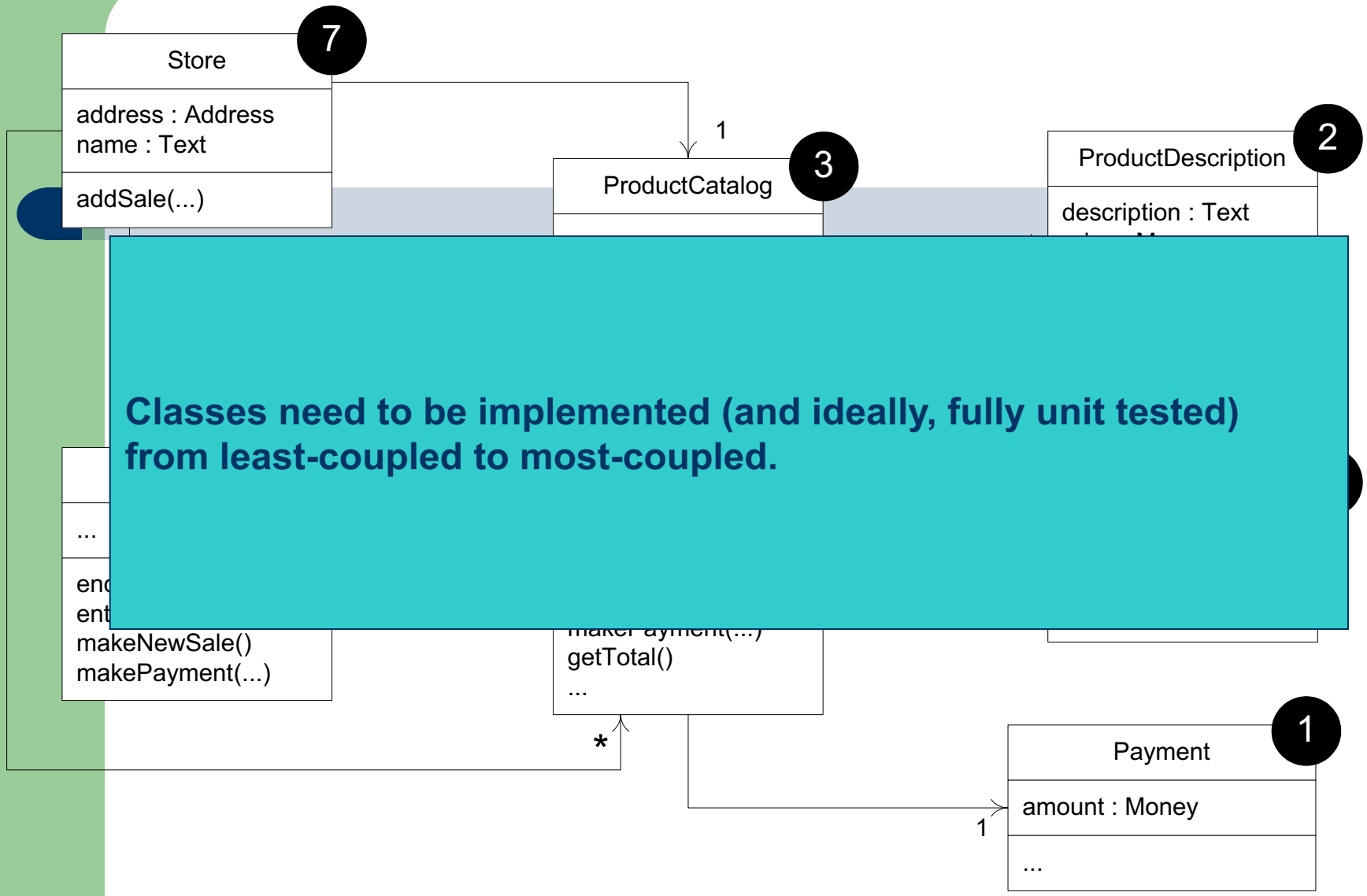
60-322
Object-Oriented Analysis and Design

