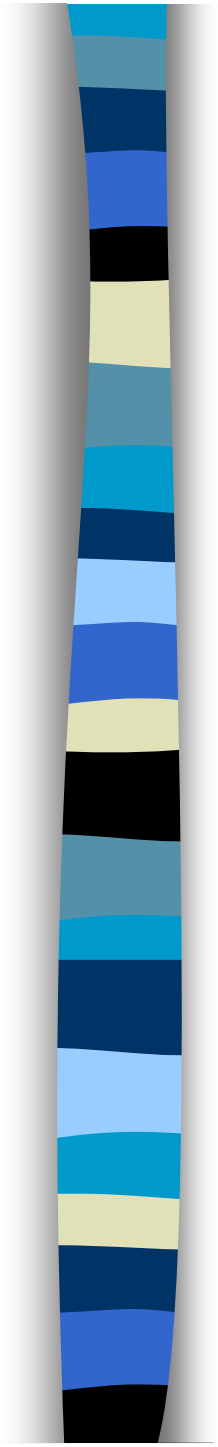


Paketi

Paketi

- Paketi i jedinice prevođenja
- Paketi i direktorijumi
- Ime paketa
- Dizajn paketa
- Primer

Paketi

- Napravljene klase i interfejsi se mogu grupisati u posebne celine koje se u Javi nazivaju ***paketi.***
- Svaka klasa i svaki interfejs pripada tačno jednom paketu.
- U isti paket obično smeštamo one referencijalne tipove koji čine neku logičnu celinu – tipove koji se koriste zajedno ili tipove koji se koriste za rešavanje iste vrste problema.
- Sve klase i interfejsi u jednom paketu (osim privatnih ugnježdenih klasa i interfejsa) su vidljivi u svim delovima paketa – mogu se koristiti bilo gde u tom paketu.

Paketi

- Da bi neugnježdena klasa ili neugnježdeni interfejs bili vidljivi izvan svog paketa, oni moraju biti deklarisani pomoću modifikatora **public**. Ostale klase i interfejsi u paketu služe samo kao pomoćni tipovi uz pomoć kojih se prave klase i interfejsi deklarisani sa public modifikatorom.
- Osim paketa koje Java programeri sami prave, postoje i standardni paketi koji se isporučuju zajedno sa Java prevodiocem, Java interpreterom i ostalim Java alatima - **Java API** (*Application Programming Interface*)
- Java API sastoji se od više desetina paketa u kojima ukupno ima više od 1000 napravljenih klasa i interfejsa deklarisanih sa public modifikatorom
- Svaki od tih referencijalnih tipova možemo koristiti za pravljenje novih referencijalnih tipova

Paketi

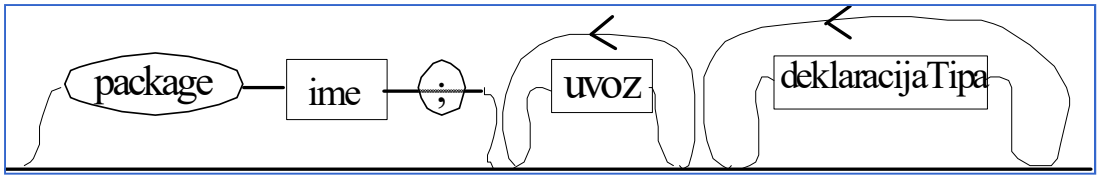
- Način grupisanja međusobno povezanih klasa i interfejsa
- Način organizovanja klasa u grupe
- Paketi sadrže klase povezane zajedničkom svrhom
- Paketi su korisni iz sledećih razloga:
  - Omogućavaju organizaciju klasa u jedinice (units)
  - Smanjuju problem konflikta naziva
  - Omogućavaju bolju zaštitu
  - Mogu se koristiti za identifikaciju klasa

Paketi

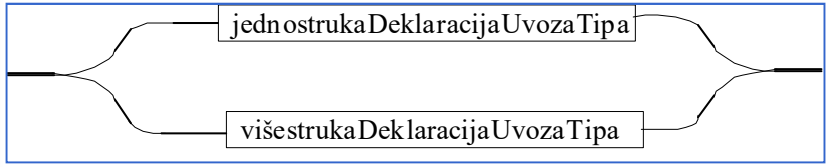
- Paketi mogu sadržati druge pakete
- Svaki nivo predstavlja malo određenije grupisanje klasa
- Biblioteka klasa u Javi sadržana je u paketu nazvanom java
- Sledeći nivo uključuje io, net, util i awt
- Java.lang
- Mehanizmi korišćenja klase koja je u nekom paketu:
  - Klase iz java.lang paketa su automatski dostupne
  - Korišćenje klase iz drugog paketa
  - Korišćenje klasa iz uvezenih paketa (komanda import)
  - Bezimeni, default paket

