

ՅՈՒՐԱԿԱՆ ԵՐԱՆԻՑՈՒԹՅԱՆ ԵՐԱՐՈՒՄ

ԻՐԱՆԻՑ

ՈՒՆԱՅԻՆ ՄԻՋԵՐՈՒԹՅԱՆ

Otvoreno računarstvo

Java

- Svijet Jave, Osnove jezika Java, Uvodni primjeri
- JNI, JAR, Paketi
- CJC, Sučelja, Iznimke, Višenitnost, Grafika
- Tokovi podataka
- Kolekcije
- Dokumentiranje koda
- Javini nasljednici

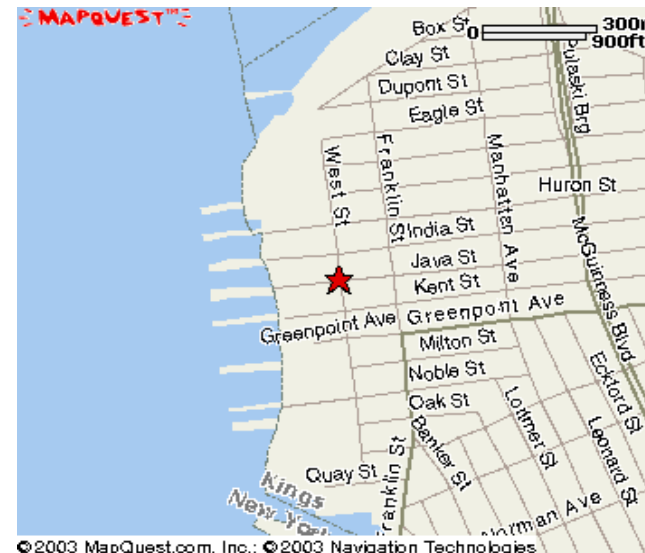
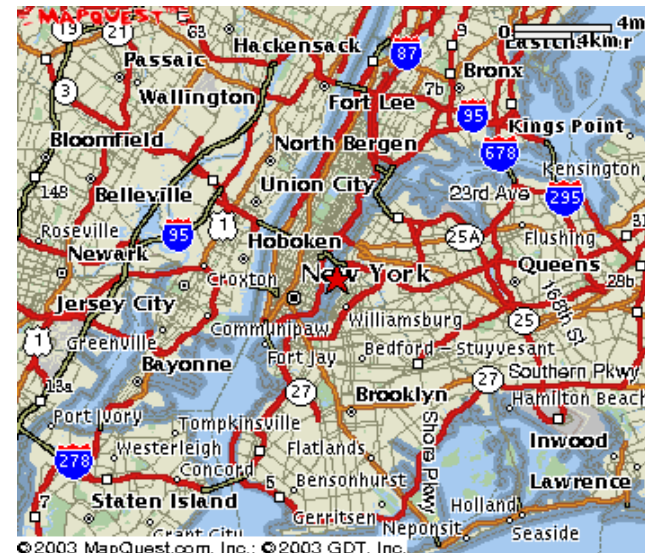
Mario Žagar



The background features large, light green letters 'O' and 'R' that are partially visible, with the 'O' on the left and the 'R' on the right, separated by a vertical white line.

Svijet Jave

Predgovor - Kojim putem dalje??



Predgovor - Kojim putem dalje??



Java 2D Graphics

By Jonathan Knudsen
1st Edition May 1999
366 pages
ISBN 1-56592-484-3

Java 2D Graphics describes the 2D API from top to bottom, demonstrating how to set line

Java Language Reference, 2nd Edition

By Mark Grand
2nd Edition July 1997
492 pages
ISBN 1-56592-326-X

This book helps you understand the

Java Swing

By Robert Eckstein,
Marc Ley & Dave Wood
1st Edition September 1998
1252 pages
ISBN 1-56592-433-X

The Swing classes eliminate

Java Cryptography

By Jonathan B. Knudsen
1st Edition May 1998
362 pages
ISBN 1-56592-402-9

Java Cryptography teaches how to write secure pro



Java Servlet Programming

By Jason Hunter with
William Crawford
1st Edition November 1998
528 pages
ISBN 1-56592-391-X

Java Servlets offer a fast

Enterprise JavaBeans

By Richard Monson-Haefel
1st Edition June 1999
336 pages
ISBN 1-56592-605-6

Enterprise JavaBeans is a thorough introduction to EJB for the enterprise software develop

Java Fundamental Classes Reference

By Mark Grand &
Jonathan Knudsen
1st Edition May 1997
1114 pages
ISBN 1-56592-241-7

The Java Fundamental Classes

Java Virtual Machine

By Jon Meyer & Troy Downing
1st Edition March 1997
452 pages
ISBN 1-56592-194-1

This book is a comprehensive programming guide for the Java Virtual Machine (JVM). It

Svijet Jave - ZAŠTO? - CILJ



- zato jer “tako dalje više ne ide!”
- dinamičan razvoj (računarstvo se razvija “prebrzo”)
- manjak znalaca
- OTVORENO RAČUNARSTVO
 - prenosivo sklopovlje
 - prenosiva programska podrška (programi, alati)
 - prenosivi ljudi, znalci

Svijet Jave - ŠTO? - filozofija (I)



- stvoriti prenosivi, mali, jednostavan kôd
- izvođenje na svim platformama od najmanje do najveće
 - JVM (Java Virtual Machine) - ME (Micro Edition), SE (Standard Edition), EE (Enterprise Edition)
- objediniti prošlost, sadašnjost i budućnost
 - C, C++, Java
- omogućiti nadogradnju, modularnost
 - Java Beans,...

Svijet Jave - ŠTO? - filozofija (II)



- totalno umrežavanje
 - paralelizam, udaljeni kod, objekti
- objektni pristup
 - varijable i postupci zajedno
- lokalni i globalni programi
 - aplikacije i appleti
- svi frontovi
 - prog. podrška (Java, JavaOS, real-time Java),
 - sklopovlje (picoJava),
 - integracija u postojeća rješenja (JVM))

Svijet Jave - KAKO? - metodologija



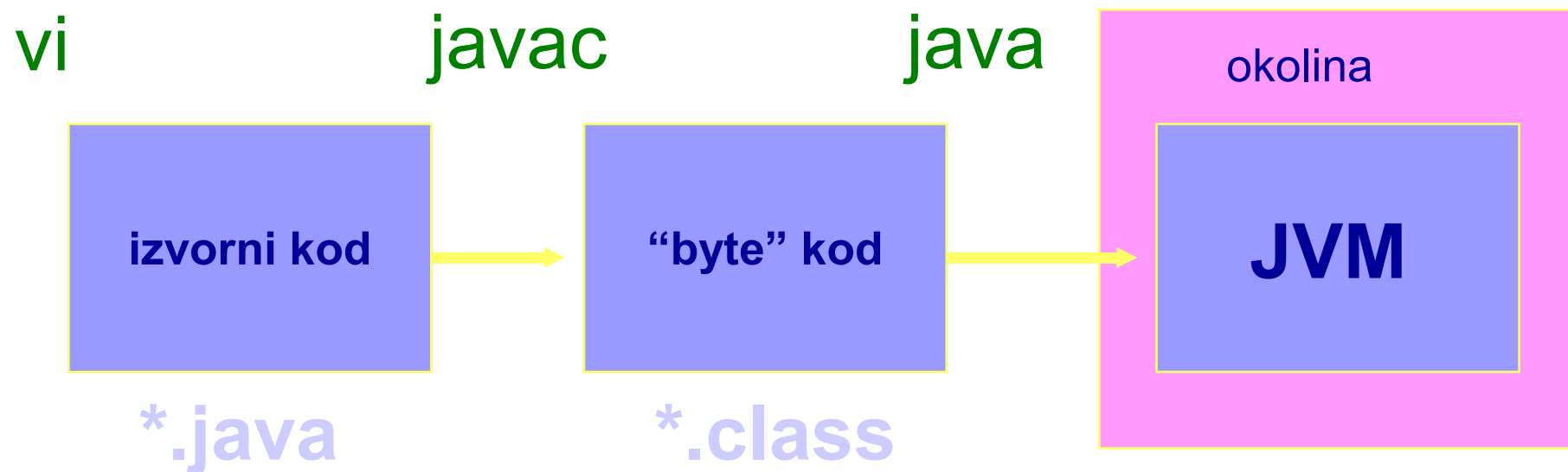
- koristi se općeniti kod - bytecode
- nadogradnja na C
- objektni pristup, jednostruko nasljeđivanje
- mrežni pristup, pomoć WWW-a
- svaki preglednik JVM
- izgradnja čip kartica
- arhitektura (trorazinska) picoJave