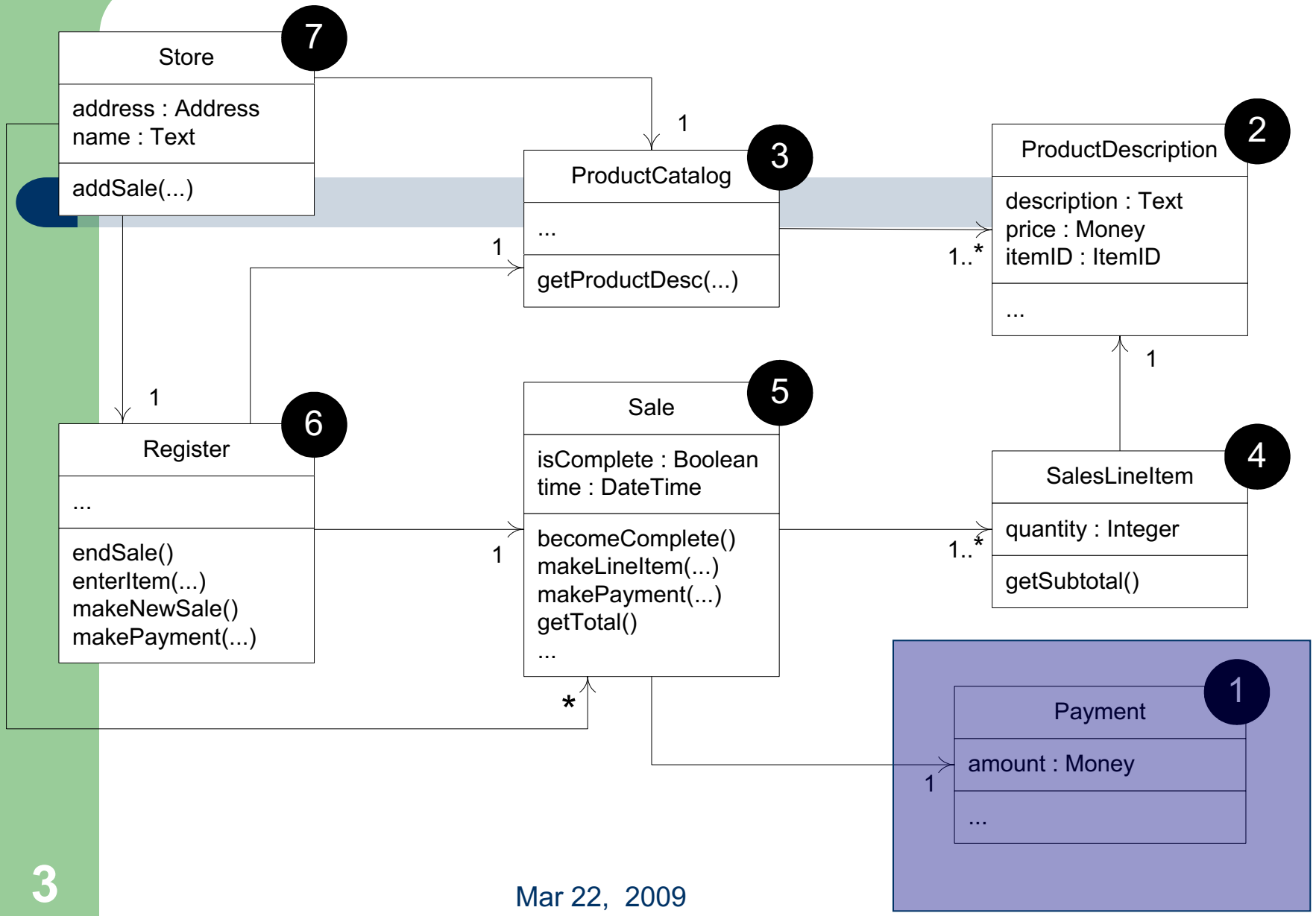
Week 12, 2009

Order of Implementation



Classes need to be implemented (and ideally, fully unit tested) from least-coupled to most-coupled.

