```java
public interface Taxable
{
   public abstract int calculateTax(int taxRate);
}

```

```java
import java.io.*; // Needed for Serializable

public abstract class Product implements Taxable, Serializable, Comparable<Product>
{

   public static final int TAXRATE = 15;
   private String barcodeNumber;
   private String name;
   private int unitPrice;

   public Product()
   {
      this("000000000000","",0);
   }

   public Product(String no, String nm, int up)
   {
      setBarcodeNumber(no);
      setName(nm);
      setUnitPrice(up);
   }

//************************************************************
//   public String getOutputString();
/* May not use toString method to display product info.
   Output strings for the two types of products differ,
   therefore the methods will have different implementations.
*/
//************************************************************

//************************************************************
   public String getOutputString()
   {
      return "Barcode "+getBarcodeNumber()+" is "+getName();
   };
/* Alternative is to implement the part that is common to all
   the subclasses
*/
//************************************************************


   public void setBarcodeNumber(String no)
   {
      barcodeNumber = no;
   }

   public void setName(String nm)
   {
      name = nm;
   }

   public void setUnitPrice(int up)
   {
      unitPrice = up;
   }

   public String getBarcodeNumber()
   {
      return barcodeNumber;
   }

   public String getName()
   {
      return name;
   }

   public int getUnitPrice()
   {
      return unitPrice;
   }

   public abstract int calculateTotalCost(int taxRate);

//*************************************************************
/*   public int compareTo(Product other)
   {
      return this.getName().compareTo(other.getName());
   }*/
/* If all the subclasses are to be sorted in the same way we can
   put the compareTo in the superclass.
   Here we want to sort on Name.
*/
//*************************************************************

}
```

```java
public class RetailProduct extends Product
{

   private String manufacturer;

   public RetailProduct()
   {
      this("000000000000","","",0);
   }

   public RetailProduct(String no, String nm, String man, int up)
   {
      super(no, nm, up);
      setManufacturer(man);
   }

   public void setManufacturer(String man)
   {
      manufacturer = man;
   }

   public String getManufacturer()
   {
      return manufacturer;
   }

   public int calculateTax(int taxRate)
   {
      return (int) (getUnitPrice()*taxRate/100.0);
   }

   public int calculateTotalCost(int taxRate)
   {
      return getUnitPrice() + calculateTax(taxRate);
   }

   public String getOutputString()
   {
      return super.getOutputString()+" from "+getManufacturer()+", unit price
"+getUnitPrice()+"c, total price "+calculateTotalCost(TAXRATE)+"c";  // The total cost
was not included in the assignment....
   }

/* If all the subclasses are to be sorted the same way we can put the compareTo method in
the
   superclass with no compareTo in the subclasses. However, if we are to sort the
subclasses differently
   we need a compareTo method in each subclass - as is the case here.
   Sorting results in all the retail products being first. Retail products are then in
alphabetical order on name.
   */
//There were two sorting questions in the assignment, this is the compareTo for the
second one
   public int compareTo(Product other)
   {
      String thisField = getBarcodeNumber().charAt(0)+getName();
      String otherField;
      char otherProductCode = other.getBarcodeNumber().charAt(0);
      if(otherProductCode == '2') // Can also use: if(other instanceof
WeightBasedProduct)
         otherField = otherProductCode+other.getName()+
((WeightBasedProduct)other).getWeight();
      else
```

```java
            otherField = otherProductCode+other.getName();
        return thisField.compareTo(otherField);
    }

    public static void main(String[] args)
    {
        RetailProduct prod1 = new RetailProduct("178030640224","Milk, 1L","Clover",1550);
        System.out.println(prod1.getOutputString());
    }
}
```

```java
public class WeightBasedProduct extends Product
{

   private int weight;

   public WeightBasedProduct()
   {
      this("000000000000","",0,0);
   }

   public WeightBasedProduct(String no, String nm, int up, int wt)
   {
      super(no, nm, up);
      setWeight(wt);
   }

   public void setWeight(int wt)
   {
      weight = wt;
   }

   public int getWeight()
   {
      return weight;
   }

   public int calculateTax(int taxRate)
   {
      return (int) (getUnitPrice() * (getWeight()/1000.0) * taxRate / 100.0); // 1000.0 is not a "magic number" it is a unit conversion constant.
