

SOURCE CODE

```
public class Client {

    public static void runTest()
    {
        Component theOrder = BuildOrder.getOrder() ;
        theOrder.printDescription();
    }
}

public class BuildOrder {

    public static Component getOrder()
    {
        Component order = new Composite( "Order" ) ;
        order.addChild(
            new Leaf("Crispy Onion Strings", 5.50 ));
        order.addChild(
            new Leaf("The Purist", 8.00 ));
        CustomBurger customBurger = new
            CustomBurger( "Build Your Own Burger" ) ;

        // base price for 1/3 lb
        Burger b = new Burger( "Burger Options" ) ;
        String[] bo =
            { "Beef", "1/3lb.", "On A Bun" } ;
        b.setOptions( bo ) ;

        // 1 cheese free, extra cheese +1.00
        Cheese c = new Cheese( "Cheese Options" ) ;
        String[] co =
            { "Danish Blue Cheese",
              "Horseradish Cheddar" } ;
        c.setOptions( co ) ;
        c.wrapDecorator( b ) ;

        // 4 toppings free, extra +.75
        Toppings t = new Toppings(
            "Toppings Options" ) ;
        String[] to =
            { "Bermuda Red Onion",
              "Black Olives", "Carrot Strings",
              "Coleslaw" } ;
        t.setOptions( to ) ;
        t.wrapDecorator( c ) ;

        // premium topping +1.50
        Premium p =
            new Premium( "Premium Options" ) ;
        String[] po = { "Applewood Smoked Bacon" } ;
        p.setOptions( po ) ;
        p.wrapDecorator( t ) ;

        // 1 sauce free, extra +.75
        Sauce s = new Sauce( "Sauce Options" ) ;
        String[] so = { "Apricot Sauce" } ;
        s.setOptions( so ) ;
        s.wrapDecorator( p ) ;

        // Setup Custom Burger Ingredients
        customBurger.setDecorator( s ) ;
        customBurger.addChild( b ) ;
        customBurger.addChild( c ) ;
        customBurger.addChild( t ) ;
        customBurger.addChild( p ) ;
        customBurger.addChild( s ) ;

        // Add Custom Burger to the ORder
        order.addChild( customBurger );
        return order ;
    }
}
```

```
public interface Component {

    void printDescription() ;
    void addChild(Component c);
    void removeChild(Component c);
    Component getChild(int i);
}

public class Leaf implements Component {

    private String description ;
    protected Double price ;

    public Leaf ( String d, Double p )
    {
        description = d ;
        price = p ;
    }

    public Leaf ( String d )
    {
        description = d ;
        price = 0.0 ;
    }

    public void printDescription() {
        DecimalFormat fmt =
            new DecimalFormat("0.00");
        System.out.println(
            description + " " + fmt.format(price) ) ;
    }

    public void addChild(Component c) {
        // no implementation
    }

    public void removeChild(Component c) {
        // no implementation
    }

    public Component getChild(int i) {
        // no implementation
        return null ;
    }
}

public class Composite implements Component {

    protected ArrayList<Component>
        components = new ArrayList<Component>() ;
    protected String description ;

    public Composite ( String d )
    {
        description = d ;
    }

    public void printDescription() {
        System.out.println( description );
        for (Component obj : components)
        {
            obj.printDescription();
        }
    }

    public void addChild(Component c) {
        components.add( c ) ;
    }

    public void removeChild(Component c) {
        components.remove(c) ;
    }

    public Component getChild(int i) {
        return components.get( i ) ;
    }
}
```

```

}

public interface PriceDecorator
{
    Double getPrice();
}

public class CustomBurger extends Composite
{
    PriceDecorator decorator = null ;

    public CustomBurger ( String d )
    {
        super(d) ;
    }

    public void setDecorator( PriceDecorator p )
    {
        this.decorator = p ;
    }

    public void printDescription() {
        DecimalFormat fmt = new DecimalFormat("0.00");
        System.out.println( description + "
        "=fmt.format(decorator.getPrice()) );

        for (Component obj : components)
        {
            obj.printDescription();
        }
    }
}

public abstract class LeafDecorator extends Leaf
implements PriceDecorator
{
    PriceDecorator wrapped ;

    public LeafDecorator( String d ) {
        super( d ) ;
        this.wrapped = null ;
    }

    public void wrapDecorator( PriceDecorator w )
    {
        this.wrapped = w ;
    }

    public Double getPrice() {
        if (wrapped == null )
        {
            return price ;
        }
        else
        {
            return price + wrapped.getPrice() ;
        }
    }

    abstract public void setOptions(
        String[] options ) ;
    abstract public String getDescription() ;

    @Override
    public void printDescription() {
        System.out.println( getDescription() ) ;
    }
}

public class Burger extends LeafDecorator
{
    private String[] options ;

    public Burger( String d )
    {
        super(d) ;
    }

    public void setOptions( String[] options )
    {
        this.options = options ;
        for ( int i = 0; i<options.length; i++ )
        {
            if ( "1/3lb.".equals(options[i]) )
                this.price += 9.50 ;
            if ( "2/3lb.".equals(options[i]) )
                this.price += 11.50 ;
            if ( "1lb.".equals(options[i]) )
                this.price += 15.50 ;
            if (
                "In A Bowl".equals(options[i]) )
                this.price += 1.50 ;
        }
    }

    public String getDescription()
    {
        String desc = "" ;
        for ( int i = 0; i<options.length; i++ )
        {
            if (i>0) desc += " + " + options[i] ;
            else desc = options[i] ;
        }
        return desc ;
    }
}

public class Cheese extends LeafDecorator
{
    private String[] options ;

    public Cheese( String d )
    {
        super(d) ;
    }

    // 1 cheese free, extra cheese +1.00
    public void setOptions( String[] options )
    {
        this.options = options ;
        if ( options.length > 1 )
            this.price +=
                (options.length-1) * 1.00 ;
    }

    public String getDescription()
    {
        String desc = "" ;
        for ( int i = 0; i<options.length; i++ )
        {
            if (i>0) desc += " + " + options[i] ;
            else desc = options[i] ;
        }
        return desc ;
    }
}

