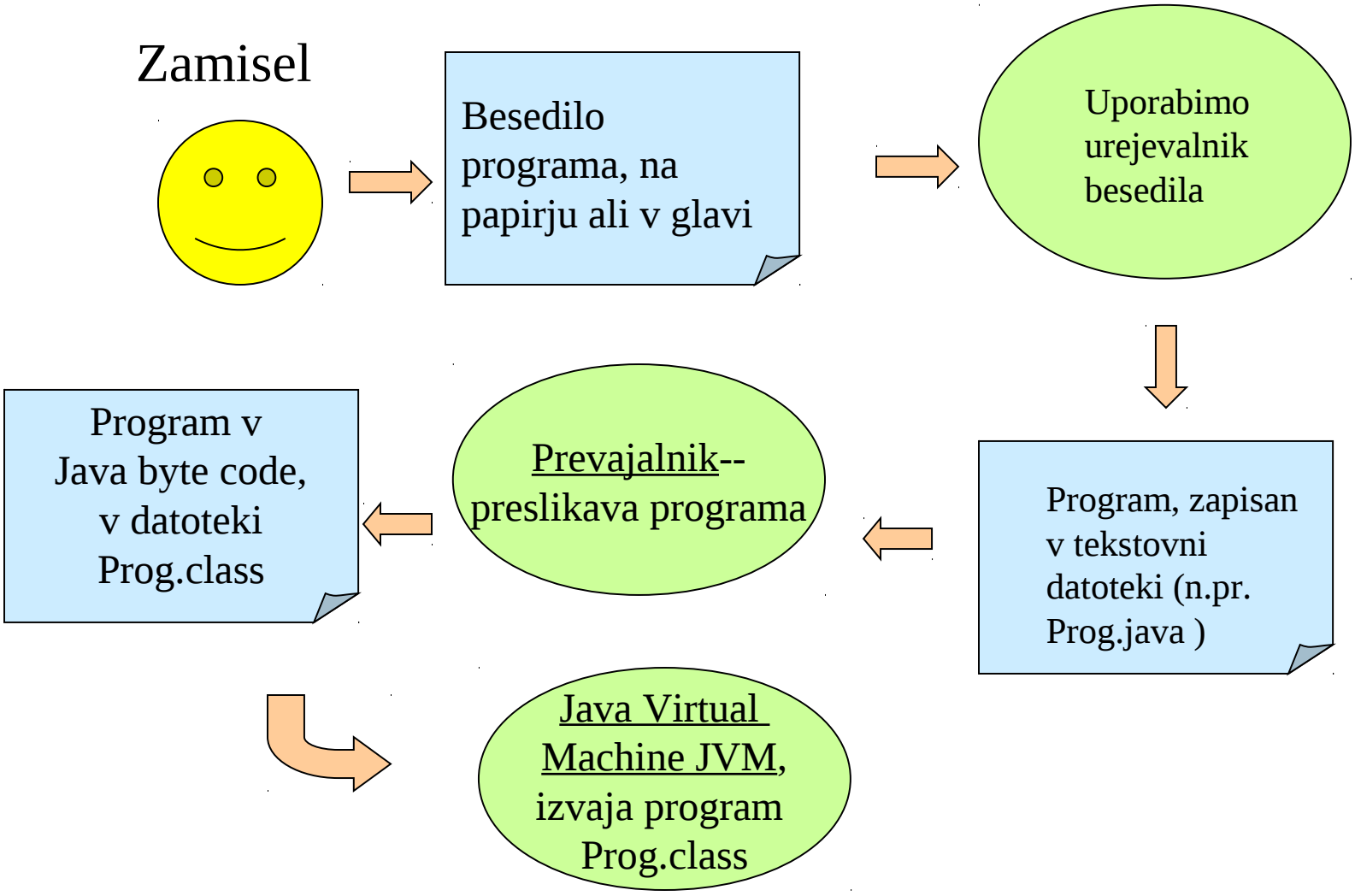
Razvoj programov

Proces programiranja



Programiranje in reševanje problemov

Koraki za pisanje programov
upoštevajo korake za reševanje problemov

- Definiranje
- Razmislek
- Načrtovanje
- Izvedba
- Popravki, izboljšave

Popravki, izboljšave

- Ali program ustreza specifikacijam? Da
- Ali so rezultati smiselni? Preverimo ročno
- Je program robusten in prijazen?
  (vsebuje komentarje za opis vhoda in izhoda)
- Ali lahko program še kako izboljšamo? Hmmm.

