Державний університет «Одеська Політехніка»
Інститут комп’ютерних систем
Кафедра інформаційних систем

Лабораторна робота №4
з дисципліни «Об’єктно-орієнтоване програмування»
Тема: «Повторне використання коду. Наслідування та композиція »

Виконав:
студент групи АІ-201
Мартинюк Д.В.

Прийняв:
доц. Годовиченко М.А.

Одеса 2021

Цели лабораторной работы

- изучить принцип наследования;
- изучить принцип композиции;
- разобраться в чем отличие наследования от композиции и когда следует применять тот или иной принцип;
- научиться отображать композицию и наследование в диаграмме классов UML.

Ход работы:

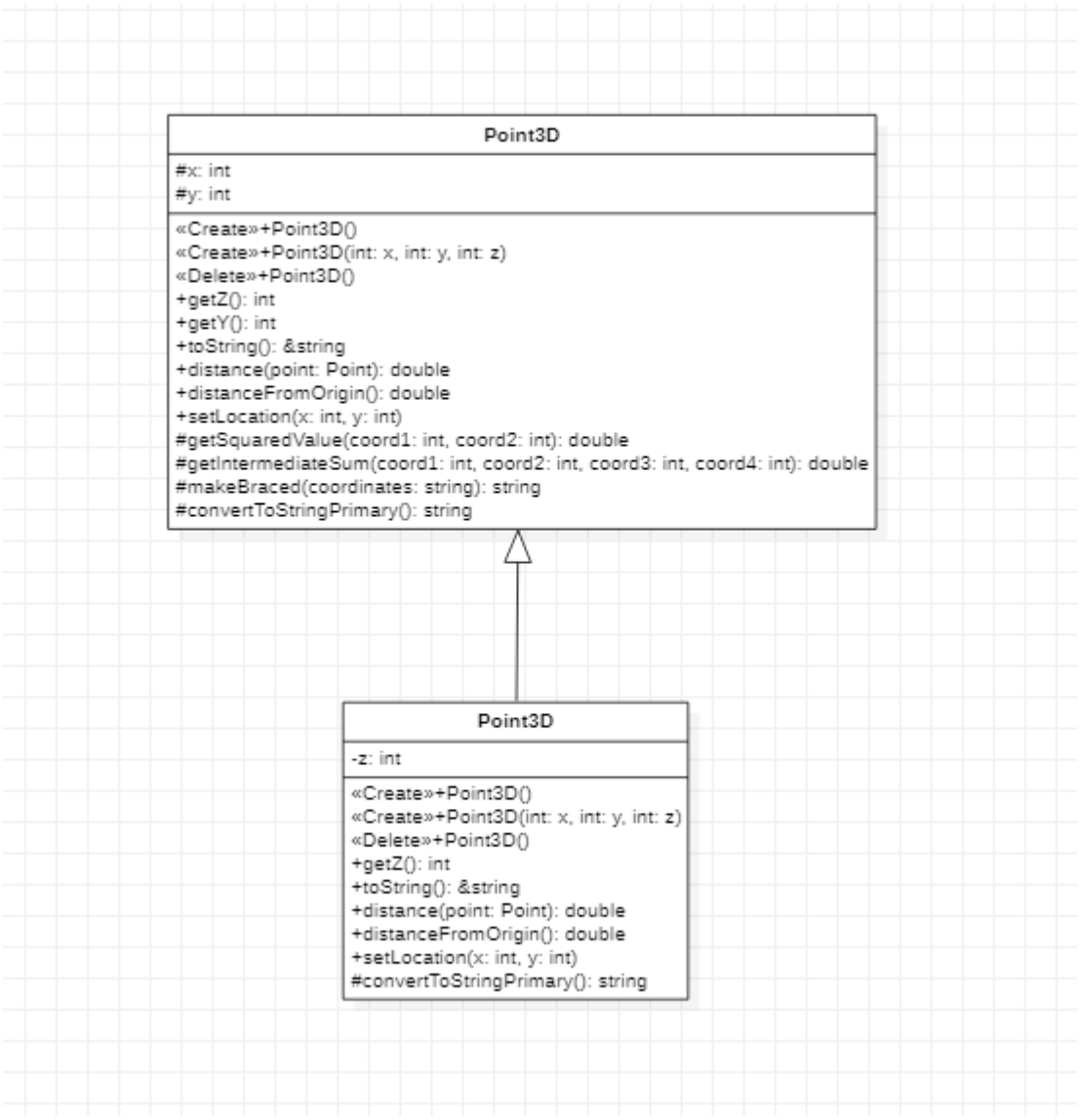
1) Создание UML-диаграмм

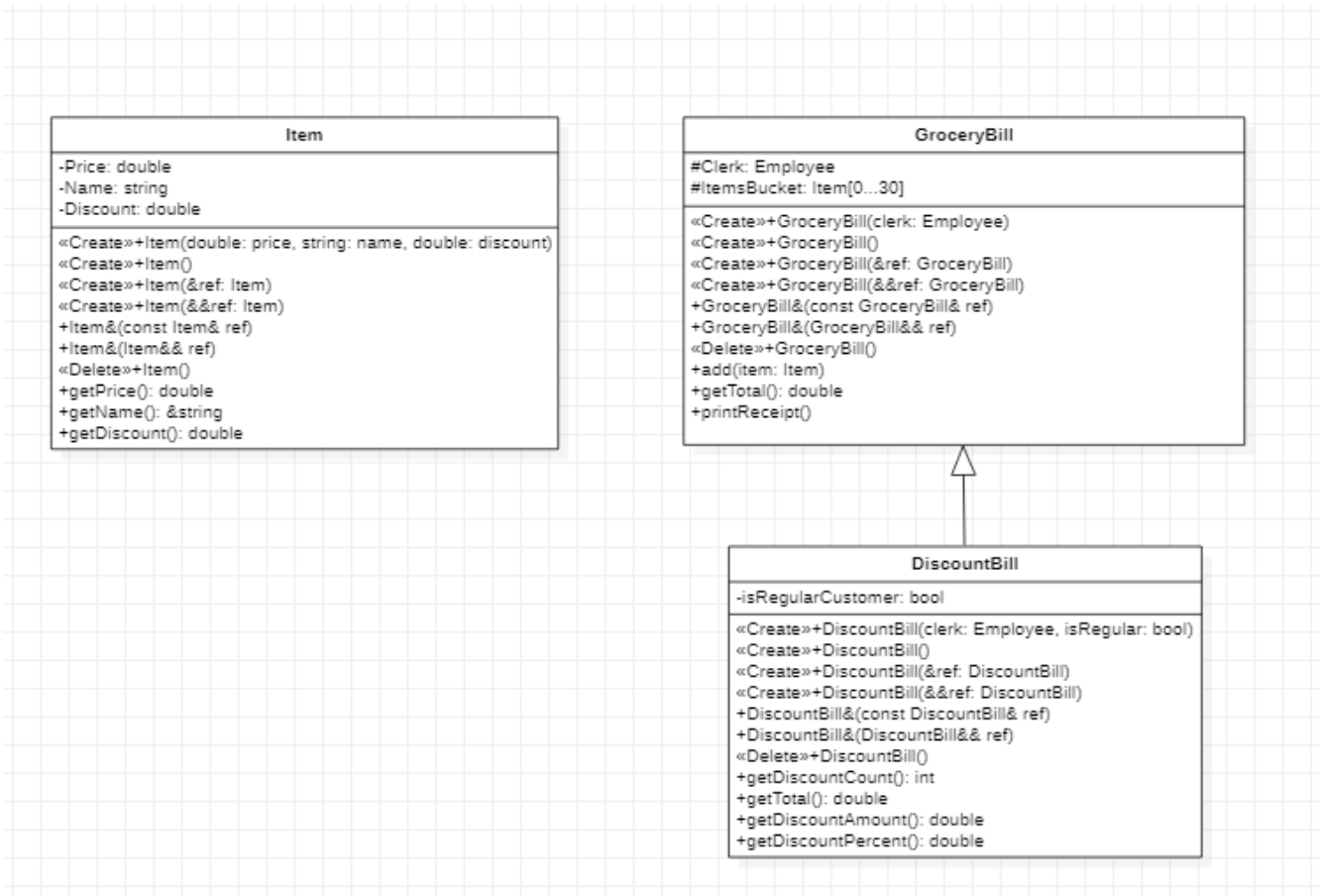
2) Класс Point3D

3) Класс DiscountBill

1. Создание UML-диаграмм

UML диаграммы к классам из лабораторной работы №2:




