Homework Manual

by

Kyle Ponte

kponte@stevens.edu

September 20, 2024

Homework Manual

Kyle Ponte
kponte@stevens.edu

This document provides the requirements and design details of the PROJECT. The following table (Table 1) should be updated by authors whenever major changes are made to the architecture design or new components are added. Add updates to the top of the table. Most recent changes to the document should be seen first and the oldest last.

Table 1: Document Update History

Date	Updates
08/22/2023	DDM: • Updated dsnManual.tex with *newcommand(s){}* for easier references of requirements, figures, and other labels.
10/25/2023	DDM: • Added chapters on use cases (Chapter **??**) and user stories (Chapter **??**).
10/11/2023	DDM: • Added chapters on requirements (Chapter **??**) and glossary.
09/18/2023	DDM: • Added chapter on development plan (Chapter **??**).
9/12/2024	Homework 1: • Added chapter called Team (Chapter 1) and a chapter called Git Homework (Chapter 2).
9/19/2024	Homework 2: • Added chapter called UML Class Modeling (Chapter 3).

Table of Contents

List of Tables

List of Figures

Chapter 1

Team
– Kyle Ponte

My name is Kyle Ponte and I am a software engineer at Stevens Institute of Technology. I'm from Bloomfield, New Jersey, and I love both playing and watching basketball. I am also very fond of traveling, and so far have ventured to places in South and Central America like Peru and the Dominican Republic. In terms of other hobbies, I also enjoy listening to music and going to the gym.