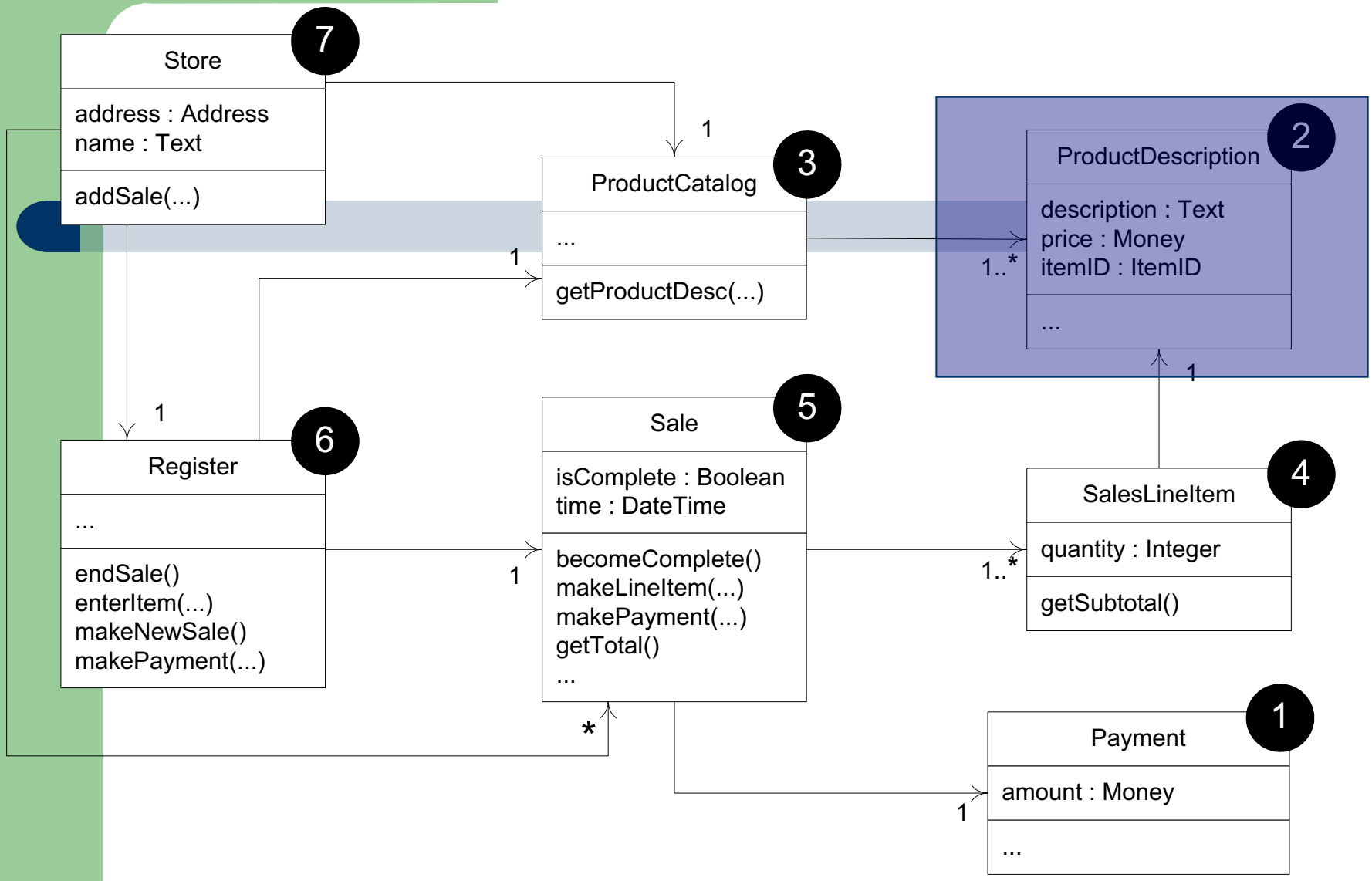
Order of Implementation



Class Payment

```
// all classes are probably in a package named
// something like:
package com.foo.nextgen.domain;


public class Payment
{
   private Money amount;

   public Payment( Money cashTendered ){ amount = cashTendered; }
   public Money getAmount() { return amount; }
}
```

