Проектиране и Тестиране на Софтуер
ТУ, кат. КС, летен семестър 2012

Лекция 22

Тема:

UML
Unified Modeling Language
Език за OOA/OOD

Съдържание:

- Методи за проектиране
  - Unified Modeling Language

Методи за проектиране

- Словесен метод
- Схеми на потока данни
- Йерархични диаграми
- Таблици на решение
- Блок схеми
- CRC карти
- Езикът UML

ООП/П

Еволюцията на СП доведе ООП/П. Водещият принцип от структурния анализ и синтез на системи *мислене чрез функции* (Thinking in Functions) остава класически, но се счита остарял и неактуален. Той се измества от наложилия се съвременен принцип *мислене чрез обекти* (Thinking in Objects), който е валиден за обектно ориентирания анализ и синтез на системи. Развиха се множество методи за ООА, ООС, ООП/П. Най-известни са:

ООП/П

1. Метод за обектно ориентирано проектиране OOD (Object Oriented Design) от Гради Буч (Grady Booch) /Rational/;
2. Метод за обектно моделиране OMT (Object Modeling Technique) от Джеймс Ръмбау (James Rumbaugh) /General Electric/;
3. Метод за обектно ориентирано софтуерно инженерство OOSE (Object Oriented Software Engineering) от Айвар Якобсон (Ivar Jacobson) /Ericson/.

ООП/П

Тримата цитирани автори започват работа за фирма Rational Software (www.rational.com) през 1994 г. Там те обединяват усилия за разработка на унифициран метод за обектно ориентирано проектиране (Unified Method). В резултат се ражда езикът за описание и моделиране на обектно ориентирани системи, обявен като UML (Unified Modeling Language).

ООП/П

- Защо UML и каква е ползата от него?
- Една голяма програма е трудно да се разбира/анализира/ по първичния код.
- Причината: текстът е много детайлизиран.
- Добре е да има окрупнена фигура, картина, образ, който описва главните части на програмата и как те работят съвместно.
- UML отговаря на тази потребност.

ООП/П

UML е набор от различни диаграми:

- Class Diagrams
- Object Diagrams
- State Diagrams
- Sequence Diagrams
- Use Case Diagrams

ООП/П

UML изпълнява разни роли:

- Помага да се разбере как работи програмата
- Помага в процеса на initial design
- Полезен е през всичките фази на software development (от initial specification до documentation, а също testing и maintenance)

UML Class Diagrams

Най-характерното за този език е, че на проектанта се предоставя възможност в унифициран диаграмен вид да изобрази класовете/обектите, с които се описва задачата за решаване.

Име на клас
Даннови компоненти
Методи

UML Class Diagrams

Диаграмата за клас/обект съдържа раздел наименование на класа, даннови компоненти и методи на класа Всеки даннов компонент или метод се предшества от префикс +, –, #. Така се описват квалификаторите за видимост public, private, protected. Атрибут за статично (static) обявяване се задава допълнително с $. Проектирането завършва в случай на проблем, който се описва само с диаграмата на един обект. Реалните проблеми се описват с множество взаимосвързани класове/обекти.

UML Class Diagrams

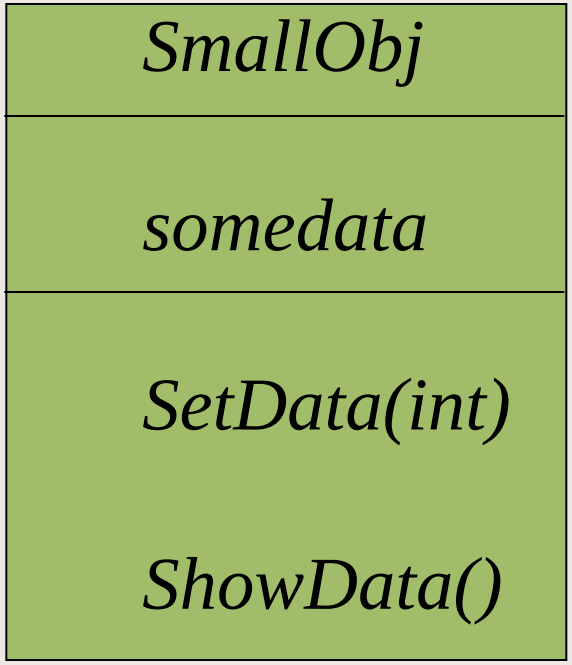
Описание на клас

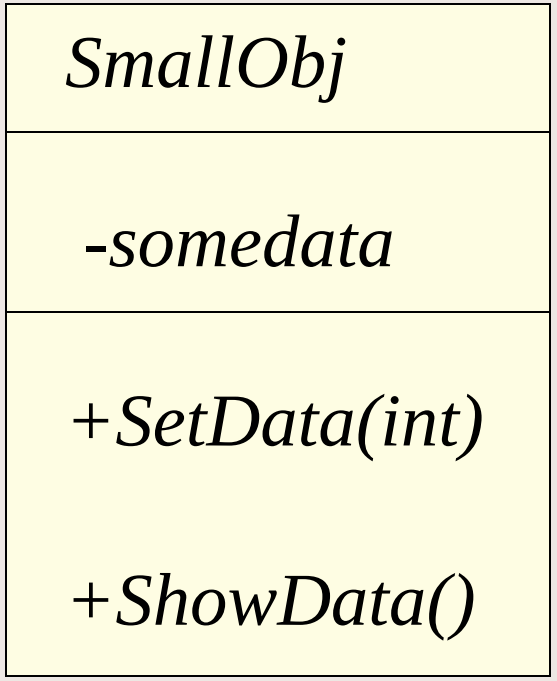
- Общ синтаксис
- Квалификатори за достъп: *private*, *public*
- Член данни
  - Обикновено *private*
- Член функции
  - Обикновено *public*