Person	Studies	**Major**
-name		-name

Figure 1.1: Example Diagram 1

Person	LivesAt	**Residency**
-name		-address

Figure 1.2: Example Diagram 2

Chapter 2

Git Homework
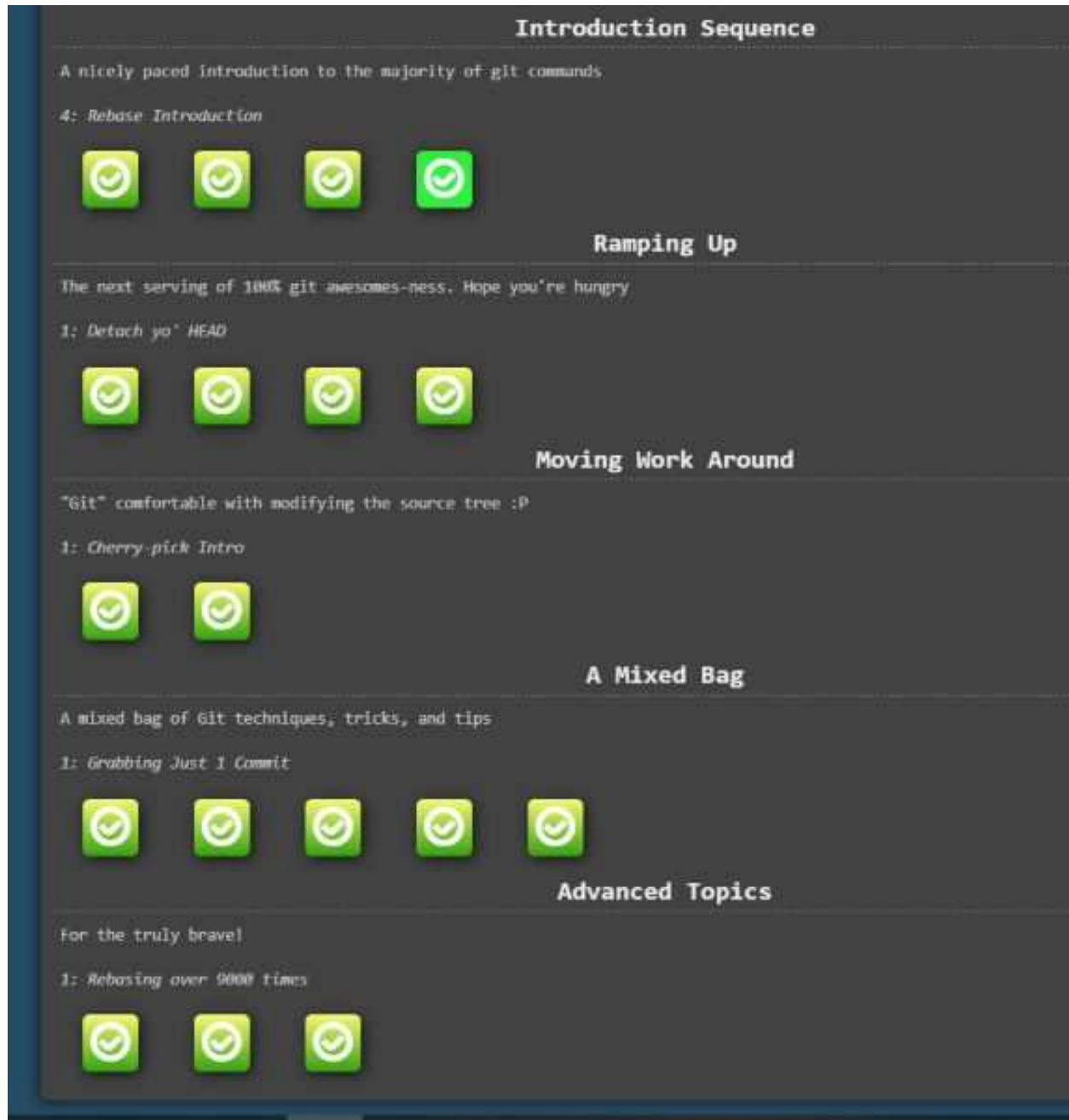*– Kyle Ponte*

Part II:

Figure 2.1: Part II HW

Chapter 3

UML Class Modeling
– Kyle Ponte

3.1   Exercise 2.1

Figure 3.1: Sample Undirected Graph
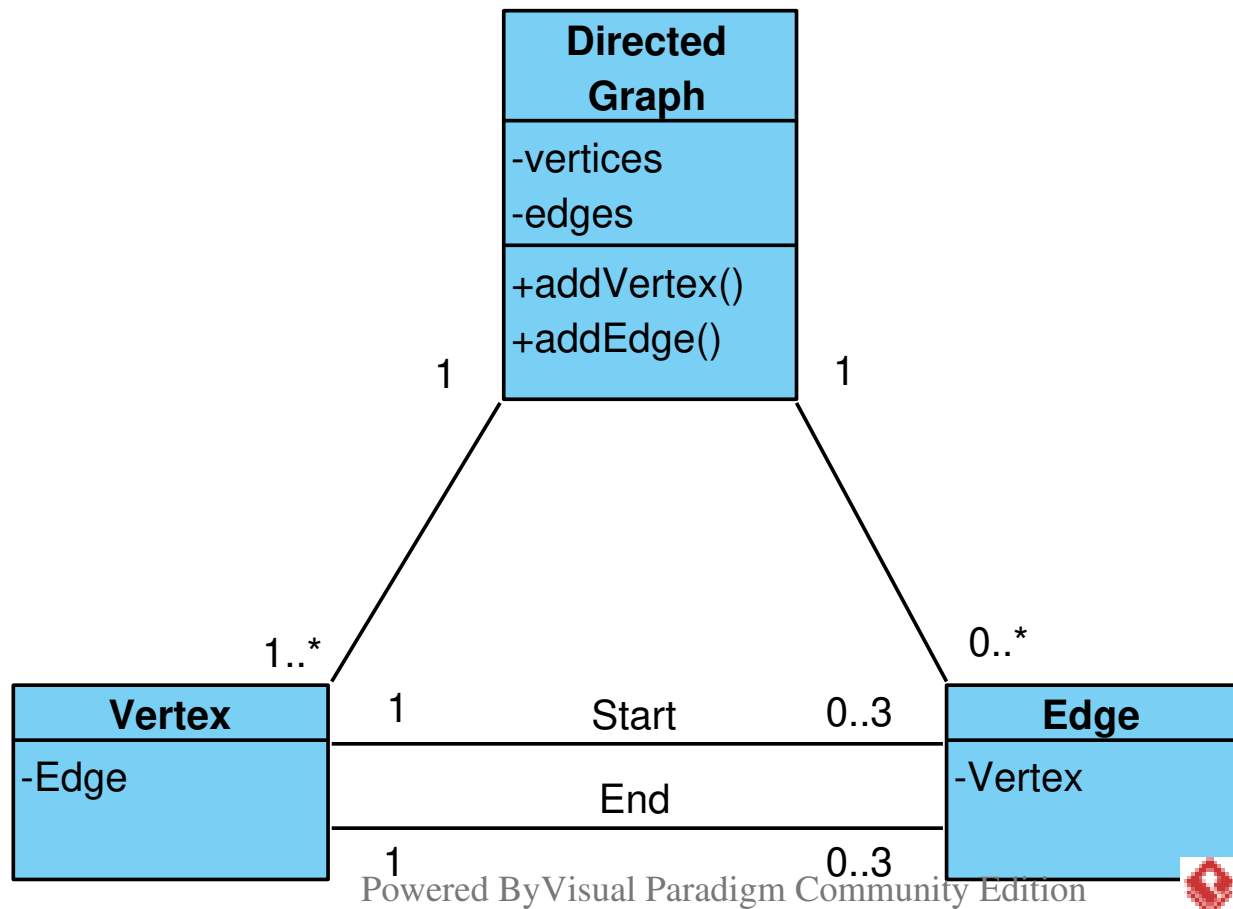
3.2 Exercise 2.2

