Matematički fakultet, Univerzizet u Beogradu
Katedra za računarstvo i informatiku

Objektno orijentisano programiranje

vežbe
školska 2016/2017

Biljana Stojanović
Nemanja Mićović
Nikola Milev

1 Karakteristike programskog jezika Java

Mašinski nezavisna – "piši jednom izvršavaj bilo gde":
- nepromenjen Java program može se pokrenuti na bilo kojoj mašini i operativnom sistemu koji podržava Javu.

Objektno-orijentisan programski jezik:
- OO programi su lakši za razumevanje jer simuliraju spoljašnji svet; jednostavnije se nadograđuju i održavaju.

2 Java aplikacije

Java aplikacije su samostalni programi (koji mogu samostalno da se izvršavaju).

3 Pokretanje Java programa (aplikacije)

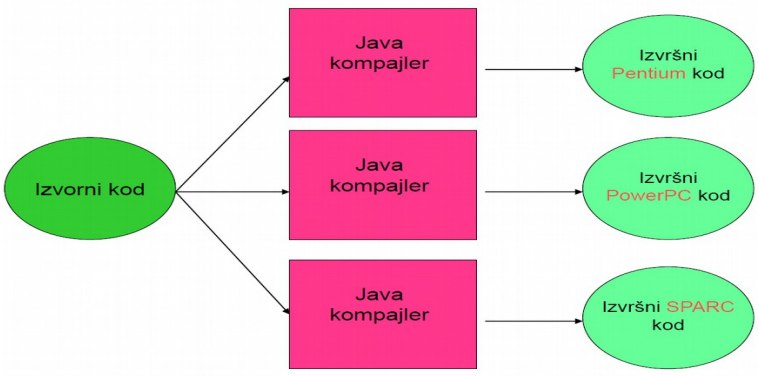
Java program se ne izvršava direktno na operativnom sistemu računara. Pokreće se u standardizovanom okruženju koje se zove **Java platforma**. Posledica toga je da se Java programi mogu pokrenuti na širokom spektru operativnih sistema (Windows, Linux, Solaris itd).

Java platforma

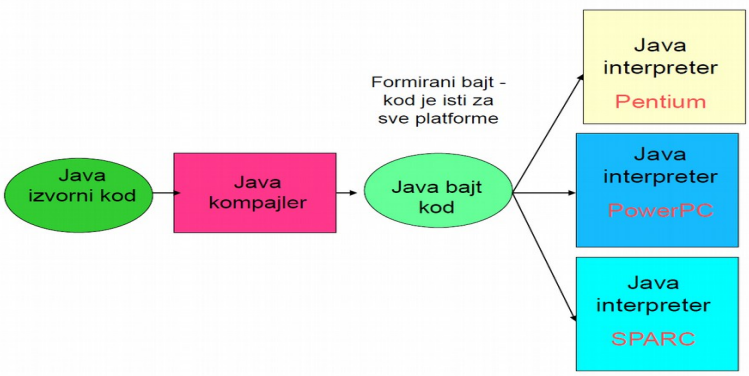
Java platforma se sastoji od dva elementa:
JAVA virtuelna mašina – JVM (osnovna komponenta)
- JVM je aplikativna virtuelna mašina, tj. softverska implementacija hipotetičkog računara (postoji samo u unutrašnjoj memoriji računara i dizajnirana je za pokretanje jednog programa (aplikacije)).
- Implementacija JVM može biti drugačija na različitim operativnim sistemima, ali je rezultat izvršavanja Java programa isti bez obzira na operativni sistem i implementaciju JVM.

JAVA application programming interface - API
- Skup komponenti koje obezbeđuju pisanje interaktivnih aplikacija u Javi (skup klasa raspoloživih programeru za upotrebu u razne svrhe, npr. za rad sa grafičkim korisničkim interfejsom i drugo).

