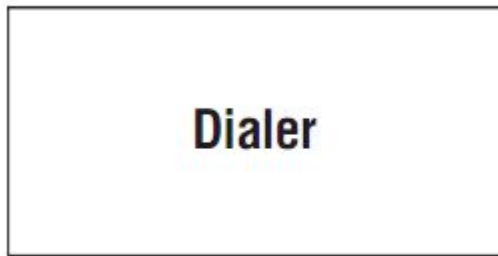


Диаграммы классов

Основные понятия

- Классы



```
public class Dialer  
{  
}
```

Значок класса

Dialer
- digits : ArrayList - nDigits : int
+ Digit(n : int) # RecordDigit(n : int) : bool

```
public class Dialer
{
    private ArrayList digits;
    private int nDigits;
    public void Digit(int n);
    protected bool RecordDigit(int n);
}
```

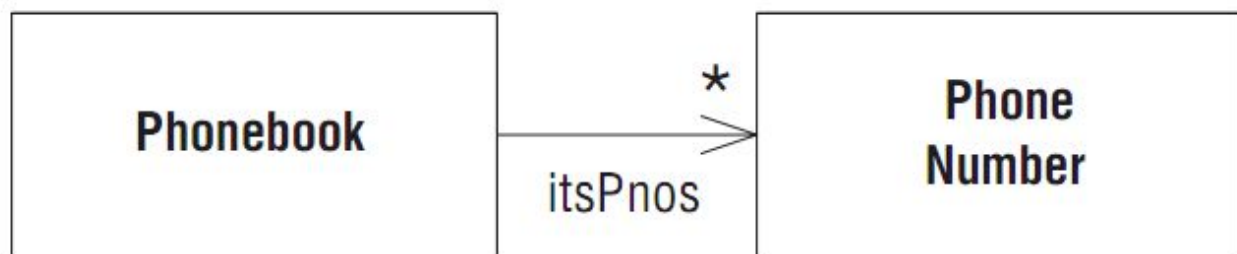
Значок класса с отделениями и соответствующий код

Ассоциация



Ассоциация

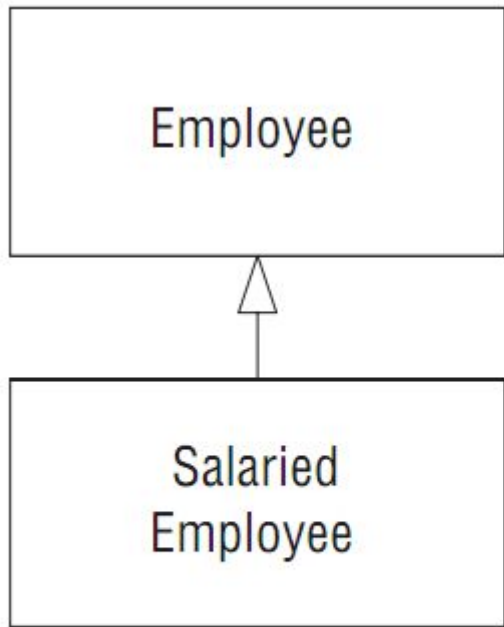
```
public class Phone
{
    private Button
    itsButtons[15];
}
```



```
public class Phonebook
{
    private ArrayList itsPnos;
}
```

Ассоциация один-ко-многим

Наследование



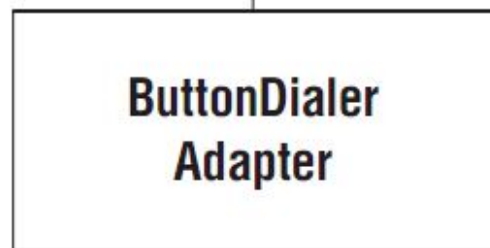
```
public class Employee
{
    ...
}
```

```
public class SalariedEmployee : Employee
{
    ...
}
```

Наследование



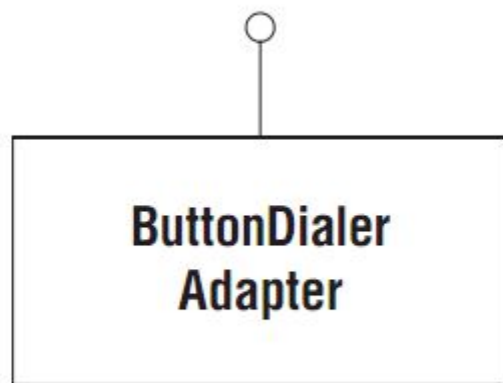
```
interface ButtonListener  
{  
    ...  
}
```



```
public class ButtonDialerAdapter  
    : ButtonListener  
{  
    ...  
}
```

