```
/**
 * 顧客
 * @see Order
 */
public class Client {
    /** 顧客番号 */
    private int m_ClientNo;
    /** 顧客名 */
    private String m_ClientName;
    /** 顧客住所 */
    private String m_ClientAddress;
    /** 顧客電話番号 */
    private String m_ClientTelephoneNo;
    /** 入会日時 */
    private Date m_JoinedDate;
    /** 関連オブジェクト：購買 */
    private Vector m_Order = new Vector();
    //
    /** クラス変数：顧客番号発番 */
    private static  int     c_client_no;
    /** クラス変数：顧客リスト */
    private static  Vector  c_member_list = new Vector();

    /** アクセス関数 */
    public  String  getClientAddress()      { return m_ClientAddress; }
    public  String  getClientName()         { return m_ClientName;}
    public  String  getClientTelephoneNo()  { return m_ClientTelephoneNo;}
    //
    /** コンストラクタ：新規顧客発番 */
    public  Client(String a_name, String a_address, String a_telephone_no, Date a_joined_date){
        //
        m_ClientNo              = c_client_no;
        m_ClientName            = new String(a_name);
        m_ClientAddress         = new String(a_address);
        m_ClientTelephoneNo     = new String(a_telephone_no);
        m_JoinedDate            = new Date(a_joined_date.getTime());
        //
        c_client_no++;
        c_member_list.addElement(this);
    }
    //
    /** コンストラクタ：新規→既存 */
    protected   Client(Client a_client){
    //
        m_ClientNo              = a_client.m_ClientNo;
        m_ClientName            = new String(a_client.m_ClientName);
        m_ClientAddress         = new String(a_client.m_ClientAddress);
        m_ClientTelephoneNo     = new String(a_client.m_ClientTelephoneNo);
        m_JoinedDate            = new Date(a_client.m_JoinedDate.getTime());
        m_Order                 = a_client.m_Order;
        //
        c_member_list.addElement(this);
    }
    /** 購買金額問い合わせ */
    public double   getSum(){
        double t_sum = 0;
        // 全ての「購買」について、「購買額」を加算する
        for ( Enumeration enum = m_Order.elements();
                enum.hasMoreElements(); ){
            t_sum += ((Order)enum.nextElement()).getOrderSum();
        }
        return t_sum;

    }
    /** 購買の登録 */
    public  void    order(Order a_order){
        m_Order.addElement(a_order);
    }
    /** 購買の探索 */
    public  Order   findOrder(String    a_order_no){
        for( Enumeration enum = m_Order.elements(); enum.hasMoreElements();){
            Order   t_order = (Order)enum.nextElement();
            if( a_order_no.equals(t_order) ) return t_order;
        }
        return null;
    }
    /** 顧客探索 */
    public  static  Client  find(String a_name, String a_address, String a_tel_no){
        //
        for( Enumeration enum = c_member_list.elements();
                enum.hasMoreElements();){
            Client t_client = (Client)enum.nextElement();
            // 顧客を発見した
            if (    t_client.getClientName().equals(a_name) &&
                    t_client.getClientAddress().equals(a_address) &&
```

```
                t_client.getClientTelephoneNo().equals(a_tel_no) )
            return t_client;
        }
        // 該当する顧客が存在しない。
        return null;
    }
}
```

```
/**
 * 新規顧客
 * @see Client
 * @see OldClient
 */
public class NewClient extends Client {
    /** 新規顧客期間 */
    private Date m_expire;
    /** 関連オブジェクト：既存顧客（私を紹介した人） */
    private OldClient m_OldClient;
    //
    /** 新規顧客リスト */
    private static  Vector  c_member_list = new Vector();

    /** アクセス関数 */
    public  Date    getExpireDate()     { return m_expire;}
    public  OldClient   getIntroducer() { return m_OldClient;}
    /** コンストラクタ */
    public NewClient(String a_name, String a_address, String a_telephone_no, Date a_joined_date, OldClient a_old_client){
        super(a_name, a_address, a_telephone_no, a_joined_date);
        m_expire        = expire(a_joined_date);
        m_OldClient     = a_old_client;
        //
        c_member_list.addElement(this);
    }
    /** 新規顧客期間問い合わせ */
    public boolean  is_expired(){
        // 常に期間終了を返す。
        return true;
    }
    /** 削除 */
    public  static  void    delete(NewClient    a_client){
