类图基本符号可拆分为虚线，箭头，实线，空心右三角，实心右三角，空心菱形和实心菱形。由这些基本的图形进行组合构成了类图的基本符号。这里要注意这几个符号的顺序，代表了类与类之间关系的耦合程度。越向右耦合度越高。

其中虚线+箭头是表示即依赖的关系,实线+箭头表示关联的关系，虚线+空心右三角表示 implements,实线+空心右三角表示的是泛化，即类的继承关系。实线+空心菱形表示的是聚合的关系，实线+实心菱形则表示组合的关系。

另外一点是在看类图的时候要注意。类图的思想其实也还没有脱离面向对象的思想，以某个类为中心，有些线是射入的而有些线是射出的。射入的线表示的是这个类被哪些类所调用而射出的线则表示该类调用了哪些类，包括泛化，关联，依赖，聚合和组合四种关系。这类似于离散数学中有关图部分的描述。

1. 类（Class）：使用三层矩形框表示。

第一层显示类的名称，如果是抽象类，则就用斜体显示。

第二层是字段和属性。

第三层是类的方法。

注意前面的符号，‘+’表示 public，‘-’表示 private，‘#’表示 protected。

2. 接口：使用两层矩形框表示，与类图的区别主要是顶端有<<interface>>显示 。

第一行是接口名称。

第二行是接口方法。

3. 继承类（extends） ：用空心三角形+实线来表示。

4. 实现接口（implements） ：用空心三角形+虚线来表示

5. 关联（Association） ：用实线箭头来表示，例如：燕子与气候

6. 聚合（Aggregation） ：用空心的菱形+实线箭头来表示

聚合：表示一种弱的‘拥有’关系，体现的是 A 对象可以包含 B 对象，但 B 对象不是 A 对象的一部分，例如：公司和员工

组合（Composition）：用实心的菱形+实线箭头来表示

组合：部分和整体的关系，并且生命周期是相同的。例如：人与手

7. 依赖（Dependency） ：用虚线箭头来表示，例如：动物与氧气

8. 基数 ：连线两端的数字表明这一端的类可以有几个实例，比如：一个鸟应该

有两只翅膀。如果一个类可能有无数个实例，则就用‘n’来表示。关联、聚合、组合是有基数的。

类之间的关系

UML 把类之间的关系分为以下 5 种.

- 关联：类 A 与类 B 的实例之间存在特定的对应关系
- 依赖：类 A 访问类 B 提供的服务
- 聚集：类 A 为整体类，类 B 为局部类，类 A 的对象由类 B 的对象组合而成
- 泛化：类 A 继承类 B
- 实现：类 A 实现了 B 接口