Figure 3.2: Sample Directed Graph

3.3 Exercise 2.3

1. Window Class

   First, the Window class is associated with the ScrollingWindow, Canvas, and Panel class by being their parent. This is an example of composition association, as they can't exist without the Window class. Furthermore, the Window class has attributes x1, y1, x2, y2. Additionally, it has multiple methods, those being display, undisplay, raise, and lower.

2. Canvas Class

   Since the Canvas class is a child of the Window class, it inherits the Window class' attributes and methods. Moreover, the Canvas class is the sole parent to the Shape class, indicating that

it shares a composition association with it. To be more specific in its relationship with the Shape class, one Canvas instance associates with numerous Shape elements. It also shares an aggregation association with the ScrollingCanvas class as it is one of multiple parents. In terms the attributes, the Canvas class has cx1, cy1, cx2, and cy2. It also has two methods, being addElement and deleteElement.

3. ScrollingWindow Class

   The ScrollingWindow class is also the child of the Window class, so it inherits its methods and attributes. Furthermore, the ScrollingWindow class has two attributes, xOffset and yOffset. It also has a scroll method. It shares a composition association with the TextWindow class as it's its only parent. On the other hand, it shares a aggregation association with the ScrollingCanvas class since it has multiple parents.

4. Panel Class

   The Panel class is another child of the Window Class so it inherits its methods and attributes. It is one of the parents of the PanelItem class, making it a aggregation association. Additionally, an instance of the Panel class can be associated with either 0 or 1 instances of PanelItem. Lastly, it has one attribute, being itemName.

5. Event Class

   The Event class is a parent to both the PanelItem and TextItem class. It has an aggregation association with both as they could exist without the Event class. Furthermore, it contains an attribute called action. To be more specific in it's association with the PanelItem class, for every event it can be associated with numerous PanelItem instances. Similarly, for every event it can be associated with numerous TextItem instances. It also has a role, notifyEvent, that connects to the PanelItem class, alongside another behavior, keyboardEvent, that connects to the TextItem class.

6. Shape Class

   The Shape class is a child of the Canvas class, so it inherits all of its attributes and methods. Furthermore, Each shape instance can only associate with one Canvas instance. The Shape class is the sole parent of both the Line and ClosedShape class. This means that it shares a composition association with both of them. Finally, it contains two attributes, color and lineWidth.

7. PanelItem Class

   The PanelItem class is a child of both the Panel and Event class, so it inherits attributes and methods from both of them. Moreover, each PanelItem instance sees one Panel instance and one Event instance. It has three attributes, those being x, y, and label. It is also the only parent to the Button and ChoiceItem class, indicating that it has a composition association with both. Lastly, it is one of the parents of the TextItem class, meaning that it has an aggregation association with it.

8. TextWindow Class

   The TextWindow class is a child of the ScrollingWindow class, so it inherits each of its attributes and methods. It also has one attribute, being string. Lastly, it has insert and delete methods.

