Matematički fakultet, Univerzizet u Beogradu
Katedra za računarstvo i informatiku

Objektno orijentisano programiranje

vežbe
školska 2016/ 2017

Biljana Stojanović
Nemanja Mićović
Nikola Milev

1 Наслеђивање класа *

Концепт наслеђивања је врло битан сегмент објектно оријентисаног програмирања.
Наслеђивање је поступак којим се из постојећих праве (изводе) нове класе.
Нова класа зове се **изведена** или **потклаша**, док је постојећа класа **базна** или **суперкласа**.
Нова и постојећа класа налазе се у специфичном односу.
Хијерархија наслеђивања може бити произвољне дубине (из изведене класе могу се даље изводити нове класе).

Зашто изводимо класе из постојећих?

Главни циљ нам је да решимо неки проблем.
Наравно, имаћемо класе и објекте јер Јава без тога не може.
Објекти могу бити нешто апстрактно (тачке, кругови, дужи), а могу бити и нешто опипљивије (животиње, мачке, буве).
Може постојати и некаква хијерархија међу објектима и класама са којима се ради.
Нпр. можемо имати класу `Povrs` и класе `Pravougaonik`, `Kvadrat`, `Krug`...
И јасно је да је правоугаоник површ, као и квадрат и круг...
Или нпр. ако имамо класе `Zivotinja`, `Macka`, `Pas`, … опет: мачка је животиња, пас је такође животиња итд.
Мачка ЈЕ животиња. То значи да има сва својства животиње (све податке (атрибуте) и понашања заједничка за све животиње). Али мачка има и нешто више од тога, нешто што је својствено само мачкама, због чега је издвојена у посебну животињску врсту (нпр. мачке преду...).

У таквим ситуацијама, када су објекти двају или већег броја класа на неки начин специјализација објеката једне класе, природно је да та класа буде базна, а да се ове остале изведу из ње.
У базној класи треба да се нађу атрибути и методи заједнички за објекте свих изведених класа, док у изведене класе треба сместити оно по чему се њихови објекти међусобно разликују.

Како рећи да је једна класа изведена из друге?

То се чини кључном речју `extends` која се наводи у првом реду дефиниције класе. Нпр.

```
public class Krug extends Povrs {
    ...
}
```

Овде се каже да је класа `Krug` изведена из класе `Povrs`.
То, даље, значи да ће сваки објекат класе `Krug`, поред атрибута који се налазе у дефиницији те класе, садржати и **све** атрибуте дефинисане у базној класи `Povrs`.
Ипак, атрибути дефинисани у базној класи не морају увек бити доступни методима изведене класе.
Посебно је важно нагласити да је **приликом прављења објекта изведене класе** (`Krug`) **неопходно извршити иницијализацију и тих атрибута** базне класе (`Povrs`) који су присутни у објекту изведене класе.

Наслеђени чланови

Код наслеђивања се уводи и појам тзв. наслеђених чланова базне класе.
То нису сви чланови (атрибути и методи) базне класе, иако су сви они присутни у објекту изведене класе.
Чланови базне класе који су доступни унутар изведене класе су **наслеђени чланови**. Они који нису наслеђени су и даље део објеката изведене класе.