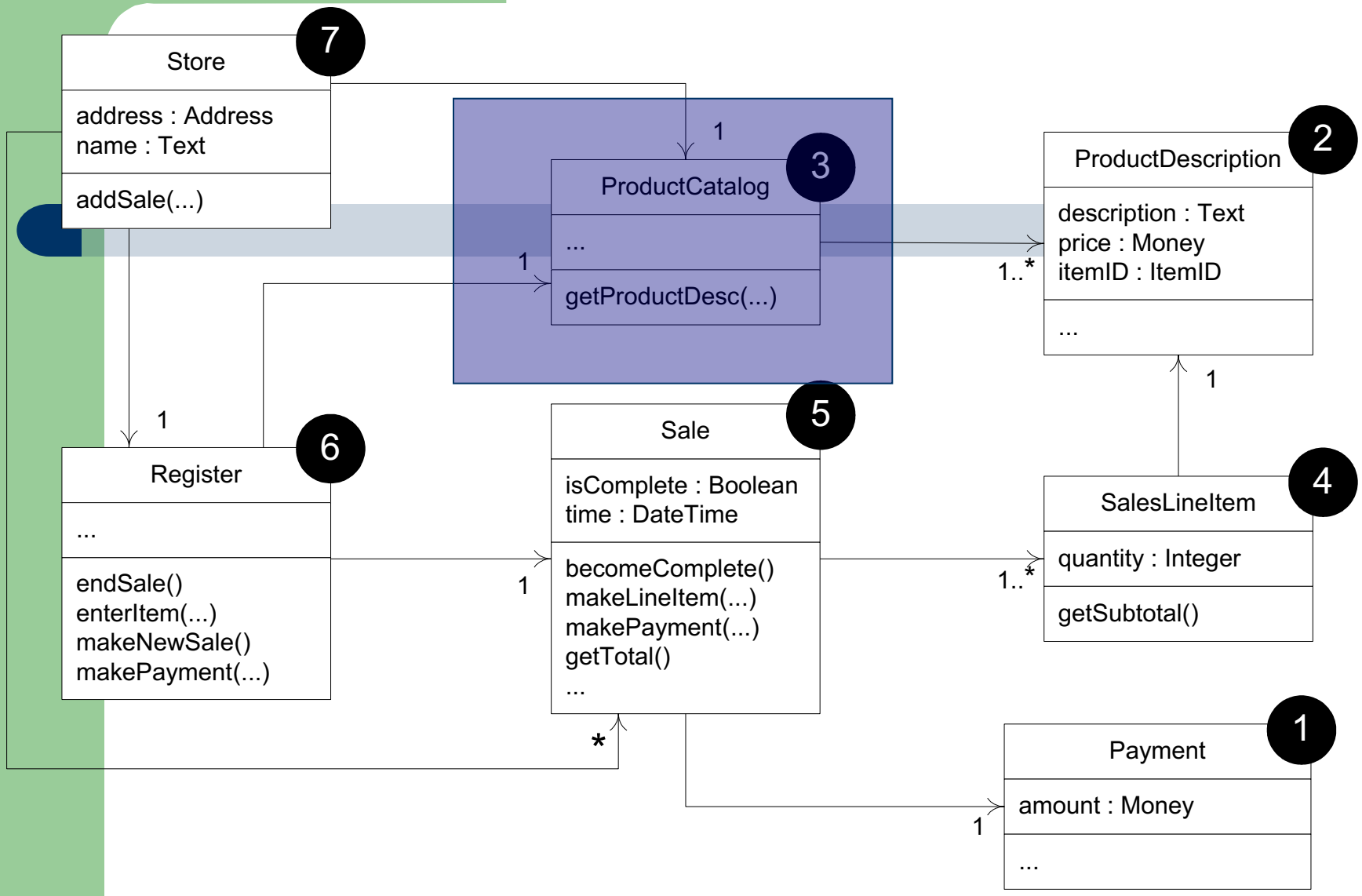
Order of Implementation



Class ProductDescription

```
public class ProductDescription
{
   private ItemID id;
   private Money price;
   private String description;

   public ProductDescription
      ( ItemID id, Money price, String description )
   {
      this.id = id;
      this.price = price;
      this.description = description;
   }

   public ItemID getItemID() { return id;    }

   public Money getPrice() { return price; }
   public String getDescription() { return description; }
}
```