Paketi i jedinice prevođenja

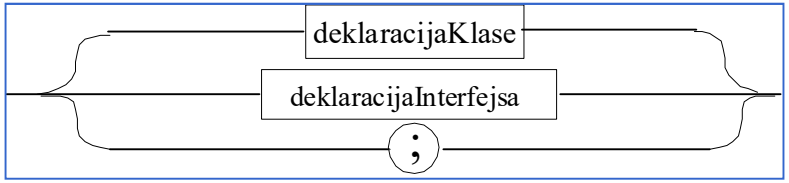
- Sve klase i interfejsi čije su deklaracije navedene u okviru neke jedinice prevođenja (fajla) pripadaju istom paketu
- Ako se ključna reč **package** i ime paketa ne navedu, tada će sve klase i interfejsi deklarisani u toj jedinici prevođenja pripadati *anonimnom* (neimenovanom) *paketu*



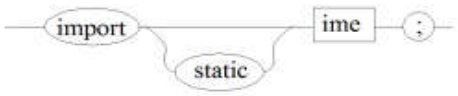
Jedinica prevođenja



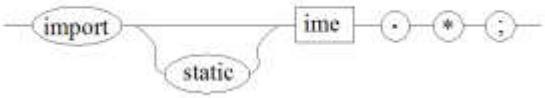
uvoz



deklaracijaTipa



jednostrukaDeklaracijaUvoza



višestrukaDeklaracijaUvoza

Primeri imena paketa

```
package aritmetika ;
package fizika.atomska ;
package MojiPaketi.DiplomskiRad.PrviDeo ;
```

Primeri uvoza

```
import java.util.Vector ;
import java.util.* ;
java.util.Vector v = new java.util.Vector() ;
```

Paketi i jedinice prevođenja

Primeri uvoza

```
import java.util.Vector ;
import java.util.* ;
java.util.Vector v = new java.util.Vector() ;
```

- Komandom **`import`** se uvoze klase iz paketa:
  - Uvoz individualne klase (import java.awt.Font)
  - Uvoz celog paketa klasa (import java.awt.*)
  - Neophodno je navesti svaki nivo hijerarhije (import java.awt.image.*)
  - import se nalazi na vrhu datoteke pre definicije klase

Primer 8.5: Statički uvoz pojedinačnih imena

```
import static java.lang.System.out;
import static java.lang.Math.cos;
import static java.lang.Math.PI;
```

Imena svih dostupnih statičkih članova klase `java.lang.Math` mogu se uvesti na sledeći način:

Primer 8.6: Istovremeni uvoz svih dostupnih statičkih članova klase

```
import static java.lang.Math.*;
```

Paketi i jedinice prevođenja

- Kao i kod imena tipova, ako se ime statičkog člana ne uveze na jedan od dva opisana načina, ono se može koristiti navođenjem kvalifikovanog imena.
- Kvalifikovano ime člana referencijalnog tipa sastoji se od kvalifikovanog imena tipa, posle kog sledi tačka i ime tog člana.
- Na primer, ako ne bismo koristili kvalifikovane oblike imena, kao u sledećem fragmentu koda.

Primer 8.7: Upotreba kvalifikovanih imena statičkih članova klase

```
double r = Math.cos(Math.PI * theta);
System.out.println(r);
```

Ako se gore pomenuta imena statičkih članova klase uvezu, isti kod mogao bi izgledati kao u sledećem primeru.

Primer 8.8: Upotreba statički uvezenih članova klase

```
double r = cos(PI * theta);
out.println(r);
```

Paketi i jedinice prevođenja

- Upotreba statičkog uvoza preporučuje se u slučajevima kada postoji potreba za **čestim** pristupom statičkim članovima iz jedne ili eventualno dve klase.
- Preterano korišćenje statičkog uvoza može dovesti do teško čitljivog programa, s obzirom da praćenje koje ime potiče iz koje klase može postati otežano – naročito ako se koristi uvoz svih statičkih članova.
- Ukoliko se koristi pravilno, statički uvoz zapravo može povećati čitljivost koda, time što odstranjuje dugačke i nepotrebne delove kvalifikovanih imena (za koje je očigledno odakle potiču i bez prisustva kvalifikovanog dela imena).