Primer priprave preprostega razreda

- Napišimo program, ki izpiše kvadrat s pomočjo primernega izpisnega znaka.
- Nato narišimo več različnih kvadratov.

- Razčlenimo problem:
  - Pomniti moramo velikost n in znak
  - Zanka
  - izpis vrstico za vrstico
  - izpis n znakov v isti vrstici

```
******    @@@@
******    @@@@
******    @@@@
******    @@@@
******
******
```

Kodiranje v Javi

```
class Square   {
    int size = 2;                    // Store for size of square
    char pattern = '*';              //  Store for character
    void draw()   {
        for(int row = 1; row <= size; row++)  {
            // to do ‘n’ (size) rows
            for(int col = 1; col <= size; col++)      //  loop to do one row
                System.out.print(pattern);            //  print one char at a time
                System.out.println();                 //  end of line move down
        }
        System.out.println();
    }  // draw method
}   // Square class
```

Uporaba razreda Square

```
public class UseSquare  {
   public static void main (String[ ] args)  {
       Square s1 = new Square();  //  the class square is in the
                                  //  default directory
       s1.draw( );                //  use the draw square defaults
       s1.size = 5;               //  Change the size
       s1.draw( );                //  Draw new square
       s1.pattern = '&';          //
       s1.draw( );                //
   } // main method
 } // UseSquare class
```

Pisec razreda, uporabnik, končni uporabnik

Pisec razreda (Class provider)

Programer-uporabnik (Class User)

Uporabnik programa (End User)

V tem pa je že bistvo objektno usmerjenega programiranja (OOP)

Programerska orodja

Dober urejevalnik teksta

lcc-win32 (prevajalnik, povezovalnik C)

Integrirano razvojno orodje za Javo

- Urejevalnik
- prevajalnik
- razhroščevalnik
- dokumentacija

Integrirano razvojno orodje za C

- Urejevalnik
- prevajalnik
- razhroščevalnik
- dokumentacija

decompiler (jave)

Eclipse(Boljše itntegrirano razvojno orodje)

Notepad++ v3.1

drjava

DEV C++

Code::Blocks

eclipse

Notepad++ - C:\INTERNO_RACUNALNISTVO\STUFF_P2\PRIMERI\SteviloEnic.java

Datoteka Urejanje Najdi Pogled Oblika Dokument Jezik Nastavitve Macro Run Plugins ?

SteviloEnic.java

```java
import java.io.*;
//*****************************************************************
public class SteviloEnic {

    static int steviloEnic(int i) {
        /* funkcija vrne stevilo setiranih bitov */
        int j, num;
        num = 0;
        for(j=0;j<32; j++){
            if(( i % 2) ==1 ) num++;
            i = i /2;
        }
        return num;
    }

//*****************************************************************
    public static void main(String[] args) throws Exception {
        /* preizkus funkcije steviloEnic */
        int i, k;
        for (i=0; i<16; i++) {
            k = steviloEnic (i);
            System.out.println ("Stevilo setiranih bitov v " + i + " je " + k + " \n");
        }
    }
}
```

Java source file | nb char : 647 | Slovenian (Slovenia) | 0 | Dos\Windows | ANSI | INS

lcc-win32: A Compiler system for windows by Jacob Navia

based on the original screenplay of Dave Hanson and Chris Fraser: A portable C compiler

Here you can download the lcc-win32 C compiler system. The system is self contained: you do not need anything else to get started programming in C in the Win32 environment. You get

1. Code generator (compiler, assembler, linker, resource compiler, librarian)
2. Integrated development environment with editor, debugger, make file generation, resource editor, etc.
3. User manual and technical documentation.