Item
-Price: double
-Name: string
-Discount: double
«Create»+Item(double: price, string: name, double: discount)
«Create»+Item()
«Create»+Item(&ref: Item)
«Create»+Item(&&ref: Item)
+Item&(const Item& ref)
+Item&(Item&& ref)
«Delete»+Item()
+getPrice(): double
+getName(): &string
+getDiscount(): double
GroceryBill
#Clerk: Employee
#ItemsBucket: Item[0...30]
«Create»+GroceryBill(clerk: Employee)
«Create»+GroceryBill()
«Create»+GroceryBill(&ref: GroceryBill)
«Create»+GroceryBill(&&ref: GroceryBill)
+GroceryBill&(const GroceryBill& ref)
+GroceryBill&(GroceryBill&& ref)
«Delete»+GroceryBill()
+add(item: Item)
+getTotal(): double
+printReceipt()
DiscountBill
-isRegularCustomer: bool
«Create»+DiscountBill(clerk: Employee, isRegular: bool)
«Create»+DiscountBill()
«Create»+DiscountBill(&ref: DiscountBill)
«Create»+DiscountBill(&&ref: DiscountBill)
+DiscountBill&(const DiscountBill& ref)
+DiscountBill&(DiscountBill&& ref)
«Delete»+DiscountBill()
+getDiscountCount(): int
+getTotal(): double
+getDiscountAmount(): double
+getDiscountPercent(): double

2. Класс Point3D

При реализации данного класса было решено помимо необходимых методов добавить вспомогательные-промежуточные для гибкости работы с классом,

данные методы являются защищенными, что не дает получить сырые промежуточные данные.

К таким методам относится **getSquaredValue()** которые возращает как результат работы часть определенного выражения из формулы для поиска расстояния(часть является шаблонной, поэтому ее можно выразить в функцию для дальнейших рассчетов для любой из точек):

```
const double getSquaredValue(int coord1, int coord2)
{
	return  (pow(coord1 - coord2, 2));
}
```

Этот же метод в дальнейшем используется для нахождения расстояния между точками в декартовой системе координат на плоскости(только X,У).

1) **getIntermediateSum()**

Данный метод тоже является вспомогательно-промежуточным. В реализации для класса Point он выполняет подстановку переданных ему параметров и вызывает 2 раза функцию с подсчетом частей уравнения(для X и для Y);

Таким образом мы можем до бесконечности наследоваться добавляя какие нибудь координаты и в переопределенном методе вызывать **getIntermediateSum() + необходимые параметры .**

```
virtual const double getIntermediateSum(int coord1, int coord2, int coord3, int coord4)
{
	return getSquaredValue(coord1, coord2) + getSquaredValue(coord3, coord4);
}
```

Также добавлены вспомогательные методы для форматированного вывода координат, они добавлены для гибкости при дальнейшем появлении новых осей координат:

```
// Returns coordinates in braced format
string makeBraced(string coordinates)
{
	return "(" + coordinates + ")";
}



// Intermediate converting to string, made for comfortable converting in derived classes
virtual const string convertToStringPrimary()
{
	return to_string(x) + "," + to_string(y);
}
```

Как результат мы избавляемся от большого количества переписываемого кода просто вызывая методы базового класса добавляя к ним необходимые изменения:

```
// Calculate distances between points/from origin
const double distance(const Point3D point)
{

	return sqrt(Point::getIntermediateSum(this->x, point.getX(), this->y, point.getY()) + getSquaredValue(this->z, point.getZ()));

}
```

3. Класс DiscountBill

Метод **getTotal()** в классе DiscountBill

Так как класс предок для DiscountBill не имел учета скидок, мы этим воспользуемся, метод вернет сумму покупок в чеке с учетом скидок только в случае если покупатель является постоянным, в противном случае мы вызовем метод предка и просто узнаем сумму покупок.

```
// Get total price of all items in the bill, but taking into account the factor that customer is regular
	const double getTotal()const override
	{

		if (this->isRegularCustomer)
		{
			return GroceryBill::getTotal() - this->getDiscountAmount();
		}

		return GroceryBill::getTotal();

	}
```

Выводы к лабораторной работе:

В ходе выполнения лабораторной работы было рассмотрено и использовано на практике принципы композиции и наследования. Смоделировано UML диаграммы классов с учетом использования принципов.