a body and the name of the interface or parent class
- If the parent is a class, the implementation will extend it
- If the parent is an interface, the implementation will implement it
- follows the operator `new` (even for interfaces), parent name and `()`

```
System.out.println(
    new Object() {   @Override
                     public String toString() {
                       return "I am anonymous";
                     }
                  } );
```

An anonymous class - implementing Comparable

- Comparator has one abstract method:

```
int compare(T o1, T o2)
```

- Can be implemented using an anonymous class

```
List<Product> products = ... // get products from somewhere
// sort on price:
Collections.sort(products, new Comparator<Product>() {
    @Override
    int compare(Product p1, Product p2) {
      return p1.price() - p2.price();
    }
  });
```

An anonymous class - assignment

- If you need to use it more than once, but don’t see the need for a stand-alone class, you can assign a reference variable the object

```
List<Product> products = ... // get products from somewhere
Comparator<Product> priceComparator =
  new Comparator<Product>() {
    @Override
    int compare(Product p1, Product p2) {
      return p1.price() - p2.price();
    }
  };
Collections.sort(products, priceComparator);
```

An anonymous class - Example program

```
import java.util.Comparator;
import java.util.List;
import java.util.Collections;
import java.util.Arrays;

public class TestAnonymous{

  public static void main(String[] args) {

    List<String> strings = Arrays.asList(args);
    System.out.println("Unsorted: " + strings);
    Collections.sort(strings, new Comparator<String>() {
                @Override
                public int compare(String first, String other) {
                  return first.toLowerCase().compareTo(other.toLowerCase());
                }
      });
    System.out.println("Sorted disregarding case: " + strings);
    Collections.sort(strings);
    System.out.println("Sorted using String's default compareTo: " + strings);
  }
}
```

```
$ javac TestAnonymous.java && java TestAnonymous a B c D e b
d
Unsorted: [a, B, c, D, e, b, d]
Sorted disregarding case: [a, B, b, c, D, d, e]
Sorted using String's default compareTo: [B, D, a, b, c, d,
e]
```

Further reading

- https://docs.oracle.com/javase/tutorial/java/javaOO/anonymousclasses.html
- https://docs.oracle.com/javase/tutorial/java/javaOO/whentouse.html