        c_member_list.removeElement(a_client);
    }

    /** メンバーリスト */
    public  static Enumeration  members(){
        return c_member_list.elements();
    }
    /** 期限算出 */
    private static  Date    expire(Date a_date){
      Calendar  t_expire    = Calendar.getInstance();
      t_expire.setTime(a_date);
      t_expire.add(Calendar.DATE,30);
      return t_expire.getTime();
    }
}
```

```
/**
 * 既存顧客
 * @see Client
 * @see NewClient
 */
public class OldClient extends Client {
    /** 集約オブジェクト：クーポン券 */
    private Coupon m_Coupon;
    /** 関連オブジェクト：新規顧客（私が紹介した新規顧客たち） */
    private Vector m_NewClient = new Vector();
    /** クラス変数：メンバーリスト */
    private static  Vector c_member_list = new Vector();

    /** コンストラクタ：新規→既存 */
    public OldClient(Client a_client){
        super(a_client);
        c_member_list.addElement(this);
    }
    /** クーポン券発行 */
    public void pressCoupon(Coupon a_coupon) {
        System.out.println("to: " + getClientName() );
        System.out.println("Subject: Coupon");
        System.out.println("sum: " + a_coupon.getSum().toString());
        System.out.println("expire: " + a_coupon.getExpire().toString());
        System.out.println("coupon#: " + a_coupon.getCouponNo());
    }
    /** 新規顧客紹介 */
    public void     introduce(NewClient a_client){
        m_NewClient.addElement(a_client);
    }
    /** メンバー? */
    public  static boolean isMember(String a_address){
```

```
        // 住所で判定する
        for( Enumeration enum = c_member_list.elements();
                enum.hasMoreElements();){
            OldClient t_client = (OldClient)enum.nextElement();
            // 同じ住所の既存顧客が存在する。
            if ( t_client.getClientAddress().equals(a_address) ) return true;
        }
        // メンバーではない
        return false;
    }
    /** find */
    public  static OldClient findOldClient(String a_name, String a_address, String a_tel_no){
        //
        for( Enumeration enum = c_member_list.elements();
                enum.hasMoreElements();){
            OldClient t_client = (OldClient)enum.nextElement();
            // 既存顧客が存在する。
            if (    t_client.getClientName().equals(a_name) &&
                    t_client.getClientAddress().equals(a_address) &&
                    t_client.getClientTelephoneNo().equals(a_tel_no) )
                return t_client;
        }
        // 該当する既存顧客が存在しない。
        System.out.println("not found:" + "name:" + a_name +",addr:"+a_address+",tel:"+a_tel_no);
        return null;
    }
    /** メンバーリスト */
    public  static  Enumeration members(){
        return c_member_list.elements();
    }

}



/**
 * クーポン券
 * @see OldClient
 */
public class Coupon {
    /** 金額 */
    private Double m_Sum;
    /** 期限 */
    private Date m_Expire;
    /** クーポン券番号 */
    private String m_CouponNo;
    /** 親オブジェクト：既存顧客 */
    private OldClient m_OldClient;
    //
    /** クラス変数：クーポン番号採番 */
    private static  double  c_coupon_no = 0;

    /** アクセス関数 */
    public Double   getSum()        { return m_Sum; }
    public Date     getExpire()     { return m_Expire;}
    public String   getCouponNo()   { return m_CouponNo;}
    /** コンストラクタ */
    public Coupon(double a_sum){
        m_Sum       = new Double(point(a_sum));
        m_Expire    = expire(new Date());
        m_CouponNo  = (new Double(c_coupon_no++)).toString();
    }
    /** ポイント算出 */
    private double  point(double a_sum){
        return a_sum * 0.2;
    }
    /** 期限算出 */
    private Date    expire(Date a_date){
      Calendar  t_expire    = Calendar.getInstance();
      t_expire.setTime(a_date);
      t_expire.add(Calendar.DATE,30);
      return t_expire.getTime();
    }
}



/**
 * 購買
 * @see OrderProduct
 * @see Client
 */
public class Order {
    /** 購買番号 */
    private String m_OrderNo;
    /** 購買日時 */
    private Date m_OrderDate;
```

```
    /** 購買額 */
    private Double m_OrderSum;
    /** 集約オブジェクト：購買商品 */
    private Vector m_OrderProduct = new Vector();
    /** 関連オブジェクト：顧客 */
    private Client m_Client;

    /** アクセス関数 */
    public  double  getOrderSum()   { return m_OrderSum.doubleValue();}