All the binaries and the associated header files etc, are contained in an auto-installable executable that will create all the needed directory structure.

License:

This software is not freeware, it is copyrighted by Jacob Navia. It's free for **non-commercial** use, if you use it professionally you have to have to buy a licence.

Professional use is:

- Related to business (e.g you use it in a corporation)
- If you sell your software.

If you plan to use lcc-win32 in courses of programming in your University, contact us for special educational rates.

C:\WWW\SIS\PREDMETI\P2\GRADIVA\PRIMERI\SteviloEnic.java

File Edit Tools Project Debugger Language Level Help

New Open Save Close Cut Copy Paste Undo Redo Find Compile Reset Run Test Javadoc

SteviloEnic.java

```java
import java.io.*;
//*********************************************************************
public class SteviloEnic {

 static int steviloEnic(int i) {
  /* funkcija vrne stevilo setiranih bitov */
  int j, num;
  num = 0;
  for(j=0;j<32; j++){
   if(( i % 2) ==1 ) num++;
   i = i /2;
  }
  return num;
 }

//*********************************************************************
 public static void main(String[] args) throws Exception {
  /* preizkus funkcije steviloEnic */
  int i, k;
  for (i=0; i<16; i++) {
   k = steviloEnic (i);
   System.out.println ("Stevilo setiranih bitov v " + i + " je " + k + " \n");
  }
 }
}
```

Interactions | Console | Compiler Output

```
Compilation completed.
```

Compiler: JDK 6.0_2

☑ Highlight source

Toggling Debugger Mode 22:7

Java - BankAccountTests.java - Eclipse SDK

File Edit Refactor Source Navigate Search Project Run Window Help

Java Debug

Package Explorer | Hierarchy

- Banking
  - org.eclipse.banking
    - BankAccount.java
      - BankAccount
        - balance
        - deposit(BigDecimal)
        - getBalance()
        - withdraw(BigDecimal)
    - InsufficientFundsException.java
  - org.eclipse.banking.tests
    - BankAccountTests.java
  - JRE System Library [jre1.5.0_06]
  - JUnit 3.8.1

Outline

- org.eclipse.banking.tests
- import declarations
- BankAccountTests
  - testDeposit()
  - testWithdraw()
  - testOverdraft()

BankAccountTests.java | BankAccount.java

```
package org.eclipse.banking.tests;

import java.math.BigDecimal;

public class BankAccountTests extends TestCase {
    public void testDeposit() throws Exception {
        BankAccount account = new BankAccount();
        account.deposit(new BigDecimal(1000));
        account.deposit(new BigDecimal(100));

        assertEquals(new BigDecimal(1100), account.getBalance());
    }

    public void testWithdraw() throws Exception {
        BankAccount account = new BankAccount();
        account.deposit(new BigDecimal(1000));
        account.withdraw(new BigDecimal(100));

        assertEquals(new BigDecimal(900), account.getBalance());
    }

    public void testOverdraft() throws Exception {
        BankAccount account = new BankAccount();
        try {
            account.withdraw(new BigDecimal(100));
```

Problems | Javadoc | Declaration | Tasks | JUnit

Finished after 0.031 seconds

Runs: 3/3 Errors: 0 Failures: 1

- org.eclipse.banking.tests.BankAccountTests [Runner:
  - testDeposit
  - testWithdraw
  - testOverdraft