Svijet Jave - ČIME? - tehnologija

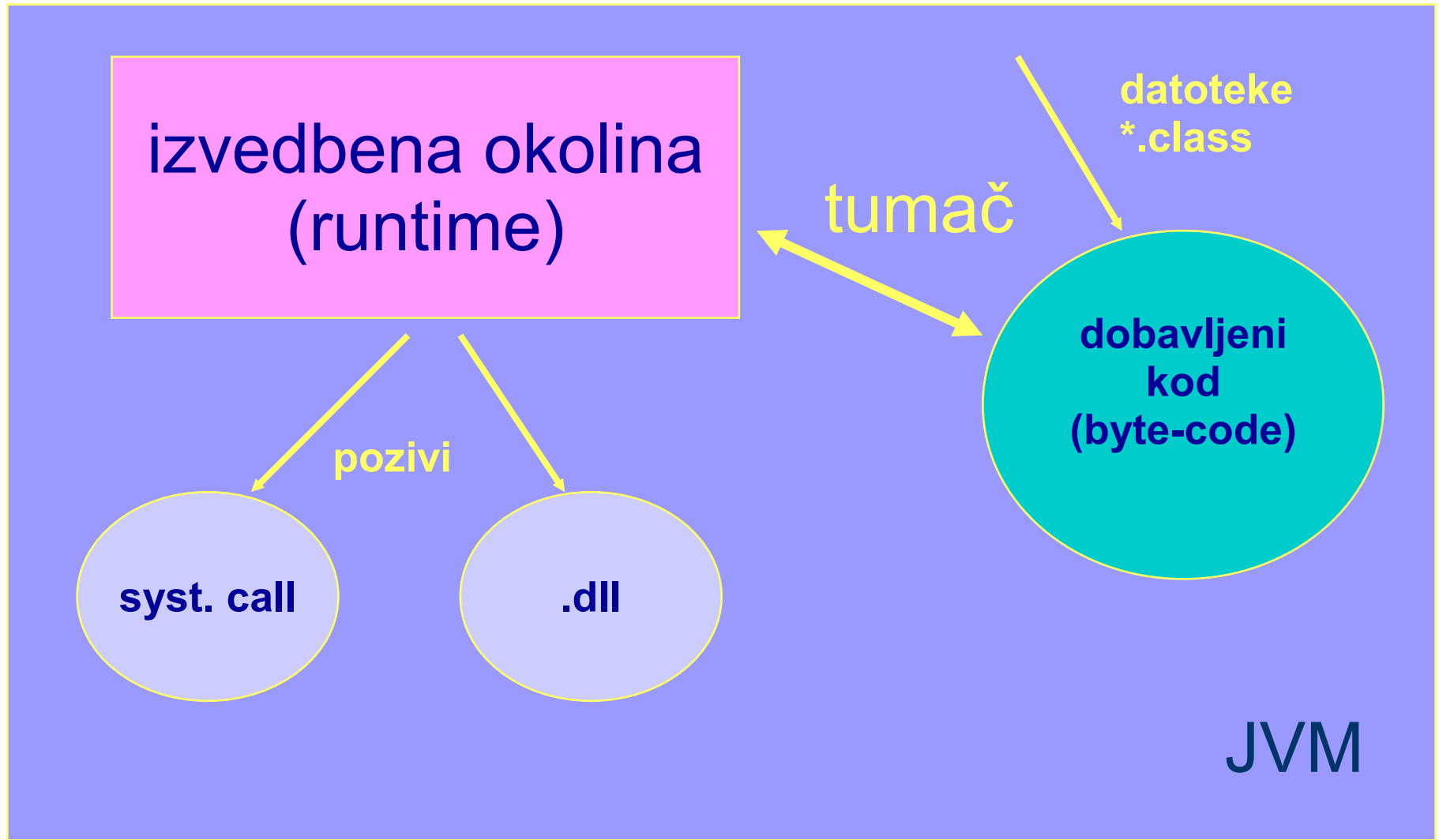


- nova generacija čipova - picoJava??
- prilagodba postojećih procesora
- JVM
- čip kartice, mobilni aparati
- skaliranje API-a
- Jezik Java, Java Beans, aplikacije, appleti, servleti, midleti, ...
- Java JINI - koncept - svaki uređaj dvije žice

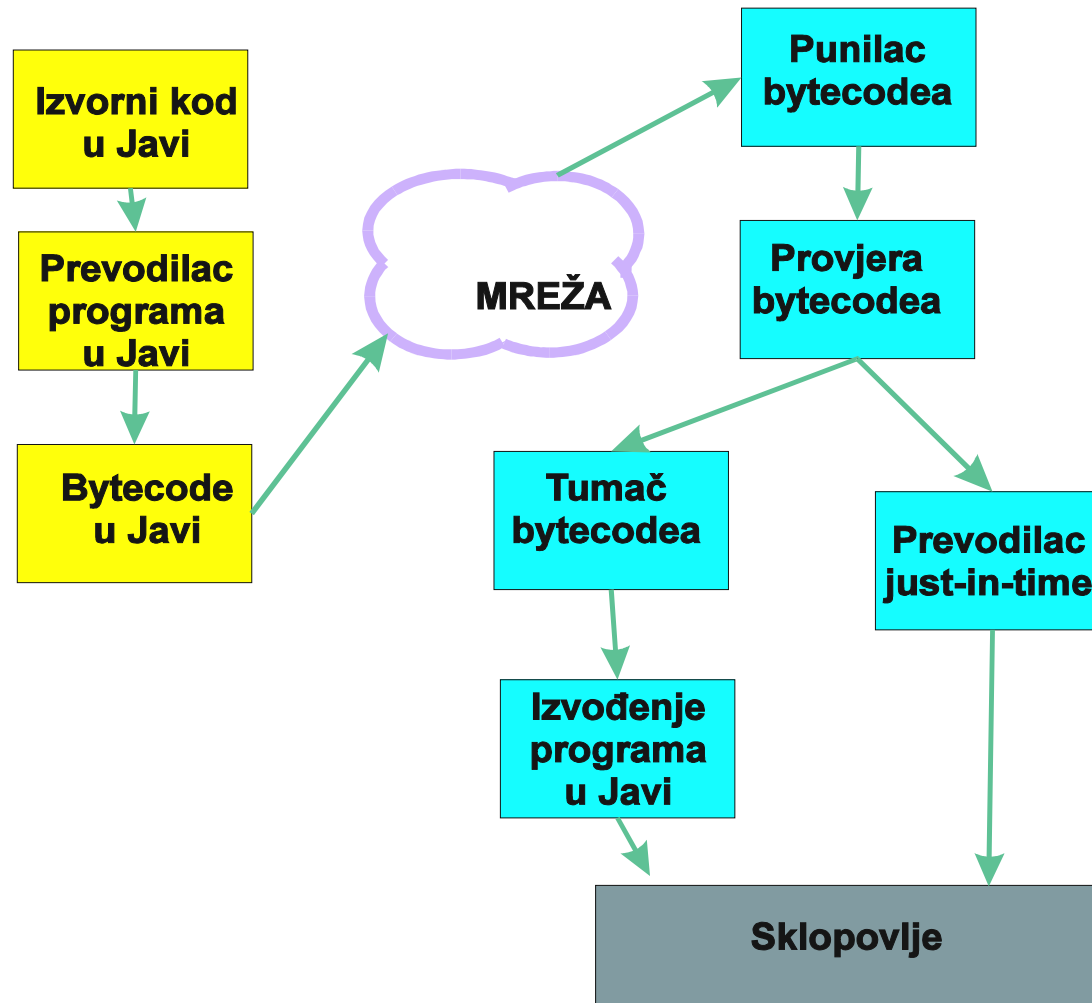
Svijet Java - JVM



Svijet Java - JVM



Izvođenje kod korisnika - Appleti, Midlet



Izvođenje appleta u Javi

The background of the slide features large, light green letters 'O' and 'R' that are partially visible, with the 'O' on the left and the 'R' on the right, both cut off by the edges of the frame.

Osnove jezika Java

Operatori (I)

- kao i C
- bolje definiran redoslijed evaluacija
- evaluacija s lijeva na desno
- asocijativnost ($6 * 4 \% 3$)
- `>>` aritmetički posmak (predznak se širi), u C-u nekad 1111., ili 000..
- `>>>` shift right and zero fill, nepotreban u C-u

Osnovni tipovi podataka



- **boolean**
 - 1 bit (false, true)
- **char**
 - 16-bitovni, Unicode (UTF-16, UTF-8), modified UTF-8
- **byte**
 - 8-bitovni, dvojni komplement (-128, +127)
- **short**
 - 16-bitovni, dvojni komplement (-32768, 32767)
- **int**
 - 32-bitovni, dvojni komplement (-2147483648, 2147483647)
- **long**
 - 64-bitovni, dvojni komplement
(-9223372036854775808, 9223372036854775807) ,

2^{64} = Bubbabyte

Osnovni tipovi podataka (II)



- **float**
 - 32-bitovni, IEEE-754 (single precision)
 - -3.4E38, +3.4E38, 6 -7 točnih znamenaka, (ovisi o broju)
- **double**
 - 64-bitovni , IEEE-754 (double precision)
 - -1.7E308, +1.7E308, 14 -15 točnih znamenaka

(kažu da je broj protona u vidljivom dijelu svemira oko 10^{78})
- `int i ;`
- literali `\b`, `\t`, `\n`, `\r`, `\\`, ...`\uxxxx` (`\u2bf3`)
- za objekte treba više:
 - Voće jabuka;
- pokazivači implicitni u Javi
 - Voće jabuka = `new Voće()`;

Java, što je što (I)

- razred
 - se sastoji iz polja (*fields, members*)
- polja
 - su podaci i metode (*data members, methods*)
- ugrađeni tipovi podataka
 - char, int, boolean,
- složeni tipovi podataka
 - razred - Auto, Voće, GeometrijskiLik,

Java, što je što (II)

- objekt
 - instanca razreda
- referentna varijabla
 - pokazivač na objekt (**mojGolf**)
- metoda konstruktor
 - **new** stvara objekt i poziva konstruktor koji inicijalizira objekt (**Auto mojGolf = new Auto();**)
- poziv metode
 - **mojGolf.potrošnja();**
- nema destruktora
 - automatski - *garbage collection*

Komponente jezika Java

- elementi jezika
- razredi, stvaranje objekata
- jednostruko naslijeđivanje,
 - korijen (root) je klasa “Object”
- podrazredi
 - `public class MojOkvir extends java.awt.Frame { ... }`
- višestruko naslijeđivanje preko mehanizma “Interface”
- čvrsta provjera “*Strongly checked type*”
 - statički i dinamički
- zaštita pristupa za razrede, podatke i metode

Komponente jezika Java (II)

- kontrolne strukture ANSI C-a
 - **polja**(*arrays*) i **nizovi**(*strings*) su objekti
- nema pristupa memorijskim adresama
 - niti C/C++ pointer aritmetike
- memoriju oslobađa sakupljač smeća (*garbage collector*)
- podrška **iznimkama** (*exceptions*)
- podrška **nitima** i kontroli paralelnog izvođenja unutar objekta

Modifikatori razreda

modifikator **class** ime.....

- **abstract**
 - mora biti proširena prije uporabe
- **final**
 - ne može se proširivati
- **public**
 - razred vidljiv drugim paketima
- **abstract public class Voće {...**
- **final public class Limun extends Voće {...**

Modifikatori varijabli - vidljivost



- **public**
 - varijabla vidljiva svima (razred mora biti također public)
- **prazno**
 - varijabla vidljiva unutar paketa
- **protected**
 - kao prazno ali varijabla vidljiva u podrazredima u drugim paketima proširenim iz ovog razreda
- **private protected**
 - varijabla vidljiva u ovom razredu i podrazredima
- **private**
 - varijabla vidljiva samo u ovom razredu

Modifikatori varijabli - način korištenja



- **static**
 - jedna varijabla za razred (ne za svaki objekt)
- **final**
 - ne mijenja vrijednost (konstanta)
- **transient**
 - za budućnost
- **volatile**
 - više izvora može mijenjati vrijednost, prilikom izvođenja potrebno osvježiti sadržaj prije čitanja
- **protected static final int br_karata = 32;**

Modifikatori metoda - vidljivost



- **public**
 - metoda vidljiva od svakud
- **prazno**
 - isto kao i za varijable
- **protected**
 - isto kao i za varijable
- **private protected**
 - isto kao i za varijable
- **private**
 - vidljiva unutar razreda, ne može biti i abstract

Modifikatori metoda - način korištenja



- **static**
 - implicitno “final”
- **final**
 - ne može biti “overriden”
- **abstract**
 - mora biti “overriden” da bude korisna
- **native**
 - nije pisana u Javi (nema tijelo već će ono biti napisano u nekom drugom jeziku)
- **synchronized**
 - samo jedna nit može izvoditi metodu u jedinici vremena, ulaz u metodu zaštićen monitorom
- **protected abstract int potrošnja() { .. }**

Ključne riječi (*keywords*) I

Built-in tipovi:

- boolean, true, false
- char
- byte, short, int, long
- float, double
- void

Ključne riječi (II)

u izrazima

- null, new, this, super

u naredbama (selekcija)

- if, else
- switch, case, break, default

u naredbama (iteracije)

- for, continue,
- do, while

Ključne riječi (III)

u naredbama (prijenos kontrole)

- return
- throw

u naredbama (niti i iznimke)

- synchronized
- try, catch, finally

Ključne riječi (IV)



promjena deklaracije (vidljivosti, namjene,..)

