**Bookstore Final Project**

**Additional Topic: GUI (Chapters 10 & 20)**

**Gabriel Venezia and Jay Nguyen**

**Problem Statement**

The Online Bookstore Management System is a Java-based project that aims to create a comprehensive system for managing the operations of an online bookstore. The project will involve creating a set of Java classes to handle different aspects of the system, including book inventory management, customer information, order processing, and more.

**CRC Cards**

| **Book** | |
| --- | --- |
| **Responsibilities** | **Collaborators** |
| Store information such as title, author, genre, price, and availability. | Inventory Class |

| **Inventory** | |
| --- | --- |
| **Responsibilities** | **Collaborators** |
| Manage the inventory of books in the online bookstore | Book Class |
| Add new books |
| Update book details |
| Search for books |
| Remove books from inventory |

| **Customer** | |
| --- | --- |
| **Responsibilities** | **Collaborators** |
| Store customer information, including name, contact details, and address | Order Class |
| Register new customers |
| Manage customers’ information |

| **Order** | |
| --- | --- |
| **Responsibilities** | **Collaborators** |
| Handle the process of placing orders | Inventory Class  Customer Class |
| Add items to order |
| Calculate total cost |
| Apply discounts |
| Generate invoice |

| **ShoppingCart** | |
| --- | --- |
| **Responsibilities** | **Collaborators** |
| Represent a customer's shopping cart | Order Class |
| Add books |
| Remove books |

| **Payment** | |
| --- | --- |
| **Responsibilities** | **Collaborators** |
| Handle the payment process, integrating with a payment gateway to process credit cards or other payment methods | Customer Class |
| Verify credit card info |
| Check if cash is equal to total cost |
| Check if cash is over/under total cost |
| Give change back |

| **Book** | | |
| --- | --- | --- |
| **Task** | **Method** | **Data Needed** |
| Constructor | Book(title, author, genre) | title author genre |
| Get the title of book | getTitle() | title |
| Get the author of book | getAuthor() | author |
| Get genre of book | getGenre() | genre |
| Get the price of the book | getPrice() | price |
| Check if the book is in stock | checkAvailable() | boolean |

| **Inventory** | | |
| --- | --- | --- |
| **Task** | **Method** | **Data Needed** |
| Add new books | addNewBooks(title, author, genre, price) | title, author, genre, price |
| Update book details (title, author, genre, price, availability) | updateBookDetails(title, author, genre, price, availability) | title, author, genre, price, availability |
| Search for books | searchBook(title, author) | title, author |
| Remove books from inventory | removeBooks(title) | title |

| **Customer** | | |
| --- | --- | --- |
| **Task** | **Method** | **Data Needed** |
| Save customer information (constructor) | Customer(name, contactDetails, address) | name, contactDetails, address |
| Get name of customer | getName() | name |
| Get address of customer | getAddress() | address |
| Get contact info of customer | getContactDetails() | contact details |
| Register a new customer | registerCustomer() | Customer |
| Set/change name of customer | setName(name) | name |
| Set/change address of customer | setAddress(address) | address |
| Set/change contact info of customer | setContactDetails(contactDetails) | contact details |
| Customers discounts | hasDiscount() | discount |

| **Order** | | |
| --- | --- | --- |
| **Task** | **Method** | **Data Needed** |
| Default constructor | Order() |  |
| Add books to order | addBooks(books) | book |
| Calculate total cost of all books | totalCost() | bookPrice |
| Apply discounts | applyDiscounts() | discount |
| Generate an invoice | toString() | book, discount, total cost |

| **ShoppingCart** | | |
| --- | --- | --- |
| **Task** | **Method** | **Data Needed** |
| Default constructor | ShoppingCart() |  |
| Add books to shopping cart | addBooks(book) | book |
| Remove books from shopping cart | removeBooks(book) | book |

| **Payment** | | |
| --- | --- | --- |
| **Task** | **Method** | **Data Needed** |
| Cash constructor | Payment(cash) | cash |
| Credit card constructor | Payment(card number, expiration date, ccv) | card number, expiration date, ccv |
| Verify credit card info | verifyCreditCard() | card number, ccv, exp date |
| Check if cash is equal to total cost | isCashEqual(cash) | cash |
| Check if cash is over/under total cost | overOrUnder(cash) | cash |
| Give change | giveChange() | change |