UML Class Diagrams

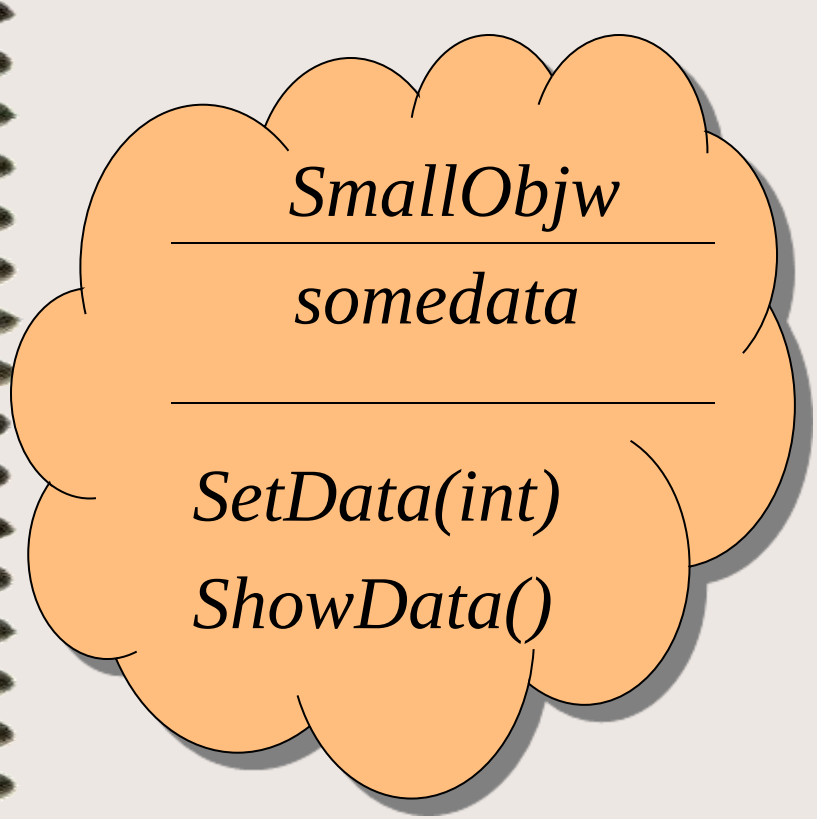
```
class SmallObj
{
 private: int somedata;

 public:  void SetData(int d) { somedata = d; }

          void ShowData() { cout << "\nData is =" <<
                                    somedata; }
};
```

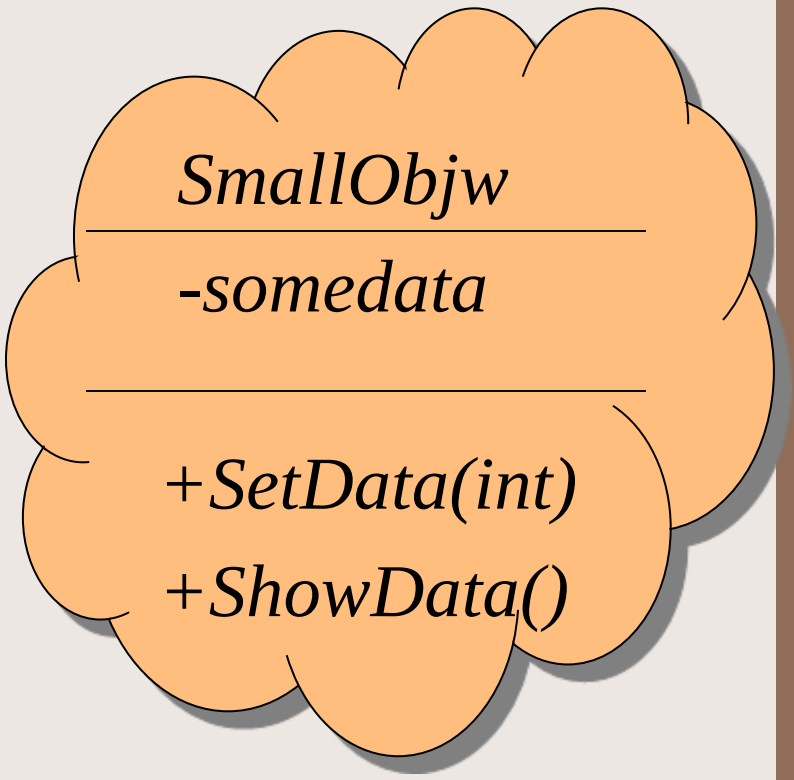
SmallObj – UML клас диаграма





SmallObj – UML клас диаграма според G.Booch





UML Object Diagrams

Приложение на клас

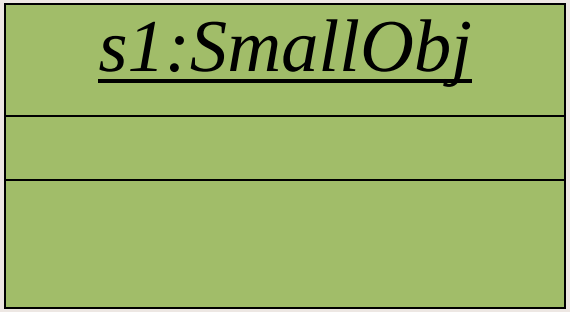
- Дефиниране на обекти
- Извикване на член функции (sending messages)

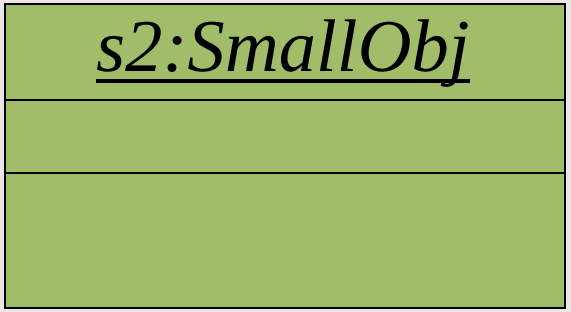
UML Object Diagrams

```
void main()
{
    SmallObj s1, s2, *ptr;

    s1.SetData(67);          s2.SetData(123);
    s1.ShowData();           s2.ShowData();

    ptr = new SmallObj; ptr->SetData(315);
    (*ptr).ShowData();
}
```

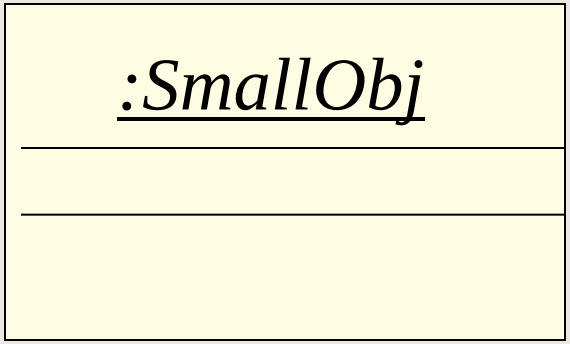
SmallObj – UML обект диаграми

За разлика от класовете, обектите се подчертават. Името на класа и обекта се разделят с двуеточие.





