PRODUCT CLASS

```
/**
 * Represents a product available in the store.
 */
public class Product implements Discountable {
   private String name;
   private double price;
   private int inventoryLevel;
   private double discountRate;

   /**
    * Constructs a new Product with the specified attributes.
    *
    * @param name          the name of the product.
    * @param price         the price of the product.
    * @param inventoryLevel the initial inventory level of the product.
    * @param discountRate   the discount rate applicable to the product.
    */
   public Product(String name, double price, int inventoryLevel, double discountRate) {
      this.name = name;
      this.price = price;
      this.inventoryLevel = inventoryLevel;
      this.discountRate = discountRate;
   }

   // Getters and Setters...

   @Override
```

```
    public double calculateDiscount() {
        // TODO: Implement discount calculation logic
        return 0;
    }
}
```

CART CLASS

```
/**
 * Represents a shopping cart containing CartItems.
 */
public class Cart {
    ArrayList<CartItem> items;

    /**
     * Constructs a new Cart with an empty list of items.
     */
    public Cart() {
        items = new ArrayList<CartItem>();
    }

    // Methods...

    /**
     * Displays the contents of the cart, including product details, price, discount, and quantity.
     */
    public void Display() {
        // Display logic...
    }
}
```

CARTITEM CLASS

```
/**
 * Represents an item in a shopping cart with a specified quantity.
 */
public class CartItem {
    private Product product;
    private int quantity;

    /**
     * Constructs a new CartItem with the specified product and quantity.
     *
     * @param product  the product associated with the item.
     * @param quantity the quantity of the product in the cart.
     */
    public CartItem(Product product, int quantity) {
        this.product = product;
        this.quantity = quantity;
    }

    // Getters and Setters...
}
```

DISCOUNTABLE INETRFACE

```
/**
 * Defines an interface for items that can provide a discount.
 */
public interface Discountable {
    double calculateDiscount();
}
```

STORE CLASS

```
/**
 * Represents a store containing products with inventory levels.
 */
public class Store {
   private Map<String, Product> products;

   /**
    * Constructs a new Store with an empty map of products and reads initial inventory from a file.
    */
   public Store() {
      products = new HashMap<String, Product>();
      this.readFromFile();
   }

   // Methods...

   /**
    * Displays the current products in the store with details like name, price, discount, and availability.
    */
   public void Display() {
      // Display logic...
   }
}
```

SALE CLASS

```
/**
 * Represents a sale transaction with a customer name, a shopping cart, and calculated total and
discount.
```

```
 */
public class Sale {
    private String customerName;
    private Cart cart;
    private double total;
    private double discount;

    /**
     * Constructs a new Sale with the specified customer name and cart.
     *
     * @param customerName the name of the customer making the purchase.
     * @param cart         the cart containing items for the sale.
     */
    public Sale(String customerName, Cart cart) {
        this.customerName = customerName;
        this.cart = cart;
    }

    // Methods...

    /**
     * Displays the details of the sale, including customer name, cart items, and total amount.
     */
    public void Display() {
        // Display logic...
    }
}
```

REPORT CLASS

```
/**
 * Represents a sales report with a list of sales, a total amount, and a list of popular products.
 */
public class Report {
    ArrayList<Sale> sales;
    double total;
    ArrayList<Product> PopularProducts;

    /**
     * Constructs a new Report with an empty list of sales, total set to 0, and an empty list of popular products.
     */
    public Report() {
        sales = new ArrayList<Sale>();
        PopularProducts = new ArrayList<Product>();
        total = 0.0;
    }

    // Methods...

    /**
     * Displays the sales report including the total sale amount and details of popular products.
     */
    public void Display() {
        // Display logic...
    }
}
```