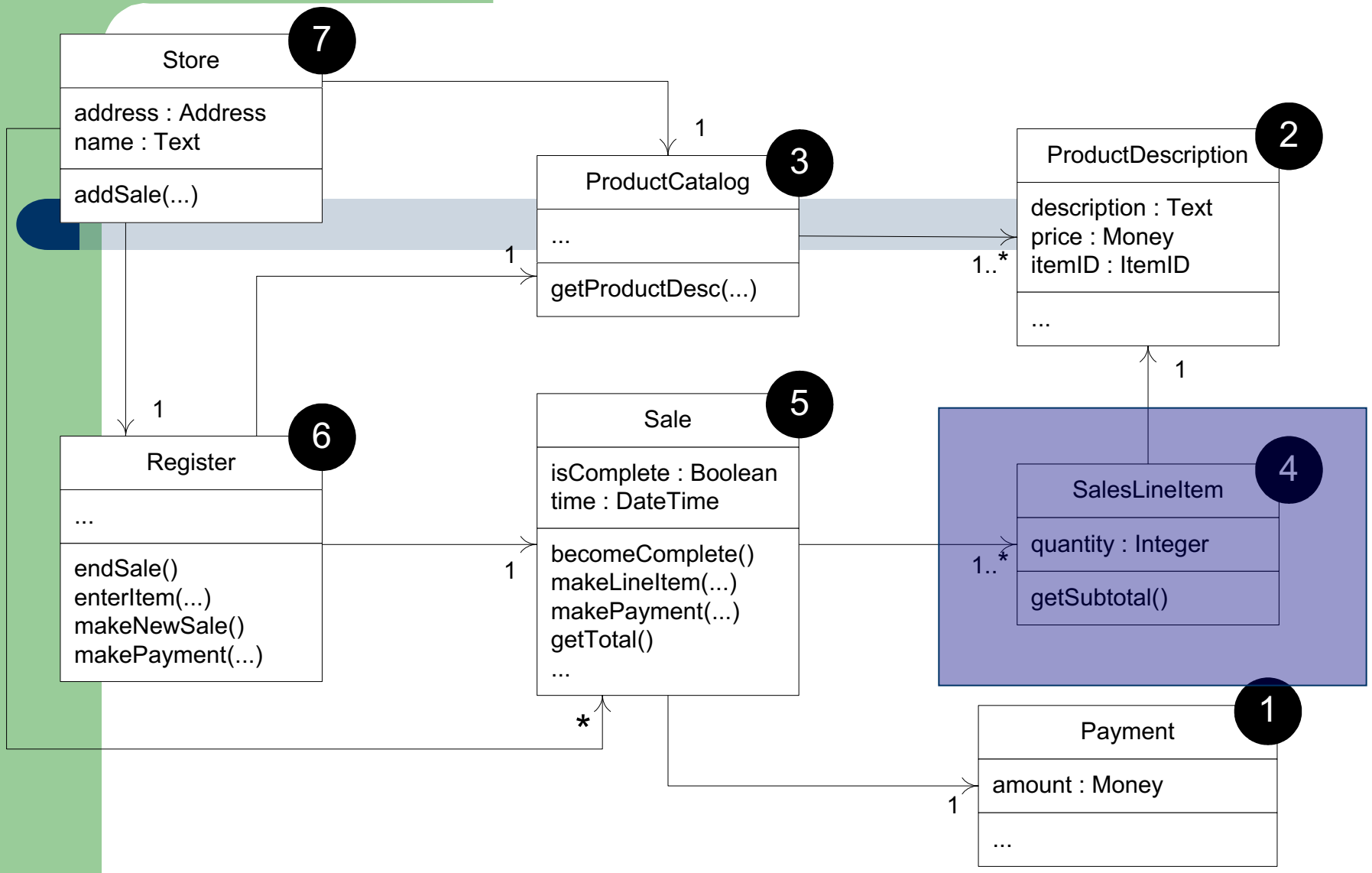
Order of Implementation



Class ProductCatalog

```
public class ProductCatalog
{
   private Map<ItemID, ProductDescription>
         descriptions = new HashMap()<ItemID, ProductDescription>;

   public ProductCatalog()
   {
      // sample data
      ItemID id1 = new ItemID( 100 );
      ItemID id2 = new ItemID( 200 );
      Money price = new Money( 3 );

      ProductDescription desc;
      desc = new ProductDescription( id1, price, "product 1" );
      descriptions.put( id1, desc );
      desc = new ProductDescription( id2, price, "product 2" );
      descriptions.put( id2, desc );
   }

   public ProductDescription getProductDescription( ItemID id )
   {
      return descriptions.get( id );
   }
}
```