Failure Trace

junit.framework.AssertionFailedError: InsufficientFundsException should have been thr

at org.eclipse.banking.tests.BankAccountTests.testOverdraft(BankAccountTests.java:

org.eclipse.banking.tests - Banking

obratni prevajalniki

Test.java - cavaj

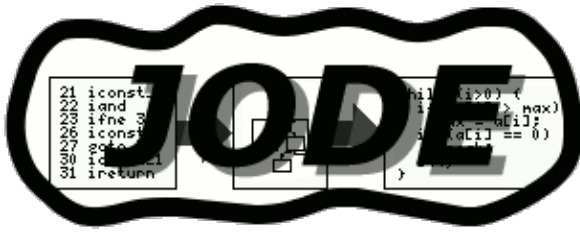
File Edit View Help

- Test
  - Test()
  - main(String args[])

```
import java.io.PrintStream;

class Test
{

    Test()
    {
    }

    public static void main(String args[])
    {
        System.out.println("Roger");
    }
}
```

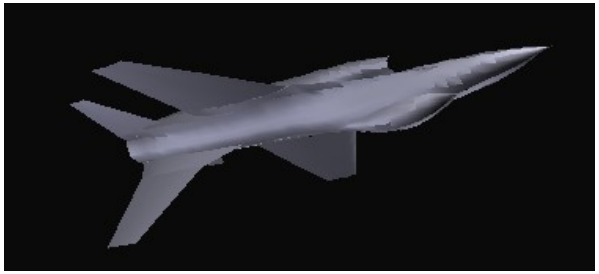
Ready | Ln 15 Col 1

Cavaj

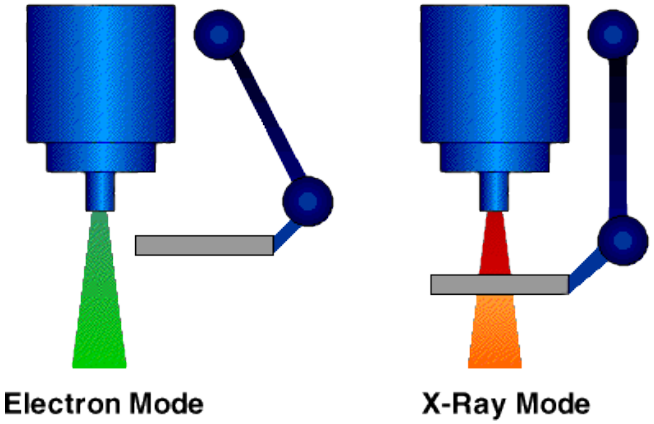


Preskušanje programov

Programska oprema lovskega letala F16 je imela napako, ki je povzročala, da se je letalo obrnilo na glavo, kadarkoli je prečkalo ekvator (napako so odkrili na simulatorju in v resnici ni nikoli prišlo do tega pojava)



Med 1985 in 1987 je računalniško krmiljena naprava Therac-25, uporabljena za obsevanje v bolnišnicah, povzročila zaradi programerske napake pacientom preveliko sevanje . Nekaj pacientov je celo umrlo.



Vrste programskih napak

- Sintaksne napake

  Neupoštevanje formata in strukture jezika, kar odkrije prevajalnik

- Napake v času izvajanja (run time)
  - Povzročijo lahko izjemne dogodke

- Logične napake

  Program se normalno zaključi, rezultati pa so napačni

Statična analiza programa

Statični programski analizatorji pregledajo kodo, ne da bi jo izvedli in ugotovijo:

- Napake v sintaksi
- kodo, ki ni dosegljiva,
- nedeklarirane spremenljivke,
- neinicializirane spremenljivke,
- neujemanje tipov parametrov,
- neklicane funkcije in procedure,
- spremenljivke, ki jih uporabljamo pred inicializacijo, možne prekoračitve mej,
- napačno uporabo kazalcev.

Zakaj preskušanje?