Отношение «реализует»

ButtonListener



Обозначение интерфейса «леденцом на палочке»

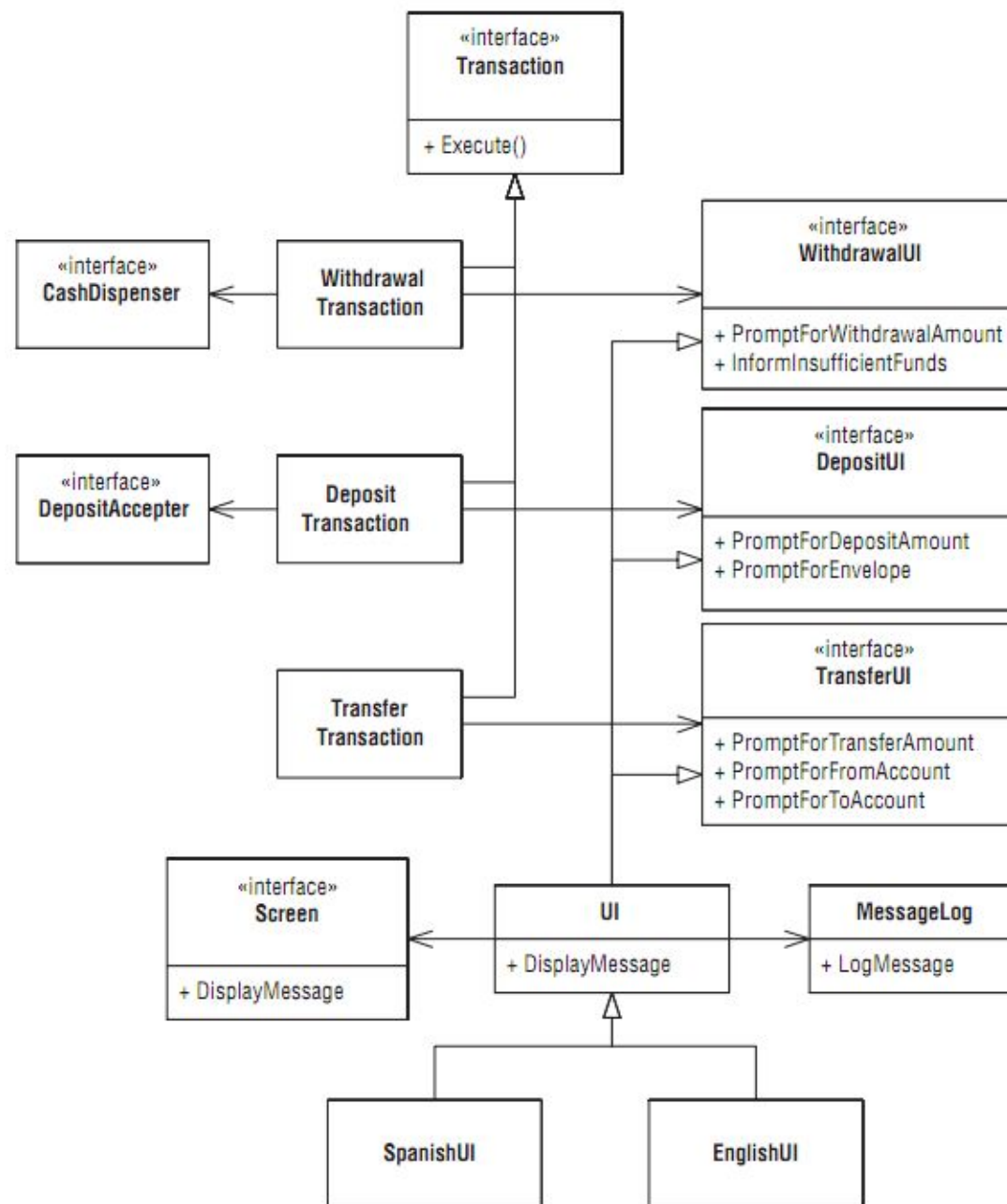
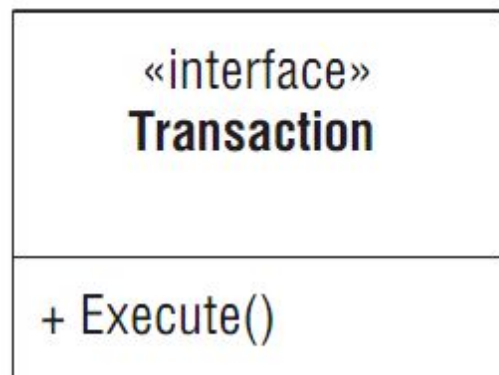