9. ScrollingCanvas Class

   The ScrollingCanvas class is a child of both the ScrollingWindow and Canvas class, meaning that it inherits attributes and methods from both of them. Apart from that, it has no unique attributes or methods.

10. Line Class

    The Line Class is a child of the Shape class so it inherits its attributes and methods. It also has four attributes, being x1, y1, x2, and y2. Lastly, it has a draw method.

11. ClosedShape Class

    Similarly, the ClosedShape class is a child of the Shape class so it also inherits its attributes and methods. It is also the sole parent to the Ellipse class, making it a composition association. On the other hand, it is one of the Polygon class' parents, meaning that they share an aggregation association. Finally, the ClosedShape class has two attributes, being fillColor and fillPattern.

12. Button Class

    The Button class is a child of the PanelItem class, indicating that it inherits all of its methods and attributes. Furthermore, it has two attributes, string and depressed.

13. ChoiceItem Class

    Next is the ChoiceItem class, which is also a child of the PanelItem class, meaning that it inherits its attributes and methods. It doens't have any unique attributes or methods itself, but has a child ChoiceEntry. It shares a composition association with the ChoiceEntry class as the child can't exist without the parent. There are two subsets of the class, as they are separated by roles/behaviors. One role is the currentChoice, which always sees one instance of the ChoiceEntry class. On the other hand, the choices role sees numerous instances of the ChoiceEntry class.

14. TextItem Class

    The TextItem class is a child of both the PanelItem and Event class, so it inherits attributes and methods from both of them. It also has two attributes, being maxLength and currentString. To be more specific in it's association with the Event class, one instance of TextItem always sees one instance of the Event class.

15. Point Class

The Point class shares an association with the Polygon class, as any instance of Point always sees one instance of the Polygon class. Furthermore, the Point class has two attributes, being x and y.

16. Polygon Class

The Polygon class is a child of the ClosedShape class so it inherits all of its attributes and methods. As mentioned before, it shares an association with the Point class. Furthmore, any instance of the Polygon class sees numerous instances of the Point class. Lastly, this class has a draw attribute.

17. Ellipse Class

The Ellipse class is also a child of the ClosedShape class, meaning that it inherits its attributes and methods. It has four attributes, being x, y, a, and b. It also has a draw method.

18. ChoiceEntry Class

The ChoiceEntry class is a child of the ChoiceItem class. As mentioned earlier, there are two subsets for potential roles, being currentChoice and choices. The currentChoice role can see either zero or one ChoiceItem instances, and the choices role can always see one instance of the ChoiceItem class.

3.4   Exercise 2.4

1. MailingAddress Class

The MailingAddress class is the sole parent of the Customer class, meaning that they share a composition association. In terms of multiplicity, an instance of the MailingAddress class could be connected to numerous instances of the Customer class. Furthmore, the MailingAddress class has an association with the CreditCardAccount Class. Additionally, one instance of the MailingAddress class can see numerous instances of the CreditCardAccount class. Lastly, this class has two attributes, being address and phoneNumber.

2. CreditCardAccount Class

The CreditCardAccount class is the only parent of the Statement class, indicating that there is a composition association between the two. When it comes to multiplicity, an instance of this class can see either zero or one instance of the Statement class. Moreover, this class shares an association with the Institution class. Furthermore, an instance of this class will always be correlated to one instance of the Institution class. Finally, this class has maximumCredit and currentBalance as attributes.

3. Institution Class

The Institution Class is only associated with the CreditCardAccount class, and each instance of this class can see either zero or one instances of the CreditCardAccount Class. It has also three attributes, those being name, address, and phoneNumber.

4. Customer Class

   The Customer class is the child of the MailingAddress class, indicating that it inherits all of its attributes. For every instance of the Customer class, it can see numerous instances of the MailingAddres class. Lastly, it has an attribute called name.

5. Statement Class

   The Statement Class is the child of the CreditCardAccount class, so it inherits all of its attributes. Each instance of this class correlates to one instance of the CreditCardAccount class. Furthermore, this class is associated with the Transaction class. For every instance of the Statement class, it either sees zero or one instances of the Transaction class. This class has multiple attributes, those being paymentDueDate, financeCharge, and minimumPayment.