Paketi i jedinice prevođenja

Primeri uvoza

```
import java.util.Vector ;
import java.util.* ;
```

- Ako se ime nekog tipa iz nekog drugog paketa ne uveze na jedan od dva gore navedena načina, ono se ipak može koristiti u jedinici prevođenja ali samo ako se navede kao kvalifikovano ime
- Kvalifikovano ime se sastoji od imena paketa kojem taj tip pripada, za kojim sledi tačka i ime tog tipa
- Npr. ako ne bismo uvezli klasu Vector iz paketa **java.util** ime ovog tipa bismo u kodu morali zapisivati sa **java.util.Vector**

Primeri uvoza

```
java.util.Vector v = new java.util.Vector() ;
```

Konflikti naziva

- Nakon uvoza klase ona se može koristiti bez naziva paketa

```
import java.util.*;
//kod
List l = new List();
```

- Slučaj više klasa sa istim nazivom iz različitih paketa

```
import java.util.*; (sadrži klasu List)
import java.awt.*; (sadrži klasu List)
```

Konflikti naziva

- Java prevodilac neće prevesti sledeći kod

```
List list = new List();
```

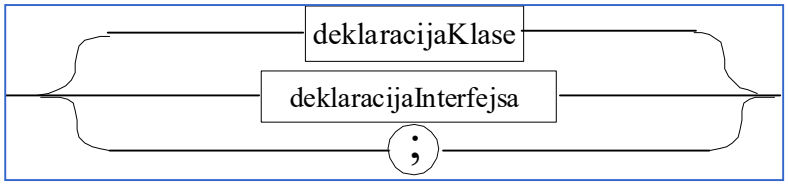
- Rešenje je u eksplicitnom navođenju punog naziva klase

```
java.awt.List jList = new java.awt.List();
java.util.List fList= new java.util.List();
```

Paketi

- Paket predstavlja jedinstveno imenovanu kolekciju klasa.
- Imena klasa jednog paketa neće se mešati sa imenima klasa nekog drugog paketa jer se imena klasa paketa kvalifikuju imenom tog paketa. Npr. puno ime klase ***String*** iz paketa ***java.lang*** je ***java.lang.String***.
- Ako bismo definisali svoju klasu sa istim imenom, ***String***, korišćenje imena ***String*** odnosilo bi se na tu našu klasu, dok bismo se standardnoj klasi ***String*** morali obraćati sa ***java.lang.String***.

Paketi



deklaracijaTipa

- Treći deo jedinice prevođenja je obično najduži. U njemu se deklarišu klase i interfejsi.
- U jednoj jedinici prevođenja može biti deklarisan proizvoljan broj referencijalnih tipova, ali najviše jedan od deklarisanih neugnježdenih tipova može biti deklarisan sa modifikatorom **public**.
- Ako u jedinici prevođenja imamo neugnježden tip deklarisan sa public, tada ime jedinice prevođenja mora biti jednako imenu tog tipa na koje se još dodaje nastavak .java
- Ako u jedinici prevođenja nema ni neugnježdenih klasa ni neugnježdenih interfejsa deklarisanih sa modifikatorom public, tada ne postoji ograničenje na ime jedinice prevođenja, sem da se ona mora završavati sa .java.

Paketi i direktorijumi

- Navođenje imena paketa kome klasa pripada

```
package Geometry;
public class Sphere {
// Details of the class definition
}
```

- package mora da bude prva naredba u fajlu (ne računajući prazne linije i komentare).
- Svi fajlovi koji sadrže definicije klasa koje pripadaju paketu Geometry **moraju biti smešteni u direktorijumu koji nosi ime samog paketa**, dakle, Geometry.

Note the use of the `public` keyword in the definition of the `Sphere` class. This makes the class accessible generally. If you omit the `public` keyword from the class definition, the class would be accessible only from methods in classes that are in the `Geometry` package.

Note that you would also need to declare the constructors and methods in the class as public if you want them to be accessible from outside of the package.