Обект без име, достъпен с указател, се показва така:

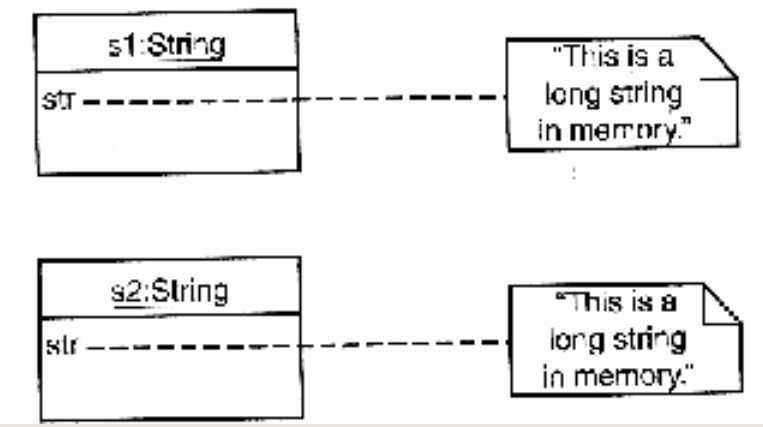


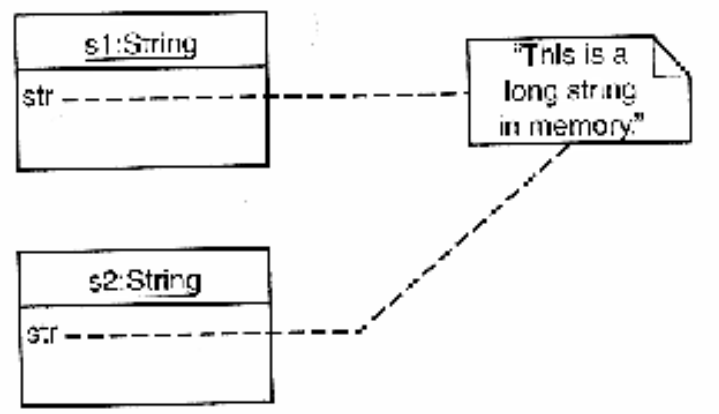
UML – Notes /бележки/

- Представят се с правоъгълник със сгънат ъгъл
- Съдържат коментар или пояснение
- Dotted line свързва с елемент, за който се отнася текстът на бележката
- Прилага се във всички видове диаграми
- Пример: Лафор 541

UML – Notes /бележки/

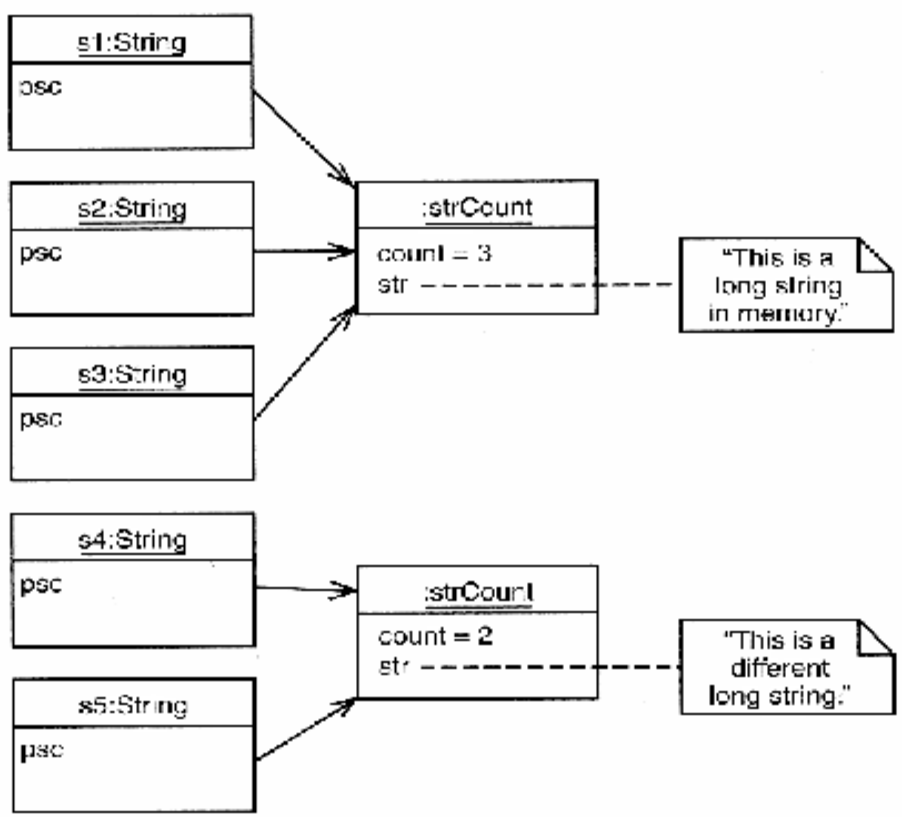
- Представят се с правоъгълник със сгънат ъгъл





UML – Notes /бележки/

- Представят се с правоъгълник със сгънат ъгъл



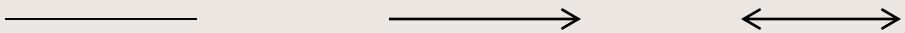
Връзки между класове

Проектирането/Моделирането продължава с връзки м/у диаграмите на класовете/обектите. Те се означават с линия, която свързва диаграмите на класовете.

Връзките се прецизират и могат да бъдат от различни видове:

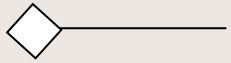
Връзки между класове

Връзка от тип асоциация (assosiation). Означава се с линия, свързваща двете клас диаграми.

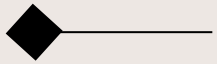


Връзка от тип наследяване (inheritance). Означава се с триъгълна стрелка.

Връзка от тип агрегация (aggregation). Означава се с линия, завършваща с празен ромб (diamond).



Връзка от тип композиция (composition). Означава се с линия, завършваща с плътен ромб (diamond).



Връзка асоциация

Пример за програми, опериращи с обекти на два класа: class time12, class time24

Лафор 351, 354

Връзки между класове:

```
  time24 t24(17, 45, 59);
  time12 t12 = t24;
// times1.cpp: routine in source –
  conversion operator: operator
  time12()
//    times2.cpp:    routine    in
  destination – 1-arg constructor
  time12(time24 t24) { } ,
  overloaded = operator
```

Връзка асоциация

Примери:

- Drivers are related to Cars
- Books are related to Libraries
- Race Horses are related to Race Tracks
- 12-hour time is related to 24-hour time

Lafore 357, fig. 8.3

Връзка асоциация

Пример

12-hour time is related to 24-hour time

GURE 8.3

IL class diagram of the TIMES1 program.

Връзка асоциация

Class association означава, че обекти от класовете, а не самите класове, са related

Два класа са в асоциация, ако обект от един клас активира член функция на обект от друг клас

Асоциация е в сила, ако атрибут от един клас е обект от друг клас.

