1

Handout A for e01 CS56 M18

Beverage.java

```
1   public class Beverage extends Product implements Edible {
2
3       private int calories;
4       private double fluidOunces;
5
6       public Beverage(int price, String name,
7                   int calories, double fluidOunces) {
8           super(price, name);
9           this.calories = calories;
10          this.fluidOunces = fluidOunces;
11      }
12
13      public int getCalories() {return this.calories;}
14      public double getFluidOunces() {return this.fluidOunces;}
15  }
```

Edible.java

```
1   /** something that can be eaten */
2   public interface Edible {
3       public int getCalories();
4   }
```

Food.java

```
1   public class Food extends Product implements Edible {
2
3       private int calories;
4       private double weight;
5
6       public Food(int price, String name,
7                   int calories, double weight) {
8           super(price, name);
9           this.calories = calories;
10          this.weight = weight;
11      }
12
13      public int getCalories() {return this.calories;}
14      public double getWeight() {return this.weight;}
15  }
```

FreeCandy.java

```
1   public class FreeCandy implements Edible {
2
3       private int calories;
4
5       public FreeCandy(int calories) {
6           this.calories = calories;
7       }
8
9       public int getCalories() {return this.calories;}
10  }
```

Product.java

```
1   public abstract class Product {
2       String name;
3       int price;
4
5       public int getPrice() { return price; }
6       public String getName() {return name;}
7
8       public Product(int price, String name) {
9           this.price = price;
10          this.name = name;
11      }
12  }
```

Code for

`TraderBobs` problem

2

Handout A for e01 CS56 M18

Handout A, p. 2

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

`int`	`compareTo(T o)`	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 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`

End of Handout