**UML Diagram**

<https://drive.google.com/file/d/1gjt2RPc9rHw0nuper-FLMlY4XpHXjkkD/view?usp=drive_link>

**Method Documentation**

public class Book {

/\*\*

Book constructor

@param title

@param author

@param genre

@param price

@param available

\*/

public Book(String title, String author, String genre, double price, boolean available)

/\*\*

Get title of book

@return title

\*/

public String getTitle()

/\*\*

Get author of book

@return author

\*/

public String getAuthor()

/\*\*

Get genre of book

@return genre

\*/

public String getGenre()

/\*\*

Get price of book

@return price

\*/

public double getPrice()

/\*\*

Check if book is available

@return true or false

\*/

public boolean checkAvailable()

/\*\*

Set title of book

@param title

\*/

public void setTitle(String title)

/\*\*

Set author of book

@param author

\*/

public void setAuthor(String author)

/\*\*

Set genre of book

@param genre

\*/

public void setGenre(String genre)

/\*\*

Set price of book

@param price

\*/

public void setPrice(double price)

/\*\*

Set availability

@param true or false

\*/

public void setAvailable(boolean available)

}

public class Inventory {

/\*\*

Add new books to the inventory

@param title

@param author

@param genre

@param price

\*/

public void addNewBooks(String title, String author, String genre, double price, boolean availability)

/\*\*

Update a book’s details

@param title

@param author

@param genre

@param price

@param availability

\*/

public void updateBookDetails(String title, String author, String genre, double price, boolean availability

/\*\*

Search for books

@param title

@param author

@return Book

\*/

public Book searchBook(String title, String author)

/\*\*

Remove books from inventory

@param title

\*/

public void removeBooks(String title)

}

public class Customer {

/\*\*

Construct new customer

@param name

@param contactDetails

@param address

\*/

public Customer(String name, String contactDetails, String address)

/\*\*

Get name of customer

@return name

\*/

public String getName()

/\*\*

Get address of customer

@return address

\*/

public String getAddress()

/\*\*

Get contact details of customer

@return contactDetails

\*/

public String getContactDetails()

/\*\*

Register a new customer

@return Customer

\*/

public Customer registerCustomer()

/\*\*

Set name of customer

@param name

\*/

public void setName(String name)

/\*\*

Set address of customer

@param address

\*/

public void setAddress(String address)

/\*\*

Set contact details of customer

@param contact details

\*/

public void setContactDetails(String contactDetails)

/\*\*

See if customer has a discount to use

@return true or false

/\*.

public boolean hasDiscount()

}

public class Order {

/\*\*

Default constructor

\*/

public Order()

/\*\*

Add books to order

@param book

\*/

public void addBooks(Book book)

/\*\*

Calculate total cost of books

@return total cost

\*/

public double totalCost()

/\*\*

Apply discounts

\*/

public void applyDiscounts()

/\*\*

Generate an invoice

@return String of invoice

\*/

public String toString()

}

public class ShoppingCart {

/\*\*

Default constructor

\*/

public ShoppingCart()

/\*\*

Add book to shopping cart

@param book

\*/

public void addBooks(Book book)

/\*\*

Remove book from shopping cart

@param book

\*/

public void removeBooks(Book book)

}

public class Payment {

/\*\*

Constructor for cash payment

@param cash

\*/

public Payment(double cash)

/\*\*

Constructor for credit card payment

@param cardNumber

@param expirationDate

@param ccv

\*/

public Payment(String cardNumber, String expirationDate, int ccv)

/\*\*

Verify credit card info to see if its usable

@return true or false

\*/

public boolean verifyCreditCard()