Paketi i direktorijumi

- Za svaku klasu i za svaki interfejs koji su deklarisani u jedinici prevođenja se u toku prevođenja kreira po jedan fajl. Prvi deo imena novog fajla jednak je imenu odgovarajuće klase ili interfejsa, a drugi je nastavak .class
- Ako u jedinici prevođenja ***A.java*** imamo deklaraciju klasa ***A*** i ***B***, koje nemaju ugnježdene klase i intefejse, i deklaraciju interfejsa ***C***, tada će prevođenjem te jedinice nastati fajlovi ***A.class***, ***B.class*** i ***C.class***. Prevođenjem ugnježdene klase nastaće fajl oblika ***ImeSpoljasnje$ImeUgnjezdjene.class***

> Ime fajla koji odgovara prevedenoj ugnježdenoj klasi (interfejsu) se gradi pomoću imena spoljašnje klase, znaka $ i imena ugnježdene klase (interfejsa). Na primer: `ImeSpoljasnje$ImeUgnjezdjene.class`.

Paketi i direktorijumi

- Direktorijum u koji Java prevodilac smešta fajlove sa ekstenzijom **.class** nastale prevođenjem neke jedinice prevođenja, se određuje na osnovu imena paketa kojem ta jedinica prevođenja pripada.
- Ime paketa se sastoji od proizvoljnog broja identifikatora međusobno razdvojenih tačkama.
- Za svaki od ovih identifikatora se pravi po jedan direktorijum sa imenom jednakim tom identifikatoru, tako da je svaki sledeći direktorijum poddirektorijum prethodog.
- Fajlovi sa ekstenzijom .class koji odgovaraju tipovima iz tog paketa se smeštaju u najdublji (poslednji) od tih direktorijuma.

Paketi i direktorijumi

- Ako je ime paketa **MojiPaketi.Posao.Projekat**,
  - tada će biti napravljen direktorijum **MojiPaketi**,
  - u okviru kojeg se pravi poddirektorijum **Posao**,
  - u okviru kojeg se pravi poddirektorijum **Projekat**.
  - Svi fajlovi nastali prevođenjem jedinica prevođenja koje pripadaju ovom paketu će biti smešteni u direktorijum **Projekat**

Paketi i direktorijumi

- Paketi su usko povezani sa strukturom direktorijuma u kojima se čuvaju.
- Možemo generisati *.class* fajl za neku klasu iz paketa u drugom direktorijumu, ali taj direktorijum takođe mora imati isto ime kao i paket.
- Ime paketa može biti složeno, npr. *geometrija.geometrija3D*, što znači da je *geometrija3D* poddirektorijum direktorijuma *geometrija.*
- Ime može biti proizvoljne složenosti, ali treba da odražava strukturu direktorijuma u koju je paket smešten.

Paketi i direktorijumi

- Na primer, ako pravimo više kolekcija klasa koje se tiču jedne iste oblasti (*geometrija*), možemo da kreiramo više paketa: *geometrija2D*, *geometrija3D*, ...

- Podrazumeva se da se klase iz prvog paketa nalaze u direktorijumu *geometrija2D*, iz drugog u direktorijumu *geometrija3D* i da su *geometrija2D* i *geometrija3D* poddirektorijumi direktorijuma *geometrija*.

Paketi i direktorijumi

- Ako želimo da uključimo neke klase iz paketa geometrija2D, pišemo:

```
import geometrija.geometrija2D.*;
```

- Ne može se pisati:

```
import geometrija.*;
```

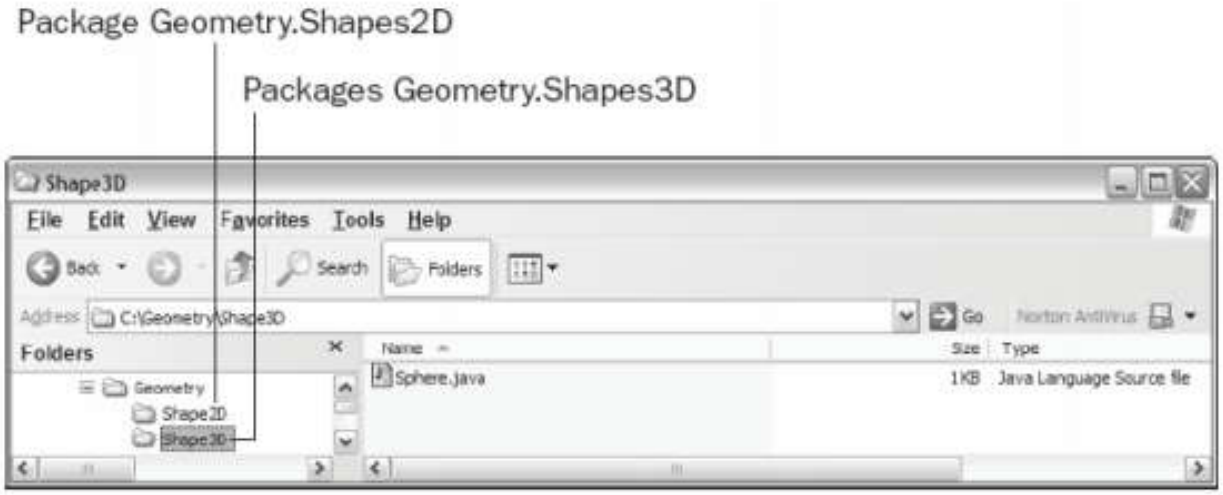
za uključivanje svih paketa iz direktorijuma *geometrija2D*.

