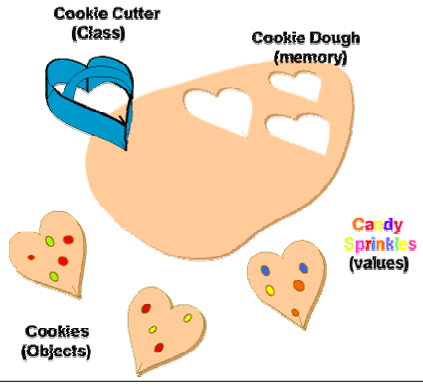
Objekti

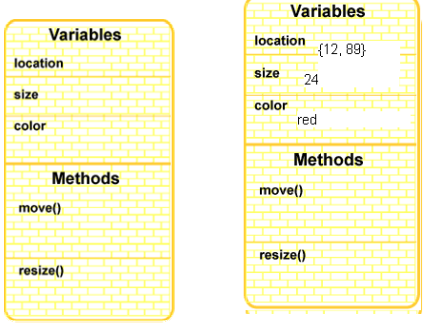
Objektno pogramiranje

Dogodkovno programiranje

Java in objekti

- Razred (class) je opis vrste objekta (načrt, kako naj bo objekt videti) – opis ideje objekta
- Primerek razreda (instanca) – konkretni objekt





class Neki {... new Neki(12, 89, 24, “red”,

Programiranje v Javi

- Sestavljanje razredov
  - Opis lastnosti objektov
  - Opis metod (“znanja” objektov)
    - Ustvarjanje objektov in njihova uporaba
    - "Ukazovanje" objektom, kaj naj počnejo
    - Objekt za izpolnitev določene naloge potrebuje druge objekte in njihove metode
    - Proženje dogodkov

Objekt in ime spremenljivke

- NekiObjekt a;
  - a je naslov kjer bo objekt (referenca na objekt)
- new NekiObjekt();
  - V pomnilniku se je naredil objekt tipa NekiObjekt() / po pravilih, ki so določena z opisom razreda NekiObjekt
- a = new NekiObjekt();
  - a kaže na novo ustvarjeni objekt

Zgled – želvja grafika

- Razred `Turtle`
  - Kako je videti "splošna" želva
  - Ko je narejen razred `Turtle` nimamo še NOBENE konkretne želve, le NAČRT, kako naj bodo želve videti
  - `new Turtle();` // ustvari se konkretna želva
    - instanca razreda `Turtle`
    - Objekt `zelvak` je primerek objekta iz razreda `Turtle`
- Kakšne lastnosti ima vsaka želva:
  - Smer
  - Položaj (x koor, y koor)
- "Stanje" te želve (v trenutku kreacije)
  - Smer (v desno (na vzhod))
  - Na sredi zaslona – v točki (380, 300)

Zgled – želvja grafika

- Kaj znajo vse želve (na katere metode se odzovejo):
  - `barva` // nastavi barvo sledi
  - `narisi` // obrni se za določen kot in pojdi naprej za določeno število korakov
  - `spi` // zaspi za nekaj trenutkov (in z zelvo tudi ves svet)
  - …
- "Ukazovanje"
  - ime_objekta.metoda(parametri)
  - `zelvak.barva(mojaBarva);`
  - `zelvak.spi(2000); // zaspi za 2 sekundi`
  - `zelvak.narisi(90, 30);` // obrni se za 90 st v levo in pojdi naprej za 30

Dedovanje

- "razširjanje" objektov
- `extends`
- Naredimo načrt za razred pametnih želv
  - Znajo risati kvadrate
- `public class PametnaZelva extends Turtle`
- Vsaka pametna želva "zna" vse, kar zna razred `Turtle` in morda še kaj

Dedovanje

- Izpeljava novih razredov iz obstoječih
- Uvajanje dodatnih metod
- Lahko tudi dodatne lastnosti (podatki)
- Primer:
  - matrike
    - Seštevanje, odštevanje, množenje
  - Kvadratne matrike / `class KvMatrike extends Matrike`
    - Ker je razred izpeljan – ni potrebno na novo pisati metod za seštevanje, odštevanje, množenje
    - Možne dodatne operacije
      - Inverz, deljenje, …

Dedovanje

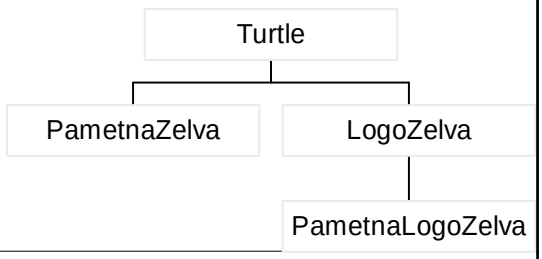
- Hierarična zgradba razredov
- Vrhnji objekt
  - Object
  - Iz njega izpeljani vsi drugi rezredi
  - Če ne napišemo `extends` …
  - `extends Object`
- "specializacija" objektov