    public  String  getOrderNo()    { return m_OrderNo; }
    /** コンストラクタ（顧客、購買番号、購買日時、購買額） */
    public  Order(Client a_client, String a_order_no, Date a_order_date){
        m_Client    = a_client;
        m_OrderNo   = a_order_no;
        m_OrderDate = a_order_date;
        m_OrderSum  = new Double(0);
        //
    }
    /** 購買商品の追加 */
    public  void    append(OrderProduct a_order_product, double a_order_sum){
        m_OrderProduct.addElement(a_order_product);
        m_OrderSum = new Double(m_OrderSum.doubleValue()+a_order_sum);
    }

}



/**
 * 購買商品
 * @see Order
 * @see CatalogProduct
 */
public class OrderProduct {
    /** 数量 */
    private Double m_Quantity;
    /** 親オブジェクト：購買 */
    public Order m_Order;
    /** 関連オブジェクト：カタログ商品 */
    public CatalogProduct m_CatalogProduct;
    /** コンストラクタ */
    public  OrderProduct(Order a_order, CatalogProduct a_catalog_product, double a_quantity){
        m_Order             = a_order;
        m_CatalogProduct    = a_catalog_product;
        m_Quantity          = new Double(a_quantity);
        //
    }

}



/**
 *カタログ
 * @see CatalogProduct
 */
public class Catalog {
    /** カタログ番号 */
    private String m_CatalogNo;
    /** 有効期間（始） */
    private Date m_StartDate;
    /** 有効期間（終） */
    private Date m_EndDate;
    /** 集約オブジェクト：カタログ商品 CatalogProduct */
    private Vector m_CatalogProduct = new Vector();
    //
    /** クラス変数：カタログ集 */
    private static Vector c_member_list = new Vector();

    /** constructor */
    public  Catalog(String a_catalog_no, Date a_start_date, Date a_end_date){
        m_CatalogNo = a_catalog_no;
        m_StartDate = a_start_date;
        m_EndDate   = a_end_date;
        //
        c_member_list.addElement(this);
    }
    /** カタログを見つける。 */
     public static  Catalog findCatalog(String a_catalog_no){
        for( Enumeration enum = c_member_list.elements();
                enum.hasMoreElements();){
            Catalog t_catalog   = (Catalog)(enum.nextElement());
            if ( t_catalog.m_CatalogNo.equals(a_catalog_no)){
                return  t_catalog;
            }
        }
        return null;
    }
    /** カタログ商品を見つける。 */
     public CatalogProduct findCatalogProduct(String a_catalog_product_no){
        for( Enumeration enum = m_CatalogProduct.elements();
                enum.hasMoreElements();){
            CatalogProduct  t_catalog_product   = (CatalogProduct)(enum.nextElement());
```

```
            if ( t_catalog_product.getCatalogProductNo().equals(a_catalog_product_no)){
                return  t_catalog_product;
            }
        }
        return null;
    }
    /** カタログ商品登録 */
    public  void    register(CatalogProduct a_catalog_product){
        m_CatalogProduct.addElement(a_catalog_product);
    }

}



/**
 * カタログ商品
 * @see Catalog
 * @see OrderProduct
 */
public class CatalogProduct {
    /** カタログ商品番号 */
    private String m_CatalogProductNo;
    /** 価格 */
    private Double  m_Price;
    /** 親オブジェクト：カタログ */
    private Catalog m_Catalog;

    /** アクセス関数 */
    public  String getCatalogProductNo()    {return m_CatalogProductNo;}
    public  double getPrice()               {return m_Price.doubleValue();}
    /** コンストラクタ */
    public  CatalogProduct(Catalog a_catalog, String a_catalog_product_no, double a_price){
        m_Catalog           = a_catalog;
        m_CatalogProductNo  = a_catalog_product_no;
        m_Price             = new Double(a_price);
        //
        a_catalog.register(this);
    }
}



/**
 * クーポン券発行マネージャ
 * @see Coupon
 * @see OldClient
 * @see NewClient
 */
public class CouponManager {
    /** 起動 */
    public void pressCoupon() {
        // 全ての新規顧客について
        Enumeration t_new_clients   = NewClient.members();
        Vector  t_delete_candidate  = new Vector();
        for (;t_new_clients.hasMoreElements();){
            NewClient   t_new_client = (NewClient)(t_new_clients.nextElement());
            OldClient   t_introducer    = t_new_client.getIntroducer();
            // 新規顧客期間を問い合わせる
            if ( t_new_client.is_expired() ){