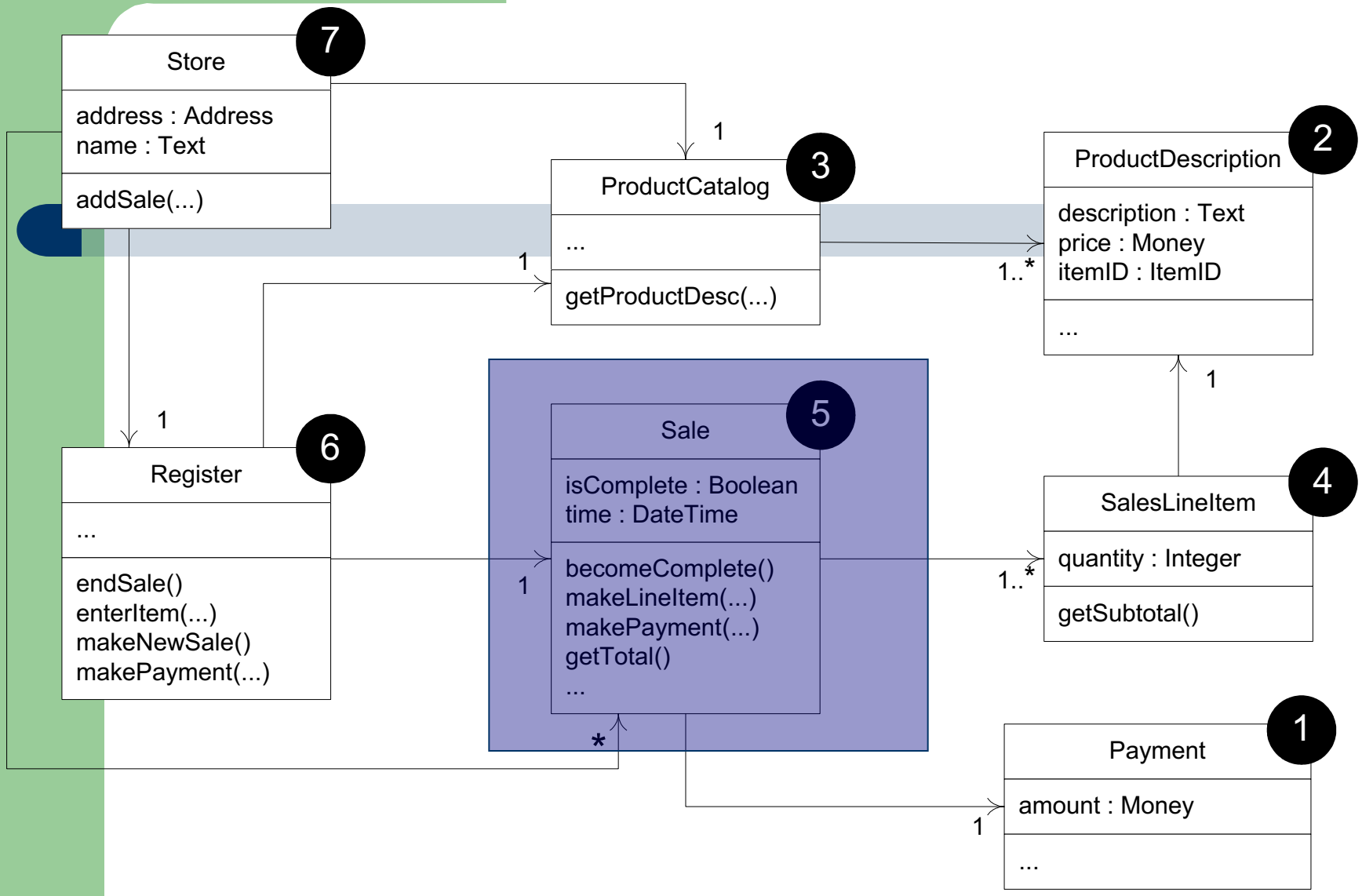
Order of Implementation



Class SalesLineItem

```
public class SalesLineItem
{
    private int     quantity;
    private     ProductDescription     description;

    public SalesLineItem (ProductDescription desc, int quantity )
    {
        this.description = desc;
        this.quantity = quantity;
    }
    public Money getSubtotal()
    {
        return description.getPrice().times( quantity );
    }
}
```