Razred Turtle

- Konstante (razredne):
  - RDECA, MODRA, CRNA, SIVA, RUMENA, ROZA, ORANZNA, ZELENA, VIJOLICNA, BELA
- Turtle.BELA
- /** Nastavimo barvo pisanja na dano barvo */
- `public void barva (Color danaBarva)`
- /** Na mestu želve napiši sporočilo */
- `public void napisi (String sporocilo)`
- /** Pocakaj za cakaj milisekund. */
- `public void spi (int cakaj)`
- /** Obrni se levo za levo stopinj; premakni se za naprej korakov.*/
- `public Turtle premakniSe (double levo, double naprej)`
- /** Obrni se levo za levo stopinj; premakni se za naprej korakov in puscaj sled.*/
- `public Turtle narisi (double levo, double naprej)`

“Pametna” Logo želva

- `public class PametnaLogoZelva extends LogoZelva`
- Dodatne metode
- Hierarhija objektov:



“Prava” Logo želva

- Logo želve dejansko pri premikanju ne puščajo vedno sledi za sabo
- Pisalo: dvignjeno/spuščeno
- Naredimo tovrstno logo želvo
- Kam jo uvrstiti v našo hierarhijo objektov?
- Ker nam metode iz LogoŽelva ne ustrezajo, izpeljimo kar iz razreda Turtle!

this

- Dostop do objekta znotraj načrta
- `this.ime`
  - Dostopamo do polja/lastnosti ime "tekočega" objekta
- `this.ploscina()`
  - Na trenutnem objektu izvajamo metodo ploscina
- `this` lahko izpustimo

"Prava" Logo želva

- `public class PravaLogoZelva extends Turtle`
- Potrebujemo dodatno stanje!
  - `private boolean pisalo = true;`
    `// pisalo je spusceno`
- Metode, ki dvigajo/spuščajo pisalo
- `PravaLogozelva.java`

Ključni pojmi OOP

- Združevanje (enkapsulacija)
  - Podatki in metode so združeni v predmetu
  - Ožje: “ščitenje” podatkov/metod pred neposrednim dosegom
- Dedovanje (inheritance)
  - Iz razreda izpeljemo nov razred
  - “prevzem” lastnosti in metod
- Večličnost (polimorfizem)
  - Objekti se lahko obnašajo v različnih vlogah

Primitivni tipi / objekti

- Primitivni tipi
- Osnovni tipi
  - int
  - double
  - boolean
  - char
  - ...
- Ne moremo narediti sami!
- Objekti
  - "dobimo z Javo"
    - String
    - Integer
    - Color
    - ...
  - Naredimo sami
  - Vsi nadgradnja
  - Če ni extends ... : extends Object

Objekti

- Imajo neka stanja (podatki)
- Nekaj znajo:
  - Reagirajo na določene metode
- Stanja: spremenljivke
- Znanje: metode

Primer objekta

- EMŠO
- Podatki (stanja)
  - Številka
- Metode
  - Povej spol
  - Povej rojstni datum
  - ...

EMŠO

```
public class EMSO {
   String stevilka;
   public boolean zenska
   { // ali je oseba s to stevilko zenska
      ...
   }
   public Datum rojDan
   { // roj. dan osebe s to stevilko
      ...
   }
   ...
}
```

Opis, kako zgleda poljuben EMSO objekt

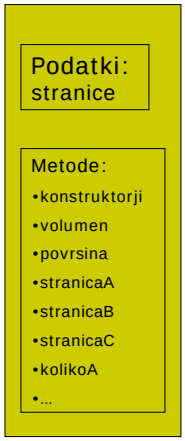
EMŠO

```
EMSO jaz = new EMSO();
jaz.stevilka = "2102962500087";
System.out.println(jaz.zenska());
```

Uporaba konkretnega EMSO objekta

Razred Kvader

- Ne pozabi: razrede poimenujemo z veliko začetnico!
- Podatki,
  - komponente, polja, ...
  - opisujejo stanje objekta
- Metode
  - Konstruktorji (posebne metode)
    - Kaj se zgodi ob new
  - "znanje" objektov tega razreda
    - volumen, površina, ...
  - Metode za ravnanje s podatki
    - Nastavi podatek
    - Vrni podatek
  - toString