- static
- abstract, final
- private, protected, public
- transient, volatile

Ključne riječi (V)

namijenjene metodama i razredima

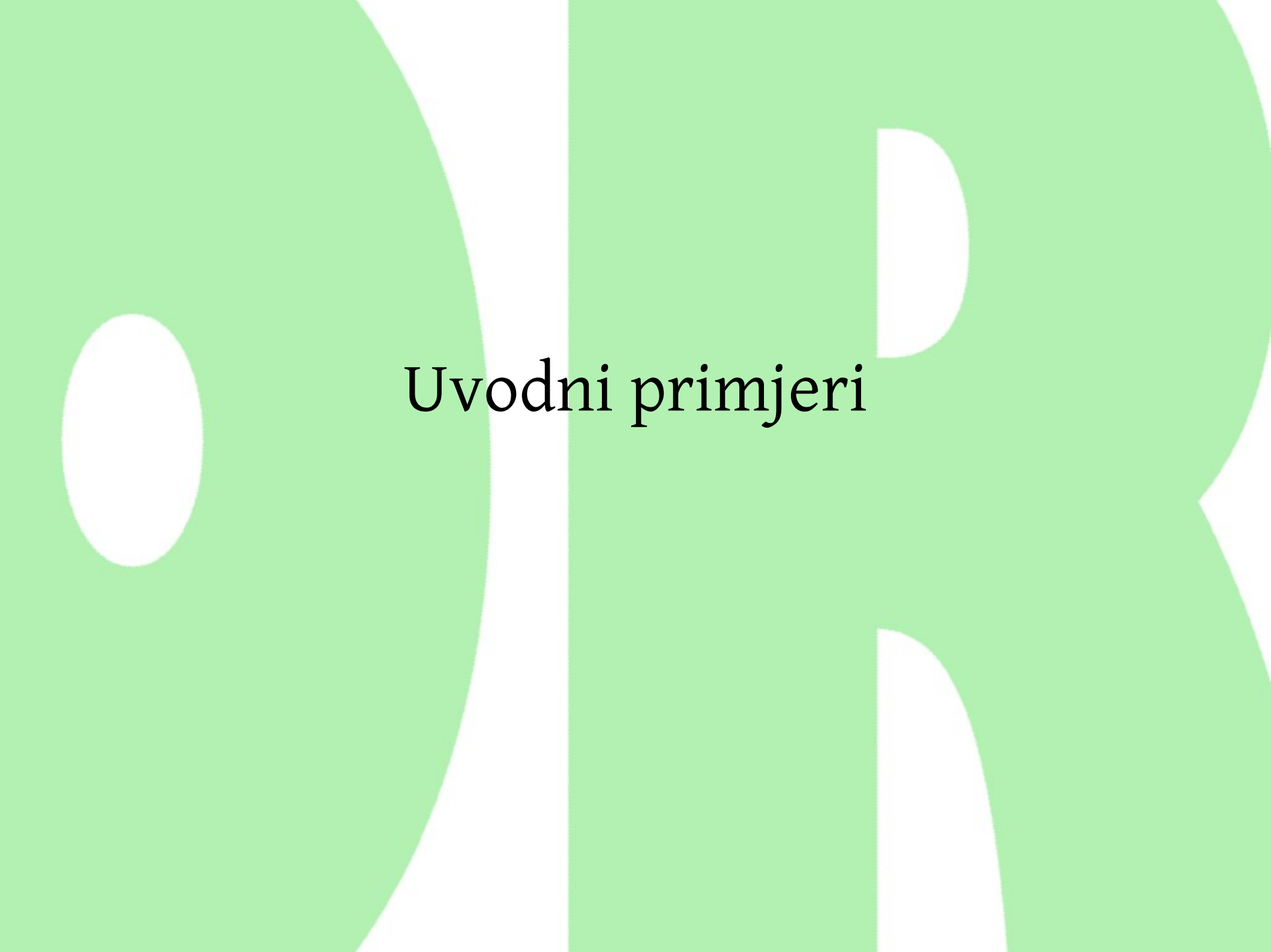
- `class`, `instanceof`, `throws`, `native`

namijenjene blokovima većim od razreda

- `extends`
- `interface`, `implements`
- `package`, `import`

za budućnost

- `cast`, `const`, `future`, `generic`, `goto`
- `inner`, `operator`, `outer`, `rest`, `var`

The background of the slide features large, light green letters 'O' and 'R' that are partially visible, with the 'O' on the left and the 'R' on the right, both rendered in a simple, sans-serif font.

Uvodni primjeri

Java - primjer

- Pozdrav svima!

```
% vi Pozdrav.java
```

```
//Primjer 1
```

```
class Pozdrav{
    public static void main(String args[]){
        System.out.println("Pozdrav svima!");
    }
}
```

```
% javac Pozdrav.java
```

```
% java Pozdrav
Pozdrav svima!
```

Java - primjer*



```
public class PentiumBug {  
    public static void main(String args[]) {  
        double x,y,z;  
  
        x = 4195835.0;  
        y = 3145727.0;  
        z = x - (x / y) * y;  
  
        System.out.println("Rezultat je " + z);  
    }  
}
```

```
% java PentiumBug  
Rezultat je 0.0
```

*1993. Intel inside - Pentium
1994-10-30 g. T. Nicely bio je nezadovoljan s dijeljenjem,
ponekad, ponegdje to je 256.000000 :-)

Jezik Java - okolina

- Stabilne, moćne biblioteke, paketi (*packages*)
- grafička sučelja za sve platforme
- jednostavna izgradnja GUI klijenata
- slike i audio
- podrška za obradu slika (filtari)
- dinamički dohvat razreda
- mrežno programiranje
- događaji (*events*), iznimke (*exceptions*),....

Java je - kao švicarski nožić:

- programski jezik
- simulacija sklopovlja, sklopovlje
- mrežno računarstvo
- aplikacije, applet, midlet, JavaCard,...
- JavaBeans, JNI, RMI,
- alati (JDK, Eclipse, NetBeans,....)
- glavna osobina Jave
 - Svijet Jave (skup modula - dijelova koji se upotpunjuju)
- dobri temelji za nadogradnju
- PRIJE SVEGA filozofija, metodologija!



Java Native Interface (JNI)

Veza Java i okoline

- Svijet Java - prenosivost, otvorenost
 - Kako ga povezati s okolinom (sklopovljem)?
 - Kako pročitati "stanje linije 3 na portu 8"?
 - Može li JVM biti u potpunosti napisan u jeziku Java?
- Vrijedi samo za aplikacije (ne applete - sigurnost)
 - Narušavanje prenosivosti
- R: Java Native Interface (JNI)

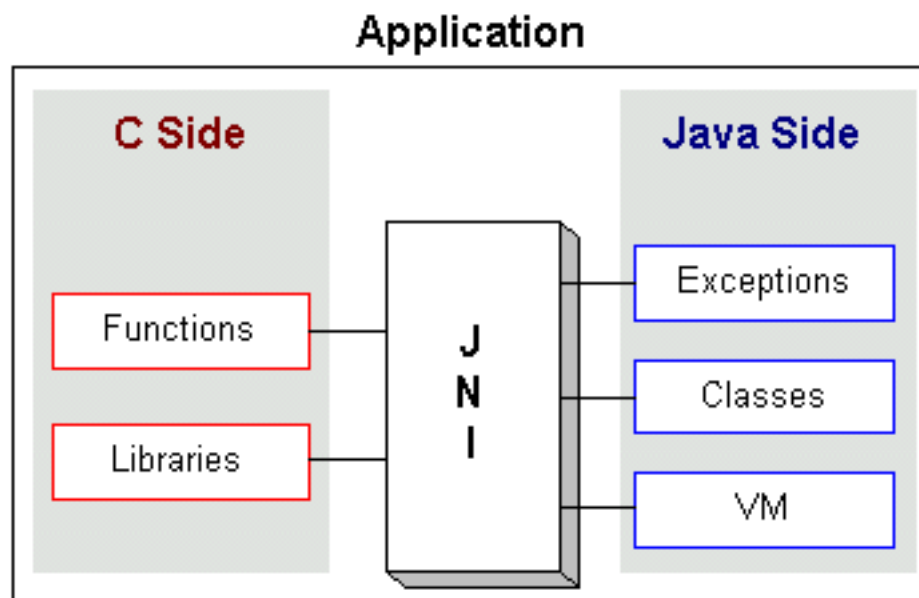
NativeIO.java:

```
class NativeIO{
    public native byte readIO(int adr);
    public native void writeIO(int adr, byte value);
    static{
        System.loadLibrary("native");
    }
}
```

Veza s C-om/C++-om/.... okolinom



- Što ako:
 - trebamo pristup sklopovlju ili
 - želimo koristiti postojeću biblioteku funkcija napisanu u C/C++a i dalje želimo koristiti Javu?
- Java Native Interface (JNI) omogućuje povezivanje s vanjskim kôdom



- JNI omogućuje dvosmjerno povezivanje:
 - pozivanje vanjskog koda iz Java programa
 - vanjske metode označavaju se ključnom riječi *native*, npr. (iz *java.lang.System*):

```
public static native long currentTimeMillis();
```
 - alatom *javah* generira se header datoteka (.h) koja se koristi za pisanje C/C++ koda
 - C kod, zajedno s generiranim headerom, kompajlira se u dinamičku biblioteku (.DLL)
 - manipulaciju Java objektima iz vanjskog koda
 - JNI definira funkcije u programskom jeziku C za manipulaciju nizovima znakova, poljima, poziv metoda, pristup varijablama objekta, etc ...



Java ARchive (JAR)

JAR - Java ARchive

- Sažeta ZIP datoteka
 - smanjuje količinu podataka za prijenos
- Slična naredbi **tar** u UNIX-u
 - **tar -cvf ime, tar -xvf ime, tar -tvf ime**
 - **jar -cvf ime.jar ime1.class ime2.jpeg**
 - **jar -xvf ime.jar**
- Unutar HTML-a (appleta) **ARCHIVE=ime.jar**
- Microsoft CAB (MSZIP) - cabinet datoteke

```
/primjeri/Properties.java
```

The background features large, light green letters 'O' and 'R' that are partially cut off by the edges of the frame. The 'O' is on the left and the 'R' is on the right, with a white vertical line separating them.