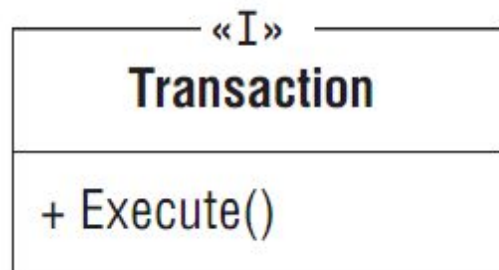
Диаграмма классов банкомата

```
public abstract class UI : WithdrawalUI, DepositUI, TransferUI
{
    private Screen itsScreen;
    private MessageLog itsMessageLog;
    public abstract void PromptForDepositAmount();
    public abstract void PromptForWithdrawalAmount();
    public abstract void InformInsufficientFunds();
    public abstract void PromptForEnvelope();
    public abstract void PromptForTransferAmount();
    public abstract void PromptForFromAccount();
    public abstract void PromptForToAccount();
    public void DisplayMessage(string message)
    {
        itsMessageLog.LogMessage(message);
        itsScreen.DisplayMessage(message);
    }
}
```

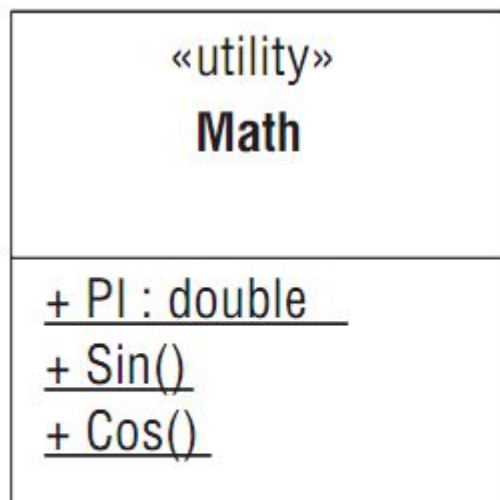
Детали: Стереотипы классов



```
interface Transaction  
{  
    public void Execute();  
}
```



Стереотип класса «interface»



```
public class Math
{
    public static readonly double PI =
        3.14159265358979323;
    public static double Sin(double theta){...}
    public static double Cos(double theta){...}
}
```

Стереотип класса «utility»

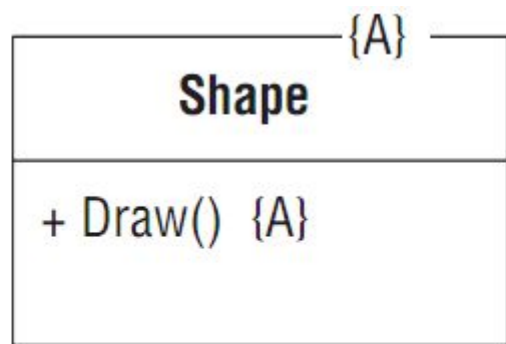
Абстрактные классы

Shape
- itsAnchorPoint
+ Draw()

Shape {abstract}
- itsAnchorPoint
+ Draw() {abstract}

```
public abstract class Shape
{
    private Point itsAnchorPoint;
    public abstract void Draw();
}
```