                // 新規顧客期間が終了している場合、購買金額を問い合わせる。
                double t_sum = t_new_client.getSum();
                    // 新規顧客を既存顧客とするため、生成。
                new OldClient((Client)t_new_client);
                // 購買金額があった場合
                if ( t_sum > 0 ){
                    // 「クーポン券」を「生成」
                    Coupon t_coupon = new Coupon(t_sum);
                    // 「紹介者」に「クーポン券」を「発行」する。
                    t_introducer.pressCoupon(t_coupon);
                }
                // 購買金額が無かった場合
                else{
                    // クーポン券発行不可レター出状
                    letter(t_introducer.getClientName(),t_new_client.getClientName());
                }
                // 新規顧客の削除
                t_delete_candidate.addElement(t_new_client);
            }
        }
        for(Enumeration enum = t_delete_candidate.elements();
                enum.hasMoreElements();){
            NewClient.delete((NewClient)(enum.nextElement()));
        }
    }
    /** クーポン券発行不可レターを出状 */
    private void    letter(String a_introducer_name, String a_name){
        System.out.println("to: " + a_introducer_name );
        System.out.println("Subject: Regret, You Get No Coupons!");
        System.out.println("name: " + a_name);
        System.out.println("has bought nothing.");
    }
}
```

```
/**
 * 注文マネージャ
 * @see OrderForm
 * @see Order
 * @see OrderProduct
 * @see Client
 */
public class OrderManager {
    /** 処理する */
    public void order(OrderProductForm  a_order_product_form) {
        // 在庫情報を調べる
        // 購買情報を登録
            // 購買情報取得
        OrderForm   t_order_form    = a_order_product_form.getOrderForm();
        String      t_order_no      = t_order_form.getOrderNo();
        Client      t_client        = t_order_form.getClient();
            // 既存の購買情報か？
        Order   t_order = t_client.findOrder(t_order_no);
        if( t_order == null ){
            t_order = new Order(t_order_form.getClient(), t_order_form.getOrderNo(), t_order_form.getOrderDate());
            t_client.order(t_order);
        }
            // 価格算出
        Catalog t_catalog   = Catalog.findCatalog(a_order_product_form.getCatalogNo());
        CatalogProduct  t_catalog_product   = t_catalog.findCatalogProduct(a_order_product_form.getCatalogProductNo());
        double t_sum = a_order_product_form.getQuantity() * t_catalog_product.getPrice();
        // 購買商品情報を登録
        OrderProduct    t_order_product = new OrderProduct(t_order,t_catalog_product , a_order_product_form.getQuantity());
        t_order.append(t_order_product, t_sum);
    }

}




/**
 * 紹介マネージャ
 * @see NewClient
 * @see OldClient
 */
public class IntroductionManager {
    /** 処理する（紹介顧客、新規顧客の名前、住所、電話番号） */
    public void introduction(OldClient a_old_client, String a_name, String a_address, String a_telephone_no) {
        // 新規顧客か？
        boolean t_is_new_client = OldClient.isMember(a_address);
        // 新規顧客条件を満たしていれば、新規顧客として登録
        if ( !t_is_new_client ) {
            Date t_joined_date  = new Date();
            NewClient t_client = new NewClient(a_name, a_address, a_telephone_no, t_joined_date, a_old_client);
            a_old_client.introduce(t_client);
        }
        // 新規顧客として認められなかった場合、クーポン券発行不可レターを出状
        else{
            letter(a_old_client.getClientName(), a_name, a_address, a_telephone_no);
        }
    }
    /** クーポン券発行不可レターを出状 */
    public  void    letter(String a_introducer, String a_name, String a_address, String a_telephone_no){
        System.out.println("to: " + a_introducer );
        System.out.println("Subject: Regret, You Get No Coupons!");
        System.out.println("name: " + a_name);
        System.out.println("address: " + a_address);
        System.out.println("telephone: " + a_telephone_no);
        System.out.println("is already client of us.");
    }

}




/**
 * タイマ
 * @see CouponManger
 */
public class Timer {
    /** 発火 */
    public static   void    fire(){
        // 「クーポン券発行マネージャ」を「起動」する。
        CouponManager t_coupon_manager  = new CouponManager();
        t_coupon_manager.pressCoupon();
    }
}
```