关联（Association）

关联指的是类之间的特定对应关系，在 UML 中用带实线的箭头表示。按照类之间的数量对比，关联

可以分为以下三种：

- 一对一关联 - 一对多关联 - 多对多关联

注意：关联还要以分为单向关联和双向关联

依赖（Dependency）

依赖指的是类之间的调用关系，在 UML 中用带虚线的箭头表示。如果类 A 访问类 B 的属性或者方法，

或者类 A 负责实例化类 B，那么可以说类 A 依赖类 B。和关联关系不同，无须在类 A 中定义类 B 类型的属性。

聚集（Aggregation）

聚集指的是整体与部分之间的关系，在 UML 中用带实线的菱形箭头表示。

聚集关系还可以分为两种类型：

- 被聚集的子系统允许被拆卸和替换，这是普通聚集关系。
- 被聚集的子系统不允许被拆卸和替换，这种聚集称为强聚集关系，或者组成关系。

注：强聚集（组成）可用带实线的实心菱形箭头表示。

泛化（Generalization）

泛化指的是类之间的继承关系，在 UML 中用带实线的三角形箭头表示。

实现（Realization）

实现指的是类与接口之间的关系，在 UML 中用带虚线的三角形箭头表示。

以下是 GOF 设计模式中的描述：

箭头和三角表示子类关系。

虚箭头线表示一个类实例化另一个类的对象，箭头指向被实例化的对象的类。

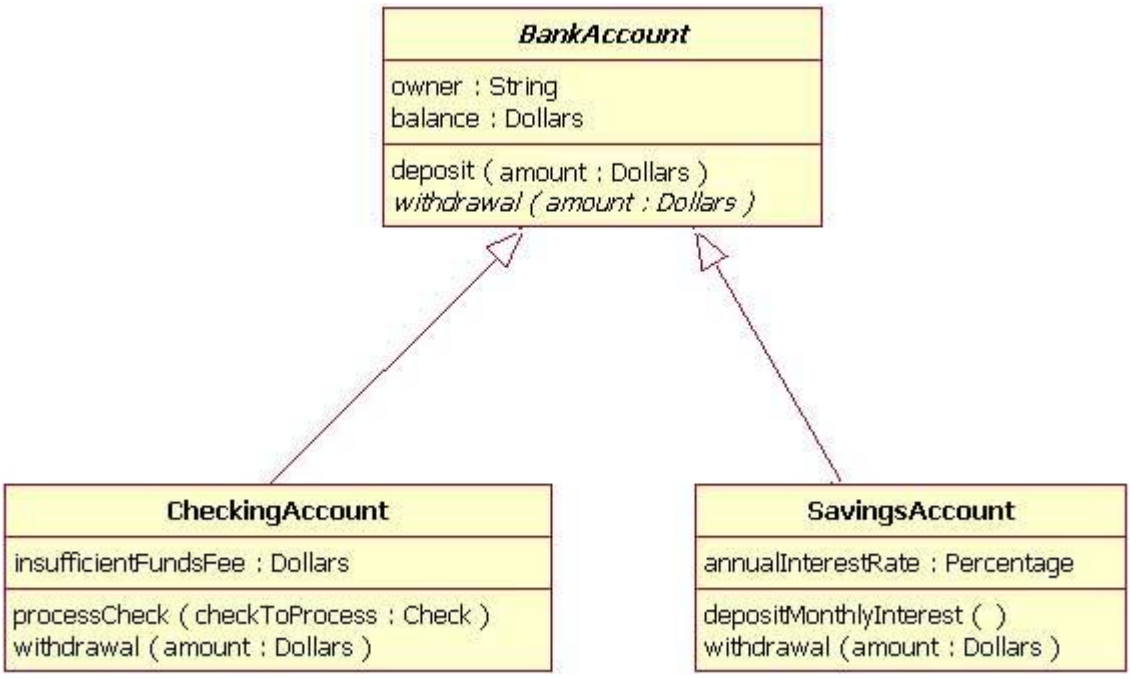
普通的箭头线表示相识（acquaintance 也叫关联或者引用），意味着一个对象仅仅知道另一个对象。相识的对象可能请求彼此的操作，但他们不为对方负责，它只标示了对象间较松散的耦合关系。

尾部带有菱形的箭头线表示聚合（aggregation），意味着一个对象拥有另一个对象或者对另一个对象负责。一般我们称一个对象包含另一个对象，或者是另一个对象的一部分。聚合意味着聚合对象和其所有者具有相同的生命周期。

抽象类名以斜体表示，抽象操作也以斜体表示。图中可以包括实现操作的

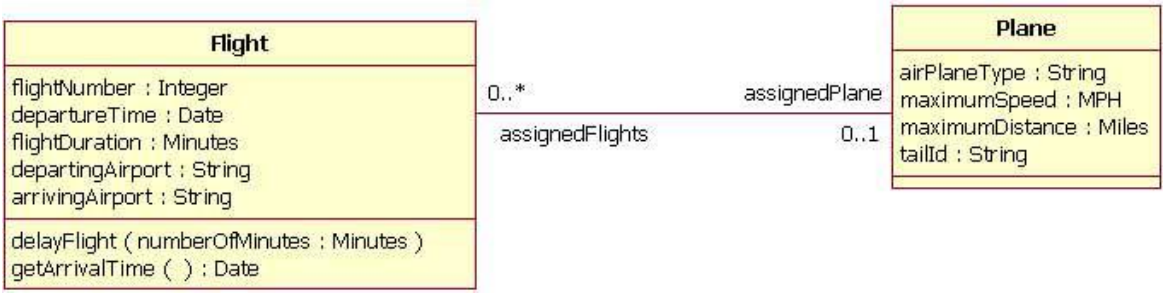
伪代码，代码将出现在带有褶角的框中，并用虚线将该褶角框与代码所实现的操作相连。

图一:



此实线箭头表示，继承，从一个非接口类的继承.