Paketi

Paketi (*Packages*)

- **paket** omogućuje organizaciju razreda i sučelja u logičnu cjelinu
- paket = skup povezanih tipova, uređuje:
 - prava pristupa
 - prostor imena
- pod tipovima se misli na:
 - razrede
 - sučelja
 - pobrojenja (enumeracija)
 - anotacije

Paketi - prostor imena

- Ako će se koristiti javno, ime razreda mora biti jedinstveno
- Jednostavan način za osigurati jedinstvenost
 - nazvati paket po vlastitoj domeni u obrnutom redoslijedu
 - npr. za razrede primjera s geometrijskim likovima:
 - *hr.fer.rasip.or.primjeri.likovi*
- puno ime razreda Citac tad bi glasilo:
 - *hr.fer.rasip.or.primjeri.likovi.Citac*
- a nalazio bi se u kazalu:
 - *hr/fer/rasip/or/primjeri/likovi*

Paketi - kontrola pristupa

- razredi u paketu zaštićeni su od pristupa izvana
- dvije razine kontrole pristupa:
 - za razred
 - public
 - razred vidljiv iz svih drugih razreda, iz bilo kojeg paketa
 - package-private (ne navodi se)
 - razred vidljiv samo unutar svojeg paketa
 - za member varijablu
 - public
 - package-private (ne navodi se)
 - slično kao i za razrede (varijabla vidljiva ili ne izvan vlastitog paketa)
 - private
 - varijabla vidljiva samo unutar vlastitog razreda
 - protected
 - varijabla vidljiva i podrazredima iz drugih paketa

Paketi - tablica vidljivosti



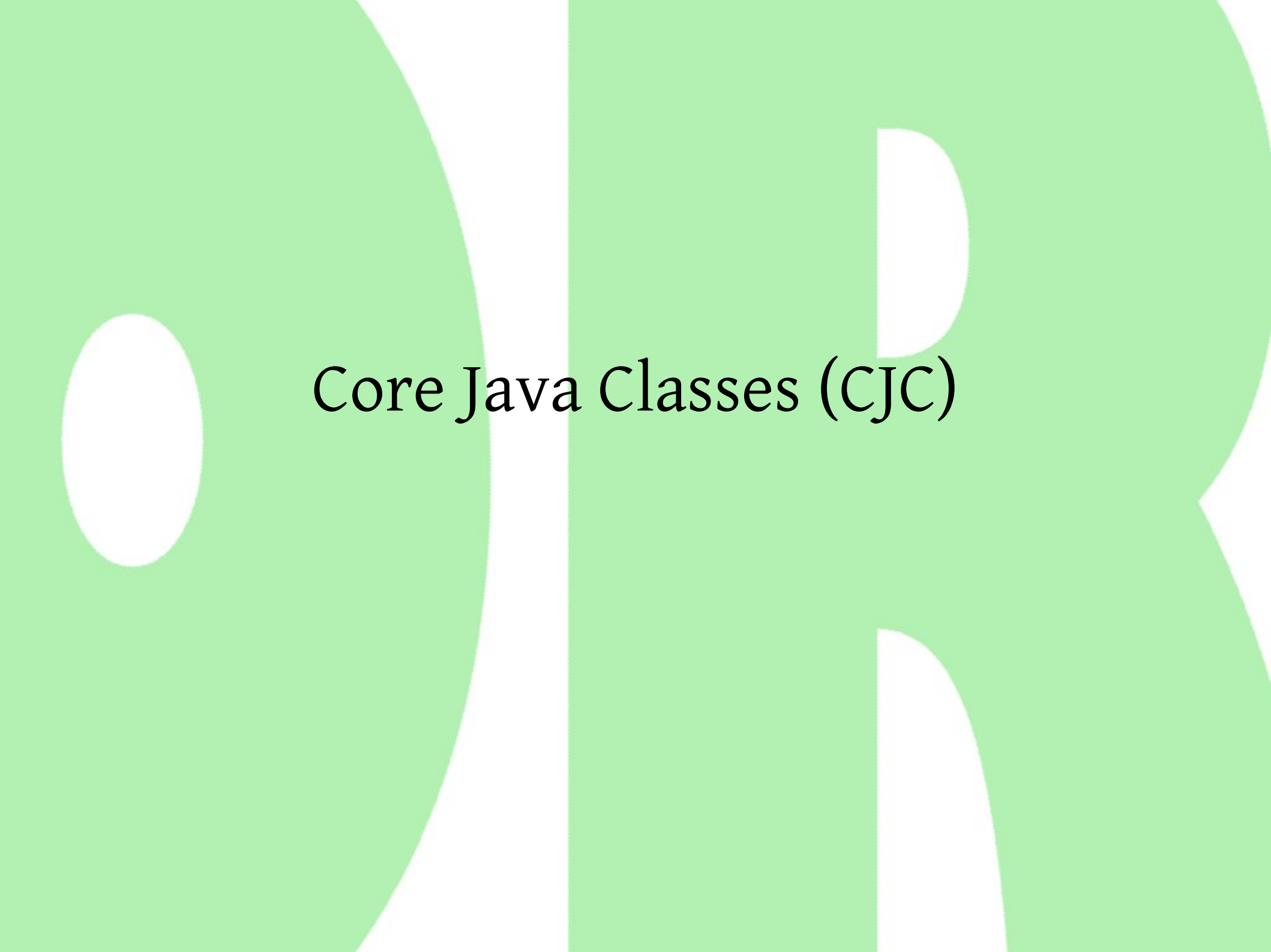
Objekt je:	default	public	protected	private	private protected
vidljiv klasama iz istog paketa	da	da	da	ne	ne
vidljiv podklasama iz istog paketa	da	da	da	ne	ne
vidljiv klasama iz drugih paketa	ne	da	ne	ne	ne
vidljiv podklasama iz drugih paketa	ne	da	ne	ne	ne
nasljeđuju ga podklase istog paketa	da	da	da	ne	da
nasljeđuju ga podklase drugih paketa	ne	da	da	ne	da

Paketi - korištenje



- svi razredi predstavljaju paket
 - po definiciji sve u standardni paket (ako nema imena)
- U WindowsXX
 - varijabla CLASSPATH .;\mojeklase;c:java\classes;...
- *import*
 - umjesto unosa cijelog imena razreda
 - ako puno ime, prevoditelj prije gotov
 - uobičajeno:
 - `import java.awt.*;` za sve razrede ili
 - `import java.lang.Math;` za pojedini razred

`class MojaMatematika extends java.lang.Math`



Core Java Classes (CJC)

Core Java Classes

Sastavni dio Javine okoline

- resursi sustava (razred *System*)
- iznimke, događaji,...
- niti
- postavljanje programskih atributa
 - naredbeni redak za jednokratne promjene,
 - svojstva (properties) za trajne promjene
- rad sa znakovima (specifičnost Jave):
 - poljima (*Arrays*), znakovnim nizovima (*Strings*), U/I tokovima podataka (*Input/Output Data Streams*),...

Polja u Javi



- Polja i znakovni nizovi su osnovni razredi u Javi
 - za usporedbu, u C-u kontinuirana grupa podataka s pristupom preko imena polja i relativne adrese (prvi element `lme[0]`)
- U C-u nema provjere granica
 - moguć je pristup preko pokazivača
 - izvor grješaka
- U Javi (ako izvan granica) poruka:
`java.lang.ArrayOutOfBoundsException`

Polja u Javi



- Stvaranje polja (deklaracija - tip i ime):
 - `int[] mojePolje; // polje cijelih brojeva`
- dodijela memorije:
 - `mojePolje = new int[velicina];`
- velicina je `int` dakle “samo” 2^{32} elemenata (2.147.483.647)
 - `int mojePolje = { 1, 2, 3,...};`

Polja u Javi



- vrijedi:
 - `char[] mojePolje = new char[200];` i
 - `char mojePolje[] = new char[200];`
- Pažnja:
 - `int a[], b, c;`
 - `int[] a, b, c; // nije isto`

Polja u Javi



- višedimenzionalna polja:
 - `float mojePolje[] [] = new float [10][20];`
- O polju dodatne informacije kao npr:
 - `mojePolje.length;` `// duljina polja`
- Nije postupak (metoda) dakle ne:
 - `mojePolje.length();` `//KRIVO!!`

Znakovni nizovi u Javi

- znakovni niz je objekt razreda `String`
- postoji i razred `StringBuffer`
- dio paketa `java.lang`
- za usporedbu u C-u je znakovni niz definiran kao polje znakova koje završava s `'\0'` (NULL)
- znakovi 16-bitovni zapis (Unicode)
- Konstruktor:
`String mojNiz = new String("abc");`

Znakovni nizovi u Javi - String



- Duljina:

`mojNiz.length();` // razlika od polja

- Zbrajanje:

`String mojNiz2 = mojNiz + "cde";` // abccde
(preopterećivanje operatora "+")

- Uspoređivanje znakovnih nizova:

`mojNiz.equals(mojNiz2);`
`mojNiz.equalsIgnoreCase(mojNiz2);`

- Nije dobro:

`mojNiz == mojNiz2;` // KRIVO!!
(uspoređuju se referentne varijable - pokazivači)

Znakovni nizovi u Javi - String



- Pronalaženje znakovnih nizova:

```
mojNiz.indexOf('b');           // 2
```

```
mojNiz.indexOf("bc");          // 2
```

```
mojNiz.lastIndexOf('c');       // 3
```

- Sortiranje nizova:

```
mojNiz.compareTo(mojNiz2)
```

- Svaki objekt ima neke karakteristične postupke
 - `equals` - usporedba jednakosti objekata
 - `getClass()` - finalni postupak, vraća razred objekta iz kojeg se poziva
 - `toString` - vraća reprezentaciju objekta u obliku znakovnog niza
 -

Znakovni nizovi - razred StringBuffer

- Objekt razreda **StringBuffer**
 - ima svoj kapacitet, sadržaj i veličinu
 - sličan razredu **String** ali dozvoljava umetanje i promjenu znakova, dodavanje znakova na kraj niza
- Konstruktori:
 - **StringBuffer mojNiz = new StringBuffer(20);**
 - **StringBuffer mojNiz = new StringBuffer("abc cde fgh");**
 - **mojNiz** - abc cde fgh
 - **mojNiz.length()** - 11
 - **mojNiz.capacity()** - 20 // za prvi slučaj
 - ...

