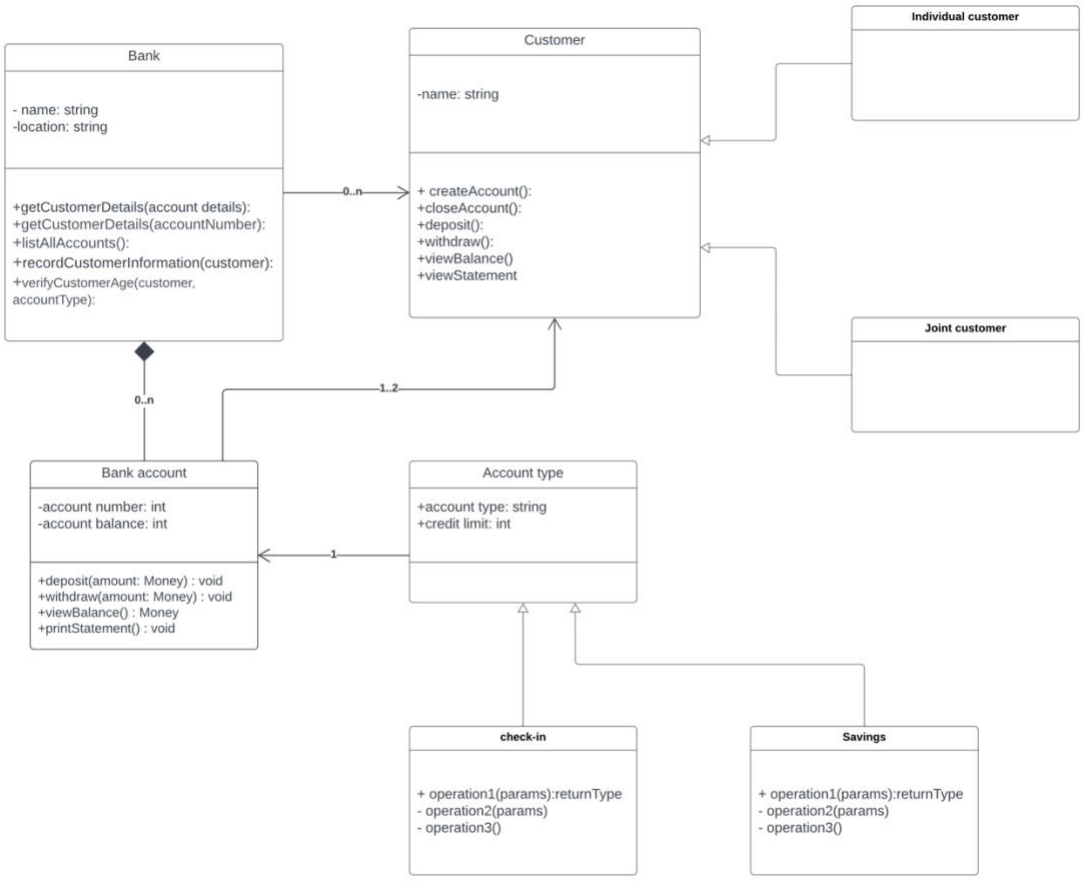
1

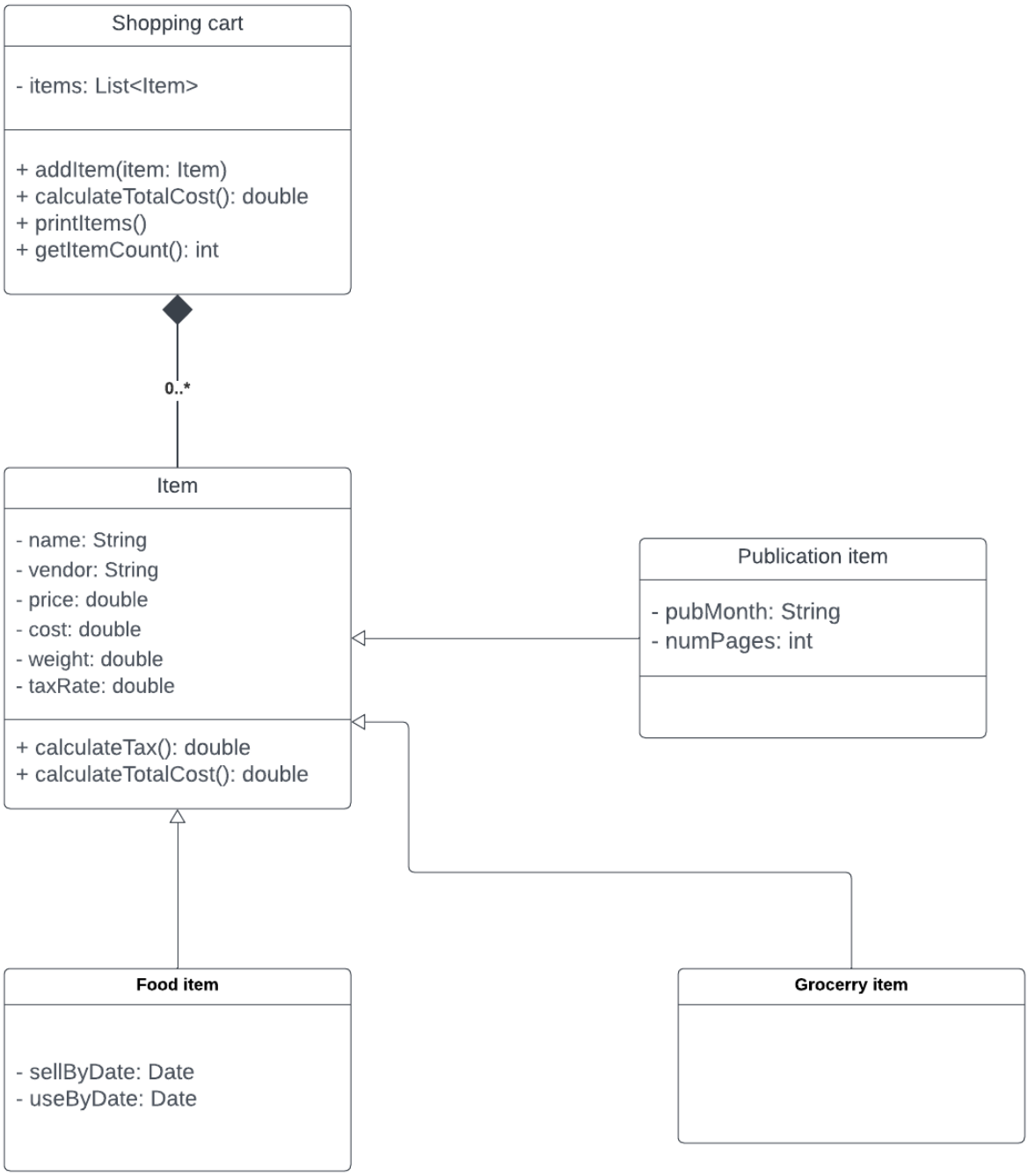


In this UML diagram:

- The "Bank" class is shown at the top.
- "Bank" has a composition relationship with "Account" since it creates and manages accounts.
- "Customer" has an association relationship with "Account" (0..n) because a customer can own multiple accounts.
- "Account" has an association relationship with "Customer" (1..2) because one account can be owned by up to two customers.
- "Account" also has an association relationship with "Account Type" to represent the type of account.

This UML diagram outlines the relationships between the classes and the multiplicity of these relationships. It provides a visual overview of how the classes are connected in the banking system.

2.



a description of the classes in the provided UML diagram and their relationships:

1. **ShoppingCart:**
    - Description: Represents a shopping cart that can hold multiple items.

- Attributes:
    - **items**: A list of items in the shopping cart.
- Methods:
    - **addItem(item: Item)**: Adds an item to the shopping cart.
    - **calculateTotalCost()**: Calculates the total cost of all items in the cart, including tax.
    - **printItems()**: Prints all items in the cart, including their price, tax, and total cost.
    - **getItemCount()**: Returns the number of items in the cart.

2. **Item:**
    - Description: Represents a generic item with common attributes.
    - Attributes:
        - **name**: A string that holds the name of the item.
        - **vendor**: A string that stores the name of the vendor for the product.
        - **price**: A double that stores the current selling price of the item.
        - **cost**: A double that stores the current cost price of the item.
        - **weight**: A double that stores the weight of a single item.