- Preskušanje programa na načrtovan in strukturiran način s ciljem, da probleme odkrijemo
- Koda, ki jo napišemo, običajno ni takoj pravilna
- Pravilnost težko dokažemo, posebno v zelo velikih programih

Preskušanje

- Preskušanje naj bi izločilo čimveč “hroščev”
- Preskušanje lahko dokaže prisotnost hroščev, ne pa njihove odsotnosti
- Pomaga nam preskušanje posameznih metod
- Pozitivno preskušanje: zagotoviti pravilno delovanje programa, če so podatki pravilni
- Negativno preskušanje: zagotoviti, da bo program znal rokovati z napačnimi podatki

Preskušanje

- Preizkusiti bi morali vse primere potekov programa
- Običajno ne moremo preskusiti vseh kombinacij vhodnih podatkov
- Izbrati moramo testne podatke, s katerimi bi našli večino napak v čim krajšem času
- Pomagamo si lahko z napotki in heuristiko

Problemi preskušanja

- Tehnični:
  - Preskušanje je težavno in dolgotrajno
  - Preskušanje moramo dobro pripraviti
  - Preskušanje moramo ponavljati (regresijsko preskušanje)
- Psihološki:
  - Programerji preskušajo le primere, ki so jih predvideli
  - Izvajajo le pozitivne teste
  - Na preskušanje vplivajo naše izkušnje

Napake med tekom programa(run-time)

Običajno jih povzročajo:

- Problemi z alokacijo pomnilnika
- Uporaba kazalca z vrednostjo null
- Neskončne zanke
- Logične- algoritmične napake

Te napake rešujemo tipično z razhroščevanjem

Primer logične napake

Logične napake najtežje odkrijemo. Program se prevede pravilno, rezultati pa so napačni.

Naslednja koda naj bi povzročila izpis 5 zvezdic, izpiše pa le eno. Zakaj?

```
for(int i = 1; i <= 5; i++);
        System.out.print('*');
```

Napake zaokroževanja

- Včasih da program napačne numerične odgovore.
- Do tega lahko pride zaradi napake zaokroževanja (*rounding error*), ki izhaja iz omejene natančnosti računalnika.

```
double x = 0.1;
while ( x != 0.2) {
        x += 0.01;
        System.out.println(x);
}
```

- Ta program bi krožil neprekinjeno. Spremenljivka x nikar ne doseže vrednosti točno 0.2 (x zgreši vrednost 0.2 za približno 8.3×10^{-17}).
- Bolje je, če stavek while zamenjamo z:

```
while ( x <= 0.2) {
```

Razhroščevanje - debugging

- Iskanje in odpravljanje napak v programu
- Razhroščevanje pride po preskušanju:
  - Ko napako opazimo, poiščemo njen vzrok in ga odstranimo
- Kako razhroščujemo
  - Opazovanje
  - Branje kode
  - Izvajanje na papirju
  - Uporaba razhroščevalnih izpisnih stavkov
  - Uporaba razhroščevalnika

Razhroščevanje s testnimi izpisi

- Vstavimo preproste izpisne stavke na ključne položaje v naši kodi
  Na primer:

```
printf(“Pred vstopom v zanko”);
................
printf(“Po izstopu iz zanke”);
```

- Ponovno prevedemo in izvedemo. Glede na zadnji testni izpis lahko ugotovimo, kje približno je program obtičal.

- Z dodatnimi testnimi izpisi lahko napako še bolj podrobno lociramo.

Razhroščevanje s testnimi izpisi

- Lahko izpisujemo tudi vrednosti različnih spremenljivk in opazujemo, če se spreminjajo v napačne vrednosti

- Koristna je uporaba pogojnega prevajanja, ki vklaplja ali izklaplja testne izpise

```
if(debug) printf( )
```

- Podobno dosežemo tudi z zakomentiranjem testnih izpisov

```
// printf ( )
```

JBuilder 9 - C:/WWW/SIS/COMPUTING/Java/JAVA_TECAJ/PRIMERI/NormirajPolje.java

File Edit Search View Project Run Team Wizards Tools Window Help

untitled1.jpx

- untitled1.jpx
  - <Project Source>
  - NormirajPolje.java

- Imports
- NormirajPolje
  - main(String[] args)
  - normirajPolje(double[] p, int n)