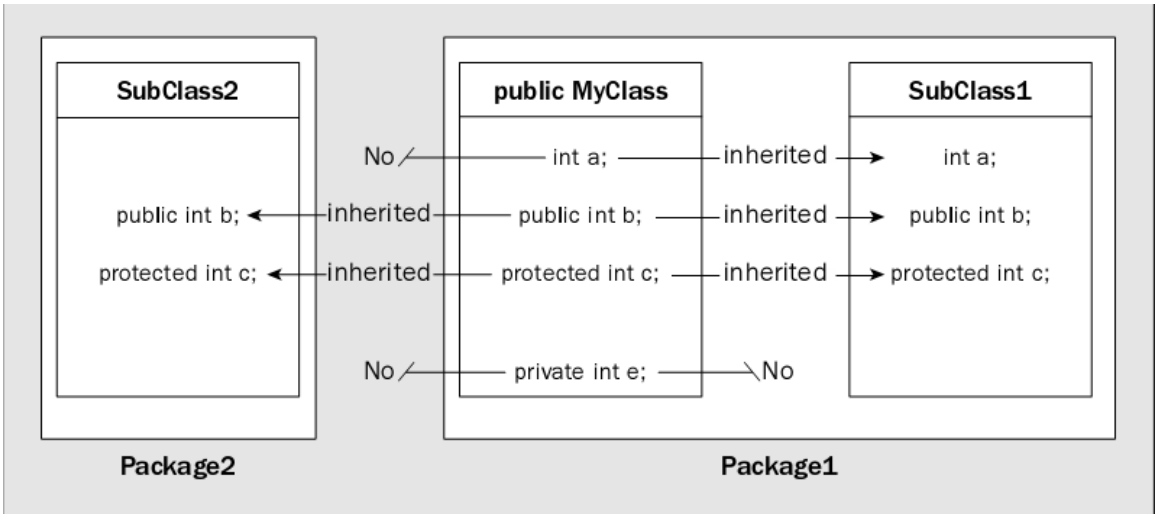
---

* *Укључује и садржај из материјала Марије Милановић*

Када се базна и изведена класа налазе у истом пакету, не наслеђују се само они чланови који су у базној класи дефинисани са приступним атрибутом `private`.

Ако се, пак, базна и изведена класа налазе у различитим пакетима, неће бити наслеђени ни они чланови базне класе који имају пакетно право приступа.

Дакле, `public` и `protected` чланови базне класе наслеђују се увек, без обзира на размештај класа по пакетима.



А шта добијемо када наследимо неки члан базне класе?

То је као да се дефиниција тог члана налази у дефиницији изведене класе.
Практично, наслеђеном члану базне класе можемо приступати навођењем само његовог имена.
Он је равноправан са члановима који су дефинисани у самој изведеној класи.

Одавде, између осталог, следи:
како нам, по правилу, **атрибути** класа (што ће се односити и на базну) имају приступни атрибут `private`, они нису наслеђени чланови базне класе.
За њихову иницијализацију користи се позив конструктора базне класе на начин који ће ускоро бити описан, док ако су нам у изведеној класи потребне њихове вредности, морамо их дохватити позивом одговарајућих метода `get*()` базне класе.

Скриверни чланови класе

Може се (случајно) десити да постоји атрибут са истим именом и у базној и у изведеној класи (тип и приступни атрибут у овој причи нису од значаја). Тада је атрибут базне класе скривен истоименим атрибутом изведене класе. Навођењем само имена атрибута обраћамо се атрибуту изведене класе. Да бисмо референсирали истоимени наслеђени атрибут базне класе, морамо његово име квалификовати кључном речју `super`.
Ово није пожељан приступ при дефинисању атрибута изведене класе. Препорука је да се њихова имена разликују од оних из базне класе.

Наслеђивање метода

Методи базне класе (са изузетком конструктора) наслеђују се по истом принципу као и атрибути.
Конструктори базне класе никада се не наслеђују!
То значи да их у конструктору изведене класе не можемо звати на начин на који то чинимо нпр. у тест-класи (навођењем њиховог имена и листе аргумената унутар пара облих заграда). Али то не значи да их не можемо звати уопште.
Штавише, приликом прављења сваког новог објекта изведене класе
НЕОПХОДНО ЈЕ У КОНСТРУКТОРУ ИЗВЕДЕНЕ ПОЗВАТИ КОНСТРУКТОР БАЗНЕ КЛАСЕ, који ће извршити иницијализацију атрибута базне класе присутних у објекту изведене класе.

Ако не позовемо конструктор базне класе из конструктора изведене, компајлер ће покушати то да уради за нас.

Позив конструктора базне класе у конструктору изведене врши се помоћу кључне речи `super` као имена метода и то мора да буде прва наредба у телу конструктора изведене класе.

Ако то није случај, компајлер ће имплицитно убацити позив подразумеваног конструктора базне класе:

```
super();
```

а уколико подразумевани конструктор не постоји у базној класи, то ће резултовати грешком при компајлирању.

Дакле, прва наредба у телу конструктора изведене класе је УВЕК позив конструктора базне класе (имплицитни позив подразумеваног конструктора од стране преводиоца или одговарајући експлицитни позив).
Ако се позива конструктор базне класе различит од подразумеваног, наводи се и одговарајући број аргумената у складу са дефиницијом конструктора:

```
super(...);
```

Пример 1 (paket nasledjivanje, klase Povrs, Krug):

Класа `Krug` наслађује само методе `getCentar()` і `toString()` из класе `Povrs`, јер се конструктори, копи-конструктор и `private` инстанцна променљива (нестатички атрибут) `centar` не наслеђују.