Абстрактные классы



Неофициальное обозначение абстрактных классов

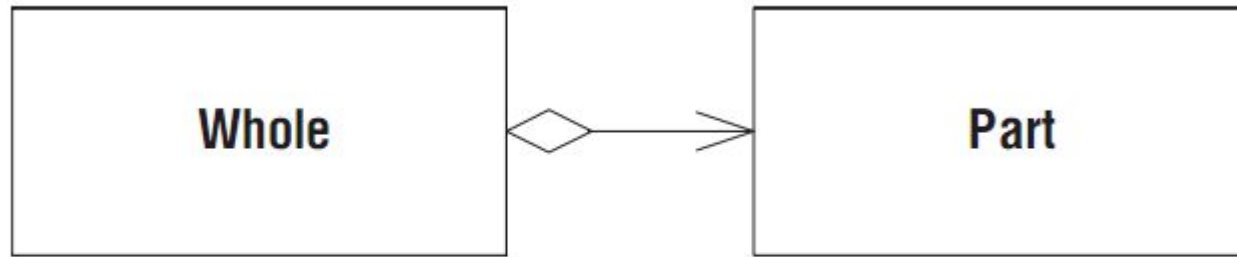
Свойства

- {author=Martin, date=20020429, file=shape.cs, private}

Shape
{author=Martin, date=20020429, file=shape.cs, private}