        - **taxRate**: A double representing the tax rate charged on the item.
    - Methods:
        - **calculateTax()**: Calculates the tax amount for the item.
        - **calculateTotalCost()**: Calculates the total cost of the item, including tax.
3. **PublicationItem:**
    - Description: Represents an item of type "Publication" with additional attributes.
    - Attributes:
        - **author**: A string that holds the name of the author.
        - **pubMonth**: A string that stores the publication month.
        - **numPages**: An integer representing the number of pages.
    - Inheritance: Inherits from the **Item** class.
4. **FoodItem:**
    - Description: Represents an item of type "Food" with additional attributes.
    - Attributes:

- **sellByDate**: A date indicating the sell-by date of the food item.
- **useByDate**: A date indicating the use-by date of the food item.

- Inheritance: Inherits from the **Item** class.

5. **GroceryItem:**
    - Description: Represents an item of type "Grocery," which is a general grocery item.
    - Inheritance: Inherits from the **Item** class.

The relationships between the classes are as follows:

- The **ShoppingCart** class contains a list of **Item** instances. It has an association relationship with the **Item** class, indicating that it can contain multiple items.
- The **Item** class serves as the superclass for specialized item types: **PublicationItem**, **FoodItem**, and **GroceryItem**. These subclasses inherit attributes and methods from the **Item** class.

The **calculateTax()** and **calculateTotalCost()** methods in the **Item** class are overridden in the subclasses to provide specialized implementations based on the item type.

This design allows for modeling a shopping cart that can hold various types of items, each with its specific attributes and behavior.