NormirajPolje

```java
import java.io.*;
//*****************************************************************
public class NormirajPolje {

        //*************************************************************
        static void normirajPolje(double p[], int n) {
                /* funkcija normira polje p z n realnimi stevili */
                double max;
                int i;
                /* najprej poiscemo maksimalni element */
                max = p[0];  /* predpostavka, da je prvi element maksimalen*/
                for (i=0; i<n; i++)
                        if (p[i]> max) max = p[i];
                /* nato se vse elemente polja normiramo (delimo) s to vrednostjo */
                for (i=0; i<n; i++) p[i] = p[i]/max;
        }

        //*************************************************************
        public static void main(String[] args) throws Exception {

                double polje[] = {10.0, 20.0, 22.0, 15.0, 30.0 };
                int num;
                int i;
                /* preizkus funkcije normirajPolje */
                num = 5;  /* stevilo elementov v polju */
                System.out.print("Pred normiranjem:");
                for(i=0;i<num;i++) System.out.print(polje[i] + " ");
                normirajPolje(polje, num);
                System.out.print("\nPo normiranju:   ");
                for(i=0;i<num;i++) System.out.print(polje[i] + " ");
        }
}
```

NormirajPolje.java | Insert | 31:9 | CUA

Source | Design | Bean | Doc | History

start | C:\WWW\SIS\P... | Programiranje 2... | razvoj_program... | testiranje_sw.ppt | JBuilder 9 - C:/... | 18:40

Nekaj misli o napakah

“Teorija je nekaj, o čemer nekaj veš, vendar ne deluje.

Praksa je nekaj kar deluje, pa ne veš zakaj.

Programiranje združuje teorijo in prakso:
nič ne dela in ne veš, zakaj...”

-- Anonymous

“If debugging is the process of removing bugs then programming must be the process of putting them in”

Dokumentiranje programa

- Zakaj bi pisali dokumentacijo?
- Vrste dokumentacije
  - Za programerja, ki je kodo (na primer razred) napisal
  - Za programerja, ki kodo (razred) uporablja
  - Za uporabnika aplikacije

Dokumentiranje: dogovori in stil

Dokumentiranje programa je pomembno za ponovno uporabo in vzdrževanje…nedokumentirana koda je praktično neuporabna!

- Kaj dokumentirati?
- Kako napisati dokumentacijo?
- Podpora programskih orodij?
- Napol formalna dokumentacija?

Priporočilo: dokumentacijo pišimo v angleščini, navodila v slovenščini

Dokumentacija za uporabnike razredov

- Java nudi avtomatski sistem dokumentacije, ki tvori za vsak razred datoteko HTML.

- To zagotavlja program javadoc

- Uporaba programa javadoc

  javadoc [*options*] *filenames*

  javadoc [*options*] *packagename*

Struktura in dokumentacija programa

Programske datoteke

Pišimo posamezne razrede v ločenih datotekah.

Tega se držimo tudi pri ne-javnih razredih.

Izjema je enkratna uporaba ne-javnih razredov, ki jih ni smiselno uporabljati izven danega konteksta.

Vsako datoteko začnemo z:

Imenom datoteke in drugimi podatki, tudi o avtorju

Datumi in seznamom sprememb in njihovih avtorjev

Če je datoteka glavna vhodna točka paketa, kratko opišemo še namen paketa.

Struktura in dokumentacija programa

Paketi (packages)

- Nov javanski paket za vsak samostojen projekt ali skupino s povezano funcionalnostjo
- Datoteka index.html v vsakem direktoriju s kratkim opisom namena in strukture paketa

Primer programske datoteke

```
/*
   Datoteka: primer.java
   Datum        Avtor       Spremembe
   Sep 1  03    S.Divjak    Prva verzija
   Sep 13 03    S.Divjak    Popravki dokumentacije
*/

package demo;
import java.util.NoSuchElementException;
```

Sintaksa komentarjev Javadoc

- Začnejo z /** in končajo z */

- Te komentarje program javadoc pretvori v dokumentacijo HTML.
- Ti komentarji lahko uporabljajo oznake HTML, kot na primer
  <PRE> in <TT>,
  ne pa strukturnih oznak kot <H2> ali <HR>.