Podatki

- Stanja, ki spadajo k objektu
- Način dostopa do stanj:
  - Privatne spremenljivke: `private`
  - `private int x;`
- Privatne spremenljivke
  - Dosegljive le znotraj razreda
  - Nihče izven razreda jih ne vidi!
  - `public` spremenljivke lahko vidijo vsi!
  - Enkapsulacija na nivoju podatkov:
    - način hranjenja podatkov je "zasebna" zadeva objekta

Kvader - podatki

- Podatki:
  - Stranice: a, b, c
  - Odločitev o načinu predstavitve
    - Kako bo imel objekt predstavljene stranice
    - Ena od možnosti:
      - 3x spremenljivka tipa int
    - Druga možnost
      - Tabela s tremi polji
    - Tretja možnost: ….
- `private int a; // stranica a`
- Kvader_0.java

Zakaj enkapsulacija?

- Zakaj `private`?
- testKvader1.java
- Kje so težave?
- Če želimo uporabljati `imeObjekta.imePodatka`
  - Poznati način predstavitve podatka
  - Ni enostavne možnosti kontrole pravilnosti podatkov!
  - Težave, če kasneje ugotovimo, da bi bilo potrebno predstavitev spremeniti!

Zakaj enkapsulacija?

- Način predstavitve uporabnika NE zanima – s `private` mu preprečimo dostop
- Kako potem do podatkov:
  - Pripravimo ustrezne metode
- Če se podpisi metod ne spreminjajo – ni težav s spreminjanjem razreda
  - Vsi programi, ki uporabljajo objekte določenega razreda, si podatke izmenjujejo le preko metod
  - Če se klic ne bo spremenil, je vseeno, kako je spremenjen razred!

Sprememba interne predstavitve podatkov

- Namesto treh spremenljivk – niz
- Sprememba metod
- Ali je potrebno popraviti program, kjer smo uporabljali razred?
- Ne, če nismo spremenili imen metod, tipov in števila parametrov, tipa rezultata, ki ga vrne metoda

Private / public

- `Cas kdaj = new Cas();`
- `kdaj.ura = 12;`
- Ne gre!
  - testCas.java
  - Je razlog res le napačno poimenovanje?
  - testCas1.java
- Razred `Cas1`, kjer so spremenljivke `public`
- `Cas1 kdaj = new Cas1();`
- `kdaj.h = 12;`
- Gre! testCas2.java

Private / public

- Kaj, če se premislimo glede določenih omejitev
- Npr. – ugotovimo, da bi bilo dobro zagotavljati "pravilnost" časa.
- Popraviti (pregledati) vse programe, ki uporabljajo `Cas1`, če nismo kje neposredno nastavili spremenljivke na napačno vrednost
- Če uporabljamo razred `Cas`, kontrolo vstavimo le v metode tega!

Private / public

- Kaj, če se premislimo glede poimenovanja ali načina, kako objekt hrani podatke
- Denimo, da bomo raje čas hranili v obliki `"hh:mm:ss"`
- Popraviti vse programe, ki uporabljajo `Cas1`
- Če uporabljamo razred `Cas`, spremenimo le tega!

Public / -

- Če vsa določila spustimo
- Dostop na nivoju paketa (package)
- Dejansko (naš način uporabe)
  - Isto kot `public`!
- Cas2.java, testCas3.java
- Obstaja še:
  - `protected`
  - Vidno tudi v razredih, ki razširjajo dani razred
  - … `class R1 extends R2`
    - V razredu `R1` vidne `public` in `protected` spremenljivke objektov razreda `R2`
    - Spremenljivke objektov razreda `R2` z določilom `private` v `R1` niso vidne

Public / private za metode

- Enaka pravila (glede vidnosti) so tudi pri metodah (določila `public`, `private`, …)
- `private int bla(…)`
  - Metodo lahko uporabljamo le znotraj razreda!
- `public int ble(…)`
  - Metodo lahko uporabljamo kjerkoli!

public vs. private vs. protected

- `public`: javno, vsi dostopajo
- `private`: le v okviru razreda
  - Tudi razred, ki je razširitev tega razreda ne more do komponent
  - Včasih preveč omejujoče
- `protected`
  - Dostopamo lahko v okviru razreda IN izpeljanih razredov (posredno in neposredno)

Dostop

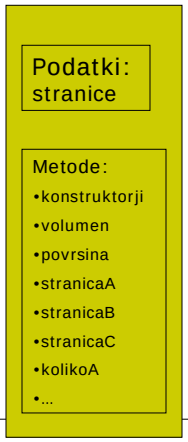
- Pri tvorbi razreda
- Spremenljivke z dostopom `private`
- Metode na voljo uporabnikom: `public`
- "Pomožne" metode: `private`
- `Protected` zaenkrat ne bomo uporabljali
- Kvader_0.java

Kvader – metode za delo s podatki

- Za podatke za katere je smiselno, da jih uporabnik vidi:
  - Metode, ki povejo vrednost podatka
  - `povejA(), …`
- Za podatke za katere je smiselno, da jih uporabnik spremeni:
  - Metode, ki spremenijo vrednost podatka
  - Možnost kontrole
  - `stranicaA(), …`
- Kvader.java (oglejmo si komentarje!)