/\*\*

Check if cash is equal to the total cost

@param cash

@return true or false

\*/

public boolean isCashEqual(double cash)

/\*\*

Check if cash is over/under the total cost

@param cash

@return true or false

\*/

public boolean overOrUnder(double cash)

/\*\*

Give back change

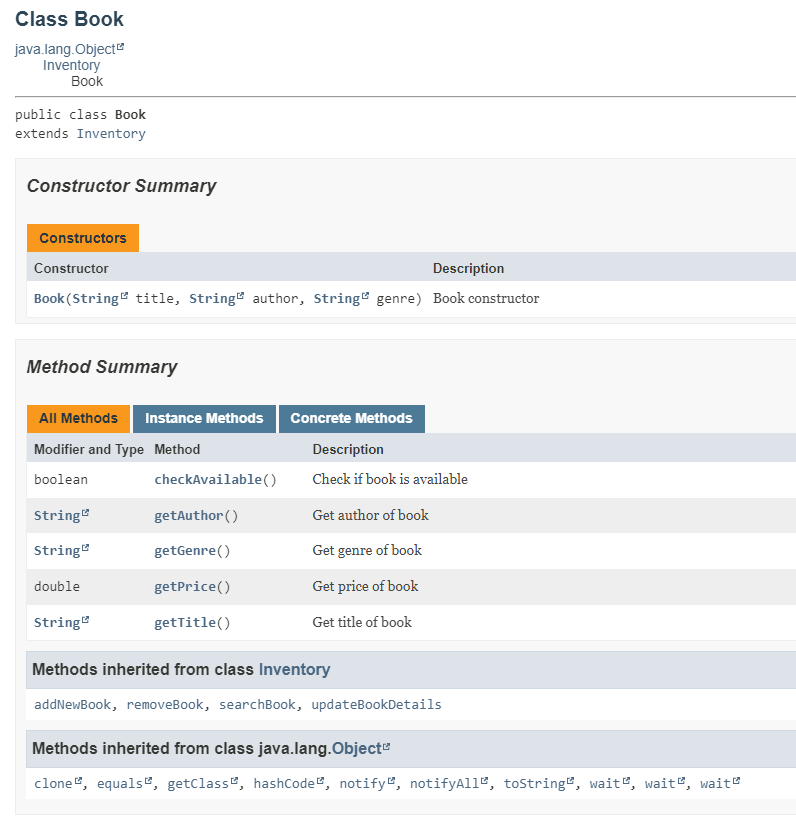
\*/

public void giveChange()