Znakovni nizovi - razred StringBuffer

- ...
- `mojNiz.charAt(n);` // n-ti znak u nizu
- `mojNiz.charAt(n, 'c');` // n-ti znak u 'c'
- `mojNiz.insert(n, "tekst");`
// ubaci "tekst" nakon n-tog znaka
- `mojNiz.append("tekst");` // dodaj na kraj
- `mojNiz.append("a=").append(5).append(";").toString()`
// a=5;

Osobine sustava (*Properties*)



- Okolina - stroj, kazalo, boje, font itd. su atributi sustava (*runtime attributes*)
- Atributi vezani uz program (veličina prozora, opcije kod pokretanja) (*preferences*)
- Kontrola preko
 - properties
 - command line arguments
 - applet parameters

Osobine sustava

- Kako iz Javine aplikacije saznati:
 - na kojem operacijskom sustavu se izvodi, ili
 - koji VM je izvodi (1.1, 1.4 ili 1.5?), ili
 - korisničko kazalo?
- Prilikom izvođenja, JVM doznaje informacije o sustavu na kojem se izvodi
- Razred *System* čuva objekt tipa *Properties* (generički Javin razred za baratanje postavkama) popunjen osobinama sustava

Osobine sustava



- Svojstva programa
 - `java.util.Properties`
- Parovi ime/vrijednost (objekti razreda String)
 - `java.lang.System`, `java.lang.Runtime`
- Postupci:
 - `static Properties getProperties();`
`Properties osobineSustava = System.getProperties();`
 - `static String getProperty(String imeSvojstva);`
`System.getProperty("file.separator");`
 - `static void setProperties(Properties prop);`

`/primjeri/Properties.java`

Osobine sustava (varijable okoline)



```
file.separator=\
java.awt.graphicsenv=sun.awt.Win32GraphicsEnvironment
java.home=C:\Program Files\Java\jre1.6.0_03
java.io.tmpdir=c:\Temp\
java.runtime.version=1.6.0_03-b05
java.specification.name=Java Platform API Specification
java.specification.vendor=Sun Microsystems Inc.
java.specification.version=1.6
...
os.arch=x86
os.name=Windows XP
os.version=5.1
path.separator=;
sun.cpu.isalist=pentium_pro+mmx pentium_pro pentium+mmx pentium...
sun.desktop=windows
sun.management.compiler=HotSpot Client Compiler
sun.os.patch.level=Service Pack 2
user.country=HR
user.dir=D:\EclipseWrkSp\Kuku
user.home=C:\Documents and Settings\korisnik
user.language=hr
user.name=korisnik
```

/Primjeri/PropList

java.util.Date

- nadnevak i vrijeme
- postupci Date(),
- Date(long datum) (ms od 1.1.1970.)
- int getDate(), int getHours(),..... void setSeconds(int sekunde)
- System.currentTimeMillis();
- String toGMTString(),
- String toLocaleString();

/primjeri/Nadnevak.java

java.util.Random




- slučajni brojevi
- `Random()` // sjeme je trenutni broj milisekundi
- `Random(long sjeme)`
- `nextDouble()`, `nextFloat()`, `nextInt()`, `nextLong()`

`/primjeri/Slucajni.java`

java.lang.Runtime



- razred Runtime - interakcija sa sustavom
- exec, exit, freeMemory, totalMemory
- gc - garbage collector
- itd.

The background features large, light green letters 'O' and 'R' that are partially visible, with the 'O' on the left and the 'R' on the right, both cut off by the edges of the frame.

Sučelja

Sučelja (eng. interfaces)

- Ugovor između razreda i vanjskog svijeta
 - razred koja implementira sučelje obećaje ponuditi ponašanje objavljeno tim sučeljem
- Zanima nas određena funkcionalnost, a NE način na koji je ostvarena
 - npr. kod kupnje auto radia zanima nas da li ima standardni (ISO) konektor, a ne električna shema)
- Sučelja ne dijele hijerarhiju razreda
 - više razreda može istovremeno implementirati isto sučelje (višestruko nasljeđivanje u Javi)
 - korištenje slično apstraktnim razredima

Sučelja



- Sučelje se sastoji od
 - konstanti i
 - predložaka (potpisa, praznih definicija metoda) koje nemaju tijela (razred ih mora implementirati)
- Razredi se nasljeđuju, sučelja implementiraju
`class A extends B implements C`
- Svako sučelje u zasebnoj datoteci
- Namjena - podržavanje dinamičkog pronalaženja metoda tijekom izvođenja programa

Sučelja



```
interface suc1{  
    tip varijabla1 = vrijednosta;  
    tip varijabla2 = vrijednostb;  
    tip metoda1 (...);  
    tip metoda2 (...);  
}
```

```
interface I{  
    void pisi(int a);  
}  
class Test implements I {  
    public void pisi(int a){  
        System.out.println(a);  
    }  
}
```

- Metoda mora biti public
- Ako razred ne implementira sve metode iz sučelja, mora biti deklariran kao abstract

```
abstract class A implements B{
```

```
...
```

```
}
```

/primjeri/Sucelja_primjeri

Sučelja



java.awt.event.MouseListener

```
public interface MouseListener extends EventListener {  
    /**  
     * Invoked when the mouse button has been clicked (pressed  
     * and released) on a component.  
     */  
    public void mouseClicked(MouseEvent e);  
    /** ...  
    public void mousePressed(MouseEvent e);  
    public void mouseReleased(MouseEvent e);  
    public void mouseEntered(MouseEvent e);  
    public void mouseExited(MouseEvent e);  
}
```

```
public class Okvir implements MouseListener{  
    public void mouseClicked(MouseEvent e)  
    {  
        if (e.getButton() == MouseEvent.BUTTON1) { ... }  
    }  
    // ...  
}
```

Sučelja

- Razred *Okvir* mora implementirati sve metode sučelja *MouseListener* ili mora biti proglašen apstraktnim
- Razred u Javi:
 - nasljeđuje samo jedan razred (implicitno `java.lang.Object`)
 - implementira nula ili više sučelja
- Sučelje se ne može instancirati, kao ni apstraktni razred
- Zato apstraktni razred može implementirati svoje metode, a sučelje ne.

Sučelja

- jednom definirano sučelje ne mijenjati!
- zašto?
 - svi razredi koji ga implementiraju više se ne bi prevodili ispravno (budući da više ispravno ne implementiraju to sučelje)
 - rješenje = promijeniti sve razrede – ne može se baš uvijek
- ako je potrebna dodatna funkcionalnost, sučelje se proširuje novim sučeljem, npr:

```
public interface MojMouseListener
    extends MouseListener{
    public void mouseDoubleClicked(MouseEvent e);
}
```

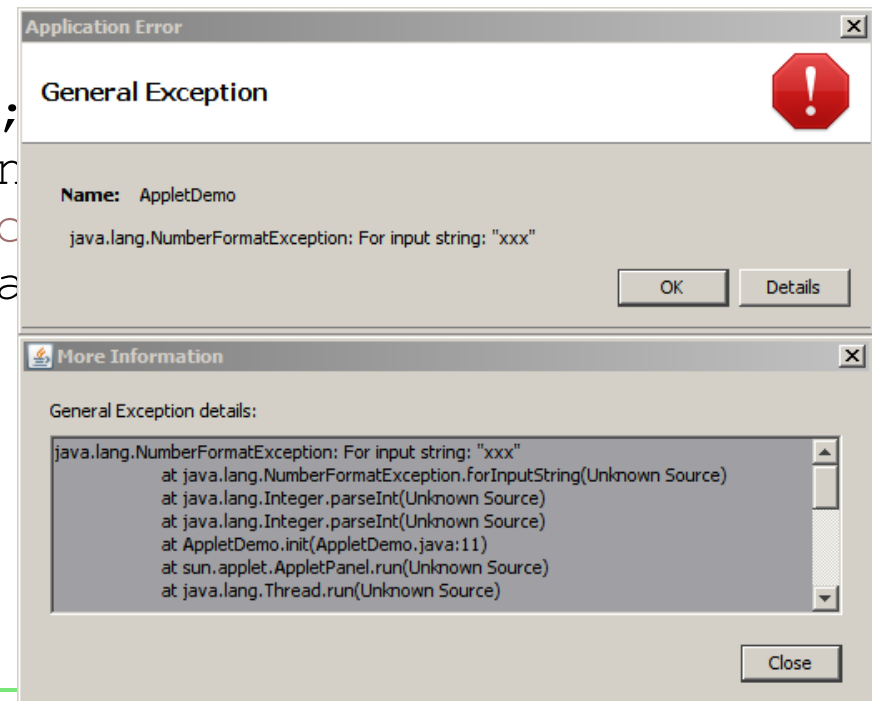
The background of the slide features large, light green letters 'O' and 'R' that are partially visible, with the 'O' on the left and the 'R' on the right, both rendered in a simple, sans-serif font.

Iznimke

Iznimke (Exceptions)

- Što bi se dogodilo da je kao parametar appleta iz prošlog predavanja predan niz znakova koji se ne može protumačiti kao broj?

```
public class AppletDemo extends Applet
{
    int brojKvadratica, delay;
    public void init() {
        setForeground (Color.green);
        brojKvadratica=Integer.parseInt(
            getParameter("BrojKvadratica"));
        delay=Integer.parseInt(getPara
    }
    . . .
}
```



Iznimke

- Dogodila bi se grješka i applet bi se prestao izvoditi
- Iznimke u Javi omogućuju sistematsko upravljanje grješkama
- ispravan način pretvaranja parametra:

```
try{  
    brojKvadratica=Integer.parseInt(getParameter("BrojKvadratica"));  
} catch (NumberFormatException nfe) {  
    brojKvadratica=100;  
}
```

/primjeri/AppletDemo.java

- Metoda *parseInt* podiže instancu razreda *NumberFormatException*
- Izvođenje programa nastavlja se naredbama iz *catch* bloka

Iznimke

- Što znači "podizati" iznimku?

```
public static int parseInt(String s, int radix)
    throws NumberFormatException {
    //...
    // provjera da li je niz 's' broj ili ne, ako nije:
    throw new NumberFormatException(s);
    ...
}
```

java.lang.Integer

- Upravljanje iznimkama u Javi postiže se:
 - blokovima *try-catch*
 - ključnim riječima *throw* i *throws*
 - razredom *Exception* i razredima koji ga nasljeđuju

Vrste iznimaka

- Dvije kategorija iznimaka:
 - generalne (*general*)
 - izvršne (*run-time*)
- Generalne iznimke moraju biti obrađene
 - poziv metode koja podiže iznimku mora biti unutar *try-catch* bloka, ili
 - metoda mora eksplicitno specificirati da iznimku prosljeđuje sljedećoj (metodi koja ju je pozvala) ključnom riječi *throws*
- Izvršne (*run-time*) iznimke ne moraju se obrađivati
 - obraditi one za koje postoji realna šansa da se pojave

Blok try-catch



```
try{
    //... kod koji podize iznimke
} catch ( NumberFormatException e){
    //...
} catch (IOException e) {
    //...
} catch (Exception e) {
    //...
} finally { // nije obavezan
    //...uvijek se izvodi, cak i ako nije bilo iznimke
}
```

The background features large, light green letters 'O' and 'R' that are partially visible, with the 'O' on the left and the 'R' on the right, both having white cutouts.