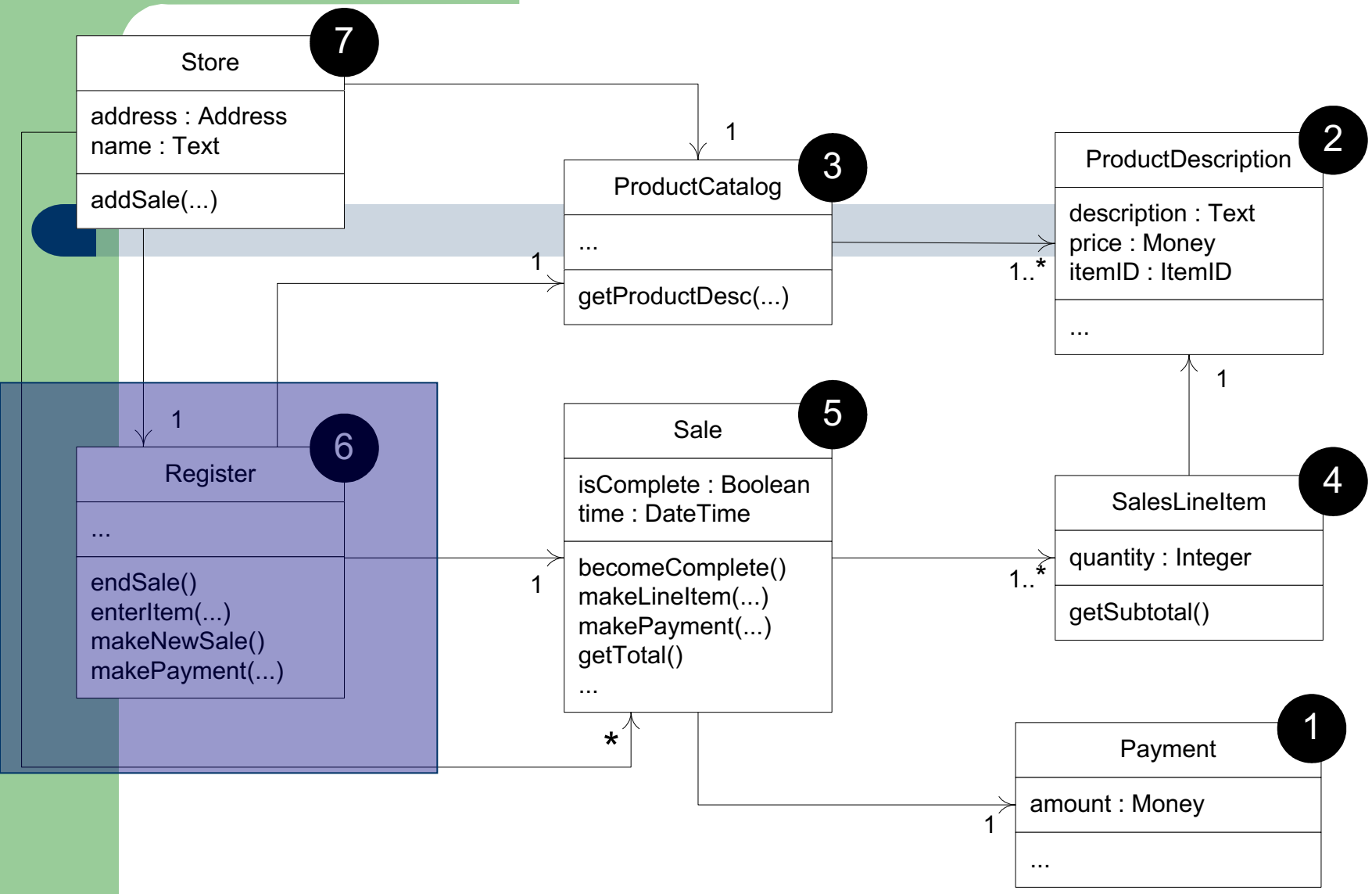
Order of Implementation



Class Sale

```
public class Sale
{
    private List<SalesLineItem> lineItems =
                                new ArrayList()<SalesLineItem>;
    private Date date = new Date();
    private boolean isComplete = false;
    private Payment payment;

    public Money getBalance()
    {
        return payment.getAmount().minus( getTotal() );
    }

    public void becomeComplete() { isComplete = true; }

    public boolean isComplete() { return isComplete; }

    public void makeLineItem
        ( ProductDescription desc, int quantity )
    {
        lineItems.add( new SalesLineItem( desc, quantity ) );
    }

    public Money getTotal()
    {
        Money total = new Money();
        Money subtotal = null;

        for ( SalesLineItem lineItem : lineItems )
        {
            subtotal = lineItem.getSubtotal();
            total.add( subtotal );
        }
    return total;
    }

    public void makePayment( Money cashTendered )
    {
        payment = new Payment( cashTendered );
    }
}
```