- Znak * se odnosi na sve klase jednog paketa.

Paketi i direktorijumi

```
package Geometry.Shapes3D;
public class Sphere {
   // class definition
   }
```

```
package Geometry.Shapes2D;
public class Sphere {
   // class definition
   }
```



- You can compile the source for a class within a package and have the `.class` file that is generated stored in a different directory, but the directory name must still be the same as the package name.

Paketi i direktorijumi

- Kada se prevodi jedinica prevođenja koja pripada nekom neanonimnom paketu, tada se Java prevodiocu mora navesti i opcija **–d** kojom se određuje ime direktorijuma koji se smatra korenskim (početnim) za taj paket.
- Na primer, ako jedinica prevođenja **razno.java** pripada paketu **MojiPaketi.Posao.Projekat**, i ako je prevodimo komandom

javac -d *c:\sviPaketi razno.java*

tada će Java prevodilac sve fajlove kreirane ovim prevođenjem staviti u direktorijum ***c:\sviPaketi\MojiPaketi\Posao\Projekat***

- Ako direktorijumi koji odgovaraju identifikatorima u imenu paketa ne postoje ispod direktorijuma c:\sviPaketi, onda će ih Java prevodilac sam kreirati.

Paketi i direktorijumi

- Direktorijum koji se navodi iza –d u pozivu Java prevodioca može biti naveden i korišćenjem relativne putanje, na primer javac –d../prog kk.java
- Jedinica prevođenja koja pripada anonimnom paketu se ne mora prevoditi sa –d opcijom
- Ako se opcija –d ne navede, fajlovi koji su rezultat prevođenja te jedinice prevođenja će biti smešteni u isti direktorijum u kojem se nalazi i ta jedinica prevođenja

Paketi i direktorijumi

- Fajlovi sa ekstenzijom **.class** nastali prevođenjem se mogu kompresovati korišćenjem posebnog alata za kompresovanje koji se isporučuje zajedno sa Javom i zove se **jar** ili pomoću nekog drugog alata za kompresovanje koji koristi **zip** format za kompresovanje podataka (na primer WinZip).
- Kompresovani paketi se mogu koristiti kao i svi ostali paketi, uz uštedu memorijskog prostora.
- Kompresovanjem paketa se kompresuju i odgovarajući direktorijumi u kojima se paketi čuvaju.
- Fajl dobijen kompresovanjem jednog ili više paketa se može čuvati u bilo kom direktorijumu (ne postoji ograničenje na ime direktorijuma kao kod nekompresovanih paketa).

Traženje paketa

- Da bi prevedeni delovi paketa koji nisu deo Java API mogli biti pronađeni, mora se nekako navesti ime direktorijuma koji je korenski za taj paket ili putanja i ime fajla sa kompresovanim paketom (**.jar** ili **.zip** fajl).
- Java alatu možemo saopštiti gde treba da traži naše pakete:
  - korišćenjem **-classpath** opcije iza koje se navode korenski direktorijumi za pakete i/ili fajlovi koji sadrže kompresovane pakete,
  - korišćenjem **CLASSPATH** sistemske promenljive u kojoj se pamte korenski direktorijumi za pakete i/ili fajlovi koji sadrže kompresovane pakete

Primeri korišćenja classpath opcije

```
javac -classpath .;\java\mojiPaketi;..\nekiPaketi.jar -d . novaKlasa.java
```

Traženje paketa

- Imena direktorijuma i fajlova, koji se navode iza **-classpath** opcije ili u definiciji sistemske promenljive **CLASSPATH** se međusobno razdvajaju simbolom koji zavisi od operativnog sistema koji koristimo
- Na operativnom sistemu Windows koristi se tačka-zarez (;) a na Linuxu i ostalim verzijama Unixa se koristi dvotačka (:)
- Sistemsku promenljivu **CLASSPATH** u jednom od Windows operativnih sistema možemo definisati **SET** komandom kao u sl. primeru