6. Transaction Class

   The Transaction class is associated with the Statement class. Each instance of this class always connects to one instance of the Statement class. This class is also the parent to the CashAdvance, Interest, Purchase, Fee, and Adjustment class. It shares a composition association with all of these classes as well. Finally, it has three attributes, being transactionDate, explanation, and amount.

7. CashAdvance Class

   The CashAdvance class is the child of the Transaction class, meaning that it inherits its attributes.

8. Interest Class

   Similarly, the Interest class is another child of the Transaction class, resulting in it inheriting its attributes.

9. Purchase Class

   The Purchase Class is also a child of the Transaction class and inherits all of its attributes. It also has a child, the Merchant class, and shares a composition association with it. Furthermore, every instance of the Purchase class correlates to one instance of the Merchant class.

10. Fee Class

    The Fee Class is another child of the Transaction class, meaning that it inherits its attributes. It also has a unique attribute, being feeType.

11. Adjustment Class

    The Adjustment Class a child of the Transaction class, indicating that it inherits all of its attributes.

12. Merchant Class

The Merchant Class is the child of the Purchase class, resulting in it inheriting whatever attributes and methods the Purchase class might have. Lastly, it has one unique attribute, being name.

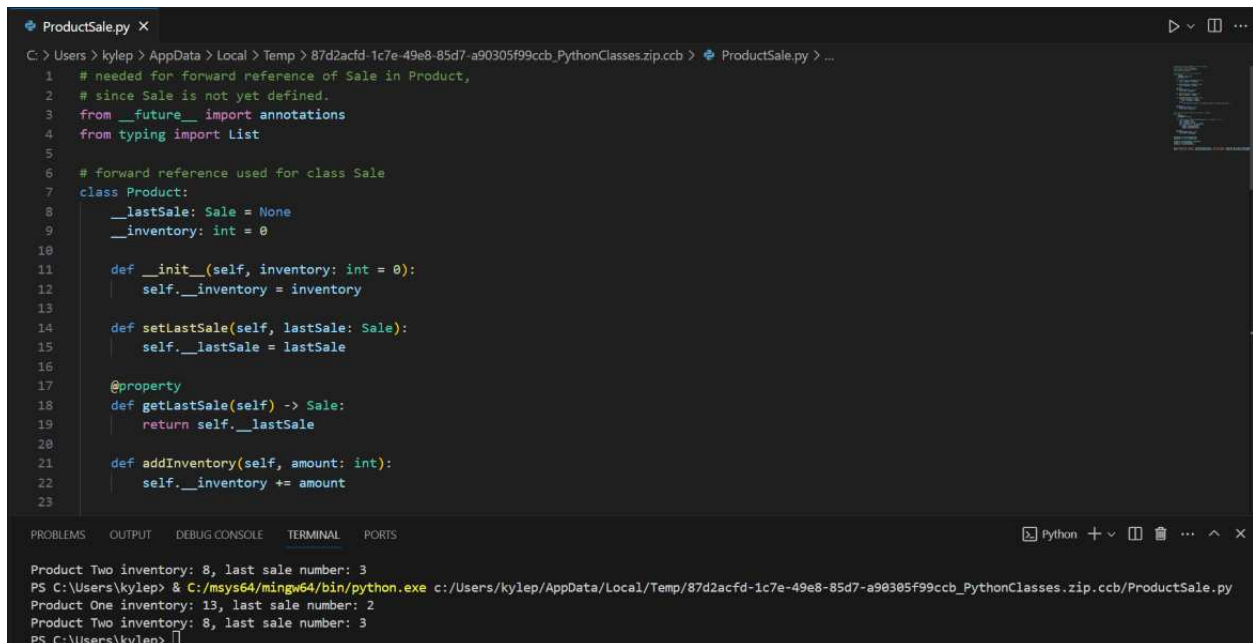## 3.5   Class - to - Code Modeling

```python
# needed for forward reference of Sale in Product,
# since Sale is not yet defined.
from __future__ import annotations
from typing import List

# forward reference used for class Sale
class Product:
    __lastSale: Sale = None
    __inventory: int = 0

    def __init__(self, inventory: int = 0):
        self.__inventory = inventory

    def setLastSale(self, lastSale: Sale):
        self.__lastSale = lastSale

    @property
    def getLastSale(self) -> Sale:
        return self.__lastSale

    def addInventory(self, amount: int):
        self.__inventory += amount

    def removeInventory(self, amount: int):
        if self.__inventory >= amount:
            self.__inventory -= amount
        else:
            raise ValueError('There is not enough inventory to remove this
    amount.')

    @property
    def getInventory(self) -> int:
        return self.__inventory


# no forward reference needed since Product is defined
class Sale:
    __saleTimes = 0
    __saleNumber: int = 0

    def __init__(self, product: List[Product]):  #, saleNumber: int = 1):
        Sale.__saleTimes +=1
        self.__product = product
```

```
43            self.__saleNumber = Sale.__saleTimes
44            for product in self.__product:
45                product.removeInventory(1)
46                product.setLastSale(self)
47
48        @property
49        def getSaleNumber(self) -> int:
50            return self.__saleNumber
51
52
53  productOne = Product(inventory=15)
54  productTwo = Product(inventory=10)
55
56  saleOne = Sale([productOne, productTwo])
57  saleTwo = Sale([productOne])
58  saleThree = Sale([productTwo])
59
60
61  print(f"Product One inventory: {productOne.getInventory}, last sale number: {
        productOne.getLastSale.getSaleNumber}")
62  print(f"Product Two inventory: {productTwo.getInventory}, last sale number: {
        productTwo.getLastSale.getSaleNumber}")
```

