

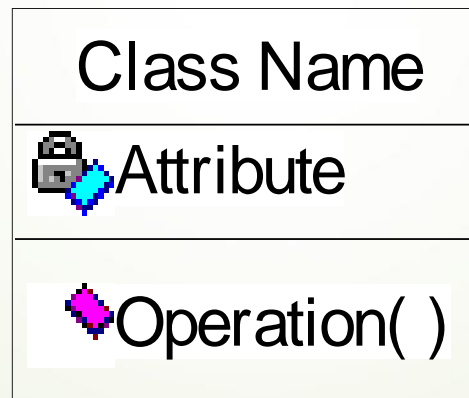


# Class диаграмми

Упражнение 2

# Class диаграми

- **Дефиниция - Клас диаграмите** се използват, за да показват някои от класовете и пакетите на класовете в системата, която ще разработим. Те дават **статична картина** на пръчетата в системата и връзките между тях.



Анотация на UML за клас

# Видове класове

- За всеки клас може да се зададе stereotype. Има три основни stereotype в UML:
  1. **Boundary classes** ( граничен клас)- форми, репорти, интерфейси към хардуер и др.
  2. **Entity classes** - класове, които съдържат информация, която ще записвате в БД.
  3. **Control classes** – класове, които отговарят за усилията на другите класове в модела.

Модификатори за достъп?!?

# Работа с атрибути

- Атрибутът (член-променлива) е част от информацията, свързана с класа. Един клас може да има един или повече атрибути.
- Спецификация на атрибута:

***Visibility name : data-type-expression = initial value***

Как се вижда даден атрибут от другите класове: **public, private, protected** и **package**.

Името на атрибута

Типът на данните, за този атрибут. **Класовете също са типове!**

Въвежда се стойност по подразбиране, ако има такава!

- За всеки атрибут може да се определи как се съдържа в класа – **по стойност или по адрес**.

# Работа с операции

Операцията (метод) е поведение, свързано с класа. Всяка операция има три части: **име на операцията, параметри на операцията, и тип на връщаната стойност от операцията.** В UML имат следният вид:

***Operation Name (argument1: argument1 data type, argument2: argument2 data type, .....): return type***

# Интерфейс

- **Интерфейсът** е клас, който **съдържа само декларации на операции**. Класовете, които го реализират, трябва да предоставят телата на тези операции. Всеки клас може да реализира множество интерфейси, както и един интерфейс, може да се реализира от множество класове.



Interface

# Връзки между класове

Разграничаваме четири основни типа връзки, за връзки между класовете:

- associations,
- dependencies,
- aggregation,
- generalization.

# Associations връзки (Двупосочна)

- Този връзка показва семантична връзка между класовете. Тя позволява един клас да знае за public операциите и атрибутите на друг клас. Например на фигурата имаме association връзка между класа House и класа Person.



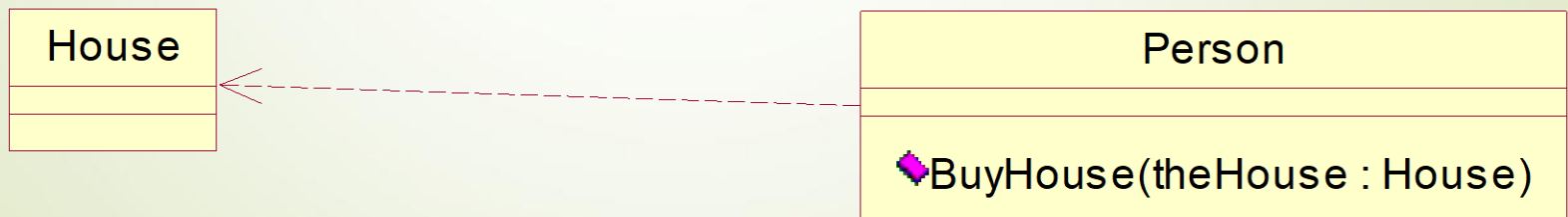
- Directed association** връзки – еднопосочна. Единият клас знае за public атрибутите и операциите на другия клас, но вторият не знае за първия!





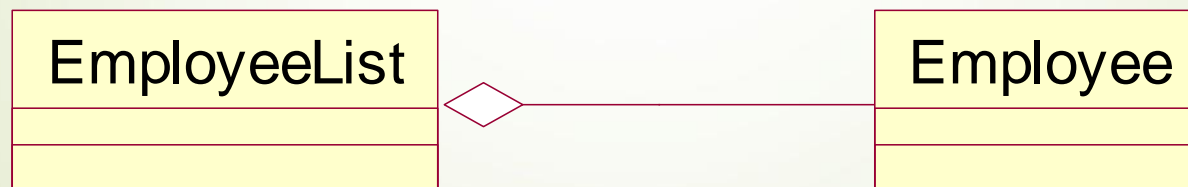
# Dependencies връзки

- Dependency е винаги едностранна и показва, че един клас зависи от дефиниции в друг клас. Примерът показва, че клас Person съдържа в себе си метод, който изцяло зависи от дефиниции от клас House – buyHouse().



