

Task 1

1. Thing class

...amming\Code\Projects\SalesSystem\SalesSystem\Thing.cs

1

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace SalesSystem
8 {
9     public abstract class Thing
10    {
11        protected string _number;
12        protected string _name;
13
14        public Thing (string number, string name)
15        {
16            _number = number;
17            _name = name;
18        }
19
20        public abstract void Print();
21        public abstract decimal Total();
22
23        public string Number
24        {
25            get { return _number; }
26        }
27        public string Name
28        {
29            get { return _name; }
30        }
31    }
32 }
33
```

2. Transaction class

...\Code\Projects\SalesSystem\SalesSystem\Transaction.cs

1

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace SalesSystem
8 {
9     public class Transaction : Thing
10    {
11
12        private decimal _amount;
13
14        public Transaction(string number, string name, decimal amount) :
15            base(number, name)
16        {
17            _amount = amount;
18        }
19
20        public override void Print()
21        {
22            Console.WriteLine($"{_number}, {_name}, ${Total()}");
23        }
24        public override decimal Total()
25        {
26            return _amount;
27        }
28    }
29 }
```

3. Batch class

```
...aming\Code\Projects\SalesSystem\SalesSystem\Batch.cs 1
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace SalesSystem
8 {
9     public class Batch : Thing
10    {
11
12        private List<Thing> _items;
13
14        public Batch(string number, string name) : base (number, name)
15        {
16            _items = new List<Thing>();
17        }
18
19        public void Add(Thing item)
20        {
21            _items.Add(item);
22        }
23        public override void Print()
24        {
25            Console.WriteLine($"Batch sale: {Number}, {Name}");
26            if (_items.Count > 0)
27            {
28                foreach (Thing item in _items)
29                {
30                    item.Print();
31                }
32                Console.WriteLine($"    Total: ${Total()}");
33            }
34            else
35            {
36                Console.WriteLine("Empty Order.");
37            }
38        }
39
40        public override decimal Total()
41        {
42            decimal _total = 0;
43            foreach (Thing item in _items)
44            {
45                _total += item.Total();
46            }
47            return _total;
48        }
49    }
```

