Inner Classes for Component-Oriented Programming: an Example

Mira Mezini

```
// ------------------------------------------------
// ------------------------------------------------

// APPLICATION CODE
// ----------------

 class HWProduct {

    protected String name;
    protected float price, salePrice;
    protected Tax[] taxes;
    // protected HashTable discount;
    // protected Producer producer;
    // protected Date entered;

    Tax[] taxes() {return taxes;}
    float salePrice() {return salePrice;}
    float saleDiscount() {return 0;}
    float regPrice() {return price;}
    // float regDiscount(int qty) {return discount.lookup(qty);}
    float regDiscount(int qty) {return qty * (float) 0.02;}

    // Producer producer() {retrun producer;}

    public HWProduct (String prodName, float pr, float salePr, Tax[] tax) {
       name = prodName;
       price = pr;
       salePrice = salePr;
       taxes = tax;
```

```
    }

}


class Tax {
   protected float salePercentage;
   // protected HashTable productCategories;

   float taxCharge(HWProduct prod, float unitPr, int qty) {
      // HashKey key = new HashKey(prod, qty);
      // float categPercentage = productCategories.lookup(key);
      // return unitPr * categPercentage * salePercentage;
         return unitPr * salePercentage;
    }

    public Tax(float perc) { salePercentage = perc; }
}

 class Quote {

   protected HWProduct prod;
   protected int qty;
   protected Customer cust;
   // protected Employee enteredBy;
   // protected Date orderEntryTime;
   // protected Date deliveryTime;

   int quantity() {return qty;}
   HWProduct prod() {return prod;}
   Customer customer() {return cust;}

   public Quote(HWProduct product, Customer customer, int quantity) {
     cust = customer;
     prod = product;
     qty = quantity;
    }
 }


 class Customer {

   protected String name;
   // protected String address;
```

```
    // protected HashTable negotiatedProdPrices;
    protected float negotiatedProdPrices;
    protected float negotiatedDiscountPercentage;

    // float negProdPrice(HWProduct prod) {
    //     negotiatedProdPrices.lookup((Object) prod);
    //  }

    float negProdPrice(HWProduct prod) {
        float pr = prod.regPrice();
        return (pr - pr * negotiatedProdPrices);
     }

    float negProdDiscount(HWProduct prod, int qty) {
       return qty * negotiatedDiscountPercentage;
      }

    String getName() {return name;}

    public Customer(String custName, float negProdPrices, float negDiscPerc) {
      name = custName;
      negotiatedProdPrices = negProdPrices;
      negotiatedDiscountPercentage = negDiscPerc;
     }
   }


// ------------------------------------------------------------------------
// ------------------------------------------------------------------------

// PRICING COMPONENT
// -----------------

 abstract class LineItemParty {
      abstract int quantity();
      abstract ItemParty item();
      abstract PricerParty pricer();
      abstract CustomerParty customer();

      public float price() {
         float discount;
         float basicPrice, unitPrice;
         PricerParty pricer = pricer();
         CustomerParty customer = customer();
```

```
        int qty = quantity();
        ItemParty item = item();
        basicPrice = pricer.basicPrice(item);
        discount = pricer.discount(item, qty, customer);
        unitPrice = basicPrice - (discount * basicPrice);
        return unitPrice + item.addCharges(unitPrice, qty);
    }
  }


  abstract class ItemParty {

     float addCharges(float unitPrice, int qty) {
        float total = 0;
        for (int i = 0; i < charges().length; i++)
             total += charges()[i].cost(qty, unitPrice, this);
        return total;
      }

    abstract ChargerParty[] charges();
  }


  interface PricerParty {
     float basicPrice(ItemParty item);
     float discount(ItemParty item, int qty, CustomerParty cust);
   }

  interface ChargerParty {
     float cost (int qty, float unitPr, ItemParty item);
   }

  interface CustomerParty {}



// ---------------------------------------------------------------------
// ---------------------------------------------------------------------


interface AdapterObject {
   Object enclosed();
 }
```

```
// REGULAR PRICING
// ---------------

 class MyQuote extends Quote {

      public class RegLineItemParty extends LineItemParty implements AdapterObject {

         protected PricerParty pricer() {
                 return prod.regPricerAdapter;}
        protected ItemParty item() {return prod.regItemAdapter;}
        protected CustomerParty customer()
                 {return cust.regCustAdapter; }
        protected int quantity() {return ((Quote) enclosed()).quantity();}
                              public Object enclosed() {return MyQuote.this;}