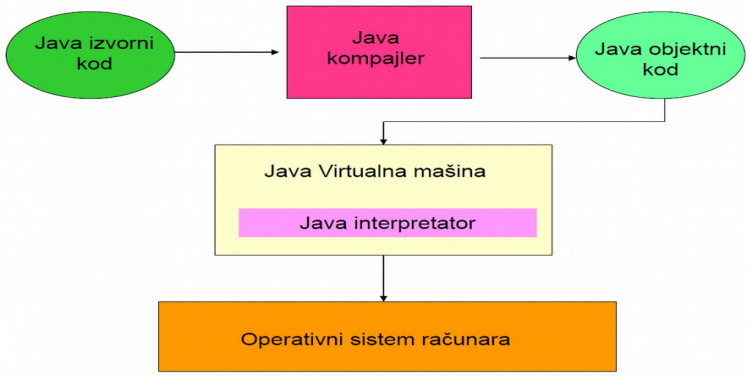
Trandicionalni način obrade izvornog programa



Način obrade izvornog Java programa



Java prevodilac (poziva se komandom *javac*) prevodi Java izvorni kod (*.java*) u tzv. bajtkod (.class) koji je binarni i isti za sve platforme ("*prevedi jednom izvršavaj bilo gde*").
Bajtkod (bytecode) su mašinske instrukcije za JVM.
Java program se izvršava pozivom *Java interpretatora (interpretera)* (komanda *java*), koji tumači i izvršava akcije specificirane u bajtkodu, unutar JVM.



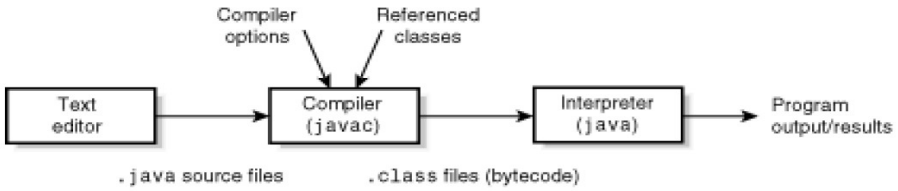
Primer pokretanja Java programa

Java izvorni kod pišemo u editoru i čuvamo sa ekstenzijom **.java** (Zdravo.java).
Prevodimo ga iz komandne linije pozivom Java prevodioca naredbom:

```
javac Zdravo.java
```

Java prevodilac generiše bajtkod program, koji je binarni ekvivalent izvornom Java programu.
Rezultujuća datoteka ima isto ime kao i izvorna, samo sa ekstenzijom **.class** (Zdravo.class).
Pozivom *Java interpretatora* izvršava se bajtkod program naredbom:

```
java Zdravo
```



Napomena: Ukoliko se izvorna .java datoteka nalazi u direktorijumu, tj. odgovarajuća klasa sadrži deklaraciju *package,* prevođenje i interpretaciju treba izvršiti na sledeći način:

- neka se klasa *Zdravo* nalazi u paketu *zdravo*, što znači da je izvorna datoteka *Zdravo.java* deo direktorijuma *zdravo.*
- pozicionirati se na roditeljski direktorijum direktorijuma *zdravo* (ako je tekući direktorijum *zdravo*, zadati komandu **cd ..**) i izvršiti naredbe:

```
javac zdravo/Zdravo.java
java zdravo/Zdravo
```

Ukoliko prevođenje ne prođe, česti uzroci mogu biti:

- promenljiva PATH nije podešena tako da uključuje putanju do programa *javac*
- ime Java izvorne datoteke ne poklapa se sa imenom Java klase (case sensitive; beline nisu dozvoljene u imenima)
- **. , ;** su bitni karakteri i moraju biti na svojim mestima
- ( ) { } i [ ] uvek dolaze u paru i ne prepliću se ...

4 Osnovni pojmovi objektno-orijentisanog programiranja

Kada se koristi programski jezik koji nije OO, rešenja se izražavaju u terminima brojeva i karaktera.

Kod OO programskih jezika na brojeve i karaktere se gleda kao na primitivne tipove podataka, a problemi se rešavaju u terminima entiteta, tj. objekata.

Objekat je integralna celina podataka i procedura za rad sa njima.
Primeri: - organizacija voznog parka u firmi. Objekti mogu biti: prevozno sredstvo (generalno posmatrano), ali i autobus, automobil, motocikl...
- geometrijski objekti: pravougaonik, kvadrat, krug...

Objektno orijentisano programiranje je programska paradigma zasnovana na skupu objekata koji dejstvuju međusobno. Glavne obrade zasnivaju se na rukovanju objektima.

Klasa je opis (specifikacija) jedne vrste objekata. Ime Java klase počinje velikim slovom. Primeri klasa: *PrevoznoSredstvo, Pravougaonik*, *Krug* itd. Definicija svake Java klase nalazi se u posebnoj datoteci koja ima isto ime kao i klasa a ekstenziju **.java**.

