Наслеђивање

незнатна модификација дела к'о бајаги скрипте (Ово је прича коју врло радо причам)

Наслеђивање је поступак којим се из постојећих праве (изводе) нове класе.
Нова класа зове се изведена или потклаcа, док је постојећа класа базна или суперкласа.
Нова и постојећа класа налазе се у специфичном односу.

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 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`.
Посебно је важно нагласити да је **приликом креирања 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.

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?

Па 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.

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 реферисали истоимени наслеђени атрибут базне класе, морамо његово име квалификовати кључном речју `super`.

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 креирања 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 бити 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.

Предефинисање (overriding) метода базне класе

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`.