



Наследяване и полиморфизъм в Java

Лектор:

инж. Божидар Бацов

Документиране на класове с javadoc

- Разглеждане на javadoc документация
- Документират се публични и protected елементи – пакети, класове, методи и полета
- Стандартни javadoc анотации - @author, @version, @since, @parameter, @return, @throws, @deprecated, @see
- Работа с инструмента javadoc

```
javadoc -d docdir package
```

```
javadoc -d docdir package1 package2
```

Преглед на домашното

- Решенията са достъпни в kenai.com
- Преглед на задачата за календара
- Преглед на задачата за класа Employee

Обзор на материала от предния път

- Класове
 - Моделират обекти от реалния свят
 - Дефинират състояние и поведение на обекти
 - Шаблони за изготвяне на обекти
- Обекти
 - Притежават конкретно състояние
 - Инстанция на клас

ООП в Java

- Състоянието се капсулира в полета(instance fields)
- Поведението се реализира посредством методи – изпълнима програмна единица

Класово състояние/поведение

- Статични полета и методи
- Достъпно без инстанция от класа
- Достъпно във всяка инстанция от класа
- Достъпно посредством инстанция
- Статичните методи могат да манипулират само статични полета

Организиране на класове

- Класовете се групират според функциите си в пакети(еквивалент на пространство на имена)
- Името на пакета + името на класа = напълно квалифицирано име на класа
- Класове от други пакети се достъпват посредством пълните им имена или чрез директивата “import”

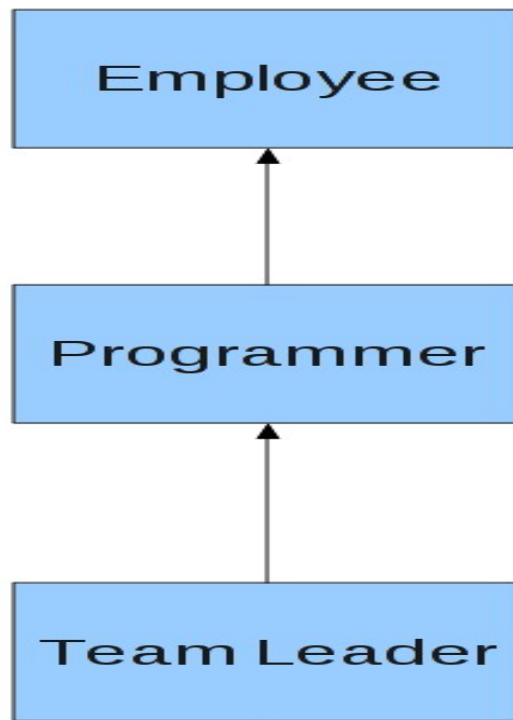
Модификатори за достъп

- `private` – достъп само в рамките класа
- `public` – достъп за всички
- `default(package)` – достъп в рамките на пакета, не съответства на ключова дума
- Могат да се прилагат на класове, полета и методи

Наследяване

- Фундаментална техника в ООП
- Основава се на идеята за изграждане на нови класове от съществуващи класове
- Преизползват се полетата и методите на съществуващите класове
- Добавя се ново поведение и състояние
- Променя се съществуващо поведение

От общото към частното



От общото към частното

- Наследяване == Разширяване
- Класът, който наследяваме се нарича суперклас(клас родител)
- Класът наследник се наричат подклас(subclass)
- Отношението между подклас и суперклас е “е”(is-a)
 - Програмиста е служител

Наследяване в Java

```
class Subclass extends Superclass {  
    fields;  
    constructors;  
    methods;  
}
```