Primeri korišćenja CLASSPATH sistemske promenljive

```
SET CLASSPATH=.;C:\posao\java\mojiPaketi;D:\noveStvari\nekiPaketi.jar
```

Traženje paketa

Primeri korišćenja CLASSPATH sistemske promenljive

```
SET CLASSPATH=.;C:\posao\java\mojiPaketi;D:\noveStvari\nekiPaketi.jar
```

- Ako se navede opcija –**classpath** prilikom poziva nekog Java alata vrednost sistemske promenljive **CLASSPATH** se neće uzimati u obzir
- Ako se opcija –**classpath** ne navede potrebne klase će se tražiti pomoću vrednosti sistemske promenljive **CLASSPATH**
- Ako se ne navede ni opcija –**classpath** niti postoji sistemska promenljiva **CLASSPATH** nestandardne klase će biti tražene samo od tekućeg direktorijuma – tj. kao da je navedena opcija –**classpath** sa tačkom (**classpath** .)
- Lokaciju paketa koji pripadaju Java API nije potrebno navoditi u okviru –**classpath** opcije ili u **CLASSPATH** sistemskoj promenljivoj

Traženje paketa

- Glavni program takođe može biti deo paketa
- Ako glavni program pripada paketu tj. postoji klasa u paketu koji sadrži metod **`public static void main (String []args)`**, tada se izvršavanje programa pokreće pozivom Java interpretera sa kvalifikovanim imenom te klase kao parametrom
- Na primer:

Primeri korišćenja classpath opcije

```
D:\posao> java –classpath C:\mojiPaketi;. Projekat.Diplomski.ProgramKlasa
```

Ime paketa

- Ime paketa se sastoji od proizvoljnog broja identifikatora međusobno razdvojenih tačkama.
- Java interpreter i svi ostali Java alati koriste sve pakete kao nezavisne celine, bez obzira na njihova imena
- (npr. ako imamo pakete sa imenima **Nauka.Matematika** i **Nauka.Matematika.Geometrija** i
- ako koristimo deklaraciju uvoza tipa **Nauka.Matematika.*** neće biti uvezeno ni jedno ime iz paketa **Nauka.Matematika.Geometrija** već samo imena svih dostupnih tipova iz paketa **Nauka.Matematika**).

Ime paketa

- Ime paketa treba birati pažljivo, da bi se izbegla mogućnost da dva programerska tima (ili dva programera) naprave dva različita paketa sa istim imenom.

- Standardna konvencija za izbor imena paketa preporučuje da prvi deo imena paketa bude obrnuta Internet adresa firme koja ili za koju se paket kreira. Na ovaj način se obezbeđuje globalna različitost imena.

Primeri imena paketa

```
Nauka.Matematika
Nauka.Matematika.Geometrija
```

Standardna konvencija za biranje imena

```
COM.SuperSoft.Simulacije.Letenje
rs.ac.ns.dmi.www.Nastava.Razno
rs.mojaFirma.mojPaket
```

Korišćenje Web adresa u imenima paketa ne znači da se ti paketi mogu pronaći na tim adresama. Web adrese se koriste samo kao pomoćno sredstvo da bi se postigla globalna različitost imena paketa.

Dizajn paketa

- Dva paketa kod kojih je ime jednog paketa prefiks imena drugog ne moraju biti ni u kakvom posebnom odnosu, i prevodilac ih tretira nezavisno.
- Ustaljena praksa je da se paket čiji je direktorijum poddirektorijum drugog paketa logički posmatra kao podpaket.
- Ako se sistem (aplikacija) sastoji od komponenti koje su predstavljene paketima tada svaka komponenta može sadržati i druge komponente predstavljene podpaketima, čime se dobija hijerarhijsko ustrojstvo sistema.
- Pri tom su komponente sistema sastavljene od jedne ili više klasa, odnosno drugih referencijalnih tipova.