4. Sales class

...amming\Code\Projects\SalesSystem\SalesSystem\Sales.cs

1

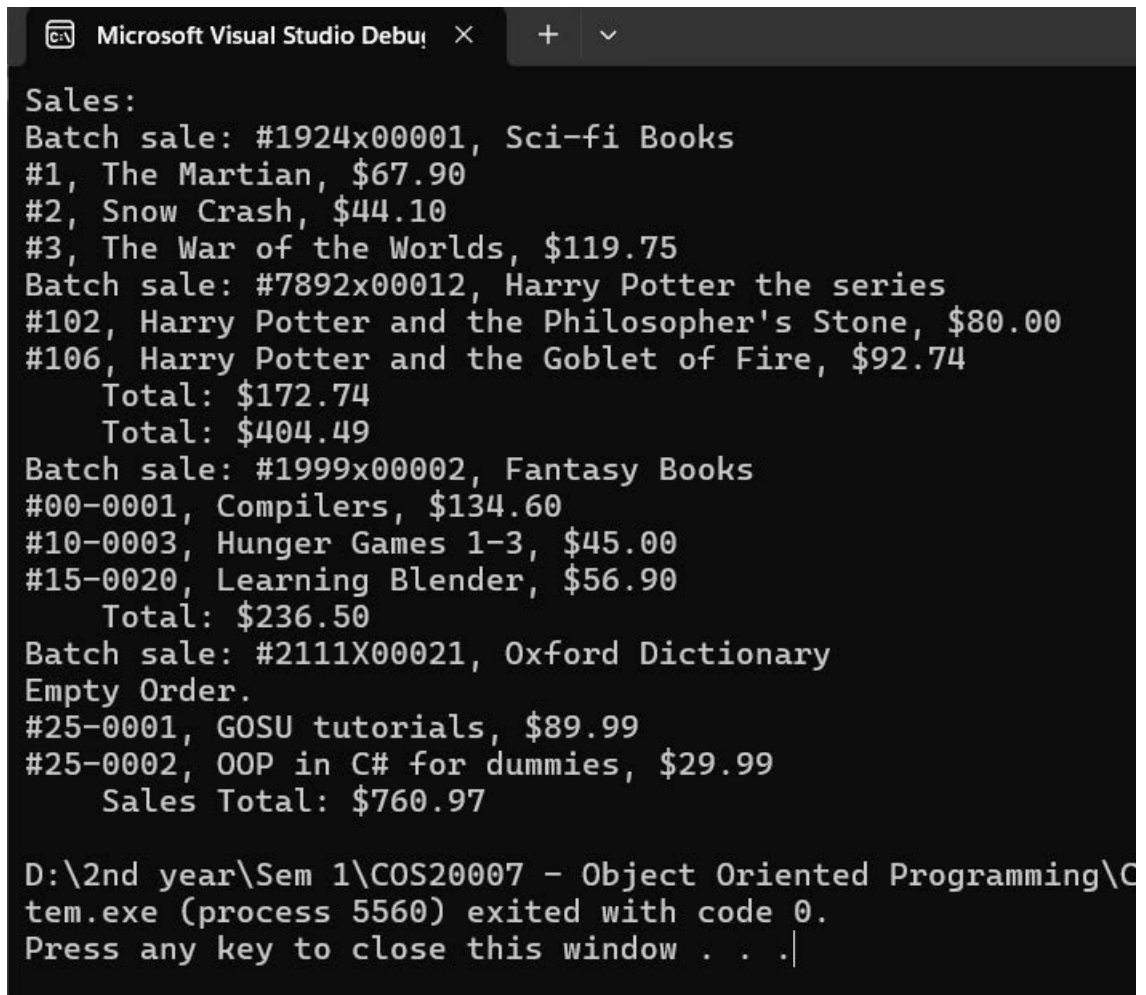
```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace SalesSystem
8 {
9     public class Sales
10    {
11        private List<Thing> _orders;
12
13
14        public Sales()
15        {
16            _orders = new List<Thing>();
17        }
18
19        public void Add(Thing order)
20        {
21            _orders.Add(order);
22        }
23
24
25        public void PrintOrders()
26        {
27            if (_orders.Count == 0)
28            {
29                Console.WriteLine("Order is empty.");
30            }
31            else
32            {
33                decimal totalSales = 0;
34
35                Console.WriteLine("Sales:");
36                foreach (Thing order in _orders)
37                {
38                    order.Print();
39                    totalSales += order.Total();
40                }
41
42                Console.WriteLine($"    Total Sales: {totalSales}");
43            }
44        }
45    }
46 }
47
```

5. Program class

```
...ming\Code\Projects\SalesSystem\SalesSystem\Program.cs 1
1 namespace SalesSystem
2 {
3     internal class Program
4     {
5         static void Main(string[] args)
6         {
7             Sales sales = new Sales();
8
9             //create 2 batch orders
10            Batch batch1 = new Batch("#1924x00001", "Sci-fi Books ");
11            batch1.Add(new Transaction("#1", "The Martian", 67.90m)); // ↗
12            // using suffix 'm' to tell the compiler this is decimal value // ↗
13            // otherwise, compiler understands 67.90 as float/double // ↗
14            batch1.Add(new Transaction("#2", "Snow Crash", 44.10m));
15            batch1.Add(new Transaction("#3", "The War of the Worlds", 119.75m)); ↗
16
17            Batch batch2 = new Batch("#1999x00002", "Fantasy Books");
18            batch2.Add(new Transaction("#00-0001", "Compilers", 134.60m));
19            batch2.Add(new Transaction("#10-0003", "Hunger Games 1-3", 45.00m)); ↗
20            batch2.Add(new Transaction("#15-0020", "Learning Blender", 56.90m)); ↗
21
22            //This batch is empty
23            Batch batch3 = new Batch("#2111X00021", "Oxford Dictionary");
24
25            //create nested batch order.
26            Batch batch4 = new Batch("#7892x00012", "Harry Potter the series"); ↗
27            batch4.Add(new Transaction("#102", "Harry Potter and the Philosopher's Stone", 80.00m)); ↗
28            batch4.Add(new Transaction("#106", "Harry Potter and the Goblet of Fire", 92.74m)); ↗
29            batch1.Add(batch4);
30
31            //create single transactions
32            Transaction transaction1 = new Transaction("#25-0001", "GOSU tutorials", 89.99m); ↗
33            Transaction transaction2 = new Transaction("#25-0002", "OOP in C# for dummies", 29.99m); ↗
34
35            //add orders to Sales
36            sales.Add(batch1);
37            sales.Add(batch2);
38            sales.Add(batch3);
```

```
39         sales.Add(transaction1);
40         sales.Add(transaction2);
41
42         //print orders
43         sales.PrintOrders();
44     }
45 }
46 }
47
```

