

UML for Java Developers

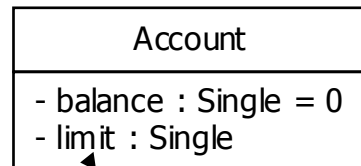
Class Diagrams

Classes

Account

```
class Account  
{  
}
```

Attributes



```
class Account
{
    private float balance = 0;
    private float limit;
}
```

[visibility] [/] attribute_name[multiplicity] [: type [= default_value]]

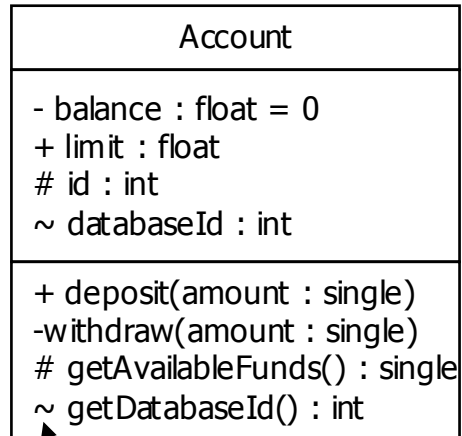
Operations

Account
- balance : Single = 0 - limit : Single
+ deposit(amount : Single) + withdraw(amount : Single)

[visibility] op_name([[in|out] parameter : type[, more params]])[: return_type]

```
class Account
{
    private float balance = 0;
    private float limit;
    public void deposit(float amount)
    {
        balance = balance + amount;
    }

    public void withdraw(float amount)
    {
        balance = balance - amount;
    }
}
```



+ = public
 - = private
 # = protected
 ~ = package

Visibility

```

class Account
{
    private float balance = 0;
    public float limit;
    protected int id;
    int databaseId;

    public void deposit(float amount)
    {
        balance = balance + amount;
    }

    private void withdraw(float amount)
    {
        balance = balance - amount;
    }

    protected int getId()
    {
        return id;
    }

    int getDatabaseId()
    {
        return databaseId;
    }
}
  
```

```
int noOfPeople = Person.getNumberO fPeople();
Person p = Person.createPerson("Jason Gorman");
```

Person
- <u>numberOfPeople</u> : int - name : string
+ <u>createPerson(name : string)</u> : Person + getName() : string + <u>getNumberOfPeople()</u> : int - Person(name : string)

Class & Instance Scope

```
class Person
{
    private static int numberOfPeople = 0;
    private String name;

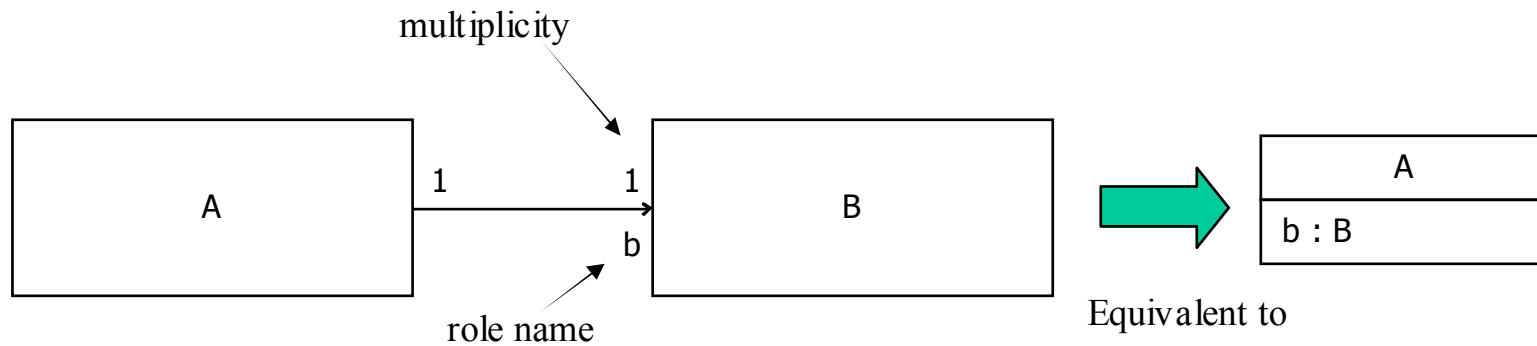
    private Person(string name)
    {
        this.name = name;
        numberOfPeople++;
    }

    public static Person createPerson(string name)
    {
        return new Person(name);
    }

    public string getName()
    {
        return this.name;
    }

    public static int getNumberOfPeople()
    {
        return numberOfPeople;
    }
}
```