- Komentarje za javadoc moramo pisati neposredno pred deklaracijo razreda, polja ali metode, ki jo opisujemo.

- Prvi stavek v takem komentarju mora biti povzetek, temu pa lahko sledi dodatne informacije, podane s posebnimi oznakami, ki začenjajo z @. Taka oznaka mora biti na začetku vrstice.

Dokumentiranje deklaracij in izrazov

Lokalne deklaracije, stavki in izrazi

- Ne komentirajmo samoumevne kode, raje se potrudimo, da bo samoumevna!
- Primer:

```
int index = -1; // -1 je zastavica, ki pomeni, da je indeks neveljaven
```

Ali, kar je pogosto boljše:

```
static final int INVALID= -1;
int index = INVALID;
```

- Konsistentno uporabljajmo enak izgled kode:
  - Število mest v zamikih kode.
  - Levi oklepaj ("{") na koncu vrstice
  - Maksimalno dolžino vrstice.
  - Zamikanje pri prelomu dolgih vrstic.
  - Deklaracijo vseh razrednih vrstic na enem mestu (na vrhu razreda).

Dogovori za imena

Paketi (packages)

- Male črke.
- Za prefikse priporočene domenske konvencije, opisane v specifikaciji Jave.

Datoteke

- Enako ime kot ime javnega razreda, ki je v njih definiran.

Razredi (classes)

- Velika začetnica, Z veliko črko začenjamo tudi interne besede v imenu razreda.

Lokalne spremenljivke

- zacnemoZmaloCrkoInterneZacetkeInternihBesedZveliko ali
- Zacnemo_z_malo_uporabljamo_podcrtaje

konstante (final)

- VELIKE_CRKE_S_PODCRTAJI

Dogovori za imena

metode

- Mala začetnica, Interne besede z veliko začetnico

Metode, ko pretvorijo objek v tip X

- toX

Metode, ki vrnejo vrednost atributa x tipa X

- X x() ali X getX().

Metode, ki spremenijo vrednost atributa x tipa X

- void x(X vrednost) ali void setX(X value).

Popoln primer

```
/*
* @(#)Blah.java       1.82 99/03/18
*
* Copyright (c) 1994-1999 Sun Microsystems, Inc.
* 901 San Antonio Road, Palo Alto, California, 94303, U.S.A.
* All rights reserved.
*
* This software is the confidential and proprietary information of Sun
* Microsystems, Inc. ("Confidential Information").  You shall not
* disclose such Confidential Information and shall use it only in
* accordance with the terms of the license agreement you entered into
* with Sun.
*/
package java.blah;
import java.blah.blahdy.BlahBlah;
```

Popoln primer (2)

```
/**
 * Class description goes here.
 *
 * @version     1.82 18 Mar 1999
 * @author      Firstname Lastname
 */
public class Blah extends SomeClass {
    /* A class implementation comment can go here. */

    /** classVar1 documentation comment */
    public static int classVar1;

    /**
     * classVar2 documentation comment that happens to be
     * more than one line long
     */
    private static Object classVar2;
```

Popoln primer (3)

```
/** instanceVar1 documentation comment */
public Object instanceVar1;

/** instanceVar2 documentation comment */
protected int instanceVar2;

/** instanceVar3 documentation comment */
private Object[] instanceVar3;

/**
 * ...constructor Blah documentation comment...
 */
public Blah() {
    // ...implementation goes here...
}
```

Popoln primer (4)

```
    /**
     * ...method doSomething documentation comment...
     */
    public void doSomething() {
        // ...implementation goes here...
    }

    /**
     * ...method doSomethingElse documentation comment...
     * @param someParam description
     */
    public void doSomethingElse(Object someParam) {
        // ...implementation goes here...
    }
}
```