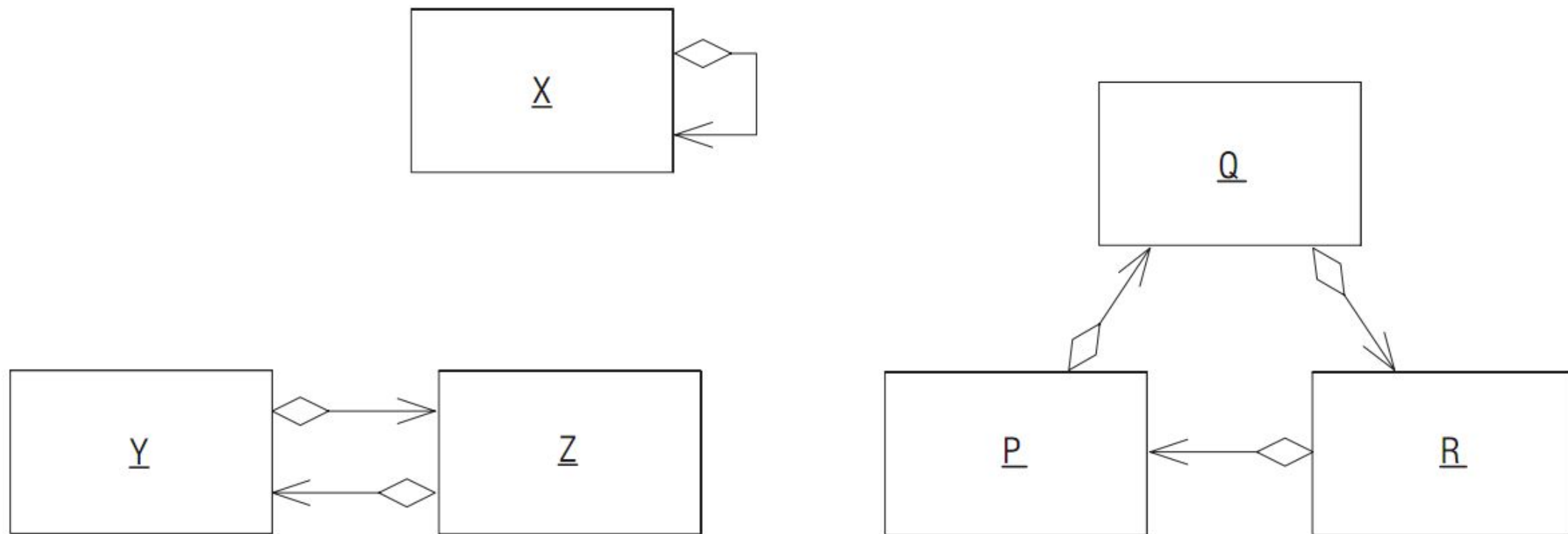
Свойства

Агрегирование



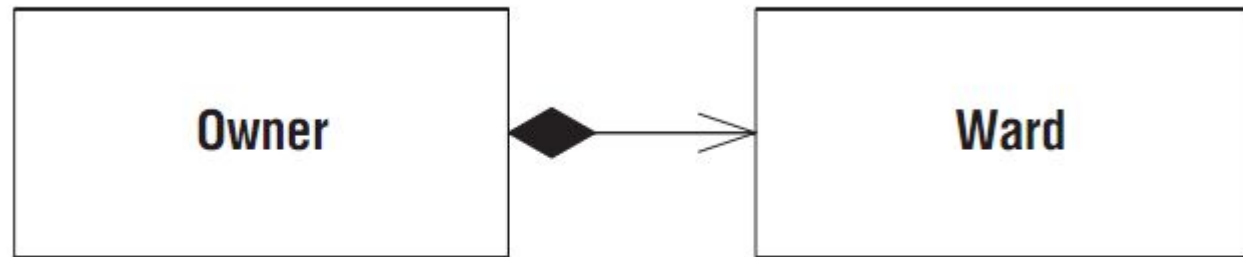
```
public class Whole
{
    private Part itsPart;
}
```

Агрегирование



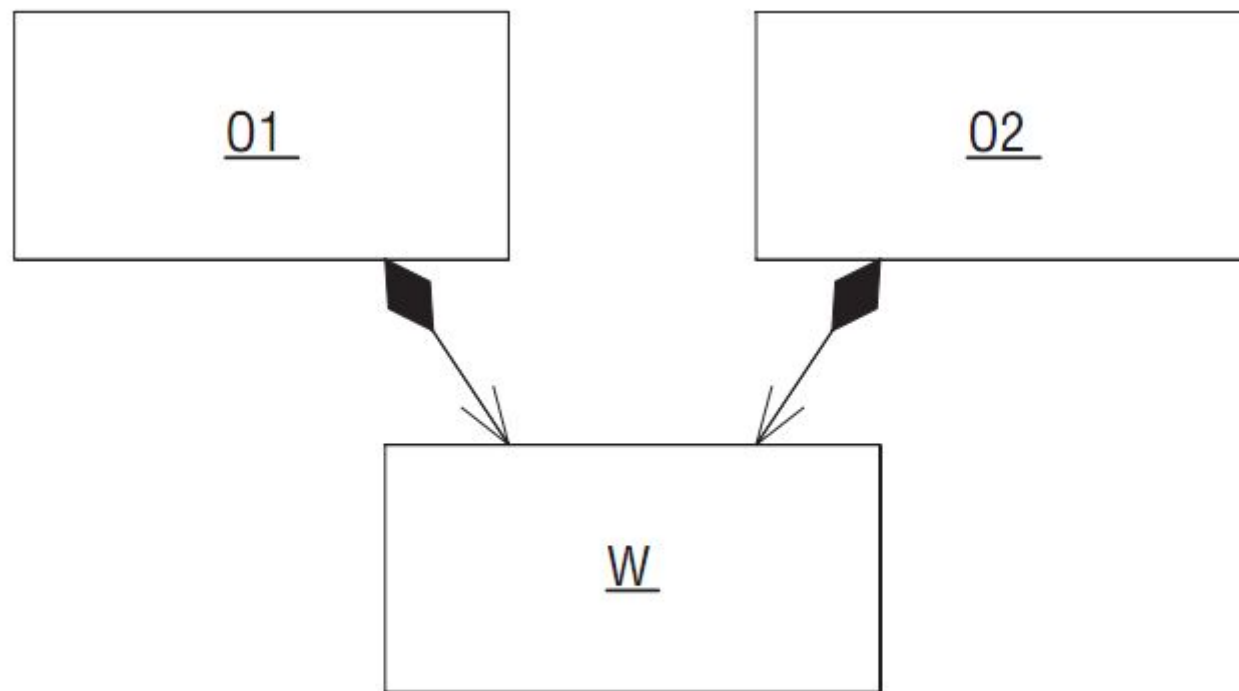
Недопустимые циклы агрегирований экземпляров