Razred Kvader

- Ne pozabi: razrede poimenujemo z veliko začetnico!
- Podatki,
  - komponente, polja, ...
  - opisujejo stanje objekta
- Metode
  - Konstruktorji (posebne metode)
    - Kaj se zgodi ob new
  - "znanje" objektov tega razreda
    - volumen, površina, ...
  - Metode za ravnanje s podatki
    - Nastavi podatek
    - Vrni podatek
  - toString



Kvader - podatki

- Podatki:
  - Stranice: a, b, c
  - Odločitev o načinu predstavitve
    - Kako bo imel objekt predstavljene stranice
    - Ena od možnosti:
      - 3x spremenljivka tipa int
    - Druga možnost
      - Tabela s tremi polji
    - Tretja možnost: ….
- `private int a; // stranica a`

Kvader – metode za delo s podatki

- Za podatke za katere je smiselno, da jih uporabnik vidi:
  - Metode, ki povejo vrednost podatka
  - `povejA(), …`
- Za podatke za katere je smiselno, da jih uporabnik spremeni:
  - Metode, ki spremenijo vrednost podatka
  - Možnost kontrole
  - `stranicaA(), …`
- Kvader.java (oglejmo si komentarje!)

Kvader - konstruktor

- Posebna metoda
- Brez tipa
- Za nastavitev začetnih vrednosti
- Ime kot je ime razreda
- Kličemo jo skupaj z `new`
- `public Kvader() { a = 1; b = 1; c = 1;}`
- Klic: `new Kvader();`
- Lahko več konstruktorjev

Konstruktorji

- Posebne metode
- Ne moremo klicati posebej
  - Le ko tvorimo objekt
  - `new`
  - Za vzpostavitev začetnega stanja
- Enako ime kot razred
- Nimajo tipa (tudi `void` ne!)
- Ni stavka `return`