Dizajn paketa

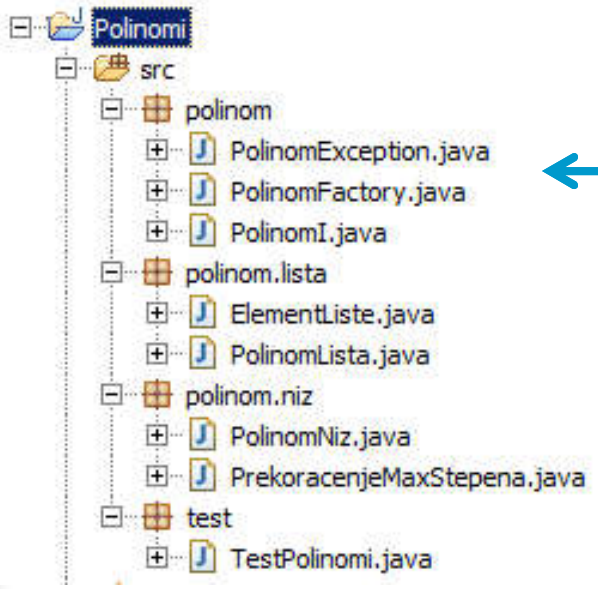
- U mnogim primenama hijerarhijsko ustrojstvo paketa je prirodan i poželjan način organizacije sistema.
- Da bi ovakav dizajn aplikacije bio uspešan, potrebno je u što većoj meri zadovoljiti princip visoke kohezije, a niske povezanosti komponenti sistema (eng. High cohesion, low coupling principle). U progr. jeziku Java ovo znači sledeće:
- **Hijerarhijsko ustrojstvo paketa**

  Jedna komponenta sistema je predstavljena jednim paketom.

  Komponente se sastoje iz podkomponenti.
- **"High cohesion, low coupling" princip**

  High cohesion: stepen zavisnosti/interakcija klasa (ref. tipova) unutar paketa je veći nego stepen zavisnosti sa klasama (ref. tipovima) van paketa.

  Low coupling: promena klase (ref. tipova) ne zahteva promenu klasa (ref. tipova) iz drugih paketa (lokalizovana propagacija promena).




```java
package test;

import polinom.PolinomException;
import polinom.PolinomFactory;
import polinom.PolinomI;

public class TestPolinomi {

    public static void main(String args[])
        throws PolinomException
    {
        PolinomI polinom1 = PolinomFactory.redak();
        polinom1.dodajMonom(2, 5.0);
        polinom1.dodajMonom(1, 4.3);

        PolinomI polinom2 = PolinomFactory.redak();
        polinom2.dodajMonom(3, 2.0);
        polinom2.saberiSa(polinom1);
    }

}
```

```java
package polinom;

import polinom.lista.PolinomLista;
import polinom.niz.PolinomNiz;

public class PolinomFactory {
    public static PolinomI gust(int maxStepen) {
        return new PolinomNiz(maxStepen);
    }

    public static PolinomI redak() {
        return new PolinomLista();
    }
}
```

```java
package polinom;

public interface PolinomI {
    void dodajMonom(int stepen, double koeficijent) throws PolinomException;
    void saberiSa(PolinomI pol) throws PolinomException;
    void pomnoziSa(PolinomI pol) throws PolinomException;
    void pomnoziSa(double skalar);
    void podeliSa(PolinomI pol) throws PolinomException;
    double izracunaj(double tacka);
    double koeficijent(int stepen);
    int najveciStepen();
}
```

Paketi Javine biblioteke

- Java raspolaže velikim brojem standardnih paketa, a najkorišćeniji su:
  - **java.lang** – osnovne karakteristike jezika, rad sa nizovima i stringovima. Klase iz ovog paketa su uvek dostupne našem programu – automatski se uključuje u naš program (klase Integer, String, Math ...)
  - **java.io** – klase za ulazno/izlazne operacije
  - **java.util** – klasa Vector (uopšteni niz), Stack, Scanner, klase za rad sa datumima, matricama i druge klase raznih vrsta
  - **javax.swing** – jednostavne za upotrebu i fleksibilne komponente za pravljenje GUI-ja (tzv. Swing komponente)
  - **java.awt** – originalne GUI komponente i osnovna podrška za neke Swing komponente

Paketi Javine biblioteke

PAKET	OPIS
java.rmi	Klase za udaljeni pristup.
java.security	Klase neophodne za izgradnju Java sigurnosnih aplikacija.
java.sql	Klase za pristup bazama podataka.
java.text	Klase koje se koriste za aplikacije koje ne zavise od jezika.
java.util	Sadrži alate i strukture podataka.