Instanca klase je konkretan objekat klase.

Instancne promenljive klase su podaci koji opisuju svojstva objekata te klase. Mogu biti promenljive primitivnog tipa ili objekti neke druge klase:

- za automobil: boja, proizvođač, model, ...
- za pravougaonik: dužine stranica (a, b)
- za krug: koordinate centra (x, y) i poluprečnik
- …

Primer:

```
class Krug {                    ← klasa Krug
  int x, y;
  double r;
}
Krug k = new Krug(5, 10, 8);      ← instanca klase Krug
```

Instancne promenljive klase *Krug* su *x*, *y* i *r*, koje u instanci *k* imaju redom vrednosti 5, 10 i 8.

Pored svojstava objekata (instancnih promenljivih), klasa opisuje i šta sve može da se radi sa objektima te klase, tj. definiše moguće operacije nad objektima klase. To se postiže blokom programskog koda koji se naziva **metod.** Sintaksno gledano, **metod** je zapravo funkcija koja je deo objekta i kojom se realizuje poruka upućena objektu.

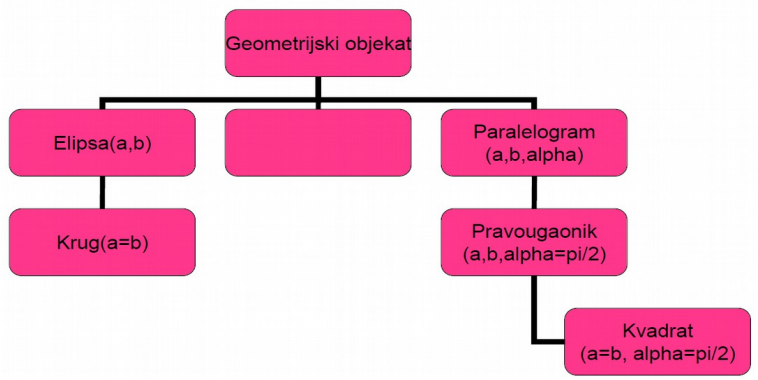
Primeri: Startuj motor, zaustavi auto, ubrzaj, …; izračunaj obim, površinu,...

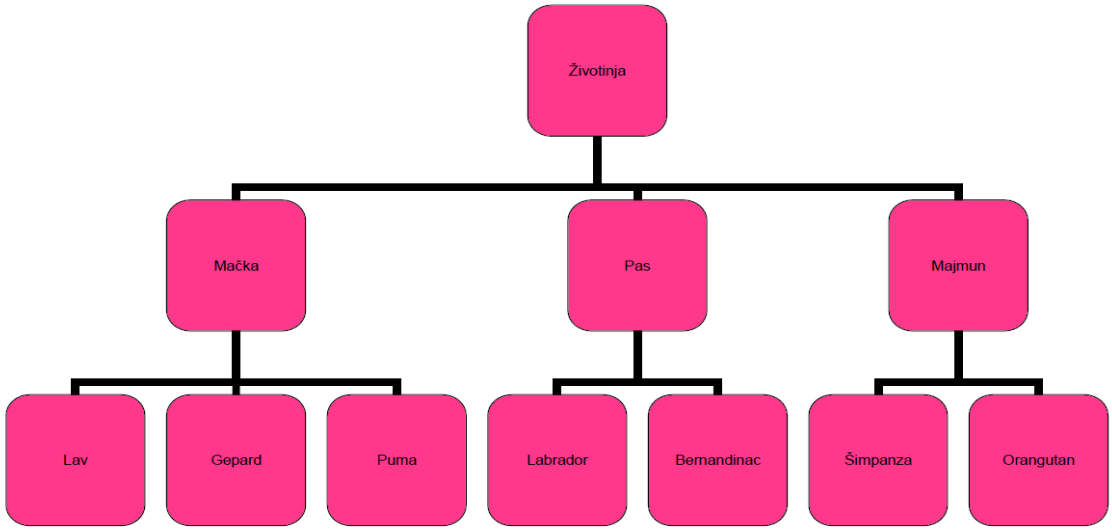
Dodavanjem novih svojstava postojećoj klasi definiše se **potklasa**. Sve instance (objekti) potklase su istovrmemeno i objekti **natklase**.
Primer: klasa A*utomobil* ima sve karakteristike klase *PrevoznoSredstvo*, ali ima i svoje dodatne karakteristike (npr. 4 točka).

