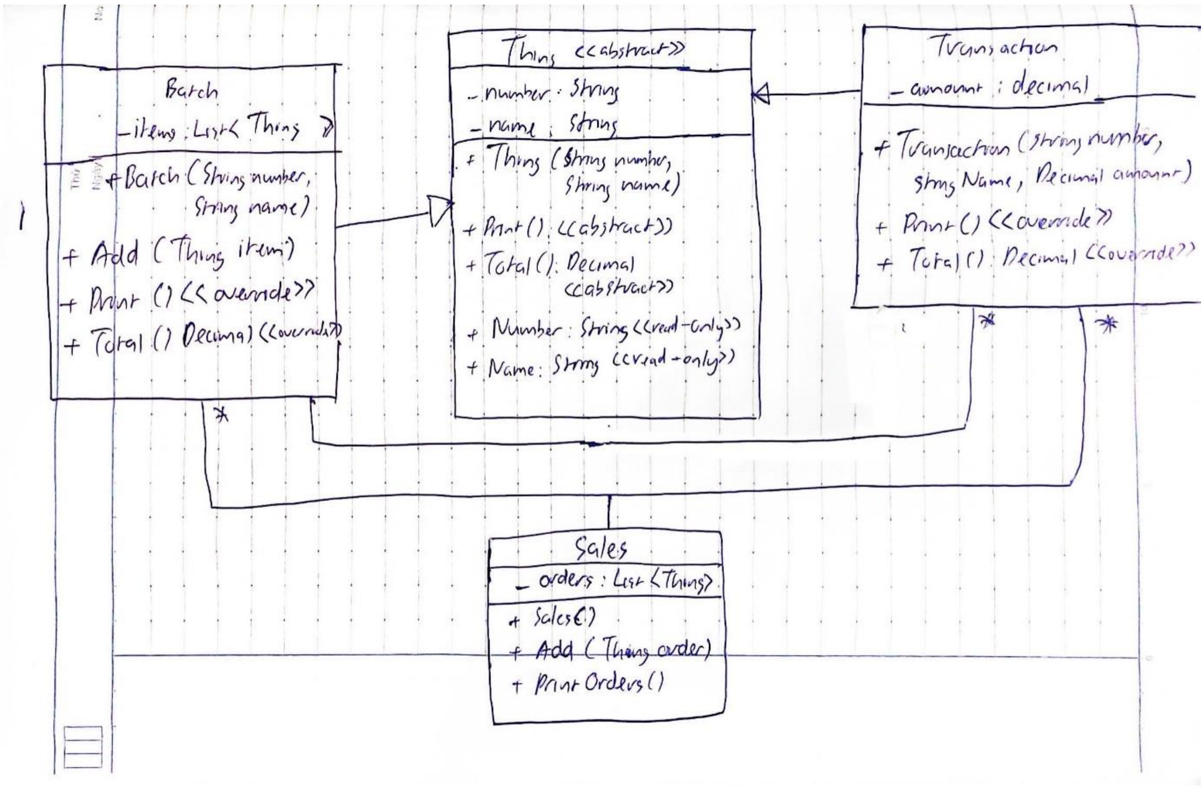
# Semester Test

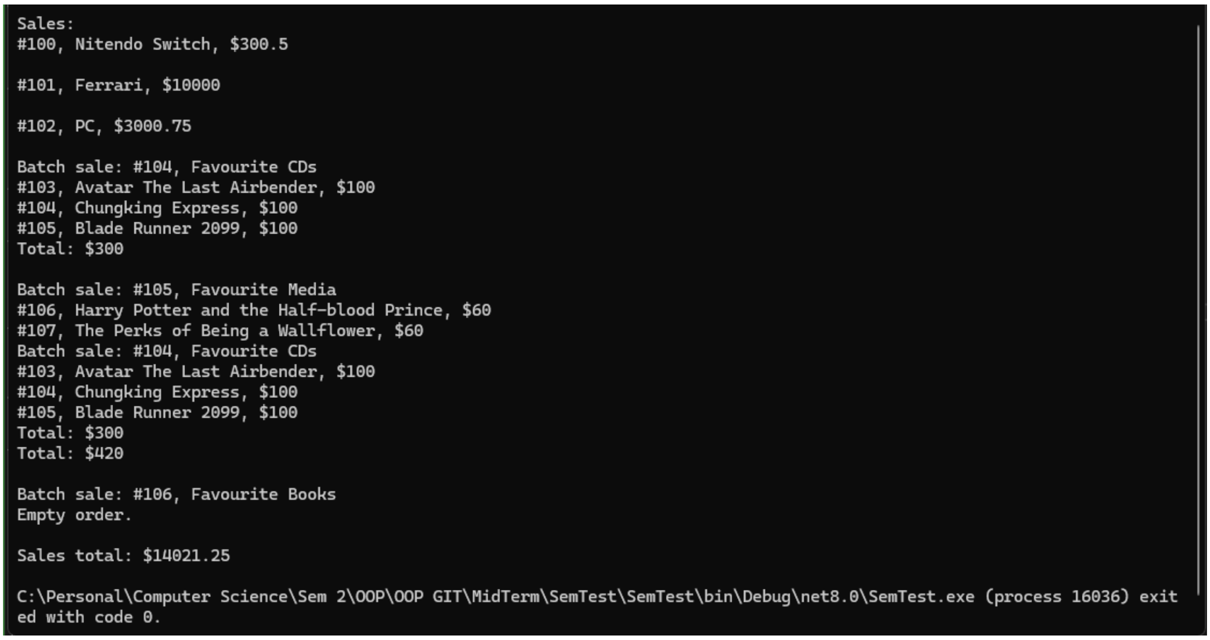
## Task 1:

UML Class Diagram:



Batch
- items : List< Thing >
+ Batch (String number, String name)
+ Add (Thing item)
+ Print () <<override>>
+ Total () Decimal <<override>>
Thing <<abstract>>
- number : String
- name : String
+ Thing (String number, String name)
+ Print () <<abstract>>
+ Total () : Decimal <<abstract>>
+ Number : String <<read-only>>
+ Name : String <<read-only>>
Transaction
- amount : decimal
+ Transaction (String number, String Name, Decimal amount)
+ Print () <<override>>
+ Total () : Decimal <<override>>
*
*
*
Sales
- orders : List <Thing>
+ Sales()
+ Add (Thing order)
+ Print Orders ()

All files are included with the submission. Here is the program output:



```
Sales:
#100, Nitendo Switch, $300.5

#101, Ferrari, $10000

#102, PC, $3000.75

Batch sale: #104, Favourite CDs
#103, Avatar The Last Airbender, $100
#104, Chungking Express, $100
#105, Blade Runner 2099, $100
Total: $300

Batch sale: #105, Favourite Media
#106, Harry Potter and the Half-blood Prince, $60
#107, The Perks of Being a Wallflower, $60
Batch sale: #104, Favourite CDs
#103, Avatar The Last Airbender, $100
#104, Chungking Express, $100
#105, Blade Runner 2099, $100
Total: $300
Total: $420

Batch sale: #106, Favourite Books
Empty order.

Sales total: $14021.25

C:\Personal\Computer Science\Sem 2\OOP\OOP GIT\MidTerm\SemTest\SemTest\bin\Debug\net8.0\SemTest.exe (process 16036) exit
ed with code 0.
```

## Task 2:

1.

- I think Polymorphism is based on Inheritance in a way. It still enables classes to inherit a parent class but allows that class to take on various forms depending on the context that it is reused in. In other words, while Inheritance relationship is “is-a”, the relationship that Polymorphism is more of “act-like”: classes can perform different actions while still being categorized as the same type.
- In Task 1, class Thing is reused and inherited by Transaction and Batch class. Each class has different extra instance variables, different overwritten methods. But ultimately, both can be categorized as an ‘order’ and can be added to the ‘Thing’ list in Sales class. How Transactions and Batch class inherited Thing’s initialization and methods is an example of Polymorphism.

2.

- The name ‘Thing’ is not suitable because it doesn’t represent both classes which inherit it (Transaction and Batch). While it can represent Transaction, Batch describes a list of items, therefore ‘thing’ doesn’t do it justice.

- I think ‘Thing’ should be renamed to ‘order’ as it represents both Transaction and Batch better.

3.

- Abstraction describes the state in which a method is declared without any implementation. The method details will be specified later by the other methods which inherit it. Therefore, abstraction allows programmers to focus on what does an object do, instead of how it does it.

- In Task 1, the ‘Thing’ class is an abstract class. Which means it cannot be initialized in other classes and also can have abstract methods (methods which don’t have any implementation but depend on child classes to give them meaning). In this case, both Print and Total methods are given different overridden implementation in Transaction and Batch class. This makes the program more scalable and easier to understand.

4.

- I think every online shopping website in real life has a somewhat similar implementation of what we did in Task 1.

- These websites have user accounts that encompass all individual purchase which might include one or many items that an user buys.