}

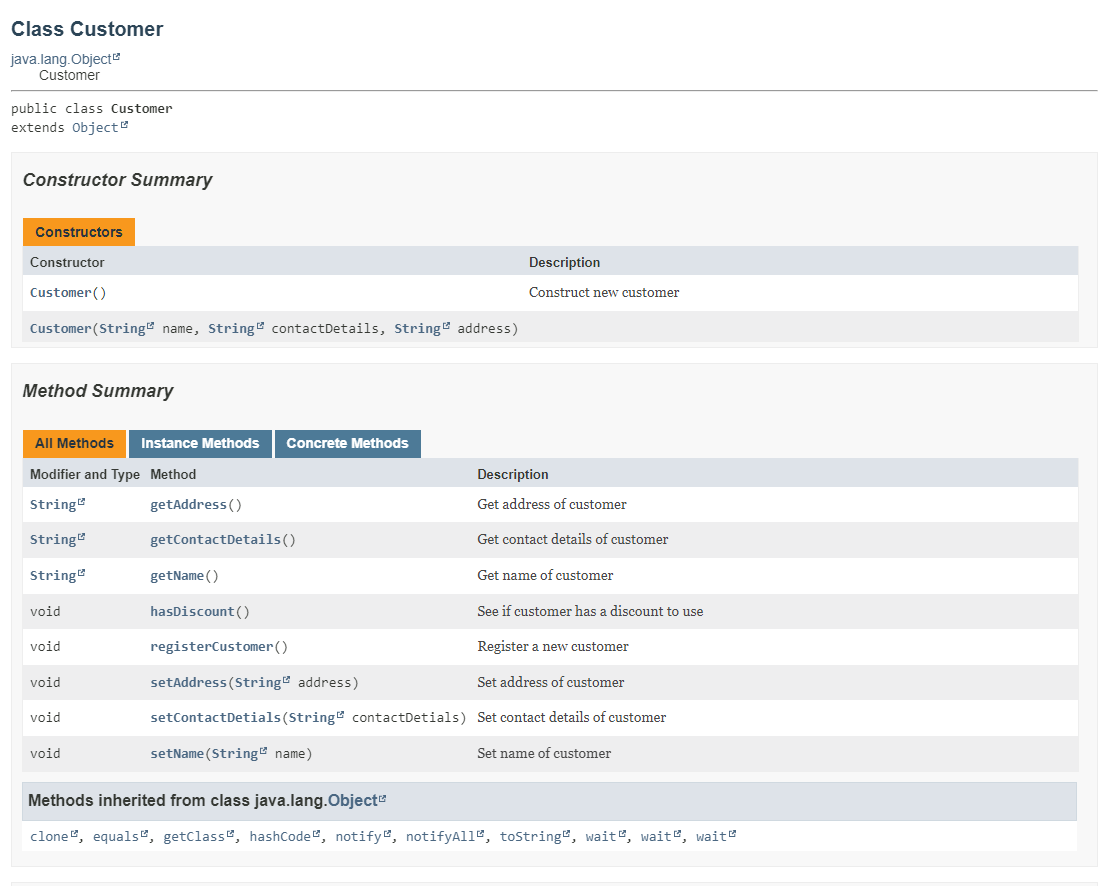
**JavaDoc**

***Book***

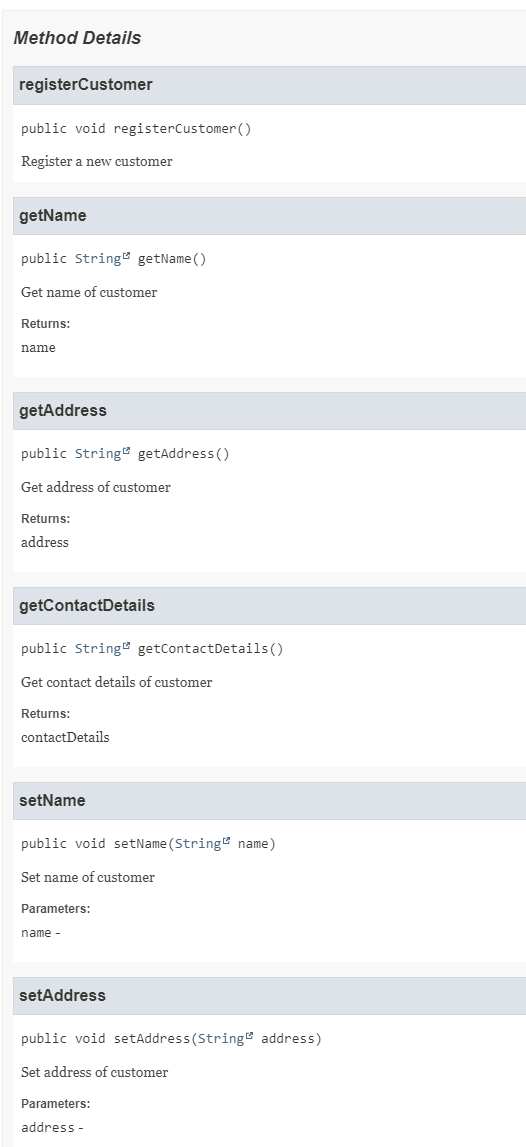




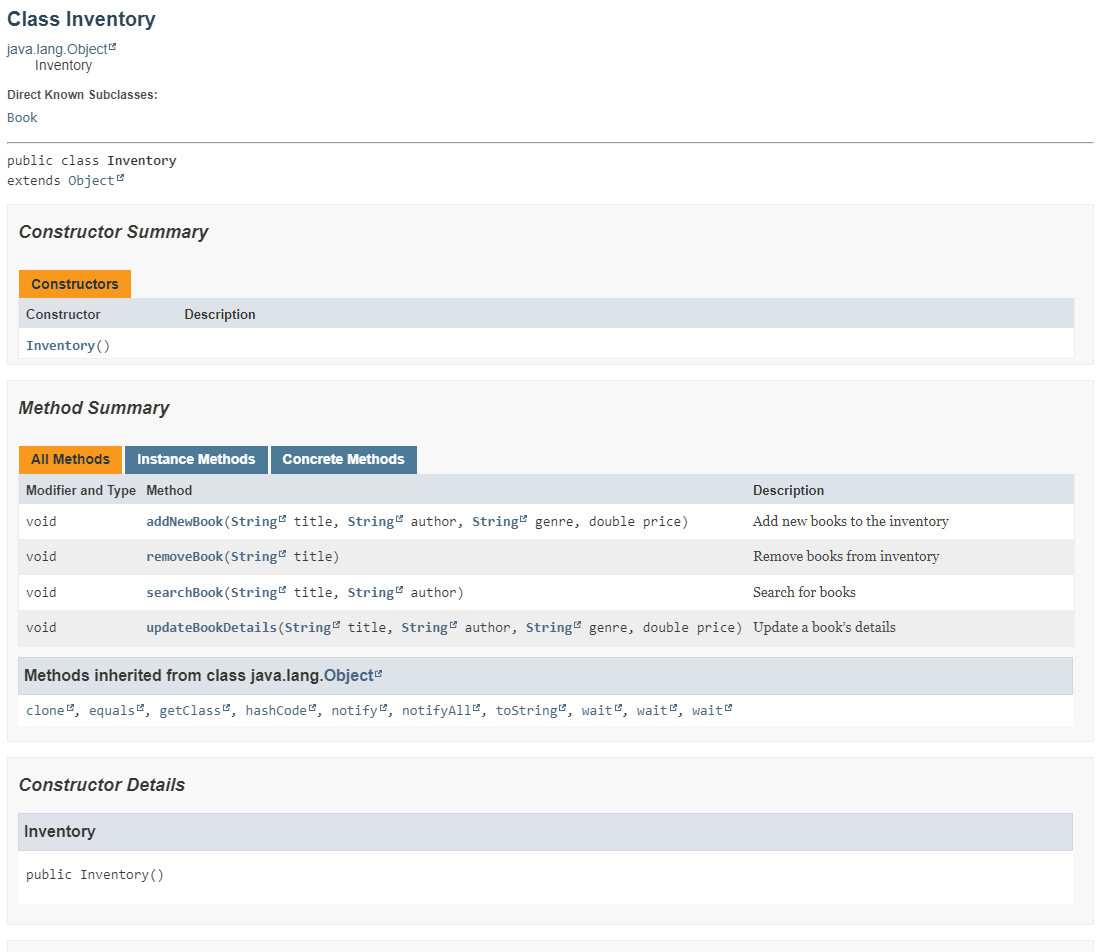
***Customer***





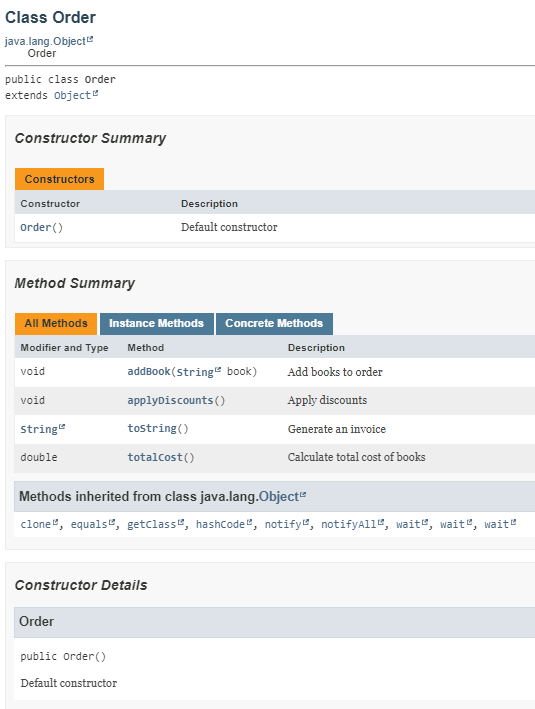


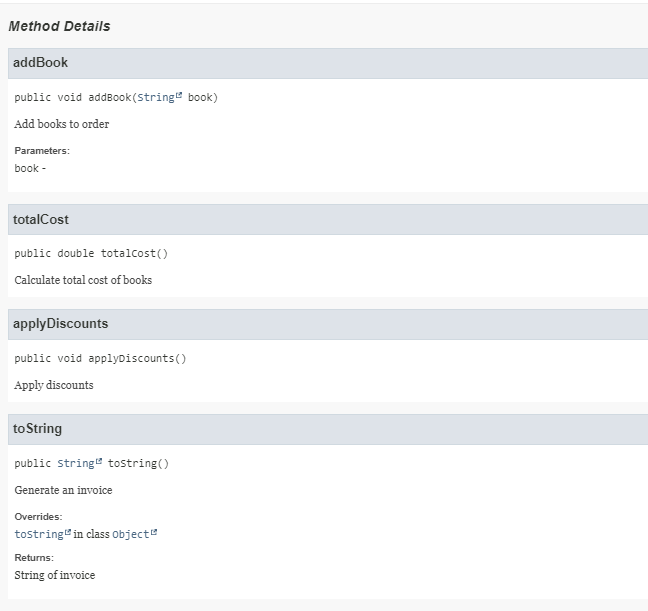


***Inventory***

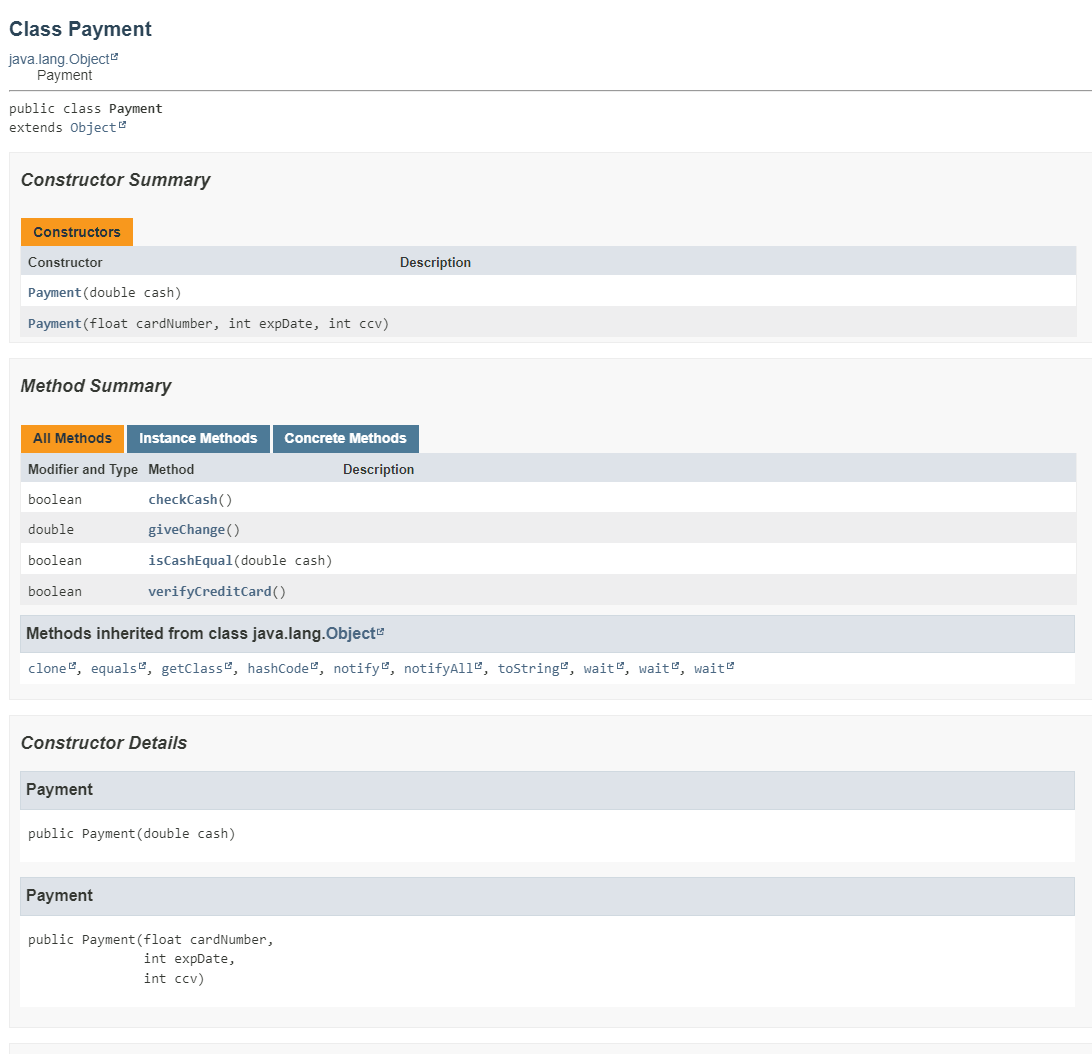
****

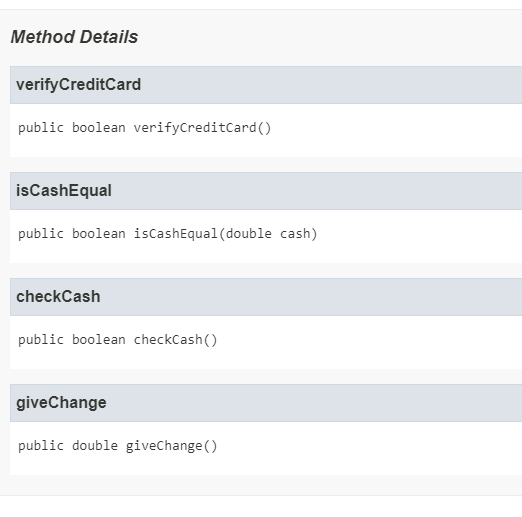
***Order***

****

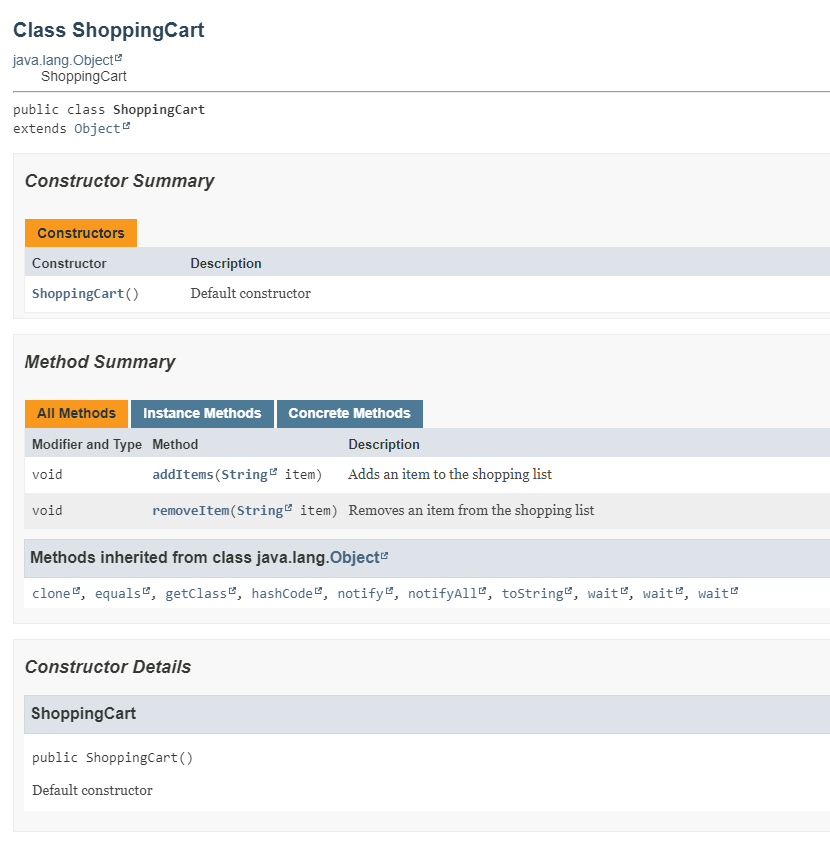
****

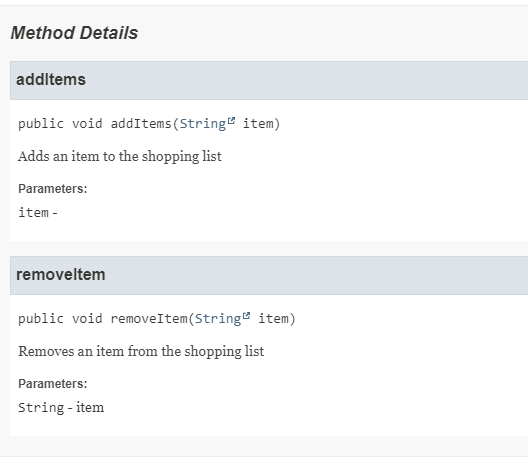
***Payment***





***ShoppingCart***

****

****

**Test Plan**

1. **Register the customer:**
   1. Test to see if customer enters valid information (name, contact details, address)