Višenitnost

Višenitnost (*Multithreading*)

- Nit, dretva, *thread*
 - osnovni kontekst izvođenja programskog kôda (*lightweight process*)
- Sadrži:
 - vlastiti stog za lokalnu pohranu podataka
 - vlastitu memoriju (*local heap*)
- Višenitnost preko
 - razreda Thread ili
 - sučelja Runnable

public class Thread extends Object implements Runnable
- Izvodi se metoda *run* u *RunnableObject*

Proces

- Proces:
 - program u izvođenju (*heavyweight process*)
 - ima:
 - vlastiti memorijski prostor
 - resurse (npr. opisnike otvorenih datoteka)
 - **jednu ili više niti**
- Globalni memorijski prostor procesa
 - dostupan svim nitima
- Svaki Javin proces ima barem jednu nit
 - onu koja izvodi metodu *main*

"Istovremeno izvođenje"

- Privid istovremenog izvođenja više niti
- Svaka nit ima prioritet (1-10, uobičajni=5)
- Različiti operacijski sustavi različito dodjeljuju procesor nitima
- Mehanizmi:
 - time slicing
 - preemption

"Istovremeno izvođenje"

- Time - slicing
 - svaka nit dobije **period vremena** u kojem se može izvršavati, zatim sve ostale niti istog prioriteta dolaze na red (operacijski sustavi Windows)
 - nit višeg prioriteta također može početi izvršavanje odmah
- Preemption
 - nit određenog prioriteta izvodi se **dok ne završi**, ili dok nit s višim prioritetom ne postane spremna za izvođenje (Sun Solaris)
 - nit višeg prioriteta "krade" procesor = *preemption*
 - mana – ako još jedna nit istog prioriteta čeka na procesor, neće se moći izvesti

Razred *Thread*

- Razred *java.lang.Thread*
 - Važniji konstruktori:
 - *Thread()*
 - *Thread(String name)*
 - Sav posao niti unutar metode ***public void run()***
 - Metoda *run()* se **NIKAD** ne poziva izravno
 - umjesto toga, poziva se *start()*
 - JVM će pozvati *run()*
 - *Pokretanje niti (dvije tehnike ostvarenja višenitnosti):*
 - naslijediti razred *java.lang.Thread* i nadjačati njegovu metodu *run()*
 - implementirati sučelje *Runnable* i metodu *run()* specificiranu sučeljem

Thread i Runnable



```
import java.lang.Thread;

class ProbnaDretva1 extends Thread {
    public void run() {
        System.out.println("Dretva[1]: Naslijedila Thread!");
    }
}
```

```
import java.lang.Runnable;

class ProbnaDretva2 implements Runnable {
    public void run() {
        System.out.println("Dretva[2]: Implementirala Runnable!");
    }
}
```

```
public class Pokretac {  
    public static void main(String[] args) {  
        ProbnaDretva1 dretva1 = new ProbnaDretva1();  
        dretva1.start();  
  
        Thread dretva2 = new Thread(new ProbnaDretva2());  
        dretva2.start();  
    }  
}
```

- rezultat izvođenja:

```
d:\src>java Pokretac  
Dretva[1]: Naslijedila Thread!  
Dretva[2]: Implementirala Runnable!
```

- metoda *run()* je pozvana?

/primjeri/Pokretac.java

Kritični odsječak

- Dio programskog kôda koji smije izvršavati samo jedna dretva
 - u Javi svaki objekt može kontrolirati ulaženje u kritični odsječak (*mutex*, *lock*)
- Blok
 - *synchronized(objektZaSinkronizaciju) {...}*
- Od Java 5.0 još više na temu višenitnosti
 - *u paketu java.util.concurrent postoje implementacije:*
 - sinkronizacijskih mehanizama – semafora, barijera, brava (*lock*)
 - kolekcija objekata sigurnih za istovremeni pristup (*HashMap*, *LinkedList*)
 - bazen niti (*thread pool*), itd.....

The background features large, light green letters 'O' and 'R' that are partially visible, with the 'O' on the left and the 'R' on the right, both rendered in a simple, rounded font style.

Grafika

Grafika

- Java
 - izvodi se na Windowsima, Solarisu, Linuxu
- Grafički podsustavi različitih operacijskih sustava potpuno različiti (ne samo izgledom već i arhitekturom)
 - npr. Unixi i Linux koriste model klijent/server X Window sustav
- Javine grafičke aplikacije isto rade i izgledaju na svim platformama

AWT, Swing

- *Abstract Window Toolkit (AWT)*
 - tanki sloj apstrakcije nad grafičkim komponentama OS-a na kojem se program izvodi
 - koristi grafičke komponente OS-a na kojem se izvodi
- **Swing**
 - od Jave 1.2
 - definira vlastite grafičke komponente
 - u potpunosti pisan u Javi (iscrtava komponente putem Java 2D API-a)
 - različiti izgledi (*look&feel*) aplikacije
 - i dalje koristi dijelove AWT-a

`/Primjeri/Swing/SwingLaF.java`

- *Standard Widget Toolkit (SWT)*
 - razvio IBM, sad Eclipse Software Foundation
 - koristi ga Eclipse razvojno okruženje
 - sličan AWT-u, također koristi grafičke komponente OS-a na kojem se izvodi
 - kompromis između lakoće korištenja (Swing) te brzine i prirodnog izgleda nativnih grafičkih komponenata (AWT)



Tokovi podataka u Javi

Tokovi podataka (paket *java.io.**)



- Tokovi bajtova (8-bitni - *byte streams*)
 - podržani preko razreda i podrazreda `java.io.InputStream` i `java.io.OutputStream`
- Tokovi znakova (16 bitni character streams)
 - implementirani su kroz razrede i podrazrede `java.io.Reader` i `java.io.Writer`
 - apstraktni razredi najviše razine vezani uz tokove znakova (*character-stream based classes*)
- Primjer prefiksa:
 - za čitanje znakova iz **datoteke** - `java.io.FileReader`
 - za čitanje bajtova (binarni podatci - slike, zvukovi) koristimo `java.io.FileInputStream`

Tokovi podataka



- Bolje je koristiti **Reader** i **Writer** (character streams):
 - skup znakova Unicode (16 bita) kodirani po normi UTF-8 (ne uvijek - serijalizacija objekata koristi nestandardni UTF-8)
 - internacionalnije - ne ovise o specifičnom kodiranju
 - zbog tehnike “buffering” efikasnije od “byte streams”.
- nego **InputStream** i **OutputStream** (byte streams):
 - skup znakova ograničenih na ISO-Latin-1 (8 bita)

Tokovi podataka - Primjer



Postupci (metode) za čitanje/pisanje su:

- `int read()`, `int read(char cbuf[])`, `int read(char cbuf[], int offset, int length)`
- `int write(int c)`, `int write(char cbuf[])`, `int write(char cbuf[], int offset, int length)`

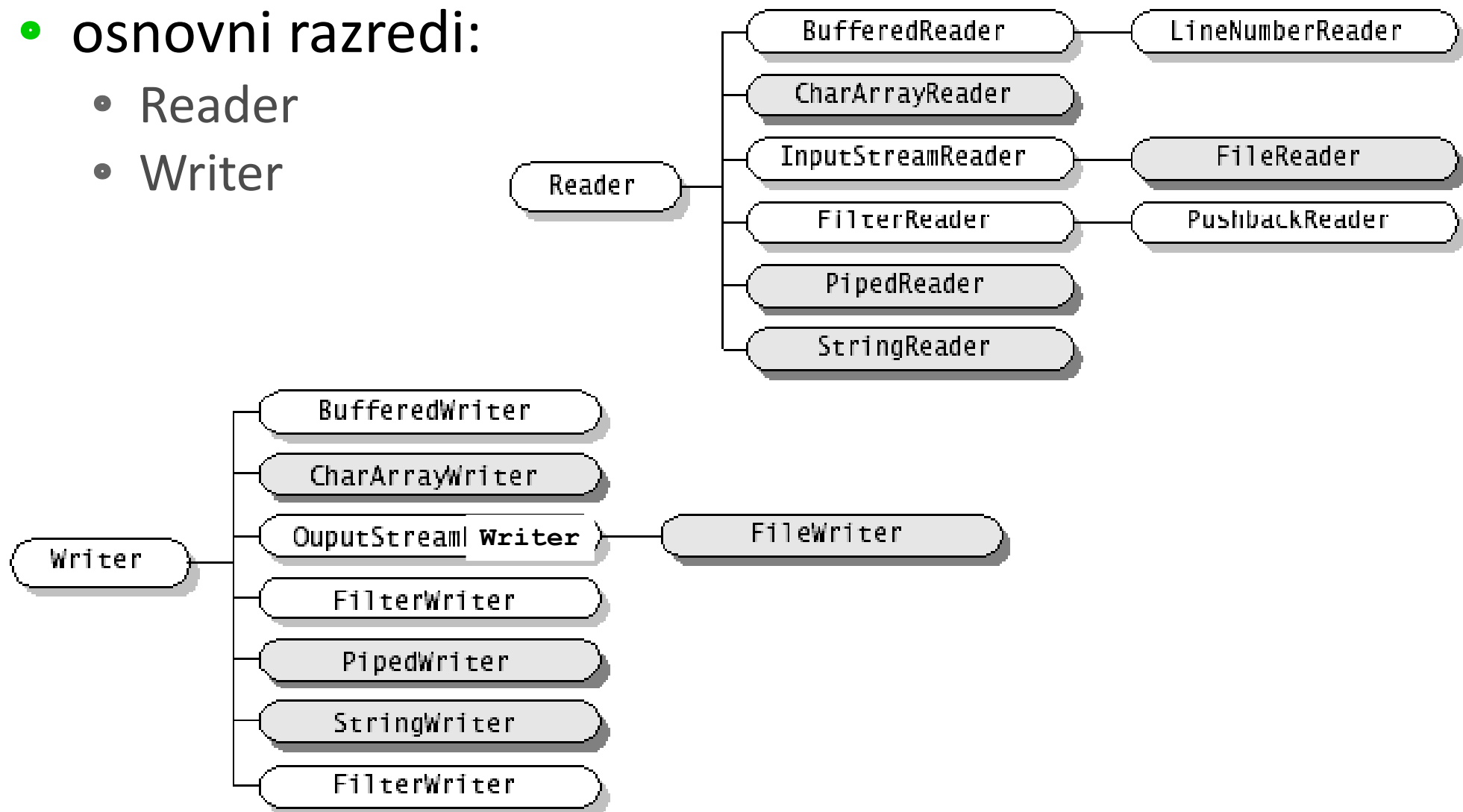
```
//ispis sadržaja datoteke(naredba cat ili type) po uzoru na C

import java.io.*;
public class Cat{
    public static void main(String args[])throws Exception{
        FileReader ulaz = new FileReader(args[0]);
        PrintWriter izlaz = new PrintWriter(System.out,true);
        char c[] = new char[4096];
        int broj = 0;
        while ((broj = ulaz.read(c)) != -1)
            izlaz.write(c, 0, broj);
        ulaz.close();
        izlaz.close();
    }
}
```