6. Program output

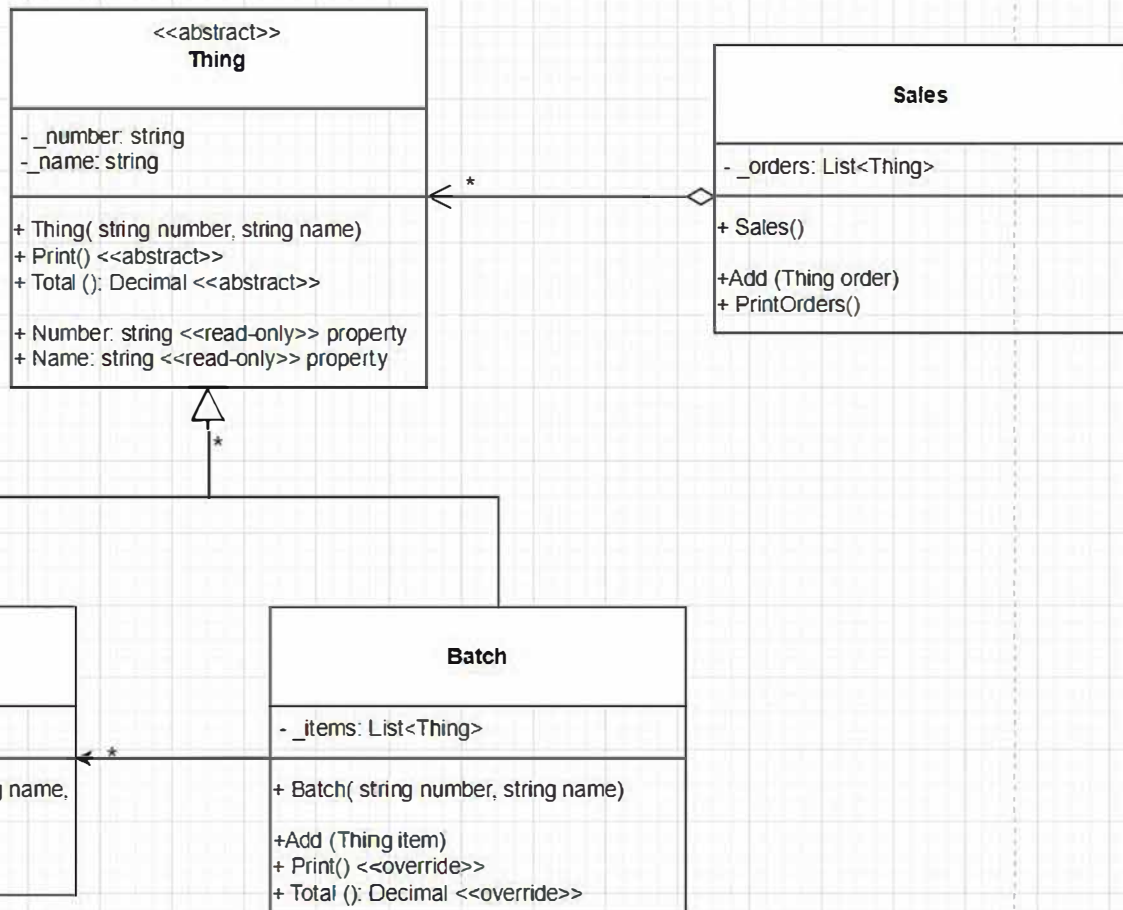


The screenshot shows the Microsoft Visual Studio Debug Console window. The title bar reads "Microsoft Visual Studio Debug Console" with a close button and window controls. The output text is as follows:

```
Sales:
Batch sale: #1924x00001, Sci-fi Books
#1, The Martian, $67.90
#2, Snow Crash, $44.10
#3, The War of the Worlds, $119.75
Batch sale: #7892x00012, Harry Potter the series
#102, Harry Potter and the Philosopher's Stone, $80.00
#106, Harry Potter and the Goblet of Fire, $92.74
    Total: $172.74
    Total: $404.49
Batch sale: #1999x00002, Fantasy Books
#00-0001, Compilers, $134.60
#10-0003, Hunger Games 1-3, $45.00
#15-0020, Learning Blender, $56.90
    Total: $236.50
Batch sale: #2111X00021, Oxford Dictionary
Empty Order.
#25-0001, GOSU tutorials, $89.99
#25-0002, OOP in C# for dummies, $29.99
    Sales Total: $760.97

D:\2nd year\Sem 1\COS20007 - Object Oriented Programming\CS
tem.exe (process 5560) exited with code 0.
Press any key to close this window . . .|
```

7. Updated UML



Task 2

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

Polymorphism, meaning "many forms," represents when different classes related by inheritance use methods to carry out different tasks. This enables a single action to be performed in many different ways according to the specialised methods in derived classes.

In task 1, polymorphism is used to treat both Batch objects and Transaction objects as Thing in the Sales class.

The **"Print"** and **"Total"** methods in the Thing, Batch, and Transaction classes are declared as abstract in the Thing class and overridden in the subclasses Batch and Transaction. This enables each subclass to offer its own implementation of these methods, crucial for printing and calculating totals tailored to each type of object.

The **"PrintOrders"** method in the Sales class iterates through a list of Thing objects and invokes their Print and Total methods. As Batch and Transaction are subclasses of Thing, they can be stored in the same list, allowing their methods to be called polymorphically. When Sales use a Thing object, it could be either a Batch or a Transaction. Polymorphism enables the order to be printed out differently depending on whether we're printing a Batch or a Transaction.