Associations

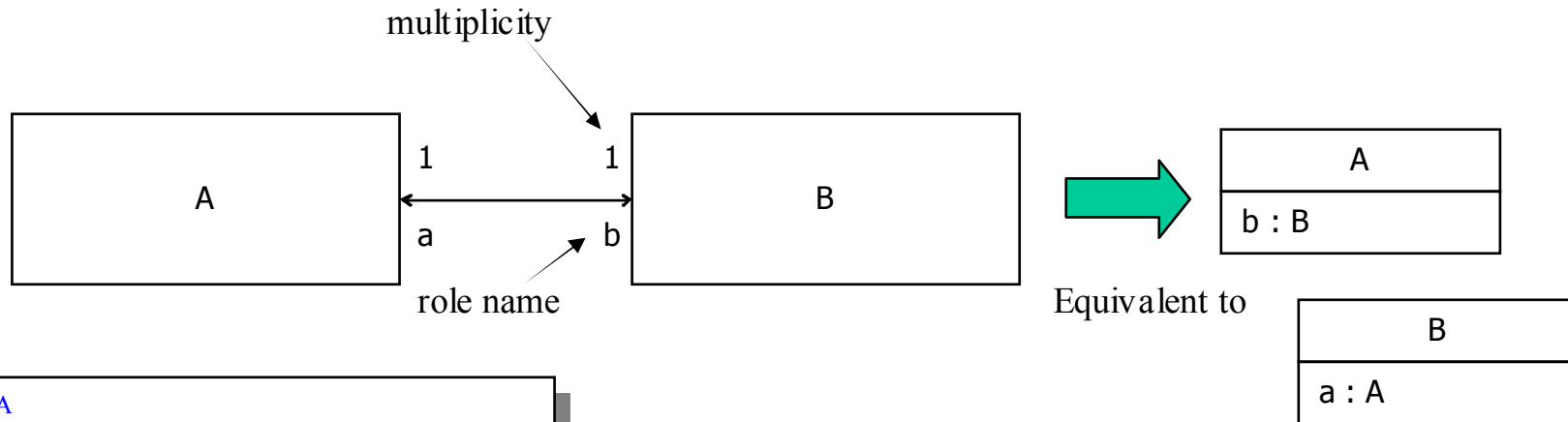


```
class A
{
    public B b = new B();
}

class B
{
}
```

```
A a = new A();
B b = a.b;
```

Bi-directional Associations



```

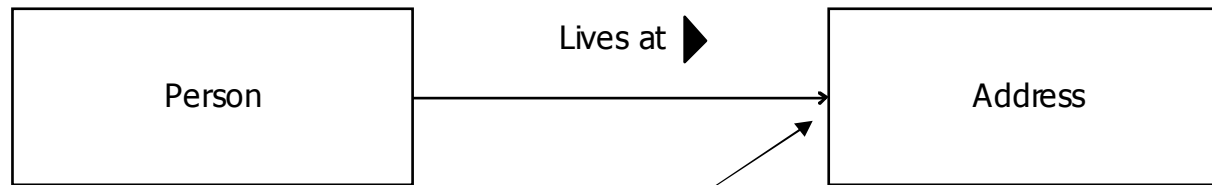
class A
{
    public B b;
    public A()
    {
        b = new B(this);
    }
}

class B
{
    public A a;
    public B(A a)
    {
        this.a = a;
    }
}
    
```

```

A a = new A();
B b = a.b;
A a1 = b.a;
assert a == a1;
    
```

