```java
package class17;

import java.util.ArrayList;

public class Driver {

    public static void main(String[] args) {
        System.out.println("************** BASE SETUP **************");
        ArrayList<Product> shoppingCart = new ArrayList();
        shoppingCart.add(new Foodstuff("Chips", 4.99));
        shoppingCart.add(new Medicine("Motrin", 8.99));
        shoppingCart.add(new HouseholdGood("Paper Towels", 2.49));
        for (Product eachOne: shoppingCart) {
            System.out.println(eachOne.toString());
        } // end for

        System.out.println("************** POLYMORPHIC INTERFACES **************");
        Taxable taxOne = new Foodstuff("Ramen Noodles", 0.99);
        Taxable taxTwo = new HouseholdGood("Toilet Paper", 1.49);
        System.out.println("The tax for " + taxOne.toString() + " is: $" +
taxOne.calcSalesTax());
        System.out.println("The tax for " + taxTwo.toString() + " is: $" +
taxTwo.calcSalesTax());

        // Because Medicine is not Taxable, it precludes us from putting everything in a
        // collection of type Taxable and calculating taxes the easy way

        // The next two lines of code are invalid
        //Taxable taxThree = new Medicine("Tylenol", 9.99);
        //System.out.println("The tax for " + taxThree.toString() + " is: $" +
taxThree.calcSalesTax());

        System.out.println("************** EXPLICIT TYPE CASTING **************");
        // This line of code is invalid, taxOne is type Taxable and can't "see" the .getName()
method
        //System.out.println("The name value contained in taxOne is: " + (taxOne.getName()));

        // By explicitly type casting it as a Product, we can now "see" the .getName() method
        System.out.println("The name value contained in taxOne is: " +
((Product)taxOne).getName());

        System.out.println("************** instanceof EXAMPLE **************");
        System.out.println("The variable taxOne is a Product: " + (taxOne instanceof Product));
        System.out.println("The variable taxOne is a Foodstuff: " + (taxOne instanceof
Foodstuff));
        System.out.println("The variable taxOne is a Taxable: " + (taxOne instanceof Taxable));
        System.out.println("The variable taxOne is a Medicine: " + (taxOne instanceof
Medicine));
        System.out.println("The variable taxOne is a HouseholdGood: " + (taxOne instanceof
HouseholdGood));

        System.out.println("************** POLYMORPHIC INTERFACES, COLLECTION & instanceof
**************");
        for (Product eachOne: shoppingCart) {
            double taxValue = 0;
            if (eachOne instanceof Taxable) {
                taxValue = ((Taxable)eachOne).calcSalesTax();
```

```
49              } // end if
50              System.out.println(eachOne.toString() + " sales tax is: " +
51                      java.text.NumberFormat.getCurrencyInstance().format(taxValue));
52          } // end for
53      } // end main
54  } // end Driver
```

Product.java

```java
package class17;

public abstract class Product {
    private final String name;
    private final double price;

    public Product(String incName, double incPrice) {
        this.name = incName;
        this.price = incPrice;
    } // end ctor

    public String getName() {
        return this.name;
    } // end getName

    public double getPrice() {
        return this.price;
    } // end getPrice

    @Override
    public String toString() {
        // % indicates a format string pattern
        // 0$ indicates the argument index
        // s indicates that the argument is a String
        // 15 represents the minimal width of the String
        return "$" + this.price + String.format("%0$"+15+"s", this.name);
    } // end toString
} // end Product
```

Foodstuff.java

```java
package class17;

public class Foodstuff extends Product implements Taxable {

    public Foodstuff(String incName, double incPrice) {
        super(incName, incPrice);
    } // end ctor

    @Override
    public double calcSalesTax() {
        return this.getPrice() * Taxable.SALESTAXRATE;
    } // end calcSalesTax
} // end Foodstuff
```

Medicine.java

```java
package class17;

public class Medicine extends Product {

    public Medicine(String incName, double incPrice) {
        super(incName, incPrice);
    } // end ctor
} // end Medicine
```

HouseholdGood.java

```java
package class17;

public class HouseholdGood extends Product implements Taxable {

    public HouseholdGood(String incName, double incPrice) {
        super(incName, incPrice);
    } // end ctor

    @Override
    public double calcSalesTax() {
        return this.getPrice() * Taxable.SALESTAXRATE;
    } // end calcSalesTax
} // end HouseholdGood
```

```java
package class17;

public interface Taxable {
    public double SALESTAXRATE = 0.085;
    public abstract double calcSalesTax();
} // end Taxable
```