Връзка асоциация

```
time24 t24(17, 45, 59);
  time12 t12 = t24;
//   times1.cpp:   routine   in
  source – conversion operator:
  operator time12()
```

Обект t12 от клас time12 извиква/активира конвертираща функция **`operator time12()`** в обекта t24 от клас time24

Връзка асоциация

```
time24 t24(17, 45, 59);
  time12 t12 = t24;
//times2.cpp:     routine     in
  destination       -       1-arg
  constructor time12(time24 t24)
  { } , overloaded = operator
```

Клас time12 е асоцииран с клас time24, т.к конвертираме обект от един клас /time24/ в обект от друг клас /time12/

Посочност на асоциация

Т.к time12 активира time24, посоката е указана със стрелка еднопосочна.

Възможна е и двупосочна асоциация.

Най-общо асоциацията се означава с линия.

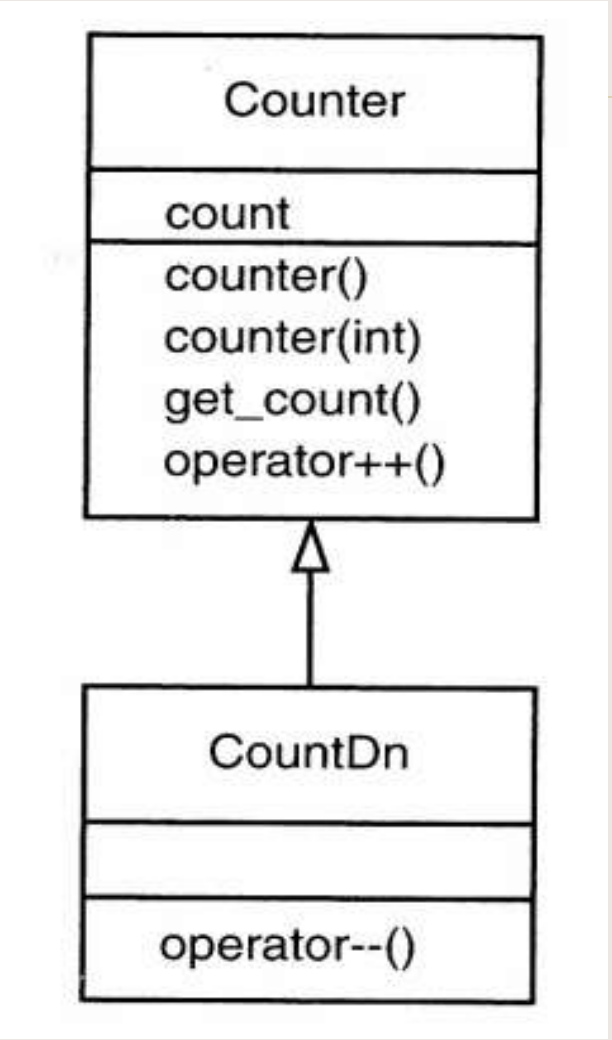
Връзка наследяване

Нарича се generalization, т.к. Родителският клас има по-общ вид от дъщерния, или обратно дъщерният клас е по-специфична версия на родителския.

Лафор 375, фиг 9.2

Връзка наследяване

Нарича се generalization



Връзка наследяване

Означава се с линия с триъгълна стрелка, която има смисъл

inherited from

derived from

is a more specific version of

Посоката указва, че породеният клас refers to методи и данни от базовия клас, докато базовият клас няма достъп до породения клас.

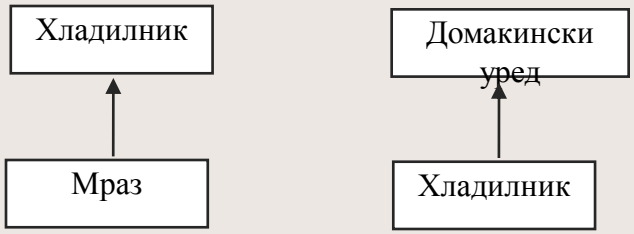
Връзка наследяване

Наследяването между базов и породен клас е връзка, която още се нарича връзка тип “е разновидност на” (“is a kind of”) при релация м/у класове и връзка тип “е” (“is a”) при релация м/у обекти. Като пример за наследяване са показани връзките между класовете Мраз, Хладилник и Домакински уред. Мраз е марка (вид) хладилник. Хладилник е вид домакински уред.

Връзка наследяване

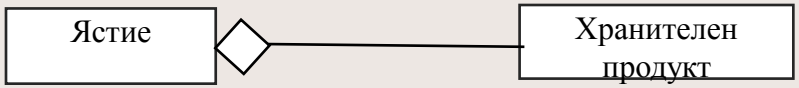
Мраз е клас, който има свои характеристики, но наследява и характеристиките на класа Хладилник.

Хладилник е клас, който има свои характеристики, но наследява и характеристиките на класа Домакински уред.



Връзка агрегация

Агрегацията е връзка от тип част – цяло, която още се нарича връзка тип “има” (“has a”) или връзка тип “е част от” (“is a part of”). Пример за връзка между класовете Ястие и Хранителен продукт. Дадено ястие има (съдържа) хранителни продукти. Даден хранителен продукт е част от едно или повече ястия.



Containership

- Classes within classes.

  Has-a relation се нарича още агрегация: Казва се:

  Library has a book.

  Invoice has a line item.

Aggregation е известна и като“part-whole” релация.

  The book is part of the library.

In OOP, aggregation may occur when an object is an attribute of another.

```
class A { … };
class B {  … A objA; … }
```

Containership

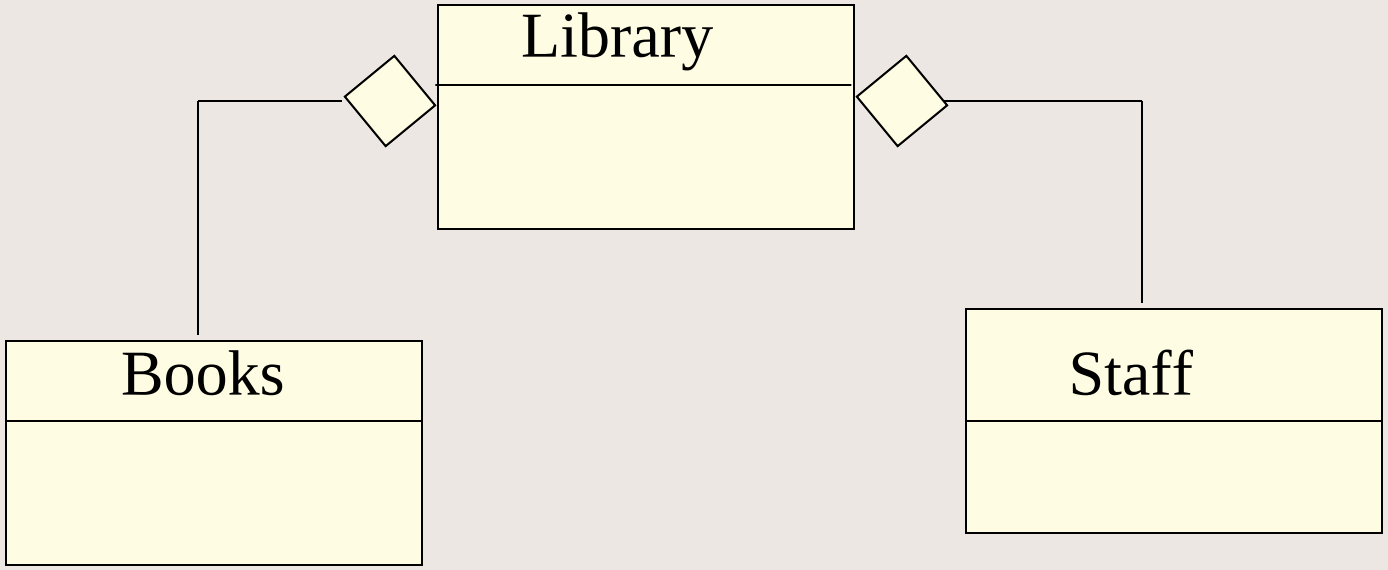
In UML, aggregation is considered a special kind of association. It's safe to call a relation an association but if class A contains object of class B, and is organizationally superior to class B, it's a good candidate for aggregation.