Композиция

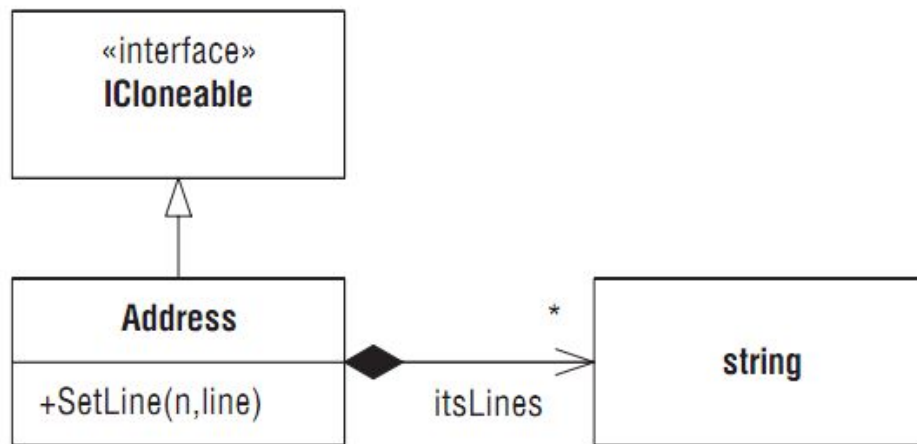


```
public class Owner
{
    private Ward itsWard;
}
```

Композиция



Недопустимая композиция



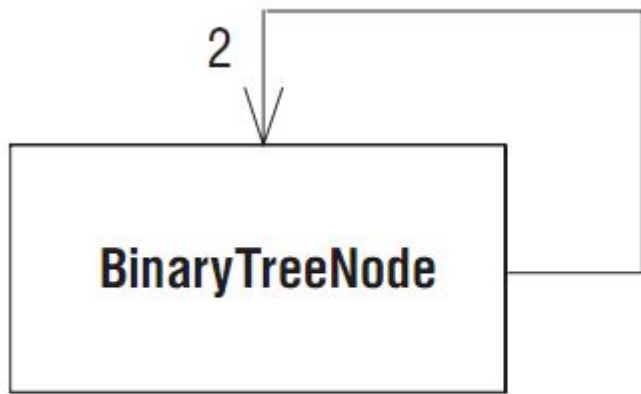
```
public class Address : ICloneable
{
    private ArrayList itsLines = new ArrayList();

    public void SetLine(int n, string line)
    {
        itsLines[n] = line;
    }

    public object Clone()
    {
        Address clone = (Address) this.MemberwiseClone();
        clone.itsLines = (ArrayList) itsLines.Clone();
        return clone;
    }
}
```

Глубокое копирование, подразумеваемое композицией

Кратность

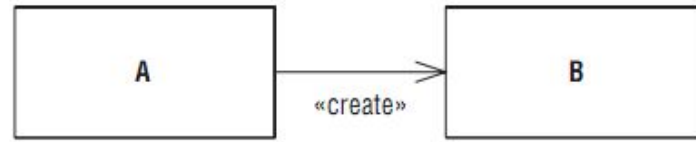


```
public class BinaryTreeNode
{
    private BinaryTreeNode leftNode;
    private BinaryTreeNode rightNode;
}
```

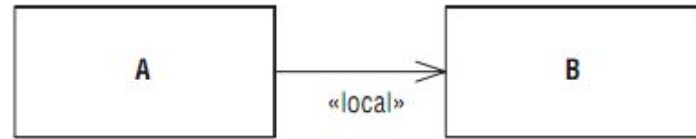
Простой пример кратности

- Цифра точное число элементов
- или $0..*$ 0 или более
- $0..1$ 0 или 1; в Java часто реализуется ссылкой, которая может быть равна null
- $1..*$ 1 или более
- $3..5$ от трех до пяти
- $0,2..5,9..*$ странно, но допустимо

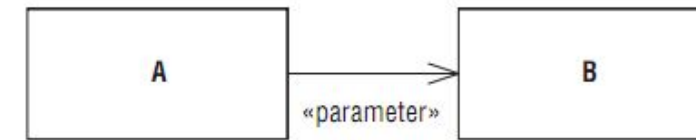
Стереотипы ассоциаций



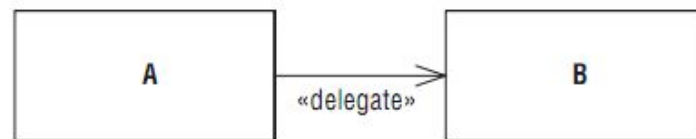
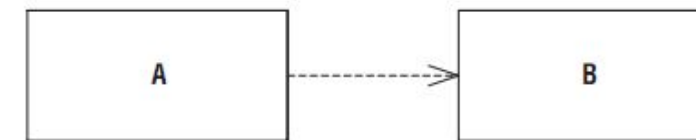
```
public class A {  
    public B MakeB() {  
        return new B();  
    }  
}
```



```
public class A {  
    public void F() {  
        B b = new B();  
        // use b  
    }  
}
```

