Contributors: Evan Romero, Sarvesh Gade, Tanmay Gupta,
Aryika Kumar, Tri Nguyen, Usman Rashid

Code Smells Writeup

1) Code Smell: In the Order.java file, the `calculateTotalPrice` method is a Bloater– it's way too long, and does multiple things. This prevents effective handling and makes the method too long and confusing.
   Fix: Extracting methods from that method, creating two private helper methods `applyGiftCardDiscount` and `applyLargeOrderDiscount`. There are now two additional methods that shorten the functionality.
2) Code Smell: Large parameter list in TaxableItem, creating a large constructor that is virtually the same as Item.
   Fix: Rather, we pass in an Item to get the taxRate within the constructor, removing this bloater.
3) Code Smell: The GiftCardItem class is virtually the same as the Item class, and is used as "instanceof" in Order.java. This falls under the dispensable lazy class code smells category, as this class "does almost nothing and isn't responsible for anything, and no additional responsibilities are planned for it."
   Fix: Remove the GiftCardItem class and simply check within the method if it is of gift card type through a property of the Order.
4) Code Smell: The method "sendOrderConfirmation" that is now in the EmailSender class after fixing the SRP principle has multiple data parameters passed in that are just fields of the Order class. This is categorized as a data clump code smell.
   Fix: Remove the parameters that are fields of the Order class and replace them with passing in an Order object itself which those fields can then be accessed through the getters in the Order class.
5) Code Smell: Checking for the hasGiftCard() is duplicative, as we check it multiple times, each time having to iterate over all of the items in the list which can be inefficient. It is also checked whenever an item is added or removed.
   Fix: To fix this we can use a boolean of hasGiftCard, that will hold the value of whether or not a gift card is present. addItem() and removeItem() would then be updated, this way we would not need to iterate through the whole item list.
6) Code Smell: In the Order.java file, for `calculateTotalPrice,` the price of the item with tax and discounts is being calculated within the same method, making it not only a bloater increasing method size but also violates the code smell of each method having a single responsibility.
   Fix: We can separate this functionality and create a new method called calculateItemTotal and then call this method from calculateTotalPrice.
7) Code Smell: In the Order.java file, once you place the order, it tries to create and send an email with order items and prices. This would violate our (S)OLID principle of Single Responsibility due to there already being an EmailSender.java that would be in charge of any email sending.
   Fix: We eliminated the code creating the email and instead invoked the method that was newly created inside of the EmailSender.java. This allows our code to become more organized