|  |  |  |  |
| --- | --- | --- | --- |
| **Object Name:** | Inventory | **Description:** | This class sets up the view for the inventory and sub-inventories which will show the information for either category-wise or a general inventory.  Depending on what the user asks for, the object’s behaviors will be implemented accordingly. |
| **Properties:** | | | |
| **Name:** | **Visibility:** | **Description:** | |
| toys  foods  books | Private  Private  Private | Object instantiation of Toy class to access methods for implementation in inventory  Object instantiation of Food class to access methods for implementation in inventory  Object instantiation of Book class to access methods for implementation in inventory | |
| **Methods:** | | | |
| **Name:** | **Visibility:** | **Parameters:** | **Description:** |
| toysInit()  foodsInit()  booksInit()  inventory()  toyStock()  foodsStock()  booksStock() | Public  Public  Public  Public  Public  Public  Public | None  None  None  None  None  None  None | Sets up and generates all values for the toy category, name (descriptions), prices, quantity, and toy's age list  Sets up and generates all values for the food category, name (descriptions), prices, quantity, and calories contained in food  Sets up and generates all values for the book category, name (descriptions), prices, quantity, and author's names  Prints out an inventory with all of information for each of the three subgroups.  Prints a portion inventory pertaining to toys  Prints a portion inventory pertaining to foods  Prints a portion inventory pertaining to books |

|  |  |  |  |
| --- | --- | --- | --- |
| **Object Name:** | Toy | **Description:** | This class extends superclass item and implements the interface of Taxable. This class generate the appropriate and necessary values for the Toys category. Description, Prices, and age limits are information included and generated in this class |
| **Properties:** | | | |
| **Name:** | **Visibility:** | **Description:** | |
| toyPrice[]  age[] | Private  Private | Array is used to hold price values for the toys which are randomly generated through the use of a for loop.  Array is used to generate a minimum age value for the toys. | |
| **Methods:** | | | |
| **Name:** | **Visibility:** | **Parameters:** | **Description:** |
| generatePrices()    calculateTax()    descriptions()    quantity()    age() | Public    Public    Public    Public    Public | None    None    None    None    None | This method generates random price values from a set range of $10-$25, stores it inside of toyPrices[].  This method implements the method defined in the interface Taxable and applies 6% tax to the prices generated in generatePrices(), then it transfers the value to the overall Inventory array.  Method defines and names what 3 objects are available in the toy’s category of the shop inventory. Sends the String to the overall inventory array.  Calls the superclass’s method to initialize the quantity in inventory for the toy category  Method randomly generates an age between 3 and 10 to state as the youngest possible age for a specific toy. |

|  |  |  |  |
| --- | --- | --- | --- |
| **Object Name:** | Food | **Description:** | This class extends superclass item and implements the interface of Taxable. This class generate the appropriate and necessary values for the Food category. Description, Prices, and number of calories are information included and generated in this class |
| **Properties:** | | | |
| **Name:** | **Visibility:** | **Description:** | |
| foodPrice[] | private | Array that holds the randomly generate values for the different food items available in the shop | |
| **Methods:** | | | |
| **Name:** | **Visibility:** | **Parameters:** | **Description:** |
| generatePrices()    descriptions()    calories() | Public    Public    Public | None    None    None | This method generates random price values from a set range of $2-$10, stores it inside of foodPrices[] It sends this value to the overall Inventory array.  Method defines and names what 4 foods will be available in the food category of the shop inventory. Sends the String to the overall inventory array.  This method randomly generates a value between 110-300, and sets is as the number of calories present in one of the food items. Sends this value to the overall Inventory array |

|  |  |  |  |
| --- | --- | --- | --- |
| **Object Name:** | Book | **Description:** | This class extends superclass item and implements the interface of Taxable. This class generate the appropriate and necessary values for the Book category. Description, Prices, and age limits are information included and generated here |
| **Properties:** | | | |
| **Name:** | **Visibility:** | **Description:** | |
| bookPrices[] | Private | Array that holds the randomly generate values for the different books available in the shop | |
| **Methods:** | | | |
| **Name:** | **Visibility:** | **Parameters:** | **Description:** |
| generatePrices()    calculateTax()    descriptions()    authors() | Public    Public    Public    Public | None    None    None    None | This method generates random price values from a set range of $12-$35, stores it inside of bookPrices[].  This method implements the method defined in the interface Taxable and applies 6% tax to the prices generated in generatePrices(), then it transfers the value to the overall Inventory array.  Method defines and names what 4 foods will be available in the food category of the shop inventory. Sends the String to the overall inventory array.  This method defines the names of the authors of the books defined in descriptions(). It sends the names to the overall Inventory array |

**Challenges you’ve faced:**

Certain challenges I have faced in this assignment consisted of one major single problem. I had slightly misinterpreted the idea of polymorphism and inheritance, hence my first rendition of this programming assignment had a fatal flaw which ended in the compiler giving me a StackOverflowError. Afterwards through an analyzation and advice, I realized my classes were looped within each other in such a manner that when I tried to pass values to an array initialized in the super class, the program went through an endless loop, and eventually crashing it.

A small minor challenge I handled was the small errors and mistakes that I had made while coding, as at times, since I was sleepy or confused, I didn’t realize what I’d done. What was originally a small error, such as the wrong logic, or a mistype, would sometimes result in some confusing and irritating error.

**Software Architecture Description:**

In essentiality, my program has 3 main classes, 2 interfaces, and 3 subclasses. The 3 main classes consist of:

The main class- ***ShoppingList*:** which renders the program, containing the main method which is where the program begins. ShoppingList contains the beginning instructions to inform the user what they can do. This class also processes the user’s input through the use of a scanner and has multiple methods which contain code which will execute depending on the user’s input.

|  |  |
| --- | --- |
| **Name of Method:** | **Description/Purpose** |
| Instruction() Instructions2() | These two methods contain instructions that the user will see upon the start of the program |
| stringCompare() | This class reads the user’s input through *Scanner* and parse it into two substrings. This method also checks the two substrings, and according to what they contain, either some boolean variables turn true/false and/or other counter variables values change. |
| addItems()  getItems()  removeItems() | These three methods are in charge of changing the contents shopping list. addItems() allows user to see more info about the item they are adding. getItems() confirms their addition to the shopping list, and removeItems() removes items from the shopping list. |

***Item*-** This class is the superclass which has the subclasses (Toy, Food and Book). This class contains a method which randomizes inventory quantity values, shell methods that do nothing in specific-as their uses are specified in the subclasses, but it also contains main variables that are used to store information through the subclasses.

***Inventory*-** This class contains the rendition of the shop inventory through the linking of all three subclasses, and creating a visually aesthetic view in the console to view items available in the shops.

By using this design, I can easily limit and remove any duplication of code. This is a sound design in which all components of the program come together to make a flawless program which runs without errors.

**Current Limitations:**

At the current moment, the program does not enable the user to eventually checkout the items listed in their shopping list.

**Improvements:**

The most significant improvement to be made to this program is to allow the user to eventually check out the items in their shopping list. Other improvements can be made as well such as showing each object's individual information as well, or maintain a larger variety of objects in inventory and having more storage space in the shopping list.