   }

   public int calculateTotalCost(int taxRate)
   {
      return (int) (getUnitPrice() * (getWeight()/1000.0) + calculateTax(taxRate));
   }

   public String getOutputString()
   {
      return super.getOutputString()+", unit price "+getUnitPrice()+"c/Kg, weight "+getWeight()+" grams, total price "+calculateTotalCost(TAXRATE)+"c";  // The total cost was not included in the assignment....
   }
   /* If all the subclasses are to be sorted the same way we can put the compareTo method in the
      superclass with no compareTo in the subclasses. However, if we are to sort the subclasses differently
      we need a compareTo method in each subclass - as is the case here.
      Sorting results in weight-based products being after retail products. Products are then in order according to name,
      products with the same name are then sorted on weight.
      */
   //There were two sorting questions in the assignment, this is the compareTo for the second one
   public int compareTo(Product other)
   {
      String thisField = getBarcodeNumber().charAt(0)+getName()+getWeight();
      String otherField;
      if(other instanceof WeightBasedProduct)
         otherField = other.getBarcodeNumber().charAt(0)+other.getName()+((WeightBasedProduct)other).getWeight();
      else
         otherField = other.getBarcodeNumber().charAt(0)+other.getName();
```

```java
        return thisField.compareTo(otherField);
    }

    public static void main(String[] args)
    {
        WeightBasedProduct prod1 = new WeightBasedProduct("245134867531","Bananas",
4300,540);
        System.out.println(prod1.getOutputString());
    }
}
```

```
import java.io.*;
import java.util.*;

public class Basket // Take note that this is my application program and that I do not
extend or implement anything!!!!!
{

   public static final int NO_OF_PRODUCTS = 25;

   static ObjectOutputStream output;

   public static void openOutput()
   {
      try
      {
         output = new ObjectOutputStream(new FileOutputStream("basket.ser"));
      } catch (IOException e)
      {
         System.out.println("Cannot open output file.");
      }
   }

   public static void closeOutput()
   {
      try
      {
         output.close();
      }
      catch (IOException e)
      {
         System.out.println("Cannot close output file.");
      }
   }

   public static void main(String args[])
   {

      Product productList[] =new Product[NO_OF_PRODUCTS];
      int count = 0;

      try
      {
         Scanner input = new Scanner(new FileReader("productdata.txt"));
         String inLine, barcode, name, manufacturer;
         int unitPrice, weight;
         char productCode;
         while (input.hasNext())
         {
            inLine = input.nextLine();
            String[] fields = inLine.split("#");
            barcode = fields[0];
            productCode = barcode.charAt(0);
            name = fields[1];
            if (productCode=='1') // Retail product
            {
               manufacturer = fields[2];
               unitPrice = Integer.parseInt(fields[3]);
               productList[count] = new RetailProduct(barcode, name, manufacturer,
unitPrice);
               count++;
            }
            else if (productCode=='2') // Weight-based product
            {
                unitPrice = Integer.parseInt(fields[2]);
                weight = Integer.parseInt(fields[3]);
                productList[count] = new WeightBasedProduct(barcode, name, unitPrice, 
weight);
                count++;
             }
             else
             {
                System.out.println("Invalid input "+inLine);
             }
          }
       }
       catch(FileNotFoundException e)
       {
          System.out.println("ERROR: "+e);
       }

       Product[] tempList = Arrays.copyOf(productList,count);
       Arrays.sort(tempList);
       for(int i = 0; i <tempList.length; i++)
          System.out.println(tempList[i].getOutputString());

       openOutput();
       try
       {
          for (int i =0; i<count; i++)
             output.writeObject(productList[i]);
       }

       catch (IOException e)
       {
          System.out.println("Problem writing product to file: "+e);
       }
       closeOutput();
    }
 }
```