Association names & role defaults

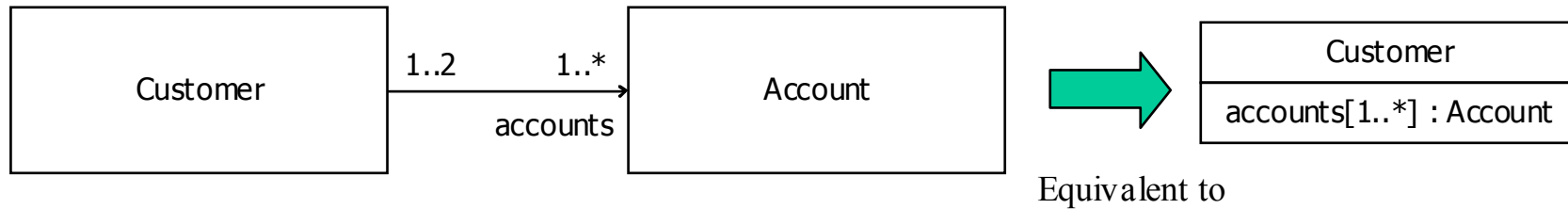


Default role name = address
Default multiplicity = 1

```
class Person
{
    // association: Lives at
    public Address address;

    public Person(Address address)
    {
        this.address = address;
    }
}
```

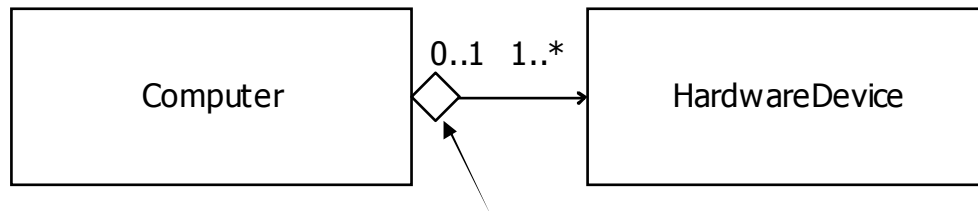
Multiplicity & Collections



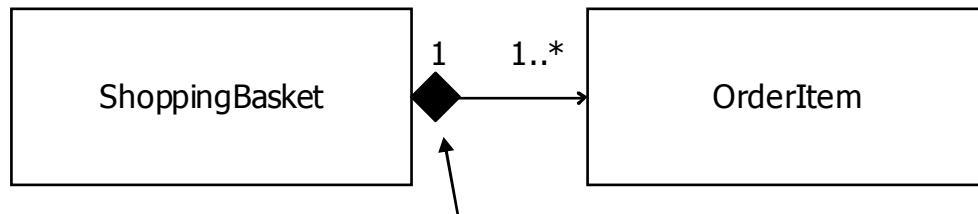
```
class Customer
{
    // accounts[1..*] : Account
    ArrayList accounts = new ArrayList();

    public Customer()
    {
        Account defaultAccount = new Account();
        accounts.add(defaultAccount);
    }
}
```

Aggregation & Composition

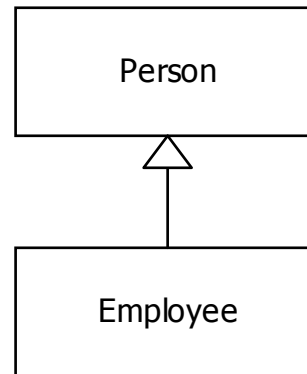


Aggregation – is made up of objects that can be shared or exchanged



Composition – is composed of objects that cannot be shared or exchanged and live only as long as the composite object

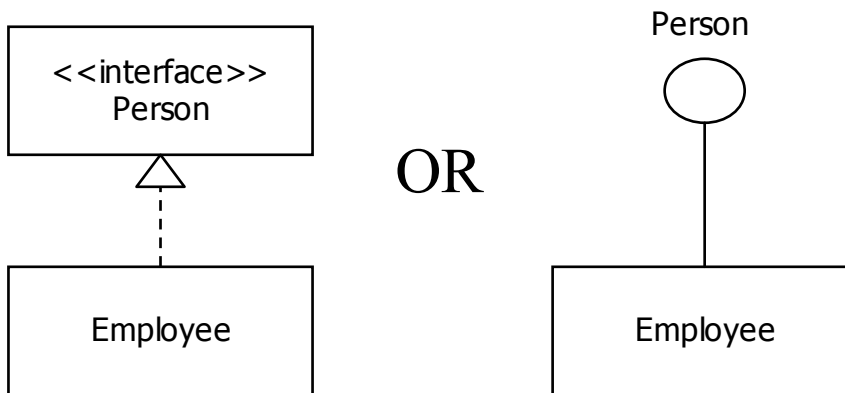
Generalization



```
class Person
{
}

class Employee extends Person
{
}
```

Realization



```
interface Person
{
}

class Employee implements Person
{
}
```

Coming soon – the complete UML for Java premium package

<http://www.parlezuml.com/tutorials/umlforjava.htm>