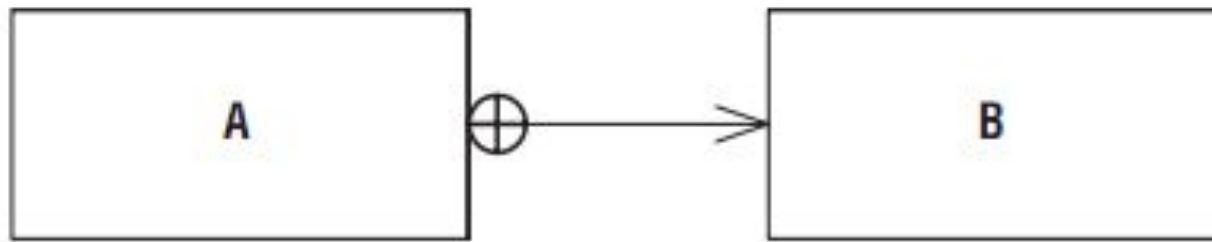


```
public class A {  
    public void F(B b) {  
        // use b;  
    }  
}
```



```
public class A {  
    private B itsB;  
    public void F() {  
        itsB.F();  
    }  
}
```

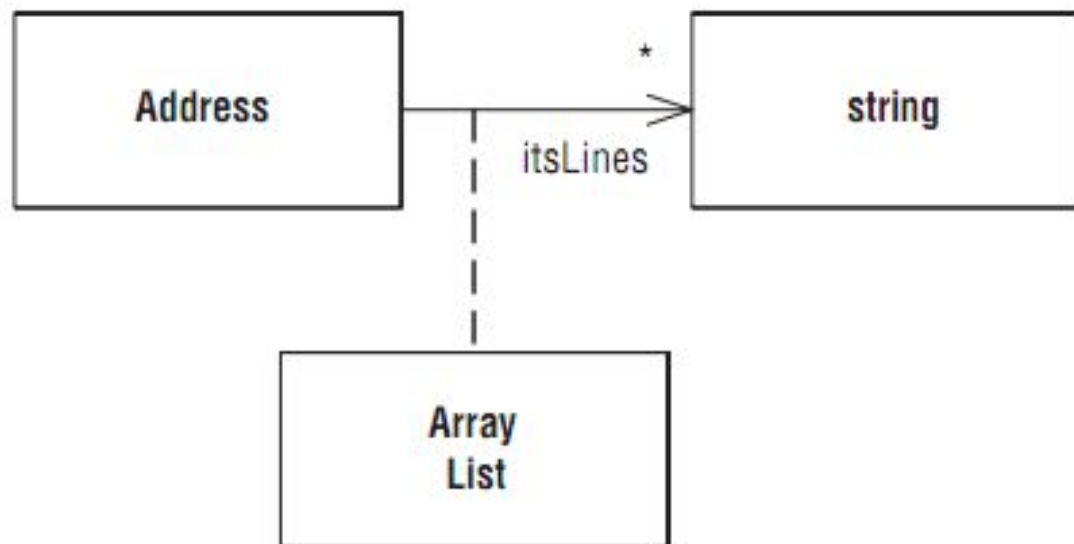
Вложенные классы



Вложенный класс

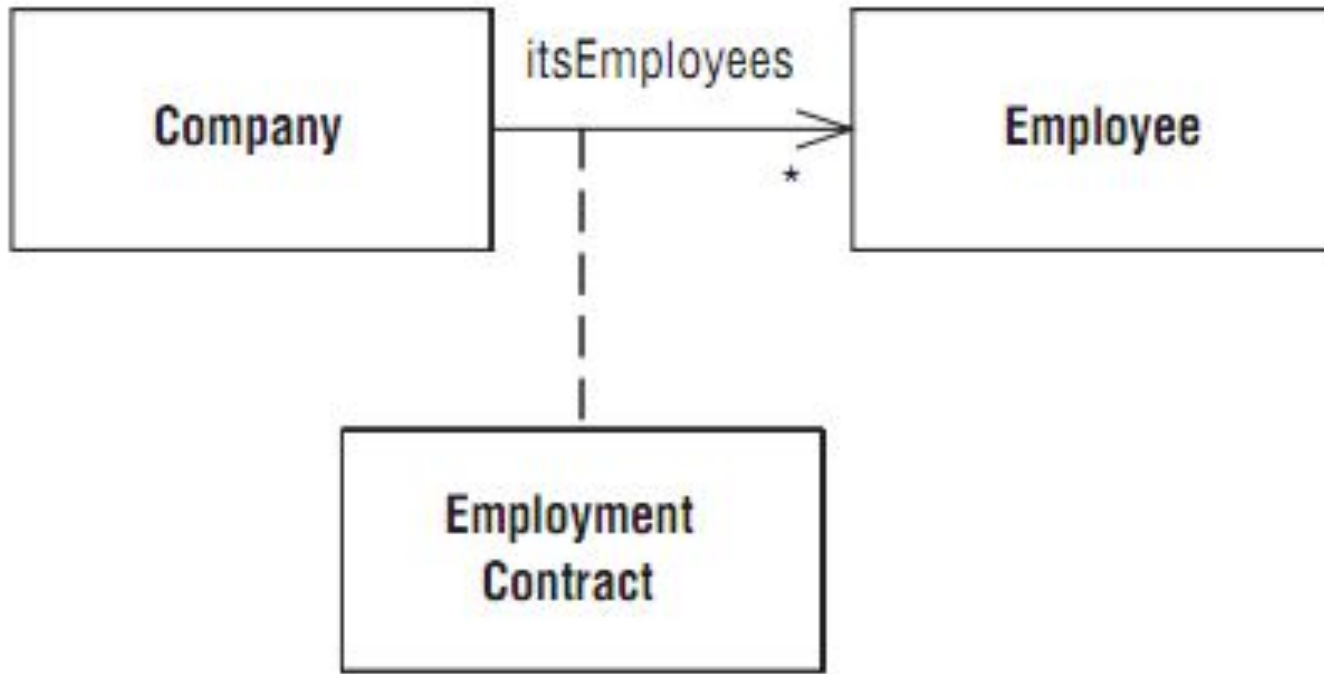
```
public class A {  
    private class B {  
        ...  
    }  
}
```


Классы ассоциаций



```