Иако се променљива `centar` не наслеђује, она јесте део објекта изведене класе, па мора бити адекватно иницијализована:

1. у конструктору класе `Krug` позивом конструктора базне класе
2. у копи-конструктору класе `Krug` позивом копи-конструктора базне класе (који мора да постоји) или позивом конструктора базне класе

У другом случају, пошто је копи-конструктор базне класе дефинисан тако да му је аргумент типа базне класе (`Povrs`), као стварни аргумент, приликом позива, може да му се проследи референца на објекат типа `Povrs`, али и на објекат произвољне потклассе класе `Povrs` (а самим тим и класе `Krug` која је из ње изведена).
Варијанта са конструктором захтева да у базној класи постоје дефинисани public методи `get*()` за сваку `private` инстанцну променљиву (у овом случају метод `getCentar()` за променљиву `centar`).

Кастовање
Када се копи-конструктору базне класе `Povrs` проследи референца на објекат класе `Krug` (или било које друге потклассе), врши се *кастовање* у базни тип `Povrs`.
Кастовање представља објекат као да је неког другог типа.
Могуће је кастовати објекат у тип (директне или индиректне) натклассе или потклассе, дакле само навише и наниже кроз неку хијерархију класа.
Кастовање навише врши се имплицитно (компајлер га убацује сâм, када је потребно). Управо се то дешава у примеру са позивом копи-конструктора базне класе `Povrs`.

Пример 2 (paket nasledjivanje2, klase Povrs, Krug, Pravougaonik, Kvadrat):

Класи `Povrs` додати су подразумевани конструктор, `public` методи `povrsina()` и `rastojanjeDoCentra(Povrs)`, који ће заједно са методима `getCentar()` и `toString()` бити наслеђени у изведеним класама.
Из класе `Povrs` изведена је још једна класа, `Pravougaonik`, а из ње класа `Kvadrat`.

Метод `povrsina()` није дефинисан у базној класи (враћа вредност 0), јер се површина рачуна различито за различите типове површи. У изведеним класама је адекватно предефинисан, за круг по формули $P=r^2\pi$, а за правоугаоник по формули `P=a*b`.

Метод `rastojanjeDoCentra(Povrs)` рачуна растојање од центра текуће до центра дате површи. Параметар метода је типа базне класе `Povrs`, те ће као стварни аргумент моћи да се проследи референца на објекат типа `Povrs`, али и референца на објекат произвољне изведене класе (`Krug`, `Pravougaonik`, `Kvadrat`), при чему се онда врши имплицино кастовање навише.

У класи `Pravougaonik` дефинисан је и метод `dijagonala()`, за рачунање дијагонале текућег правоугаоника, као и метод `opisaniKrug()`, који прави и враћа круг описан око правоугаоника.
Ови методи ће бити наслеђени у класи `Kvadrat`, јер су декларисани као `public` и нема потребе да се предефинишу у тој класи, јер се понашају исто и за квадрате.

Класа `Kvadrat` нема додатних нестатичких атрибута, јер квадрат је правоугаоник где су странице `a` и `b` једнаке дужине. Неопходно је једино да се дефинишу конструктори и да се предефинише метод `toString()`.

Приступ ненаслеђеним члановима базне класе у изведеној класи

Методи базне класе који су наслеђени у изведеној класи (имају атрибут приступа различит од `private` или су без атрибута приступа у базној класи) могу да приступе свим члановима базне класе унутар изведене класе, чак и онима који нису наслеђени.

2 Predefinisanje (overriding) metoda bazne klase

Може се дефинисати метод у изведеној класи који има **исти потпис као и неки метод базне класе**.

Претпоставка је да су нама, као и обично, методи у класама `public`, па ће ова прича важити. Иначе, прича важи када приступни атрибут метода у изведеној класи није рестриктивнији од приступног атрибута метода базне класе који има исти потпис.

Када позовемо метод просто навођењем његовог имена, позиваће се метод изведене класе, а ако желимо да позовемо базну верзију метода, морамо користити кључну реч `super`.

Пример:

Метод `toString()` класе `Krug` предефинише метод `toString()` класе `Povrs`, који се може позвати у облику `super.toString()` како би се генерисала `String`-репрезентација базног дела круга.

