# Task 2

### *Question 1: Describe the principle of polymorphism and how and where it is being used in Task 1.*

Polymorphism is the ability of a single interface (a method or property) to be used for different types of objects. It allows objects to be treated as instances of their parent class, enabling flexibility and code reuse without changing the parent class.

Polymorphism is represented through the inheritance and implementation of the Thing abstract class by its subclasses Batch and Transaction. Both Batch and Transaction classes override the abstract methods Print() and Total() from the Thing class, allowing them to provide specific implementations based on their requirement.

### *Question 2: What is wrong with the class name Thing? Suggest a better name and explain the reasoning behind your answer.*

The name "Thing" doesn't provide much context or clarity about the purpose of the class. Therefore, a name like OrderItem is more suitable since it reflects the role of the class as managing orders and the items in the orders, making the purpose of the class clearer.

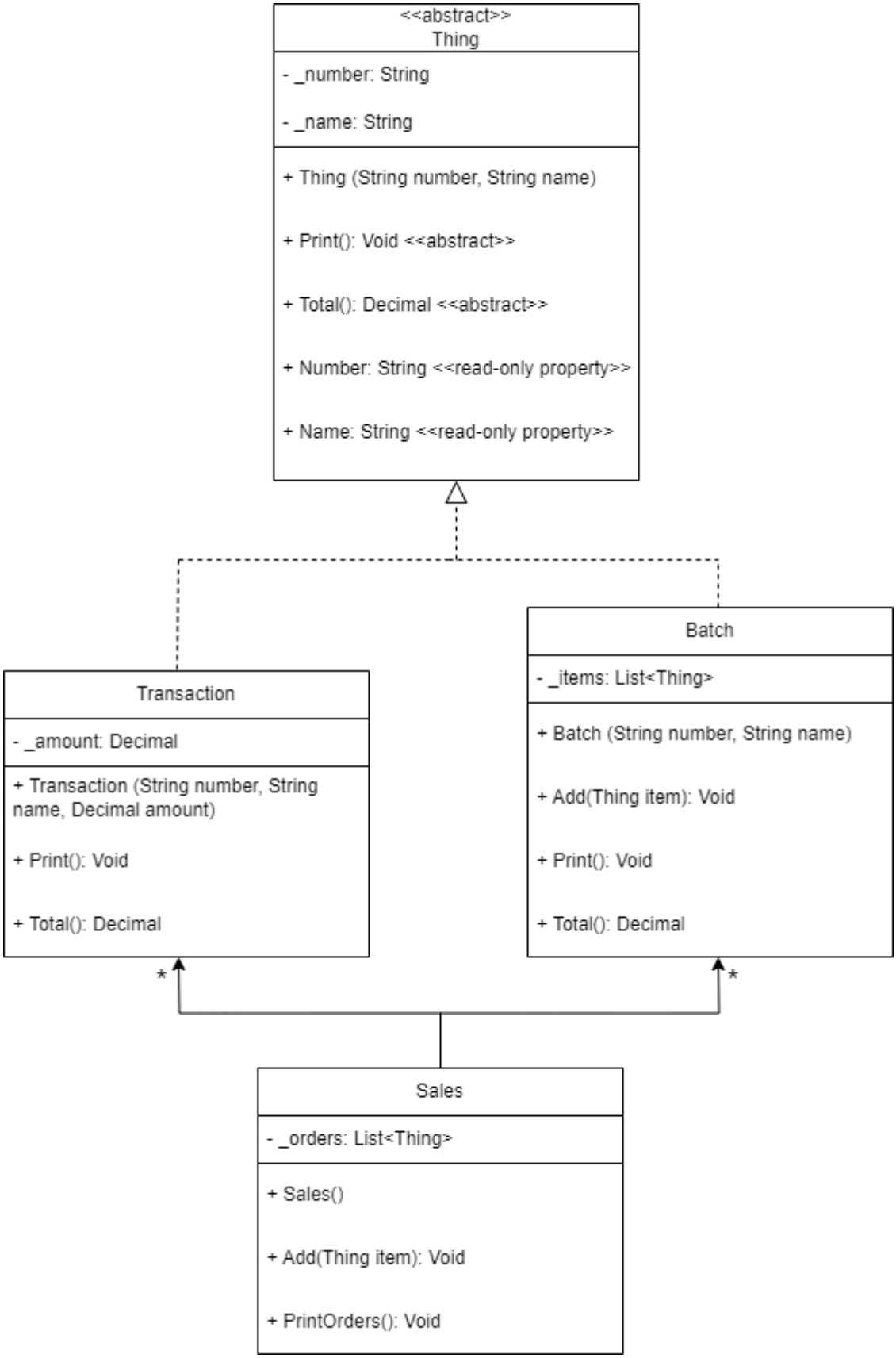
### *Question 3: What is abstraction and how and where it is being used in Task 1.*

Abstraction is a fundamental principle in OOP that involves hiding complex implementation details and focusing on functionalities or behaviors for the users to simplify complex system. It helps us build models of actual things that include necessary behaviors and offers an easy-to-use interface for working with objects.

Abstraction is used through the use of the abstract class Thing. This class defines common properties and methods that are shared by its subclasses (Batch and Transaction), while leaving the specific implementation details to these subclasses. It allows interacting with different types of objects (Batch and Transaction) without needing to know their exact private implementation.

### *Question 4: Can you think of a scenario or system design that resembles Task 1? Look at the classes and their interaction in Task 1 and identify a real-world system or approach that uses a similar relationship.*

A real-world system that has a similar relationship is an e-commerce order processing system: The Batch class can represent multi-product orders (similar to a batch contains multiple transactions, for example the Computer Science books Batch contains 3 transactions). The Transaction class can represent single-product orders. (similar to a transaction contains a book, for example the transaction #01-01 contains the book ‘Gone with the Wind’). The abstraction provided by the Thing class allows the system to handle different types of orders (multi-product orders like Computer Science books Batch and single-product order like Transaction #01-01 and Transaction #03-01), providing flexibility and scalability in managing orders. The Sales class can represent the order processing part, which generates order invoices for customers.


<<abstract>>
Thing
- _number: String
- _name: String
+ Thing (String number, String name)
+ Print(): Void <<abstract>>
+ Total(): Decimal <<abstract>>
+ Number: String <<read-only property>>
+ Name: String <<read-only property>>
Transaction
- _amount: Decimal
+ Transaction (String number, String name, Decimal amount)
+ Print(): Void
+ Total(): Decimal
Batch
- _items: List<Thing>
+ Batch (String number, String name)
+ Add(Thing item): Void
+ Print(): Void
+ Total(): Decimal
*
*
Sales
- _orders: List<Thing>
+ Sales()
+ Add(Thing item): Void
+ PrintOrders(): Void