public class Address {  
    private ArrayList itsLines;  
};
```

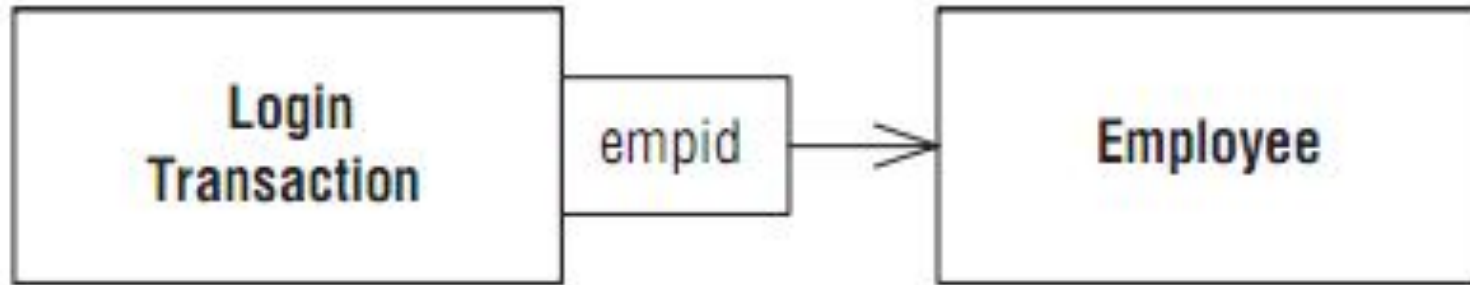
Класс ассоциации



Контракт с работником

```
public class Company
{
    private EmploymentContract[] itsEmployees;
};
```

Квалификаторы ассоциаций



Квалификатор ассоциации

```
public class LoginTransaction
{
    private string empid;
    public string Name()
    {
        Employee e=DB.GetEmp(empid);
        return e.GetName();
    }
}
```