A company might have an aggregation of employees.

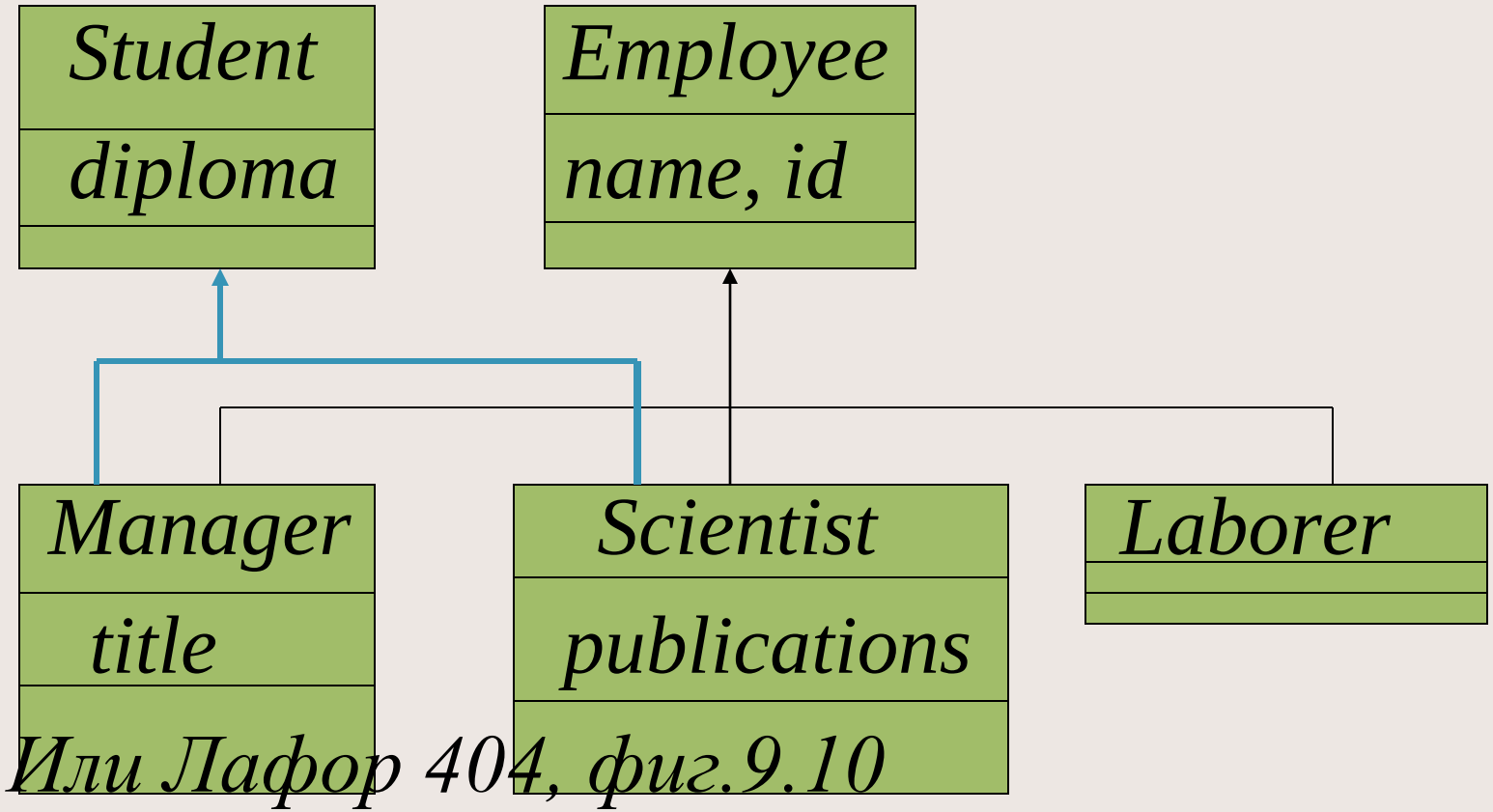
A stamp collection might have an aggregation of stamps.

Връзка агрегация

- Aggregation is shown in the same way as association in UML class diagrams except that the “whole” end of the association line has an open diamond-shaped arrowhead. Lafor415, fig9.11