图二:

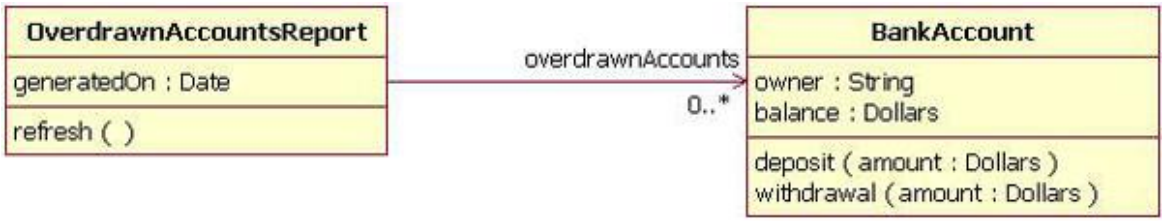


那条连线表示双向关联:

看左边, Flight 扮演 assignedFights 角色，有 0 到 1 个 Plane 跟他关联(一个航班要么取消了没有飞机,要么只能对应一架飞机)

看右边, Plane 扮演着 assignedPlane 角色，有 0 到多个 Flight 跟他关联(一个飞机可以参与多个航班，也可以停在仓库里面烂掉)

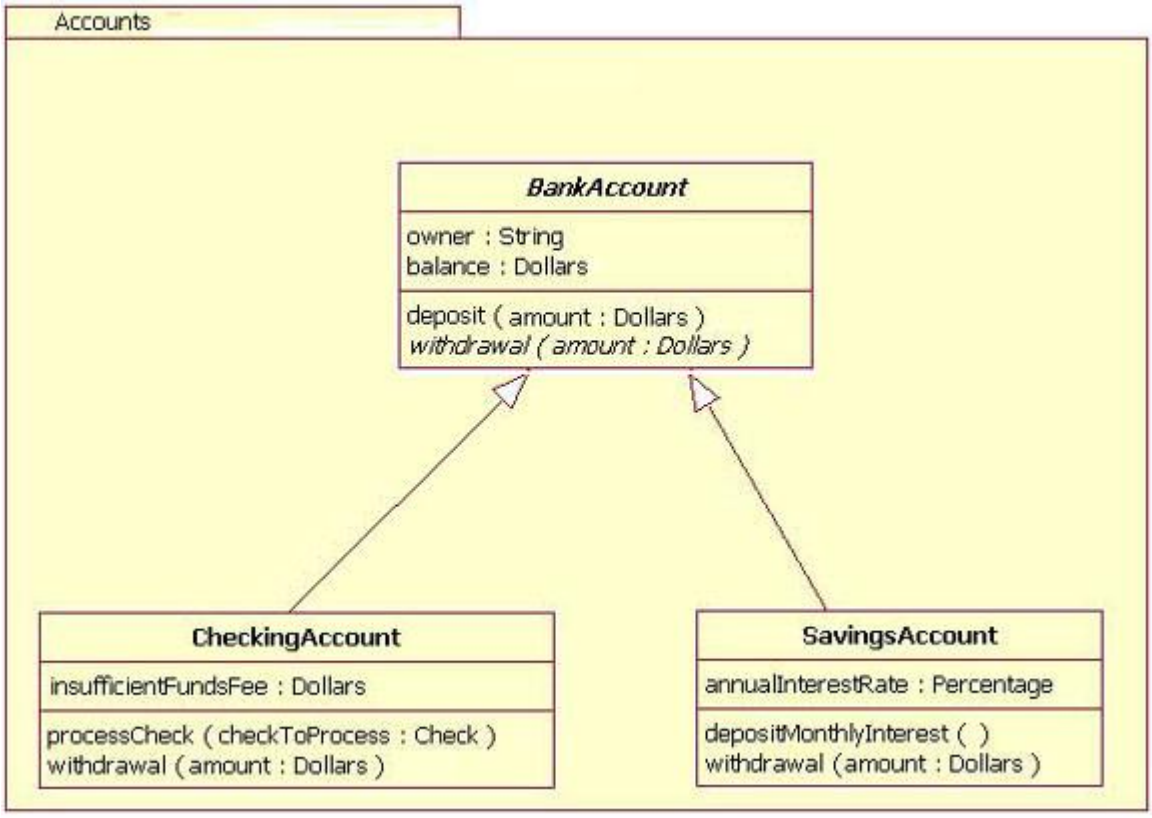
图三:



那条连线表示单向关联:

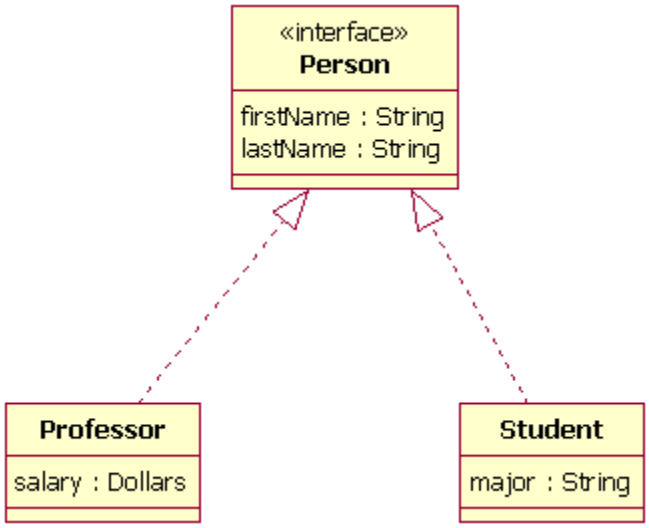
基本的意义跟上面的是一样的，唯一不同的是，右边的类对左边的类是一无所知的.

图四:



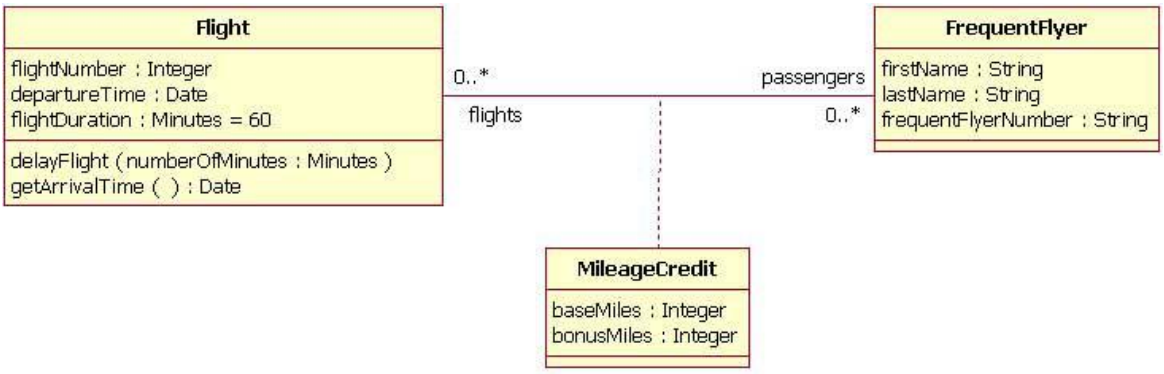
那个大的包围的框叫软件包，名字为 Account，就一些可以归类的类包装起来.

图五:



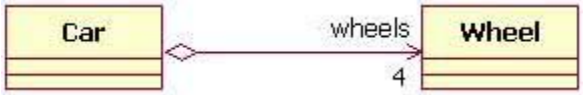
如此虚线的箭头表示实现一个接口.

图六:



水平的连线还是表示上面所说的关联，但从关联连线中引伸出来的虚线，这意味当 Flight 类的一个实例关联到 FrequentFlyer 类的一个实例时，将会产生 MileageCredit 类的一个实例.

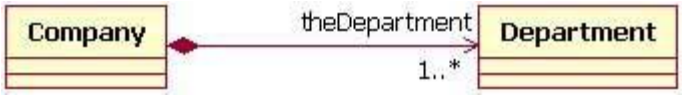
图七:



带菱形的箭头表示基本聚合，由上图知道, Wheel 类扮演 wheels 角色，聚合 4 个到 Car 对象里面去,

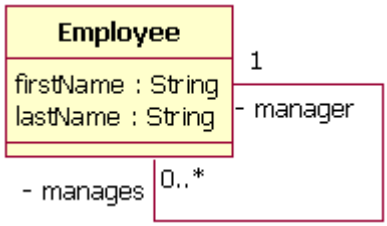
空心的菱形表示 Wheel 对象并不随 Car 的创建而创建,销毁而销毁.

图八:



意义和上面类似，唯一不同的是，实心菱形表示 Department 对象随 Company 对象的创建而创建,销毁而销毁.

图九:



表示反射关联，显示一个 Employee 类如何通过 manager / manages 角色与它本身相关。当一个类关联到它本身时，这并不意味着类的实例与它本身相关，而是类的一个实例与类的另一个实例相关。