Nasleđivanje

Prilikom projektovanja programa uočavaju se veze između pojedinih klasa, gde je bitna uloga nasleđivanja.
Nasleđivanje je postupak definisanja novih klasa na osnovu postojećih. Na taj način se uspostavljaju relacije između jedne i više drugih klasa.

Primeri nasleđivanja





5 Struktura Java programa

Java program se uvek sastoji iz jedne ili više klasa.
U Javi imamo standardnu klasu ***java.lang.Object*** koja je natklasa svih klasa.
Uobičajeno je da se definicija jedne klase nalazi u posebnoj datoteci koja ima *isto ime* kao i klasa.
Java izvorna datoteka uvek ima ekstenziju **.java**.

Klase se grupišu u skupove – *pakete*. Svaki paket čuva se u zasebnom direktorijumu.
Biblioteka klasa u Javi je kolekcija već postojećih klasa.
Korisnik sam može da pravi svoje pakete koji će sadržati odgovarajuće klase.

Java raspolaže velikim brojem standardnih paketa klasa. Najbitniji *standardni paketi*:

- *java.lang* – osnovne karakteristike jezika, rad sa nizovima i stringovima; klase iz ovog paketa su uvek dostupne, tj. automatski se uključuju u program (npr. klase Integer, String, Character, Math, ...)
- *java.io* – klase za ulazno/izlazne operacije
- *java.util* – veliki broj korisnih klasa za razne namene (Scanner, ArrayList (uopšteni niz), Vector, HashMap, ...)
- *java.awt, java.swing* – paketi za rad sa grafičkim komponentama koji su ranije korišćeni
- *javafx* – osnovni paket (koji sadrži druge potpakete) za rad sa *JavaFX* grafičkim komponentama
- *...*

Upotreba Java paketa i klasa

Upotreba klasa iz nekog paketa omogućena je ***import*** deklaracijom koja se navodi na početku programa:

```
import java.util.ArrayList; //ukljucena je samo klasa ArrayList paketa java.util
import java.util.*; //ukljucene su sve klase paketa java.util
```

Može se koristiti klasa iz nekog paketa i bez import deklaracije, ali se mora navesti njeno puno ime u programu:

```
java.util.ArrayList a;
```

Gde je biblioteka?

Standardne Java klase su spakovane u jednu kompajliranu datoteku **rt.jar** (sadrži samo .class datoteke) koja se nalazi u **jre/lib** poddirektorijumu koji se kreira prilikom instalacije JDK-a.

Standardne JavaFX klase spakovane su u datoteku **jfxrt.jar** koja se nalazi u poddirektorijumu **jre/lib/ext**.

Metod main()

Svaka Java aplikacija sadrži klasu koja definiše metod *main()*. Njegovim izvršavanjem pokreće se Java aplikacija (nakon što se ime ove klase prosledi kao argument Java interpreteru).

Metod *main* ima predefinisanu formu:
public static void main(String[] args) { … }
da bi bio prepoznat od strane Java interpretera kao metod odakle počinje izvršavanje programa.

Primer Zdravo.java

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

public – globalno dostupna klasa i metod
static – metod je dostupan i kada ne postoje objekti klase
void – metod nema povratnu vrednost
String[] args – niz argumenata komandne linije. Argumenti su objekti standardne klase *String,* kojom se opisuju niske karaktera. Klasa *String* je definisana u paketu *java.lang*.
System – standardna klasa koja omogućava rad sa standardnim ulazom/izlazom. Nalazi se u paketu *java.lang*.
out – objekat (klase *PrintSteam)* koji je član klase *System* i predstavlja standardni izlazni tok.
println() – metod koji pripada objektu *out (*metod klase *PrintStream)* koji ispisuje nisku karaktera zadatu između dvostrukih navodnika.

Poziv metoda neke klase vrši se u formi: ***ime_objekta.ime_metoda***.

6 Karakterski skup u Javi

Java je bazirana na UNICODE karakterskom skupu. Svaki karakter predstavlja se pomoću 2 bajta (16 bitova). Java je CASE-SENSITIVE (pravi se razlika između malih i velikih slova).

7 Tipovi podataka u Javi

Postoje dve vrste tipova podataka:
- primitivni tip
- referentni (objektni) tip podataka

Primitivni tipovi podataka

Celobrojni tipovi: *byte(8), short(16), int(32) i long(64)* (svi su označeni).
Realni (u pokretnom zarezu): *float(32), double(64)* (standard IEEE 754).
Logički tip: *boolean(1) (*ima dve vrednosti: *true* i *false)*.
Znakovni tip: *char(16)* (neoznačen).
U Javi ne postoji ključna reč *unsigned*.