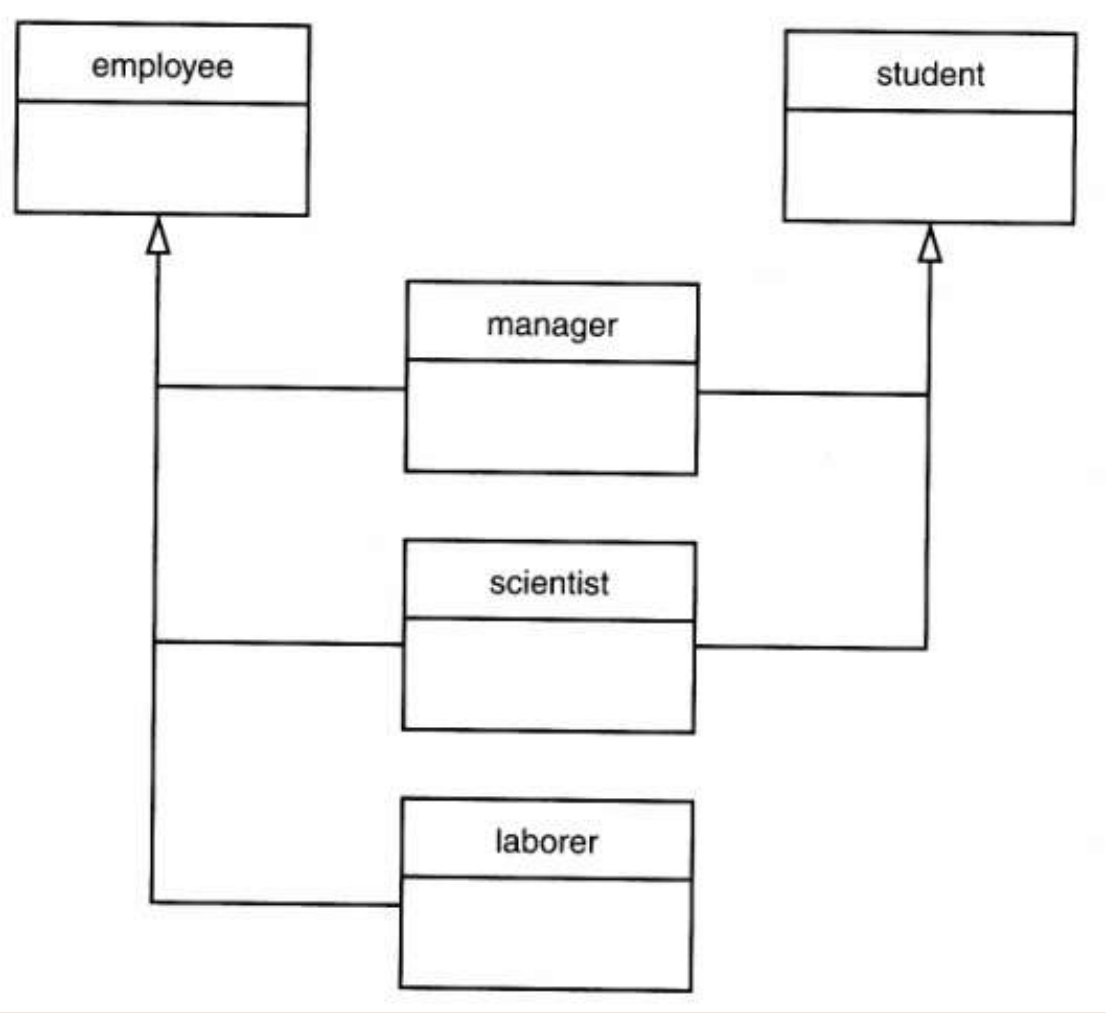


Връзка агрегация вместо наследяване

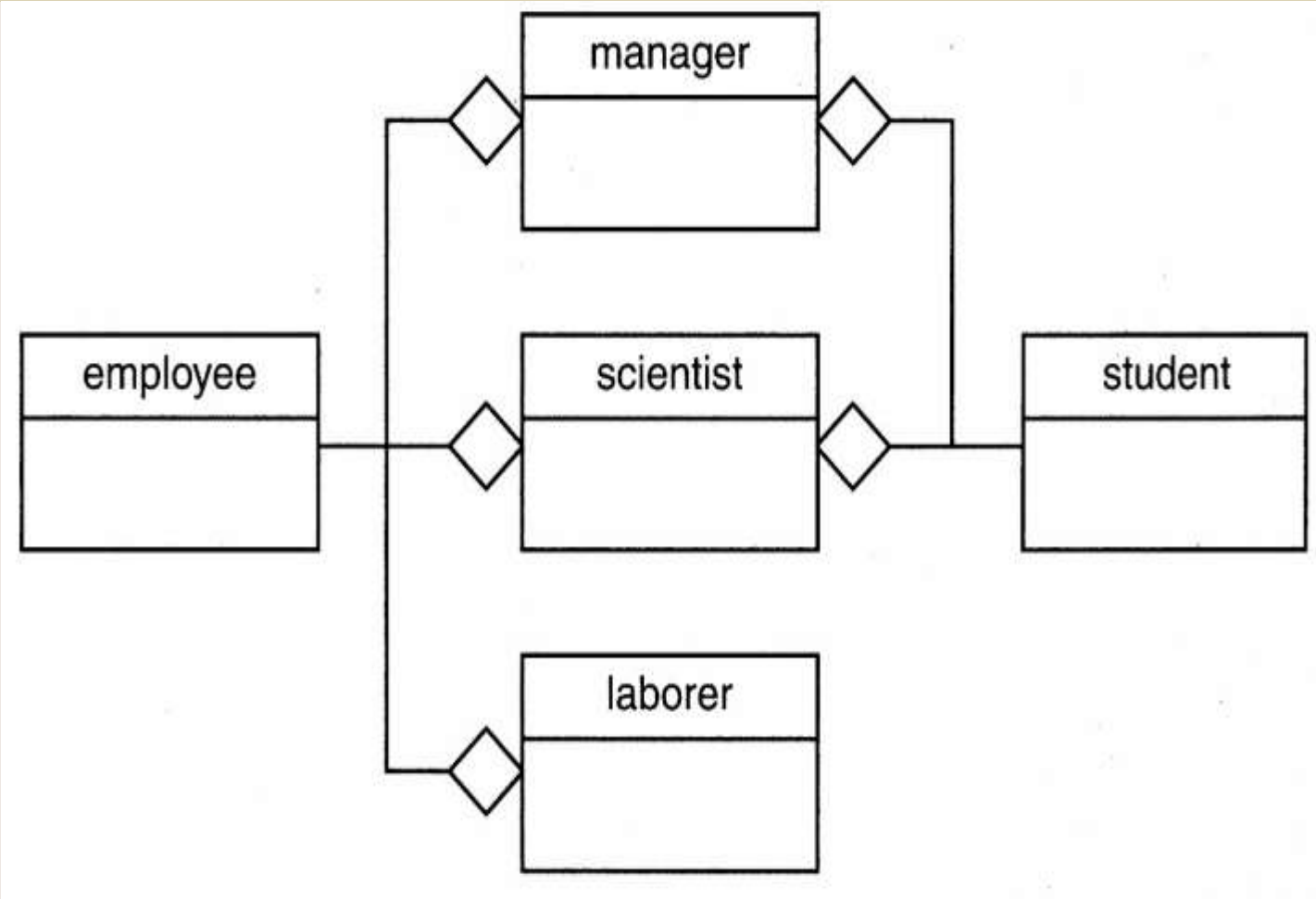


Или Лафор 404, фиг.9.10

Връзка агрегация вместо наследяване



Връзка агрегация вместо наследяване



Връзка композиция

Следващата връзка се нарича композиция и е разновидност на агрегацията. При композицията цялото строго притежава (strongly owns) своите компоненти. Всички действия с обекта от целия клас (копиране, преместване, разрушаване) влекат същите действия, които се извършват над обектите, които го съставят. Обектите компоненти са притежание на точно един определен обект от класа цяло. Затова стойността, с която се означава връзката от страната на цялото, е 1. Примерът, който е показан, описва връзка композиция между клас Жилищен блок и клас Апартамент. Жилищен блок има (съдържа) апартаменти. Апартаментът е част от жилищен блок.