   2. Verify that they are successfully registered
   3. Check existing customers
2. **Browse Books:**
   1. Check if the Browse Books button displays books in the inventory with book title, author, genre, price, and availability
3. **Add Book to Shopping Cart**
   1. Test by adding books to shopping cart by adding correct and incorrect titles and authors
   2. Check that the shopping cart updates to each new book added
   3. Make sure accurate messages are displayed when adding books
4. **View Shopping Cart**
   1. Check if the View Cart button displays books in the shopping cart with titles, authors, and prices
   2. See if there is accurate messages when cart is empty
5. **Placing Order**
   1. Test placing order with books in cart
   2. Verify order summary is accurate with titles, authors, prices, total cost, etc.
   3. Handle things when cart is empty
6. **Payment**
   1. Test the Place Order button and input both cash and credit card options
   2. Verify the program prompts user for payment method
   3. Check accuracy when inputting payment details including credit card number, expiration date, CCV, and cash
   4. Make sure the change is calculated correctly
7. **GUI**
   1. Test user interface and see if its user-friendly for the online book system
   2. Check whether the buttons work correctly

**User Guide - How to Run Online Bookstore Management System**

**Prerequisites:**

1. ***Java:***
   * Ensure that you have Java installed on your computer. You can install it here: <https://www.oracle.com/java/technologies/downloads/>
2. ***Integrated Development Environment (IDE):***
   * This allows you to access/modify and run the program.

**Steps to Run Project**

1. ***Download Project***
   * Get the project source code files and download them to your desktop.
2. ***Extract Project***
   * If it's a ZIP file, extract it to obtain the files.
3. ***Open Files in IDE***
   * Go to the project directory in IDE to import/open project files.
4. ***Run Code***
   * The main class of the project is “OnlineBookstoreGUI”, so run the file .
5. ***GUI Opens***
   * Once you run the class, the Online Bookstore Management GUI will pop up as a separate window.
6. ***Enjoy the Project***
   * Have fun tinkering with the application by providing customer information and checking out with books.
7. ***Exit the application***
   * When finished, close the GUI to exit the application.