Exercise

Given the partial Java code for a Point-Of-Sale Application, produce a Design Class Diagram with navigability and multiplicity (ignore the operations).

Point-Of-Sale

Application Class

ItemId

```
public class
    ItemId {
  private int
          id;

      . . .
}
```

Class ProductDescription

```
public class ProductDescription {
      private ItemId id;
      private double price;
      private String description;

      . . .
}
```

Class ProductCatalog

```
import java.util.HashMap;
import java.util.Map;

public class ProductCatalog {
      private Map<ItemId, ProductDescription> descriptions;

      public ProductCatalog() {
            descriptions = new HashMap<ItemId,ProductDescription>();

            //sample data
            ItemId id1 = new
            ItemId(100); ItemId id2 =
            new ItemId(200);

            ProductDescription desc1 =
                        new ProductDescription(id1, 3.00, "product 1");
            ProductDescription desc2 =
                        new ProductDescription(id2, 5.00, "product 2");

            descriptions.put(id1,desc);
            descriptions.put(id2,desc);
      }

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

Class SaleLineItem

```
public class SaleLineItem {
      private ProductDescription product;
      private int quantity;
```

```
        . . .
}
```

Class Payment

```
public class Payment {
        private double amount;

        . . .
}
```

Class Sale

```
import
java.util.ArrayList
; import
java.util.Date;
import
java.util.List;

public class Sale {
        private List<SaleLineItem> lineItems;
        private boolean isComplete;
        private Date date;
        private Payment payment;

        public Sale() {
                lineItems = new ArrayList<SaleLineItem>();
                isComplete = false;
                date = new
                Date();
                payment =
                null;
        }

        . . .
}
```

Class CashRegister

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

        . . .
}
```

Class Store

```
public class Store {
        private ProductCatalog catalog;
        private CashRegister register;

        public Store() {
                catalog = new
                ProductCatalog(); register =
                new CashRegister(catalog);
        }

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