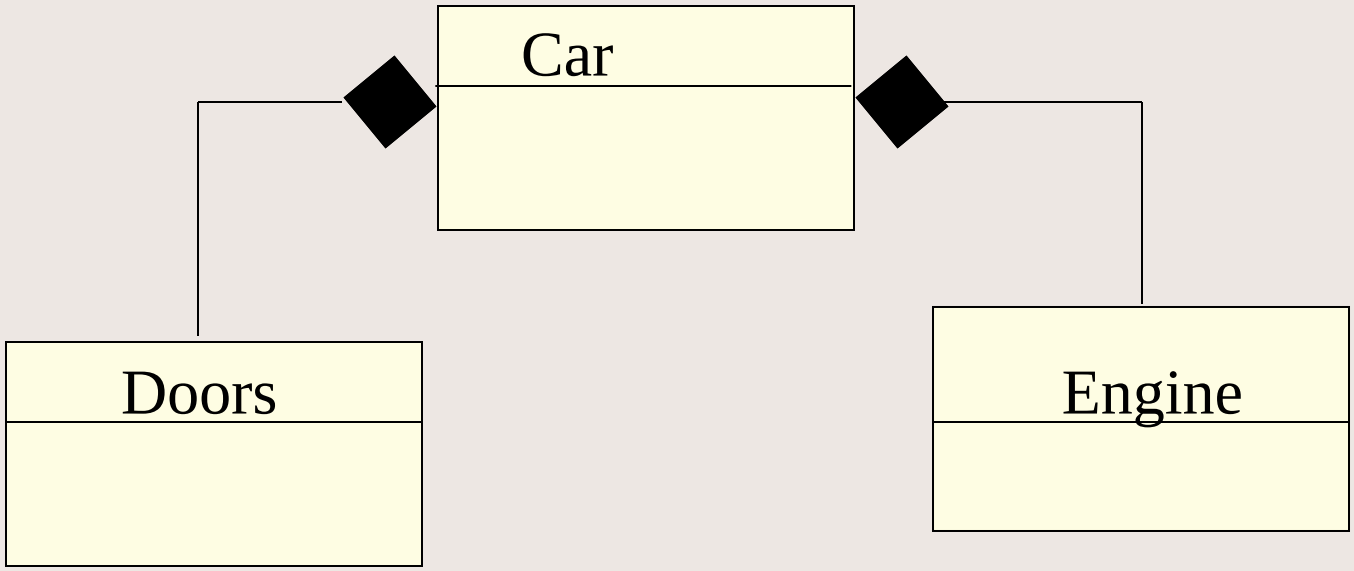
Връзка композиция

Не е възможно един конкретен апартамент да е част от няколко жилищни блока. Той принадлежи на точно един жилищен блок. По тази причина с отчитане на кардиналността описанта връзка композиция означава:

Един жилищен блок (конкретен обект) има 17 апартамента

Composition: a stronger Aggregation

- Composition has characteristics of aggregation plus:
  - The part may belong to only one a whole
  - The lifetime of the part is the same as the lifetime of the whole



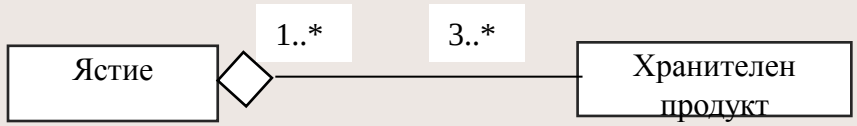
Многочисленост (multiplicity) в UML

Този тип връзки имат и характеристика кардиналност (cardinality). Връзката се надписва в двата края с числа (стойности или диапазон от стойности), които показват броя обекти от съответния клас, които участват в асоциацията. Възможни стойности за означаване са.

1	– един обект;
*	– неопределен брой обекти;
0..1	– нула или един обект;
0..*	– нула или повече обекти;
1..*	– един или повече обекти.

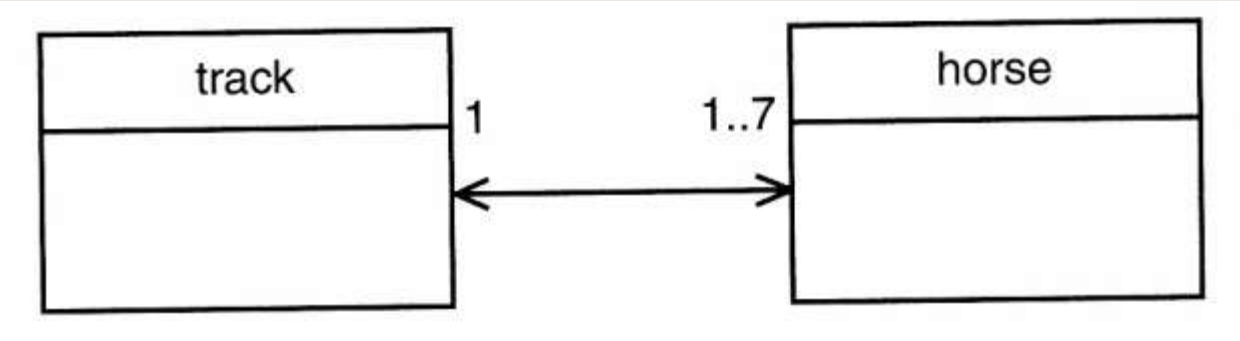
Многочисленост (multiplicity) в UML

Обогатената връзка от тип агрегация с въведена кардиналност от двете страни е показана. Тя съответства на следната схема. Едно или повече ястия съдържат три или повече хранителни продукта. Три или повече хранителни продукта са част от едно или повече ястия.