Избор приступних атрибута за чланове базне класе

- методе који чине спољни интерфејс класе треба декларисати као `public`. Све док нису предефинисани у изведеној класи, биће у њој доступни директно преко свог имена (иначе им се приступа преко `super`)
- препорука је да се инстанцне промењиве (нестатички атрибути) декларишу као `private`, а да се обезбеде `public` методи за приступ и измену
- константни атрибути (статички или нестатички) могу да буду `public` ако су намењени некој општој употреби
- `protected` чланови се најчешће користе у пакетима класа са неограниченим приступом свим члановима из истог пакета, при чему је приступ ван пакета ограничен на потклассе

3 java.lang.Object – универзална суперкласа

Уколико се експлицитно не наведе да је класа изведена из неке друге, подразумева се да је изведена из класе `java.lang.Object`, тако да је `Object` директна или индиректна суперкласа сваке друге класе.
Тиме се обезбеђује да класе које дефинишемо наслеђују чланове класе `Object`. За сада је од значаја `public` метод `toString()`, који враћа стринг-репрезентацију објекта, и о коме је било речи и раније.
Подсећања ради, стринг-репрезентација се генерише у подразумеваном облику:

```
<puno_kvalifikovano_ime_klase>@<hash_code_objekta(heksadekadno)>
```

Пуно квалификовано име класе укључује и име пакета коме класа припада. О хеш кодовима објеката ће бити речи у наставку курса. Ради се о вредности која се генерише на основу вредности референце објекта позивом одговарајућег метода (`hashCode()`).

Поред метода `toString()`, размотрићемо и `public` методе `equals()`, `hashCode()` и `getClass()`.
Документација је доступна на адреси:
https://docs.oracle.com/javase/8/docs/api/java/lang/Object.html

4 Задаци за вежбу

1. Написати базну класу `Zivotinja` која садржи један нестатички атрибут `vrsta`, који представља врсту којој животиња припада. Атрибут декларисати тако да приступ ван класе буде онемогућен.
Обезбедити:
- подразумевани конструктор (вредност атрибута vrsta поставља се на „`nepoznata`“)
- конструктор који очекује све потребне аргументе
- копи-конструктор
- метод за прављење стринг-репрезентације; нпр ако је врста пас, у облику:
  `"Ovo je pas"`

Написати изведену класу `Pas` која од атрибута додатно садржи *име* пса и његову *расу*. Атрибут декларисати тако да приступ ван класе буде онемогућен.
Обезбедити:
- конструктор који очекује један аргумент (име пса), а расу поставља на подразумевану вредност „`sarplaninac`“
- конструктор који очекује све потребне аргументе
- копи-конструктор
- метод за прављење стринг-репрезентације; нпр у облику:
  `"Ovo je pas Lesi, sarplaninac"`

Написати тест класу `TestZivotinje` и у њој направити објекте класе `Pas` (коришћењем оба конструктора), исписати податке о њима и позвати метод за оглашавање пса. Направити и копију једног од објеката и исписати његове податке, као и начин његовог оглашавања.

2. У претходном задатку потребно је урадити следеће:

а) допунити дефиницију изведене класе Pas методом за оглашавање пса:
   public void oglasiSe()
   који исписује како се пас оглашава (нпр. "AV AV")

б) Написати класу `Macka` изведену из класе `Zivotinja` која од атрибута додатно садржи *име* мачке и њену *расу*. Атрибут декларисати тако да приступ ван класе буде онемогућен.
   Обезбедити:
   - конструктор који очекује један аргумент (име мачке), а расу поставља на подразумевану вредност „`persijska`“
   - конструктор који очекује све потребне аргументе
   - копи-конструктор
   - метод за прављење стринг-репрезентације; нпр у облику:
     `"Ovo je macka Srecko, sijamska"`
   - метод за оглашавање мачке
     public void oglasiSe()
     који исписује како се мачка оглашава (нпр. "MIJAUUU")

в) Написати класу `Pudlica` изведену из класе `Pas` која нема никакве додатне атрибуте и методе.
   Једино што је потребно дефинисати јесте одговарајући конструктор и копи-конструктор.

Тест класу допунити прављењем једног објекта класе `Macka` и једног објекта класе `Pudlica`, исписати податке о њима и позвати метод за њихово оглашавање.