Znakovni tokovi

- osnovni razredi:

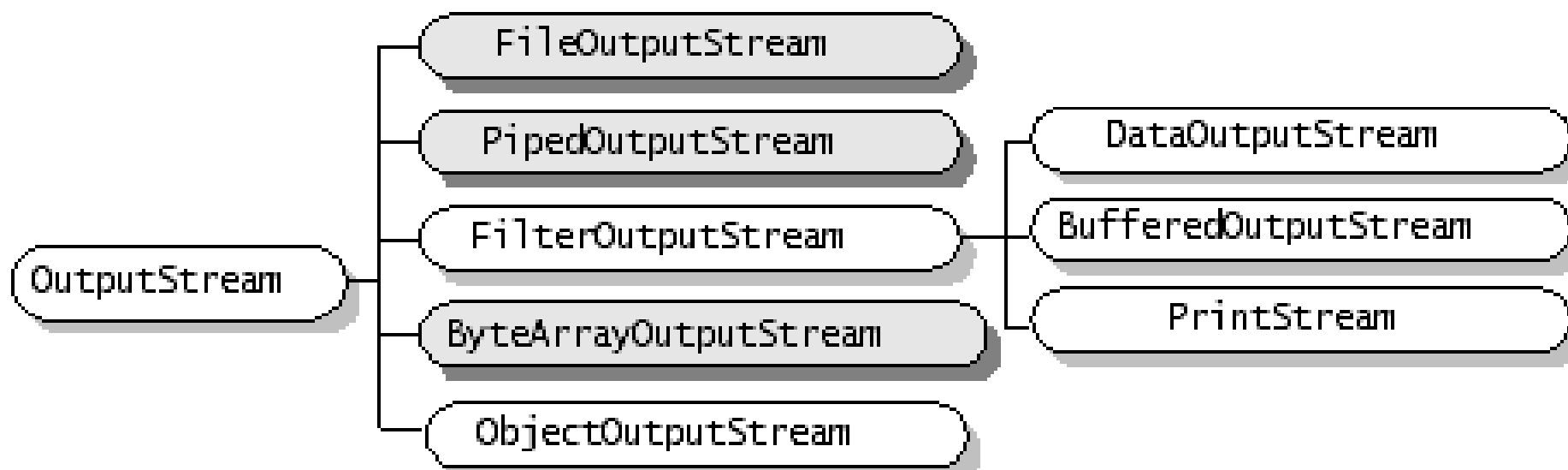
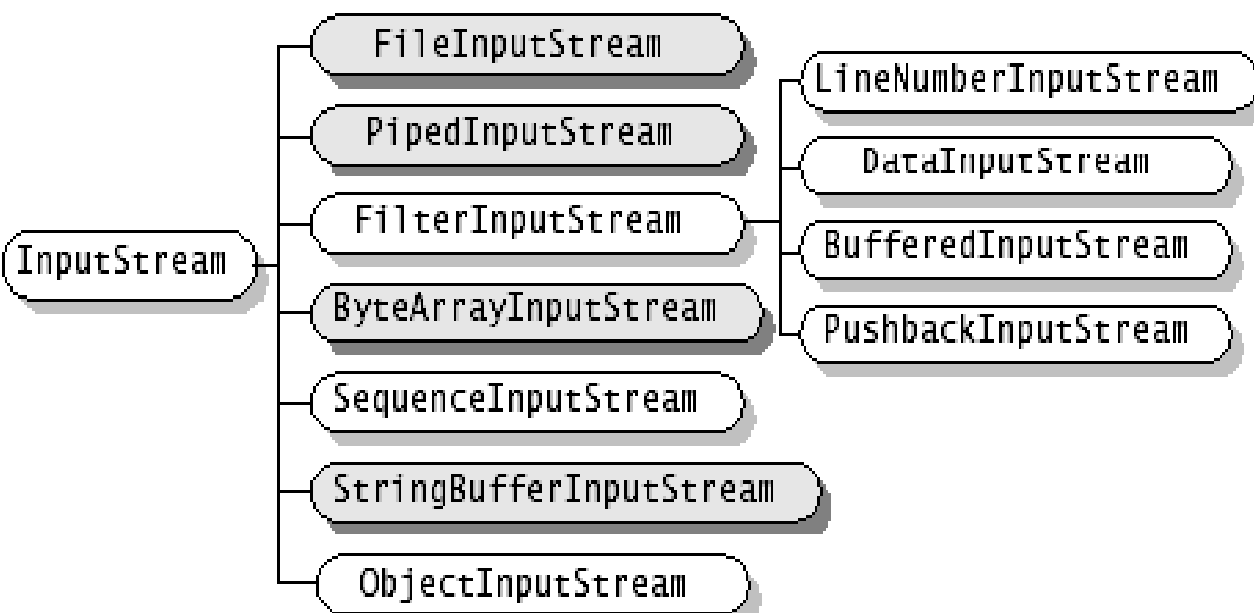
- Reader
- Writer



Podatkovni tokovi

- osnovne razredi:

- InputStream
- OutputStream



Osnovne metode

- I znakovni i podatkovni tokovi definiraju slične metode:

```
int read()
int read(char cbuf[])
int read(char cbuf[], int offset, int length)
```

java.io.Reader

```
int read()
int read(byte cbuf[])
int read(byte cbuf[], int offset, int length)
```

java.io.InputStream

```
int write(int c)
int write(char cbuf[])
int write(char cbuf[], int offset, int length)
```

java.io.Writer

```
int write(int c)
int write(byte cbuf[])
int write(byte cbuf[], int offset, int length)
```

java.io.OutputStream

Otvaranje i zatvaranje

- svi tokovi se automatski otvaraju pri stvaranju, a zatvaraju:
 - eksplicitnim pozivom *close()* ili
 - implicitno kad više nisu potrebni (*garbage collector*)

Standardni tokovi podataka

- u C-u: *stdin*, *stdout*, *stderr*
- u Javi:

Stvara ih
virtualni
stroj

```
public final class System {
    //...
    //standardni ulaz
    public final static InputStream in=...;
    //standardni izlaz
    public final static PrintStream out=...;
    //standardni izlaz za greske, ispisuje
    //na konzolu (kao i out)
    public final static PrintStream err=...;
    //...
}
```

java.lang.System

```
System.out.println("123")
```

Ulančavanje tokova podataka

- Čitanje podataka pomoću razreda Reader -možemo učitavati:
 - znak po znak
 - N znakova odjednom
- Što ako želimo učitavati redak po redak?
 - učitavati znak po znak, provjerati oznaku kraja retka?
 - učitati N znakova pa tražiti oznake kraja retka?
- NE!, postoje razredi koje to mogu bolje

Ulančavanje tokova podataka

- Tokovi podataka mogu se ulančavati
 - npr. razred *BufferedReader* može čitati podatke od *InputStreamReader*-a:

```
BufferedReader breader = null;
breader = new BufferedReader(new FileReader(f));
```

- čitanje jednog retka:

```
String redak=breader.readLine();
```

- osim dodatnih metoda, postiže se i ubrzanje rada
- i drugi tokovi se mogu ulančavati

/Primjer/Likovi/Citac.java

Rad s datotekama

- Razred *java.util.File* predstavlja **IME** datoteke ili kazala uključujući put (puni ili relativni)
- Datoteka ili kazalo ne moraju postojati
- Ima metode koje omogućuju:
 - provjere:
 - da li datoteka/kazalo postoji
 - da li je moguće čitanje ili pisanje
 - stvaranje datoteka
 - brisanje datoteka i kazala
 - dohvaćanje sadržaja kazala
 - stvaranje kazala
 - promjenu imena
 - etc ...

/Primjeri/TurnAround.java

Imena datoteka na različitim OS-ima



- Interni zapis imena datoteke ili kazala neovisan je o operacijskom sustavu
- Prilikom pretvorbe tog apstraktnog zapisa imena u niz znakova (*String*), koriste se osobine platforme na kojoj se izvodi aplikacija *System.getProperty()*
 - Linux koristi '/' za razdvajanje kazala
 - Windowsi koriste '\'
- Polje *java.io.File.pathSeparator*

```
/Primjeri/PrintThySelf.java
```

The background features large, light green, stylized letters 'O' and 'R' that are partially visible, framing the central text.

Kolekcije

Skupovi objekata (*Collections*)

- Kolekcija = objekt koji sadrži druge objekte
- Kolekcija omogućuje:
 - spremanje objekata
 - dohvat objekata
 - manipulaciju objektima
- Obično predstavlja podatke koji su grupirani i u stvarnom svijetu
 - npr. imenik je skup zapisa o kontaktima