```

```

public class Toppings extends LeafDecorator
{
    private String[] options ;

    public Toppings( String d )
    {
        super(d) ;
    }

    // 4 toppings free, extra +.75
    public void setOptions( String[] options )
    {
        this.options = options ;
        if ( options.length > 4 )
            this.price += (
                options.length-4 ) * 0.75 ;
    }

    public String getDescription()
    {
        String desc = "" ;
        for ( int i = 0; i<options.length; i++ )
        {
            if (i>0) desc += " + " + options[i] ;
            else desc = options[i] ;
        }
        return desc ;
    }
}

```

```

public class Premium extends LeafDecorator
{
    private String[] options ;

    public Premium( String d )
    {
        super(d) ;
    }

    // premium topping +1.50
    public void setOptions( String[] options )
    {
        this.options = options ;
        if ( options.length > 0 )
            this.price += options.length * 1.50 ;
    }

    public String getDescription()
    {
        String desc = "" ;
        for ( int i = 0; i<options.length; i++ )
        {
            if (i>0) desc += " + " + options[i] ;
            else desc = options[i] ;
        }
        return desc ;
    }
}

```

```

public class Sauce extends LeafDecorator
{
    private String[] options ;

    public Sauce( String d )
    {
        super(d) ;
    }

    // 1 sauce free, extra +.75
    public void setOptions( String[] options )
    {
        this.options = options ;
        if ( options.length > 1 )
            this.price += (options.length-1) * 0.75 ;
    }

    public String getDescription()
    {
        String desc = "" ;
        for ( int i = 0; i<options.length; i++ )
        {
            if (i>0) desc += " + " + options[i] ;
            else desc = options[i] ;
        }
        return desc ;
    }
}

```