Composition

Item : Represents a product
ShoppingCart : keeps track of a list of item

```
|-----------------------------------
|          Item
|-----------------------------------
| - itemName        :  String
| - itemPrice       :  double
| - itemQuantity    :  int
|------------------------------------
|   Item()
|   Item(String name, double price, int quantity)
|   Item(Item obj)
| + public setName(String name):void
| + public setPrice(double price):void
| + public setQuantity(int quantity):void
| + public getName():String
| + public getPrice():double
| + public getQuantity():int
| + public printInfo():void
| + public toString():String
| + public printItemCost():void
|-----------------------------------
```

```
          ShoppingCart
 -   cartItems   : ArrayList<Item>  // Using composition
 -   customerName : String
 -   currentDate : String

+  ShoppingCart()
+  ShoppintCart(String name, String date)
+  getName(): String
+  getDate(): String
+  getNumOfItems(): int
+  removeItem(Item): void
+  addItem(Item): void
+  modifyItem(Item) : void
+  getCost(): double
+  setName(String): void
+  setDate(String): void
+  printBill(): void
```

GoShopping:

+ show menu(): add item, remove item , print cost, exit
+ readItem()
+ main()

- shop: ShoppingCart   // composition
      shop.PublicMethod()

Loop through menu and stop when done

Address
-name: String

+ getName():String
+ setName(String):void
Address(Address obj){
    name = obj.name;
// name = obj.getName();
}

Person
- name: String - address: Address
+ getName():String + setName(String):void + getAddress(): Address + setAddress(Address t): void void setAddress(Address t) (copy constructor) { address = new Address(t);} + setStName(String temp){   address.setName(temp);}

Composition (Using Address class)
Indirect access must use an object

Address t1 = new Address("Torrance Blvd."); t1 = DF453 → Torrance Blvd. Main St.

Person   p1= new Person();   -→ name => Joe Smith   address => DFA53 → Torrance Blvd.

p1.setName("Joe Smith");

p1.setAddress(t1);

t1.setName("Main St."); → p1.setAddress(t1);

p1.setStName("Main St.);

Address
-name: String
+ getName():String + setName(String):void

Person
- name: String - address: Address
+ getName():String + setName(String):void + getAddress(): Address + setAddress(Address): void

Student extends Person
- major: String
+ getName():String + setName(String):void + getAddress(): Address + setAddress(Address):void + getMajor():String + setMajor(String):void

Composition (Using Address class)
Indirect access must use an object

Inheritance
Inheriting public and protected members and have direct access to them
Student gets all available members of person and can use them directly

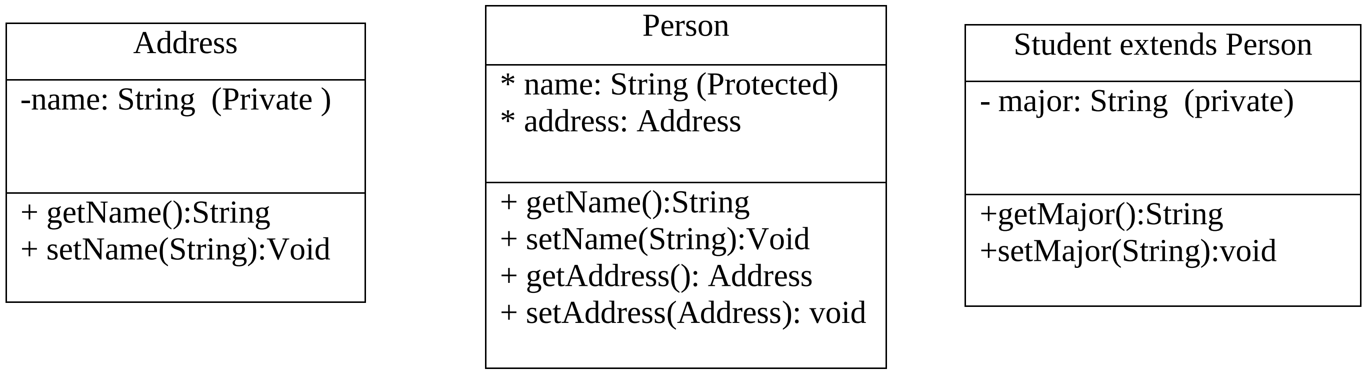
Student   s1 = new Student(); name → Joe   address → Main St,

Address = new Address(
“Main St”);

s1.setName(“Joe”);

s1.setAddress(t1);

Public	+	
Private	–	
protected	*	-- Extended classes are able to access protected elements



Base or Parent : refers to the original class
Derived or Child : refers to the class that inherited

```
Person p1 = new Person();
p1.setName("Joe");
Student s1 = new Student() ;
s1.setName("Alan");
s1.name /////  NO
```