Primer - retki vektori

Fajl `RVektor.java` iz direktorijuma `C:\tekstovi`

```
package MojiPaketi.Retke;

import java.util.Vector; // jednostavno predstavljanje liste objekata

class Element {  //pomocna klasa
  private int indeks;
  private double vrednost;

  Element(int indeks, double vrednost) { // konstruktor
    this.indeks = indeks;
    this.vrednost = vrednost;
  }

  int ind() {
    return indeks;
  }

  double vre() {
    return vrednost;
  }

  void staviVred(double vrednost) {
    this.vrednost = vrednost;
  }
  Element duplikat() {
    return new Element(indeks, vrednost);
  }
}
```

Primer - retki vektori

```
public class Rvektor {
  private Vector elem;  // svi ne-nula elementi
  private int dim;      // dimenzija vektora

  public RVektor(int dim) {  //konstruktor
    this.dim = dim;
    elem = new Vector();
  }
  public int citajDim() {
    return dim;
  }
  public double citajEl(int indeks) {  // uzima komponentu vektora
    int vel = elem.size();
    int i = 0;
    boolean nadjen = false;
    boolean jos = true;
    while ((i < vel) && jos ) {
      if ( ((Element) elem.get(i)).ind() < indeks ) {
        i++;
      } else {
        jos = false;
        if ( ((Element) elem.get(i)).ind() == indeks ) {
          nadjen = true;
        }
      }
    }
    if (nadjen) {
       return ((Element) elem.get(i)).vre();
    } else {
      return 0;
    }
  }
```

Primer - retki vektori

```
public RVektor saberiSa(RVektor r) {
  if (dim != r.dim) {
    System.err.println("Vektori imaju razlicite dimenzije!");
    return null;
  } else {
    RVektor rez = new RVektor(dim);
    int vel = elem.size();  // broj ne-nula elemenata
    int rvel = r.elem.size();
    int i = 0;
    int ri = 0;
    while ((i < vel) && (ri < rvel)) {
      Element e = (Element) elem.get(i);
      Element re = (Element) r.elem.get(ri);
      if (e.ind() < re.ind()){
        rez.elem.add( e.duplikat() );
        i++;
      } else if (e.ind() > re.ind()) {
        rez.elem.add( re.duplikat() );
        ri++;
      } else {  //treba ih sabrati
        double zbir = e.vre() + re.vre();
        i++;
        ri++;
        if (zbir != 0){
          rez.elem.add( new Element(e.ind(), zbir) );
        }
      }
    }
```

Primer - retki vektori

```
for (int indeks = 1; indeks <= dim; indeks++) {
      if ((imaJos) && (e.ind() == indeks)) {
        sb = sb.append( e.vre() );
        i++;
        if (i < vel) {
          e = (Element) elem.get(i);
        } else {
          imaJos = false;
        }
      } else {
        sb = sb.append("0");
      }
      if (indeks < dim) {
        sb = sb.append(", ");
      } else {
        sb = sb.append(" )");
      }
    }
    return sb.toString();
  }
}
```

- Ako je tekući direktorijum računara **C:\tekstovi** i ako hoćemo da korenski direktorijum novog paketa bude **C:\paketi**, tada prevođenje fajla **RVektor.java** vršimo na sledeći način:

C:\tekstovi> javac -d \paketi RVektor.java

Primer - retki vektori

Fajl `Test.java` iz direktorijuma `C:\razno\probe`

```
import MojiPaketi.Retke.RVektor;

class Test{
  public static void main(String[] args){
    RVektor a = new RVektor(10);
    RVektor b = new RVektor(10);
    a.pisiEl( 8, 12.5);
    a.pisiEl( 3, 7);
    b.pisiEl( 10, 5);
    b.pisiEl( 1, 0.5);
    b.pisiEl( 3, -7);
    RVektor zbir = a.saberiSa(b);
    double pro = a.skalarnoMnoziSa(b);
    System.out.println( a );
    System.out.println( '+' );
    System.out.println( b );
    System.out.println( '=' );
    System.out.println( zbir );
    System.out.println( "a skalarni proizvod je " + pro );
  }
}
```

Prevođenje, pokretanje i ispis

```
C:\razno\probe> javac -classpath \paketi Test.java //prevođenje

C:\razno\probe> java -classpath \paketi;. Test //pokretanje


( 0, 0, 7.0, 0, 0, 0, 0, 12.5, 0, 0 )
```

Primer - retki vektori

Prevođenje

```
C:\razno\probe> javac -classpath \paketi Test.java
```

- Kreira fajl `Test.class` u direktorijumu `C:\razno\probe`

Pokretanje

```
C:\razno\probe> java -classpath \paketi;. Test
```

- U `-classpath` opciji navodimo 2 direktorijuma.
  - `\paketi` moramo navesti da bi bila pronađena klasa `MojiPaketi.Retke.RVektor`, a
  - tekući direktorijum (.) se navodi da bi bila pronađena `Test` klasa