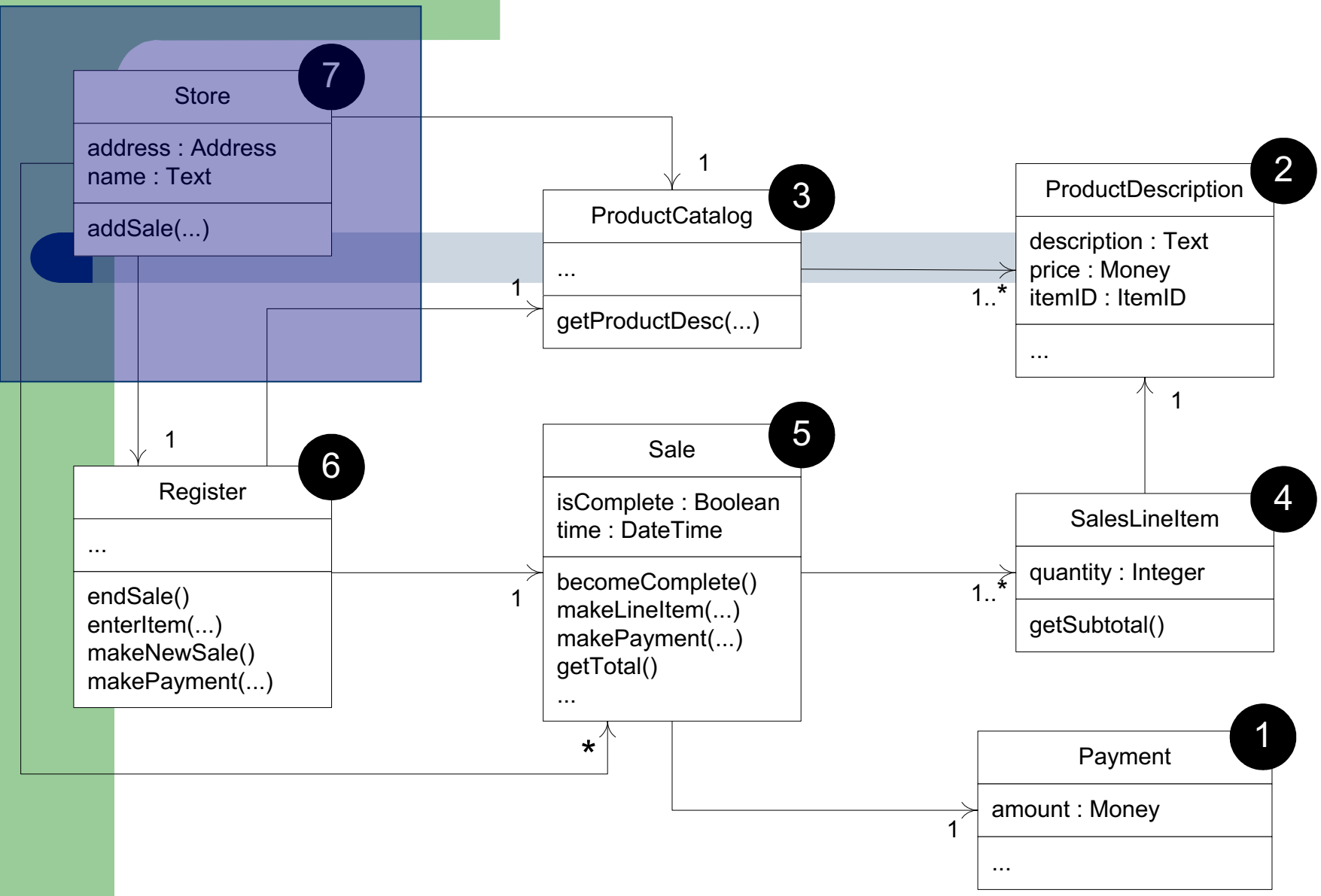
Order of Implementation



Class Register

```
public class Register
{
    private ProductCatalog catalog;
    private Sale currentSale;

    public Register( ProductCatalog catalog )
    {
        this.catalog = catalog;
    }

    public void endSale()
    {
        currentSale.becomeComplete();
    }

    public void enterItem( ItemID id, int quantity )
    {
        ProductDescription desc = catalog.getProductDescription( id );

        currentSale.makeLineItem( desc, quantity );
    }

    public void makeNewSale()
    {
        currentSale = new Sale();
    }

    public void makePayment( Money cashTendered )
    {
        currentSale.makePayment( cashTendered );
    }

}
```

Order of Implementation



Class Store

```
public class Store
{
    private ProductCatalog catalog = new ProductCatalog();
    private Register register = new Register( catalog );

    public Register getRegister() { return register; }
}
```