Beverage.java

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
```

```
public class Beverage extends Product implements Edible {

    private int calories;
    private double fluidOunces;

    public Beverage(int price, String name,
                int calories, double fluidOunces) {
        super(price, name);
        this.calories = calories;
        this.fluidOunces = fluidOunces;
    }

    public int getCalories() {return this.calories;}
    public double getFluidOunces() {return this.fluidOunces;}
}
```

1

Handout A for e01 CS56 W20

Edible.java

```
1
2
3
4
5
```

```
/** something that can be eaten */
@FunctionalInterface
public interface Edible {
    public int getCalories();
}
```

Food.java

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
```

```
public class Food extends Product implements Edible {

    private int calories;
    private double weight;

    public Food(int price, String name,
                int calories, double weight) {
        super(price, name);
        this.calories = calories;
        this.weight = weight;
    }

    public int getCalories() {return this.calories;}
    public double getWeight() {return this.weight;}
}
```

Note: `FreeCandy` and `Product` are on Handout B.

Code for

`TraderBobs` problem

2

Handout A for e01 CS56 W20

Useful Reference Items related to Sorting

Here are a few reminders of things we discussed in class, but that you might reasonably need a "reference" for if you were using them in the real world.

The interface `java.util.Comparator<T>` includes the following method signature:

`int`	`compare(T o1, T o2)`	Compares its two arguments for order. Returns a negative integer, zero, or a positive integer as the first argument is less than, equal to, or greater than the second.

The interface `java.lang.Comparable<T>` includes the following method signature:

`int`	`compareTo(T o)`	Compares this object with the specified object for order. Returns a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

The class `java.util.ArrayList<E>` includes this method:

`void`	`sort(Comparator<? super E> c)`	Sorts this list according to the order induced by the specified `Comparator`.

The class `java.util.Collections` contains the following static method:

`static <T extends Comparable<? super T>> void sort(List<T> list)`	Sorts the specified list into ascending order, according to the natural ordering of its elements.

The classes `java.lang.String` and `java.lang.Double` implement `Comparable<String>` and `Comparable<Double>`, each in the way that you would expect.

Other potentially useful methods

In `java.lang.Integer`:

`public static int compare(int i1, int i2)`	Compares the two specified int values. The sign of the int value returned matches the contract of the `compare` method in `java.util.Comparator`

In `java.lang.Double`:

`public static int compare(double d1, double d2)`	Compares the two specified double values. The sign of the int value returned matches the contract of the `compare` method in `java.util.Comparator`

End of Handout