Kvader - konstruktorji

- Standardno – konstruktor brez parametrov
- Naredi "privzeti" objekt
- Denimo – kvader s stranicami
  1 x 1 x 1
- ```
  public Kvader(){
    a = 1;
    b = 1;
    c = 1;
  }
  ```
- Kaj pa, če bi uporabniku radi ponudili več možnosti "začetnega" kvadra?
  ```

Več konstruktorjev

- Več konstruktorjev:
  - Več metod z enakim imenom
  - Je to možno?
- Preobteževanje
  - Overloading
  - Več metod z enakim imenom
- Metode se morajo razlikovati ne v imenu, ampak podpisu
- Podpis metode
  - Podpis: ime + število in tipi parametrov!
  - Return tip NI del podpisa!

Kvader.java

- Konstruktor, ki naredi kocko
- Konstruktor, kjer podamo stranice
- Konstruktor, ki naredi kopijo objekta

O konstruktorjih

- http://www.javaworld.com/jw-10-2000/jw-1013-constructors.html

Preobtežene metode

- Tudi "navadne" metode (ne le konstruktorji) so lahko preobtežene
- Možnost, da imamo enako poimenovano metodo, ki pa sprejema parametre različnega tipa
- Metoda ploscina, ki kot parameter dobi objekt iz razreda `Kvadrat`, `Krog` ali `Pravokotnik`
  - Uporaba: `ploscina(bla),` kjer je bla lahko objekt tipa `Kvadrat`, `Krog` ali `Pravokotnik`
- Tri metode, z enakim imenom, a različnimi podpisi
  - Enostavnejša uporaba kot tri metode z različnimi imeni ali ena metoda, kjer preverimo, kakšen objekt smo dobili
  - Enostavneje, če dobimo še četrti tip objekta – v "glavno" kodo ni potrebno posegati – naredimo novo metodo z istim imenom (`ploscina`) in s parametrom katerega tip je novi objekt
  - Klic je še vedno `ploscina(bla)`!

Primeri preobteženih metod

- `Math.abs()`
- Uporabimo lahko na tipu `double` in na tipu `int`
- Dve metodi!
- Math.java

Kvader - toString

- Kaj se zgodi, ko potrebujemo namesto objekta niz
- Npr. `System.out.println(new Kvader());`
- Predefiniranje metode iz razreda `Object`

Prekrite metode

- V predniku (neposrednem – `extends` … ali posrednem) definirana metoda nam ne ustreza
- Predefiniranje: prekrite metode
- Overriding
- Enak podpis metode kot pri predniku – velja naša definicija
- Predefiniranje lahko tudi preprečimo
  - O tem kdaj drugič …

Iz razreda Kvader izpeljimo razred Kocka

- ? podatki
  - Obstoječi zadoščajo (celo preveč jih je)
  - Se jih lahko "znebimo"?
- Metode:
  - Konstruktorji
    - Pripravimo nove ustrezne konstruktorje
  - `volumen`, `povrsina`
    - Lahko ostaneta
  - Metode za vračanje podatkov o stranicah?
    - V redu – objekt mora "paziti" na pravilnost podatkov
    - Morda dodamo še metodo `povejStranico`
  - Metode za nastavljanje podatkov
    - Obstajajo `stranicaA`, `stranicaB`, `stranicaC`
    - Uporabnik jih lahko uporabi in s tem povzroči zmedo, saj popravi le en podatek!
    - Nujno predefiniranje!
  - toString
    - Napišemo na novo!

Razred Math

- Kaj opazimo?
  - Tip metod?
- Static
  - statična metoda
  - Razredna metoda
- Ni vezana na objekt
- Uporabimo lahko, tudi če ni nobenega objekta tega razreda
- Klic
  - `ImeRazreda.imeMetode(parametri)`

Statične komponente

- `static`
- Obstajajo neodvisno od obstoja objekta tega razreda
- Le en primerek za cel razred!
- `Math.PI`
- Uporaba
  - Različne konstante
  - Enolična identifikacija
    - Objekt naj ima svojo serijsko številko
    - Denimo, da so te številke zaporedne
    - Vedeti, katera je bila dodeljena zadnja
    - Zadnja dodeljena številka: ni lastnost objekta, ampak cele skupine objektov
  - Vedno, ko je določen podatek skupen za vse objekte tega razreda

Statične komponente

- `public` ali `private` (odvisno od željenega načina dostopa)
- Dostop
  - ime_objekta.ime_polja ali ime_razreda.ime_polja
  - Slednje boljše (saj ta komponenta ni vezana na objekt!)
  - Prvi
- Zgled:
  - Štetje objektov

Ustvarjanje razredov: povzetek

- Konkretni objekt je primerek (instanca) iz nekega razreda
- Nastane z `new`
  - Izjema: `String`
  - Ni izjema, le drugačen zapis!
  - Podobno tabele z inicializacijo
- Posamične lastnosti objekta
  - Dostop: `private`
  - Uporabnika ne zanima način hranjenja lastnosti objekta
- Metode za delo s podatki/lastnosti
  - `vrniPodatek`, `nastaviPodatek`
- Konstruktorji
  - Kaj se zgodi ob kreaciji novega podatka

Ustvarjanje razredov: povzetek

- toString
  - Da objekt pretvorimo v niz
  - Za "predstavitev" objekta
- Metode
  - Namenjene uporabnikom: public
  - "interne" (pomožne) metode: private
- Več metod z enakim imenom
  - Preobteževanje (overloading)
- Razredi nastajajo z dedovanjem
- Prevzem vseh lastnosti in metod "starševskega" razreda
- Predefiniranje metode prednika
  - Prekrivanje
  - Overriding

Kaj zvedo o OO študenti 2. letnika FMF – visokošolski študij

- 2003/4
  - http://haka.fmf.uni-lj.si/pra-racunalnistvo-1/lekcija12/index.html
  - http://haka.fmf.uni-lj.si/pra-racunalnistvo-1/lekcija13/index.html
- 2002/3
  - http://haka.fmf.uni-lj.si/pra-racunalnistvo-1/arhiv-2002/gradivo/lekcija_19/index.html

Nekaj povezav

- Java Tutorial (preveden v slov.)
  - http://storm.uni-mb.si/vaje/os/tecaj_jave/java/objects/index.html
- (slo) http://www.jugsi.org/dokumentacija/Knjiga/2/21.html
- http://sep.stanford.edu/sep/jon/family/jos/oop/index.html
- http://www.scism.sbu.ac.uk/jfl/jflcontents.html#chap1.html
- http://www.javaworld.com/javaworld/jw-04-2001/jw-0406-java101.html