Javin okvir kolekcija

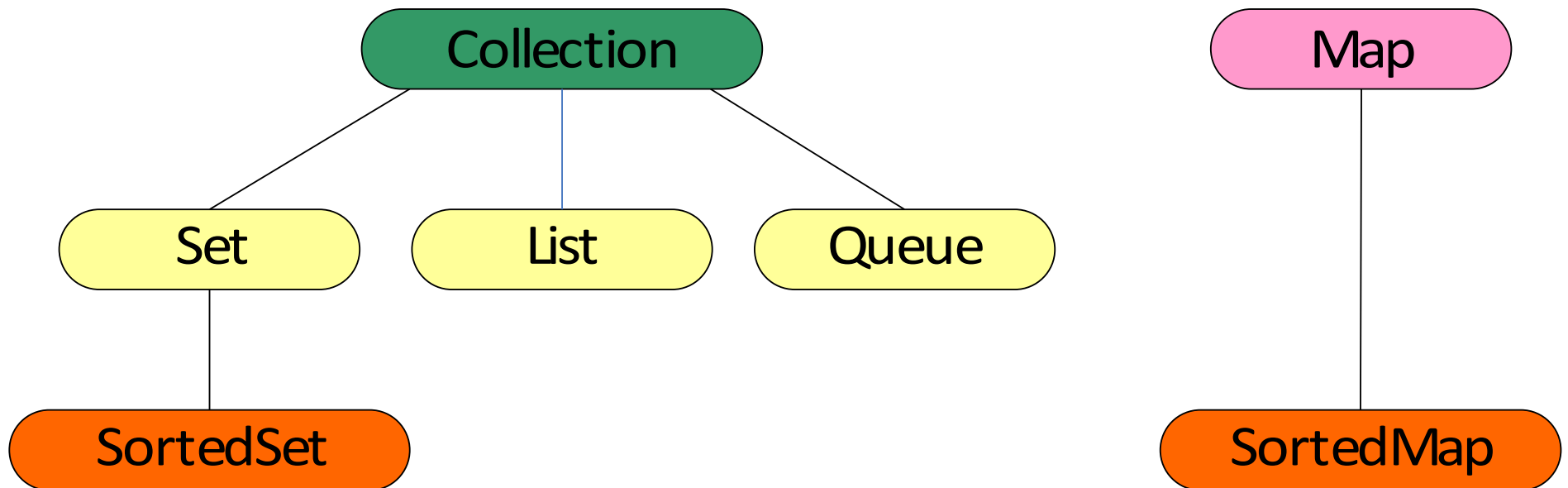


- Okvir kolekcija (*collections framework*)
 - jedinstvena arhitektura za predstavljanje i manipulaciju skupovima objekata
- Sastoji se od:
 - sučelja
 - predstavljaju razne kolekcije, omogućuju manipulaciju skupom objekata neovisno o implementaciji
 - implementacija
 - implementacije sučelja, ponovno iskoristive strukture podataka
 - algoritama
 - algoritmi za korisne operacije nad kolekcijama
 - npr. sortiranje i pretraga

Prednosti okvira kolekcija

- Olakšava kodiranje
 - ne treba implementirati vlastite kolekcije
- Povećava kvalitetu i brzinu programa
 - isprobane i optimirane standardne implementacije
- Omogućuje suradnju između nepovezanih programskih sučelja
 - različita sučelja (API) komuniciraju razmjenjujući kolekcije objekata
- Olakšava učenje i primjenu novih programskih sučelja
 - ako ta sučelja koriste standardne kolekcija (umjesto vlastitih), lakše ih je shvatiti

Osnovna sučelja



- Sortirani skup je poseban slučaj skupa, koji je poseban slučaj kolekcije
- Mapa je posebna vrsta kolekcije

Osnovna sučelja

- Collection
 - općeniti skup objekata, nema direktne implementacije
- Set
 - kolekcija koja ne dozvoljava iste elemente
- List
 - uređena kolekcija – elementi se mogu indeksirati, može sadržavati iste elemente
- Queue
 - red, organiziran po FIFO principu
- Map
 - skup parova ključ-vrijednost,
 - ne dozvoljava dva ista ključa: jedan ključ, jedna vrijednost

Osnovna sučelja

- **SortedSet**
 - skup (Set) u kojem su elementi poredani u uzlaznom redoslijedu
- **SortedMap**
 - mapa (Map) u kojoj su ključevi poredani u uzlaznom redoslijedu

Generički tipovi (Generics)

- U primjeru s geometrijskim likovima prolazili smo kroz sve objekte u skupu (Vector)
- Prije generičkih tipova:

```
...
Vector kontejner= new Vector();
...
kontejner.add(new Pravokutnik(linija));
...
for(int i = 0; i < kontejner.size(); i++)
{
    površina=((Lik) kontejner.get(i)).izracunajPovršinu();
    ...
}
...
```

/Primjeri/Likovi/Citac.java

java.lang.Object ne zna za metodu *izracunajPovršinu()* - potreban cast na *Lik*

Što ako ne znamo tip objekta pohranjenog u skupu ili pogrešno pridjelimo tip (cast)?

Generički tipovi

- U slučaju pogrešne dodjele tipa (cast) u prethodnom primjeru, prevoditelj ne bi prijavio grešku
 - rješenje = tipizirati svaki skup objekata
- Generički tipovi postoje od Jave 5 (1.5)

Generički tipovi

- prethodni primjer s generičkim tipovima:

/Primjeri/Likovi/Citac.java

```
...
Vector<Lik> kontejner= new Vector<Lik>();
...
kontejner.add(new Pravokutnik(linija));
...
for(int i = 0; i < kontejner.size(); i++)
{
    povrsina=kontejner.get(i).izracunajPovrsinu();
    ...
}
...
```

Nema dodjele tipa ni
mogućnosti greške

Metoda *Vector.get(i)* je tipizirana i
vraća objekt tipa *Lik*

Sučelje Collection

- najvažnije metode:

```
public interface Collection<E> extends Iterable<E>
{
    // Basic operations
    int size();
    boolean isEmpty();
    boolean contains(Object element);
    boolean add(E element);           //optional
    boolean remove(Object element);  //optional
    Iterator<E> iterator();
    ...
}
```

deklaracija formalnog parametra tipa

java.util.Collection

Obilaženje elemenata

- najčešće korištena funkcionalnost kolekcije – obilazak svih elemenata
- dva načina:
 - petlja for-each
 - razred Iterator

Petlja for-each

- poseban oblik for petlje
- npr:

```
class ForEach {
    public static void main(String[] args) {
        int[] brojevi = {1, 2, 3, 4, 5};
        for (int broj : brojevi) {
            System.out.println(broj + "/5");
        }
    }
}
```

/Primjeri/ForEach.java

- ispis:
 - 1/5
 - 2/5
 - 3/5
 - 4/5
 - 5/5

Iterator

- svaki skup objekata moguće je obilaziti pomoću sučelja *Iterator*:

```
public interface Collection<E> extends Iterable<E>
{
    Iterator<E> iterator();
    ...
}
```

java.util.Collection

- sučelje *Iterator*, osim pomicanja na sljedeći element (*next()*), omogućuje i provjeru da li element postoji (*hasNext()*) , i brisanje iz kolekcije (*remove()*):

```
public interface Iterator<E> {
    boolean hasNext();
    E next();
    void remove();
}
```

java.util.Iterator

```
Vector<Lik> kontejner= new Vector<Lik>(); /Primjeri/Likovi/Citac.java
...
kontejner.add(new Pravokutnik(linija));
...
for(int i = 0; i < kontejner.size(); i++)
{
    povrsina=kontejner.get(i).izracunajPovrsinu();
    ukupna+=povrsina;
    System.out.print(kontejner.get(i));
    System.out.print(" \tpovrsina=[" + (povrsina) + "]cm^2\n");
}
```

Koristimo sučelje List za
izdvajanje elemenata skupa

Ali kad bi koristili sučelje
Iterator za obilaženje skupa ...

```
for(Iterator<Lik> iterator=kontejner.iterator(); iterator.hasNext();)
{
    Lik trenutniLik=iterator.next();
    povrsina=trenutniLik.izracunajPovrsinu();
    ukupna+=povrsina;
    System.out.print(trenutniLik);
    System.out.print(" \tpovrsina=[" + (povrsina) + "]cm^2\n");
}
```


The background of the slide features large, light green, stylized letters 'O' and 'R' that are partially visible, creating a decorative frame around the central text.

Dokumentiranje kôda

Pisanja dobrog kôda

- standardi pisanja kôda:
 - pravila nazivanja elemenata jezika
 - često pred-određena samim jezikom, de-facto norma
 - pravila formatiranja i uvlačenja kôda
 - stvar osobne preferencije
- cilj – čitljiv i lako razumljiv kôd
- kako ga postići?
 - pridržavati se normi programskog jezika
 - odabrani standard konzistentno koristiti

Imena u Javi

- imena razreda i sučelja – velikim početnim slovom, sve ostalo malim
- CamelCase (camelCase) – za zapisivanje složenica od više riječi

Dokumentacija

- programer opisuje svoj uradak:
 - za sebe
 - za ostale programere
 - za korisnike
- samoopisivi programski kôd:
 - identifikatori razumljivi po imenu
 - objekti odgovaraju problemu
 - komentari:
 - prvo napisati komentar, pa onda funkciju
 - razumljivi, jezgroviti

Komentari u Javi

- "obični":
 - `/* komentar */`
 - (ne prevodi se tekst između `/*` i `*/`).
 - `// komentar`
 - (ne prevodi se tekst od `//` do kraja linije)
- komentari za dokumentaciju:
 - `/** dokumentacija */`
 - (ne prevodi se tekst između `/**` i `*/`).
 - alat javadoc koristi tekst komentara za automatsko generiranje dokumentacije

Javadoc komentar

- Javadoc komentar:
 - napisan je u jeziku HTML
 - stoji ispred deklaracije **razreda, konstruktora ili metode**
 - sastoji se od dva dijela:
 - opisa
 - skupa oznaka

```
/**
 * Returns a new string that is a substring of this string. The
 * substring begins with the character at the specified index
 * and extends to the end of this string. <p>
 * Examples:
 * <blockquote><pre>
 * "unhappy".substring(2) returns "happy"
 * "Harbison".substring(3) returns "bison"
 * "emptiness".substring(9) returns "" (an empty string)
 * </pre></blockquote>
```

substring

```

 * @param      beginIndex the beginning index, inclusive.
 * @return     the specified substring.
 * @exception  IndexOutOfBoundsException if beginIndex is negative or larger than the length of
 *      this String object.
 *
 * Examples:
 *
 * "unhappy".substring(2) returns "happy"
 * "Harbison".substring(3) returns "bison"
 * "emptiness".substring(9) returns "" (an empty string)
 */
public String substring(int beginIndex) {
    return substring(beginIndex, length());
}

```

Parameters:

beginIndex - the beginning index, inclusive.

Returns:

the specified substring.

Throws:

[IndexOutOfBoundsException](#) - if beginIndex is negative or larger than the length of this String object.

Javadoc komentari



- moguće oznake:
 - @param
 - @return
 - @throws (@exception)
 - @author
 - @version
 - @see
 - @since
 - @serial
 - @deprecated
- navode se tim redom!

Alati




- *javadoc* – dolazi s Sunovim JDK-om
- *doxygen* – otvoreni alat, prepoznaje *javadoc* komentare (između ostalih), mnoštvo izlaznih formata

<http://svn.wikimedia.org/doc/>
(dokumentacija wiki sustav Wikipedie)

Pišimo bolji kôd...

- uzorci u dizajnu kôda (design patterns) – rješenja problema koji se često susreću
- uzorci:
 - kad ih primjeniti?
 - koji problemi su prikladni?
- svaki uzorak ima:
 - jasno definiran problem koji rješava
 - ostvarenje (pseudo ili u sintaksi nekog jezika)
 - posljedice
- više: [Thinking in Patterns with Java](#)



Javini nasljednici

Sve ovo u C#?

- vrlo slični jezici
- suptilne razlike
- više: C# vs Java

A large, bold, green letter 'R' is positioned on the left side of the slide, partially cut off by the edge.

Pitanja?