       };

     public RegLineItemParty regPricingAdapter;

     public MyCustomer cust;
     public MyHWProduct prod;

     public MyQuote(MyHWProduct product, MyCustomer customer, int quantity) {
           super(product, customer, quantity);
           prod = product;
           cust = customer;
           regPricingAdapter = new RegLineItemParty();
       }
    }

  class MyHWProduct extends HWProduct {

     public class RegPricerParty implements PricerParty, AdapterObject {
         public float basicPrice(ItemParty item) {return regPrice();}
         public float discount(ItemParty item, int qty, CustomerParty cust) {
                         return regDiscount(qty); }
         public Object enclosed() {return MyHWProduct.this;}
      }

     public class RegItemParty extends ItemParty implements AdapterObject {

        public ChargerParty [] charges() {
           ChargerParty [] taxTable = new ChargerParty[taxes.length];
           for (int i = 0; i <taxes.length; i++) {
```

```
            taxTable[i] = taxes[i].regChargerAdapter;}
            return taxTable;
        }

       public Object enclosed() {return MyHWProduct.this;}
      }

    public PricerParty regPricerAdapter;
    public ItemParty regItemAdapter;

    MyTax[] taxes;

    public MyHWProduct(String prodName, float pr, float salePr, MyTax[] tax) {
        super(prodName, pr, salePr, tax);
        taxes = tax;
        regPricerAdapter = new RegPricerParty();
        regItemAdapter = new RegItemParty();
     }
  }

class MyTax extends Tax {

    public class RegChargerParty implements ChargerParty, AdapterObject {
         public float cost(int qty, float unitPr, ItemParty item) {
           return taxCharge((HWProduct) ((AdapterObject)item).enclosed(), unitPr, qty);
          }

         public Object enclosed() {return MyTax.this; }
     }

   public ChargerParty regChargerAdapter;

   public MyTax(float perc) {
      super(perc);
      regChargerAdapter = new RegChargerParty();}
 }


  class MyCustomer extends Customer {

     public class RegCustomerParty implements CustomerParty, AdapterObject {
          public Object enclosed() {return MyCustomer.this;}
         }
```

```
      public CustomerParty regCustAdapter;

      public MyCustomer(String custName, float negProdPrices, float negDiscPerc) {
           super(custName, negProdPrices, negDiscPerc);
           regCustAdapter = new RegCustomerParty();}

        }

// ---------------------------------------------------------
// ---------------------------------------------------------

// NEGOTIATED PRICING
// -------------------

class NegQuote extends MyQuote {

   public class NegLineItemParty extends MyQuote.RegLineItemParty {
       protected PricerParty pricer() {return cust.negPricerAdapter;}
       public Object enclosed() {return NegQuote.this;}
    }

   public NegLineItemParty negPricingAdapter;

   public NegCustomer cust;

   public NegQuote(MyHWProduct product, NegCustomer customer, int quantity) {
       super(product, customer, quantity);
       cust = customer;
       negPricingAdapter = new NegLineItemParty();
     }
  }

 class NegCustomer extends MyCustomer {

   public class NegCustomerParty extends MyCustomer.RegCustomerParty {
      public Object enclosed() {return NegCustomer.this;}
    }

   public class NegPricerParty  implements PricerParty, AdapterObject {

     public float basicPrice(ItemParty item) {
        return negProdPrice((HWProduct) ((AdapterObject)item).enclosed());
       }
```

```
     public float discount(ItemParty item, int qty, CustomerParty cust) {
         return negProdDiscount((HWProduct) ((AdapterObject)item).enclosed(), qty);
      }
     public Object enclosed() {return NegCustomer.this;}
    }

   NegPricerParty negPricerAdapter;

   public NegCustomer(String custName, float negProdPrices, float negDiscPerc) {
           negPricerAdapter = new NegPricerParty();}
 }


// ------------------------------------------------------------
// ------------------------------------------------------------


class Client {

  static public void main(String[] args) {
    MyTax t1 = new MyTax((float)0.05);
    MyTax t2 = new MyTax((float) 0.07);
    MyTax[] t = {t1, t2};
    MyHWProduct product = new MyHWProduct("Schreibmaschine", (float) 250, (float) 200, t);
    MyCustomer cust = new MyCustomer("Schmidt", (float) 0.5, (float) 0.02);
    MyQuote q = new MyQuote(product, cust, 10);
    System.out.println(q.regPricingAdapter.price());
    NegCustomer cust1 = new NegCustomer("Schmidt", (float) 0.05, (float) 0.02);
    NegQuote q1 = new NegQuote(product, cust1, 10);
    System.out.println(q1.regPricingAdapter.price());
    System.out.println(q1.negPricingAdapter.price());
   }
}

// Result:
// ------
// 224.0
// 224.0
// 212.8
```