Пример в NetBeans

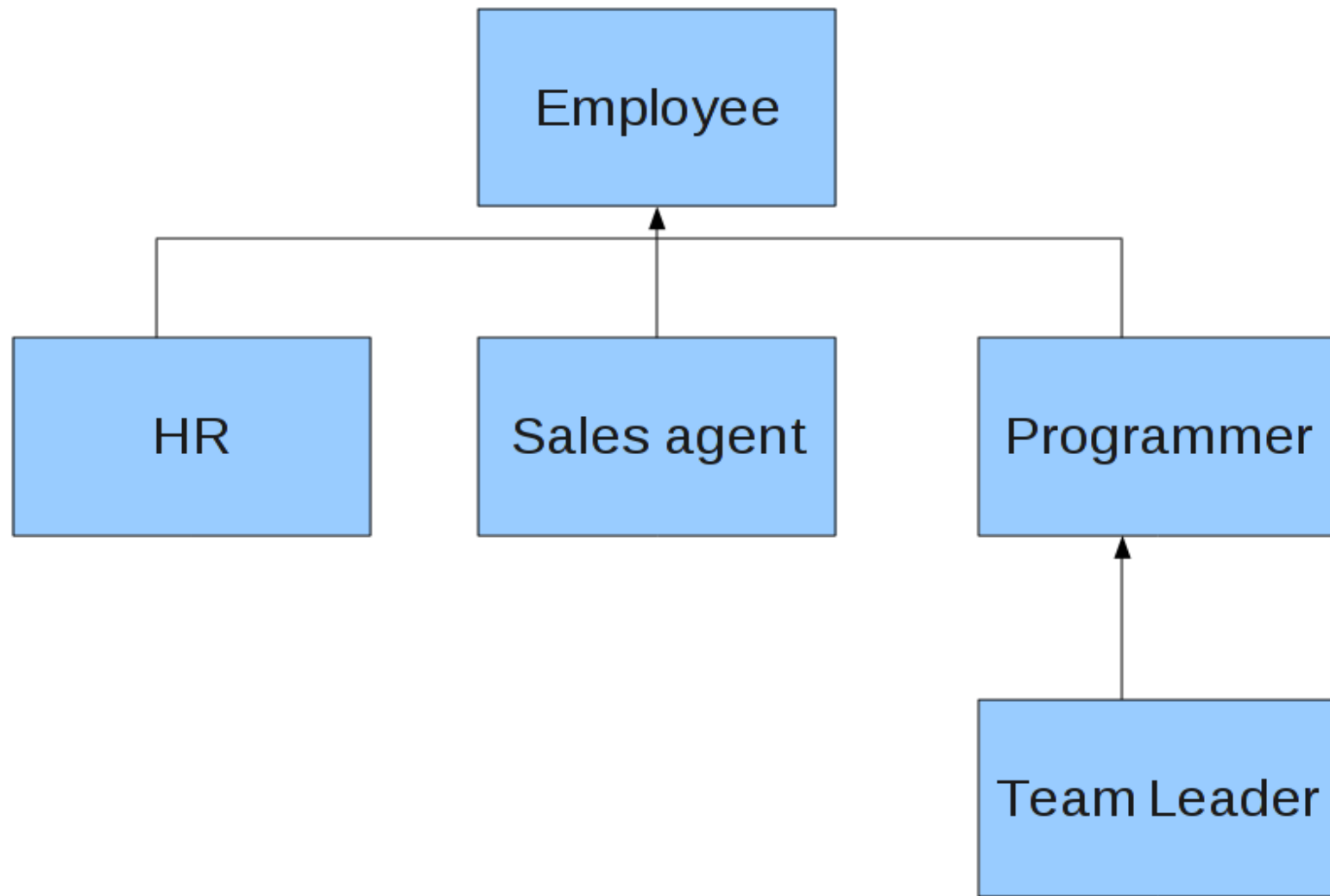
Важни моменти при наследяването

- Конструктора на подкласа извиква класа на суперкласа – това може да стане имплицитно или чрез ключовата дума “super”
- Private елементите не са достъпни директно в наследения клас
- Чрез ключовата дума “super” може да бъде извикан произволен метод от суперкласа

Йерархия на наследяването

- Единична йерархия на наследяване – един клас може да има само един суперклас

`class A extends B, C // невъзможно`



Полиморфизъм

- Техника на програмиране, при която заместяваме подкласове с техни суперкласове, където това е подходящо
Programmer pr = new Programmer();
Employee empPr = new Programmer();
- Полиморфизмът работи тясно с т.н. Късно свързване на методите (late binding)

Полиморфизъм

- Какво ни дава полиморфизма?
 - Гъвкавост – може да добавяме нови класове без да прекомпилираме кода си
 - Генеричност – като се придържам към по-общи класове, обикновено пишем по-прост код

Търсене на метод

- Търсене от компилатора(статично/ранно свързване)
- Търсене от виртуалната машина(динамично/късно свързване)
- Таблица на методите(method lookup table)

Късно свързване

- Независимо от какъв тип сте декларирали дадена променлива, виртуалната машина на Java знае истинския тип на този обект по време на изпълнение на програмата и ще извика метода от истинския тип, а не от декларирания тип

```
Employee emp = new Programmer();  
emp.toString() // вика toString() на класа  
//Programmer
```

Забраняване на наследяването и късното свързване

- `final class` – неразширяем клас
- `final method` – метод, който не може да бъде динамично свързан
- В `final class` всеки метод е имплицитно `final`
- `Final` като защитна и оптимизационна техника

Абстрактни класове

- Обратното на финални класове – създадени са да бъдат наследявани
- Не могат да бъдат инстанцирани – не могат да се създават обекти от тях
- Обикновено са на върха на йерархията на наследяване
- Съдържат един или повече метода без имплементация – абстрактни методи

Абстрактни класове

```
abstract class SomeClass {  
    abstract void doSomething();  
}
```

```
SomeClass sc = new SomeClass(); //не!
```

Пример в NetBeans

Преобразуване на обекти

- Всеки обект от подклас е и обект от суперкласа

- Обратното не е вярно

```
Programmer p = new Programmer();
```

```
Employee e = p; // работи
```

```
e = new Employee();
```

```
p = e; // не работи
```

Преобразуване на обекти

- Референция от суперклас може бъде конвертирана до референция от подклас експлицитно

```
Employee e = new Programmer();
```

```
Programmer p = (Programmer) e;
```

- Опасна операция – може да доведе до грешки по време на изпълнението на програмата

Преобразуване на обекти

- instanceof – ключова дума за проверка на тип на обект

```
if (e instanceof Programmer) {  
    p = (Programmer)e;  
}
```

- Несъвместими преобразувания са забранени от компилатора

```
Date d = (Date) e; // грешка
```

Върховния суперклас Object

- Началото на класовата йерархия в Java
- Всеки клас имплицитно наследява класа Object
- Той съдържа в себе си най-общи методи като `clone()`, `equals()`, `toString()` и т.н., които имат смисъл за всички обекти

Върховния суперклас Object

- Всяка референция може да бъде преобразуна до тип Object

```
Employee e = new Employee();
```

```
Object o = e;
```

Сравняване на обекти

- Метода equals()
- Дефиниране на подходящ метод equals()
- Връзка между equals() и hashCode()

Пример в NetBeans

Модификатора за достъп `protected`

- Еквивалент на `default` достъп + достъп в подкласовете
- Употребата му обикновено се счита за лош стил
- `protected` и `default` е желателно да бъдат избягвани

Нива на достъп - обзор

- 4 нива на достъп – private, protected, public и package(default)
- 3 модификатора на достъп – private, protected, public

Пример в NetBeans

Въпроси?