Design document for Item Class

**1. Introduction**

The code defines a C++ class **Item**, which represents an item that can be sold. An **Item** object has three attributes: name, price, and quantity. The class provides getter and setter methods for these attributes, as well as two operator overloads (**operator>>** and **operator<<**) and an equality operator (**operator==**).

**2. Class Definition**

**2.1 Class Attributes**

The **Item** class has the following attributes:

* **name**: a **string** that stores the name of the item.
* **price**: a **double** that stores the price of the item.
* **quantity**: an **int** that stores the quantity of the item.

**2.2 Class Methods**

The **Item** class has the following methods:

* **setName(string s)**: a setter method that sets the name of the item to **s**.
* **setPrice(double p)**: a setter method that sets the price of the item to **p**.
* **setQuantity(int q)**: a setter method that sets the quantity of the item to **q**.
* **getName()**: a getter method that returns the name of the item.
* **getPrice()**: a getter method that returns the price of the item.
* **getQuantity()**: a getter method that returns the quantity of the item.
* **operator>>(istream&, Item&)**: an operator overload that reads an item from an input stream.
* **operator<<(ostream&, Item&)**: an operator overload that writes an item to an output stream.

**2.3 Class Constructors**

The **Item** class has two constructors:

* **Item()**: a default constructor that creates an **Item** object with empty name, price 0.0, and quantity 0.
* **Item(string n, double p, int q)**: a constructor that creates an **Item** object with the specified name, price, and quantity.

**2.4 Equality Operator**

The **Item** class has an equality operator **operator==**, which compares two **Item** objects for equality.

**3. Conclusion**

The **Item** class provides a simple implementation of an item that can be sold, with attributes for name, price, and quantity, and methods for accessing and modifying these attributes. The class also provides two operator overloads and an equality operator, making it easier to work with **Item** objects in C++ programs.

UML diagram

+----------------+

| Item |

+----------------+

|- name: string |

|- price: double |

|- quantity: int |

+----------------+

| +Item() |

| +Item(string, double, int) |

| +setName(string) |

| +setPrice(double) |

| +setQuantity(int) |

| +getName(): string |

| +getPrice(): double |

| +getQuantity(): int |

+----------------+

| +operator>>(istream&, Item&) |

|  |
| --- |
|  |

Design document for shopping cart class

**1. Introduction**

The code defines a C++ class **ShoppingCart**, which represents a shopping cart for items. A **ShoppingCart** object is a bag of **Item** objects, with an additional attribute to store the total price of the items in the cart. The class provides methods for adding and removing items from the cart, as well as a method for getting the total price of the items in the cart.

**2. Class Definition**

**2.1 Class Attributes**

The **ShoppingCart** class has the following attribute:

* **totalPrice**: a **double** that stores the total price of the items in the cart.

**2.2 Class Methods**

The **ShoppingCart** class has the following methods:

* **getTotalPrice()**: a getter method that returns the total price of the items in the cart.
* **add(Item)**: a method that adds an **Item** object to the cart.
* **remove(Item)**: a method that removes an **Item** object from the cart.

**2.3 Class Constructor**

The **ShoppingCart** class has one constructor:

* **ShoppingCart()**: a default constructor that creates an empty **ShoppingCart** object with total price 0.0.

**2.4 Inheritance**

The **ShoppingCart** class is derived from the **Bag** class, which is a templated class for a bag of objects of type **ItemType**.

**3. Conclusion**

The **ShoppingCart** class provides a simple implementation of a shopping cart for items, as a bag of **Item** objects, with additional functionality to keep track of the total price of the items in the cart. The class provides methods for adding and removing items from the cart, as well as a method for getting the total price, making it easy to use in C++ programs.

UML diagram

+-----------------+ | ShoppingCart | +-----------------+ | - totalPrice: double +-----------------+ | + ShoppingCart() | + getTotalPrice(): double | + add(Item): bool | + remove(Item): bool +-----------------+ ^ | +------------------+ | | | +----------------+ | | Bag<ItemType> | | +----------------+ | | | | +----------------+ | |

Top of Form

Bottom of Form