Многочисленост (multiplicity) в UML

- Лафор 489, фиг10.20
- И сорс текст Лафор485, horse.cpp,



Многочисленост (multiplicity) в UML

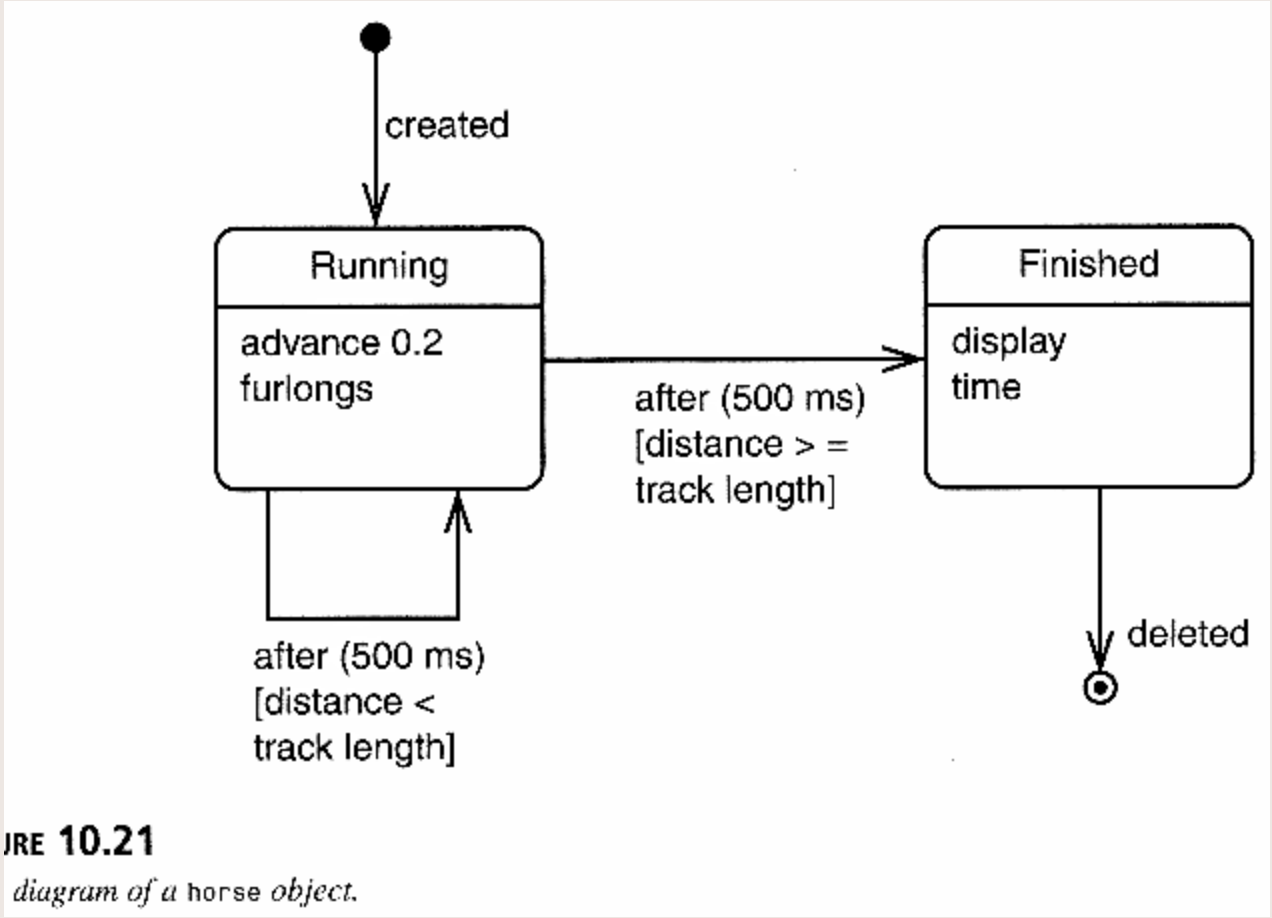
- Възможни варианти на означения:
  - 1
  - *
  - 0..1
  - 1..*
  - 2..4
  - 7,11

UML State Diagrams

- Наричат се още state chart диаграми.
- Разгледаните досега клас/обект диаграми са статични, т.к. Релациите м/у класовете/обектите не се променят при изпълнение.
- Интересно е да се следи динамиката на поведението на обекти във времето.
  - Обект се създава,
  - Отделни събития му въздействат,
  - Обектът взима решения, върши действия,
  - На края евентуално се разрушава.
- Отделните състояния се свързват с преходи.

UML State Diagrams

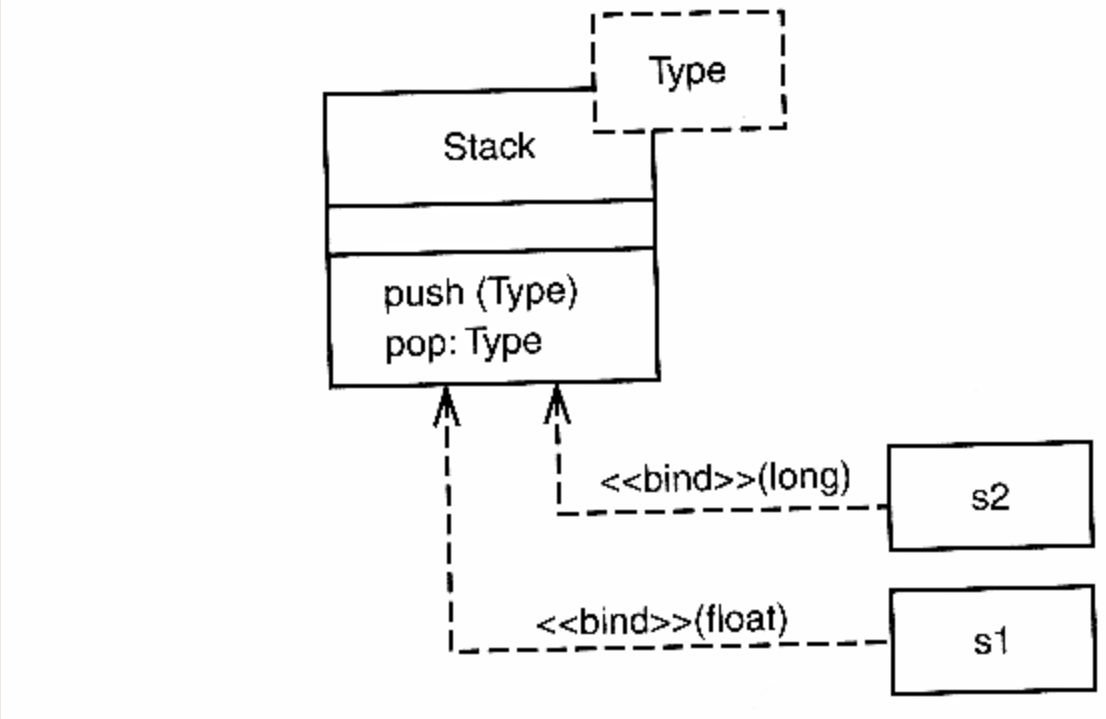
- Лафор 490,фиг10.21, Състояния и преходи м/у състояния за обект horse



URE 10.21
diagram of a horse *object.*

UML and Templates

- Лафор 702,фиг14.3, Tempstak.cpp



RE 14.3
late in a UML class diagram.

UML and Templates

- Templates /parameterized classes/ се представят с вариация на клас диаграма. Имената на template аргументите се разполагат в dotted rectangle, който се вгражда в горния десен ъгъл на клас диаграмата

Благодаря
За
Вниманието