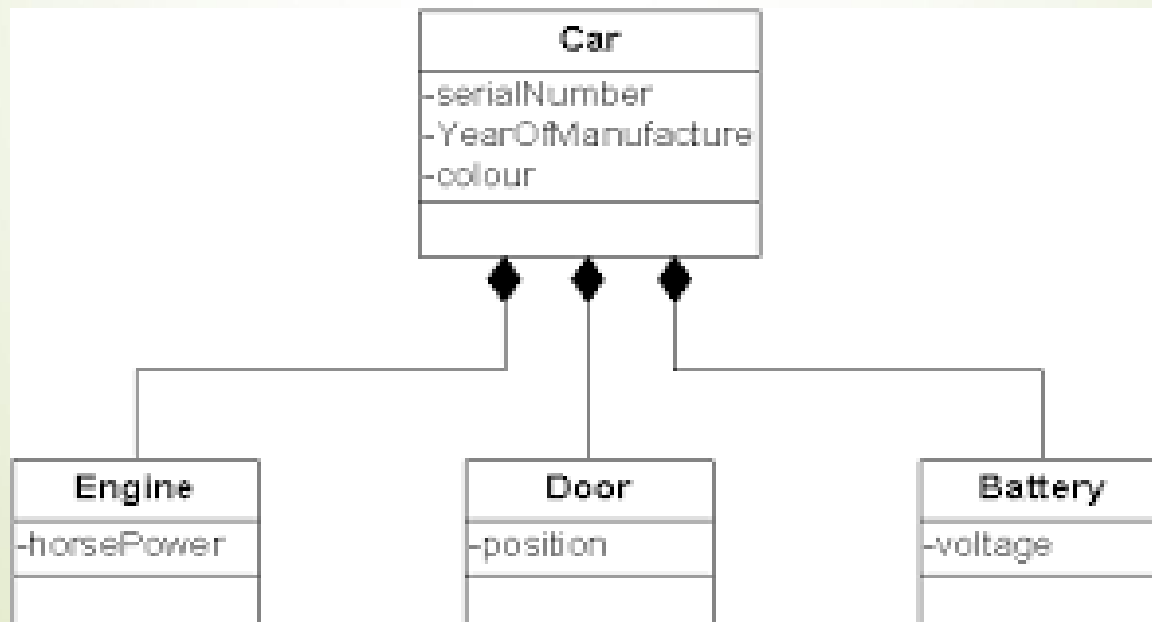
# Aggregations връзки

- **Aggregation** е строга форма на association. Това е връзка между **нещо цяло и неговата част**. Например Списък с работници се състои от работници (by value). **Агрегат** наричаме родителския клас, а компоненти наричаме агрегираните класове.



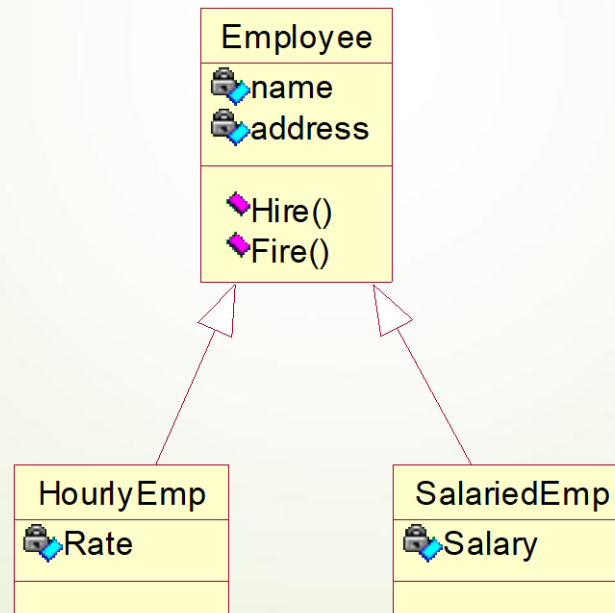
# Composition връзки

- Композицията е агрегация, при която компонентите не могат да съществуват без агрегата (родителя).



# Generalizations връзки

- Това е връзка за наследяване между два класа. Тя позволява един клас да наследи public и protected атрибути и операции на друг клас. Например





# Object диаграми

- Object диаграмата представлява граф от инстанции, които включват обекти и информационни стойности. Те показват моментно положение на състоянието на системата. Използването на тези диаграми е доста ограничено – главно се използват да показват примери на информационни структури.