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

The name **"Thing"** is too generic in this context. If the program has more abstract class, the name Thing might make it difficult to tell which feature this group of classes represents. For example, in the Shape Drawing program, Shape class is the abstract class. By looking at the name, people can understand what this class illustrates. Therefore, to reflect better the purpose of the class, I would recommend this abstract class to be named **"SalesItem"** or **"OrderItem"**. This change makes the code more understandable to the coder. For instance, Sales class would be written to have a list of **"OrderItem"** instead of having a list of **"Thing"**.

3. What is abstraction and how and where it is being used in Task 1.

Abstraction is the process of hiding complex details and showing only essential features to the external environment. An abstract class serves as a blueprint for its derived classes, providing a basic structure without specifying the implementation of its methods.

In Task 1, abstraction is implemented through the use of the Thing class. The Thing class represents sales items, hiding specific order details like Batch and Transaction, and providing common methods like Print and Total.

The Print and Total methods in Thing are abstract, so subclasses must specialise them. This hides the specifics, letting each subclass customise its behavior. The Sales class manages

a list of Thing objects, hiding individual order details and allowing uniform interaction with different order types.

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.**

Task 1 is like a basic sales system, managing different types of orders, such as batches and single transactions. It is similar to an online store with:

- **Thing** is the listed products on the platform.
- **Batch** is the orders containing many products.
- **Transaction** is a single purchase of customers.
- The **Sales** class handles order processing, total calculation, and reporting in the system.