3.6 Code Results

Figure 3.3: Code Results

Product		**Sale**
-lastSale -inventory	1..* 0..*	-saleTimes -saleNumber -product
+setLastSale() +getLastSale() +addInventory() +removeInventory() +getInventory()		+getSaleNumber()
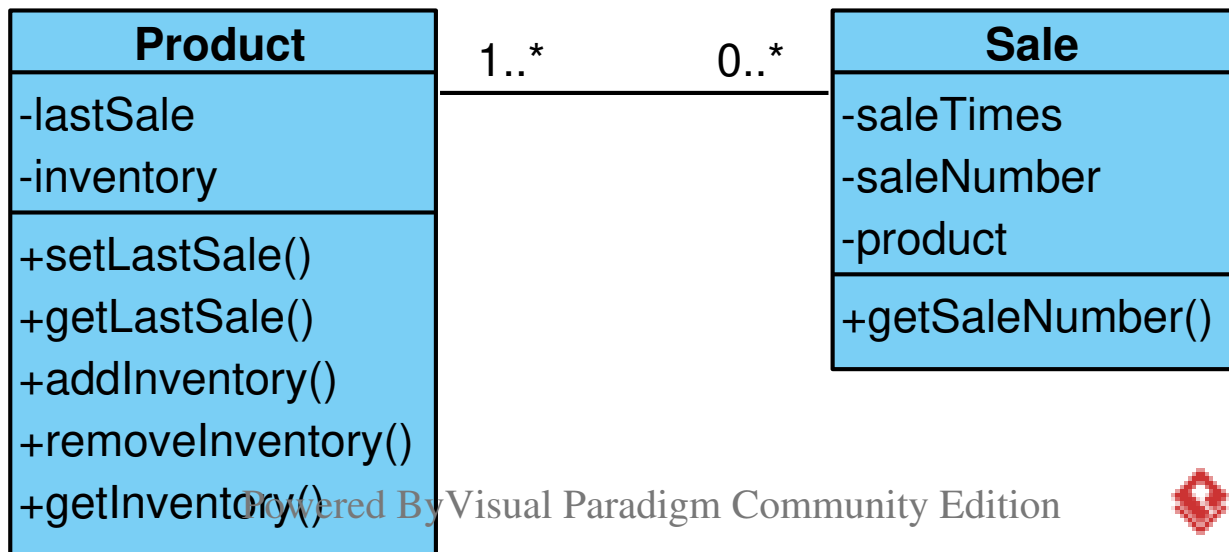
Figure 3.4: UML Diagram for Product and Sale

Bibliography

Index