CartSystem.java:

```
package shoppingCart.system;

import java.math.BigDecimal;

import shoppingCart.gui.UI;
import shoppingCart.model.Inventory;
import shoppingCart.model.UserList;

/** The entry point to the Shopping Cart application.
 *  The application performs different functions depending on who logs in.
 *  It allows a seller to maintain an inventory of items available for sale
 *  and customers to browse and add items to their cart, and purchase the contents of their cart.
 *
 *  CartSystem manages interactions between the UI and the DBManager, PaymentValidator, and UserList.
 *  It also creates the same.
 *  @author Newman Souza
 *  @author Seth Moore
 */
public class CartSystem {

   /** Constructs a CartSystem object.
    *  @precondition                Appropriate files are available for DBManager
    *  @postcondition               object created
    *  @postcondition               UI created and running
    */
   public CartSystem() {
            dbManager = new DBManager();
            paymentValidator = new PaymentValidator();
            UI ui = new UI(this);
            userList = new UserList();
                     userList = dbManager.loadUserList();
   }

   /** Creates a CartSystem
    *  @param args not used
    */
            public static void main(String[] args) {
                     CartSystem cartSystem = new CartSystem();
            }

   /** Takes a user's username and password, and, if they are a valid username/password pair,
    *  returns the User's type to the caller and loads the Inventory.
    *  Otherwise returns null.
    *  @param username              the User's username
    *  @param password              the User's password
    *  @return                      the User's type or null
    *  @precondition                username and password are valid references
    */
   public String login(String username, String password) {
            String type = userList.validate(username, password);
            if (type != null) {
                Inventory inventory = Inventory.getInstance();   // Inventory is a singleton, getInstance() guarantees that
                inventory = dbManager.loadInventory();           //   this same instance will always be used when it gets called.
            }
            return type;
   }
```

```
	/** Processes payment for for items in Customer's cart.
	 *  @param                          the customer's payment information
	 *  @return                         true if successful and false otherwise
	 *  @precondition                   none
	 *  @postcondition                  payment processed
	 */
	public boolean pay(String cardNumber, BigDecimal total) {
		boolean result = paymentValidator.validate(cardNumber, total);
		if (result) {
			saveInventory();
		}
		return result;
	}

	/** Saves the current state of the Inventory.
	 *  @precondition                   appropriate file permissions are available to DBManager.
	 */
	public void saveInventory() {
		Inventory inventory = Inventory.getInstance();
			dbManager.saveInventory(inventory);
	}

	private DBManager dbManager;
	private PaymentValidator paymentValidator;
	private UserList userList;

}
```