Referentni tip podataka

Tip objekta je određen klasom. Promenljiva klasnog tipa uvek sadrži referencu na objekat, ne i sam objekat. U Javi, referenca na objekat ne mora da bude isto što i adresa objekta u memoriji. Referenca je jednostavno vrednost na osnovu koje Java VM može da pristupi objektu.
Primer:

```
String s = "Zdravo svima". // promenljiva s sadrzi referencu na objekat klase
                           // String ciji je sadrzaj niska "Zdravo svima"
```

Drugim računima, *svim objektima se pristupa preko reference*.

8 Elementarne konstrukcije jezika Java

Elemente jezika koje prevodilac izdvaja tokom prevođenja programa kao nedeljive celine (tokene) čine:

- *identifikatori, literali, separatori, operatori, ključne reči, komentari*

Identifikatori

Identifikator je niz UNICODE karaktera proizvoljne dužine koji mora počinjati *slovom, _ ili $.* Ostali karakteri mogu biti *slova, cifre, _ ili $.* Ne smeju se koristiti ključne reči.

Literali

Eksplicitne vrednosti koje se javljaju u programu su *literali*.
Literal može biti konstantna vrednost primitivnog tipa ili objekat klase *String* (niska karaktera).

Celobrojni literali

Celobrojni literali:

- Dekadni – ne počinju cifrom 0
- Oktalni – počinju cifrom 0
- Heksadekadni - počinju cifrom 0 za kojom sledi slovo 'x' (0x) ili 'X' (0X)

Literal tipa long ima sufiks l ili L.

Realni, logički, znakovni i String literali:

Realni literali su podrazumevano tipa *double*. Može i da stoji sufiks D ili d.
Realni literal je tipa float ako sadrži sufiks F ili f (1.423, 1.423F, 2.54E8, 9E-28F,...).
Logički literali: true i false.
Znakovni literal - konstantna vrednost tipa char, tj. neoznačena 16-bitna vrednost.
Primeri: ‘a’, ‘\n’, ‘3’, ‘\\’
'\u0041' – heksadekadni kod karaktera iz UNICODE skupa (ovde, velikog slova A)
String literal – niska karaktera zadata između dvostrukih navodnika.

Separatori, operatori, komentari

Separatori: () {} [] ;
Nema operatora , kao u C-u.
Nema operatora za pristup memoriji, što znači da u Javi *nema pokazivača.*
Komentari:
C stil /*... */
Jednolinijski komentari //...
Dokumentacioni komentari

```
/**
* ...
*/
```

Podaci, promenljive, izračunavanja

Ključna reč *final* označava da se vrednost promenljive ne može menjati. Konvencija je da se konstante pišu velikim slovima.

Kastovanje – primer:

```
int a = 3, b = 5;
double c = 1.5 + b/a;
double d = 1.5 + (double)b/a;
```

Ne postoji automatsko kastovanje *boolean* tipa u celobrojni tip i obrnuto! Ako je *a* celobrojna promenljiva, ne može se pisati *if(a)*, već mora *if(a != 0)*.

Promenljiva se može uvesti bilo gde u bloku. Oblast važenja je od tog mesta do kraja bloka.

Naredbe za kontrolu toka su iste kao u prog. jeziku C (if, else, switch, for, while, do, break, continue).
Nema naredbe goto.
U Javi nema operatora sizeof, kao ni ključnih reči *typedef, struct, union, extern.*

Pravljenje objekta neke klase vrši se operatorom *new*.

9 Kratak pregled rada sa stringovima (niskama karaktera)

Postoje dve standardne klase: *String* (za konstantne stringove) i *StringBuffer* (za promenljive stringove).

Može se koristiti i deklaracija pomoću *char[]*, ali ne postoji automatska konverzija između niza i stringa. *Primer:*

```
char ime[] = {'a', 'l', 'e', 'k', 's', 'a', 'n', 'd', 'a', 'r'};  // ispravno
char ime[] = "aleksandar";  // ne sme
String ime =  "aleksandar"; // ispravno
```

Metodi

- *length()* - dužina stringa
- *charAt(int)* – pristup elementu na datoj poziciji
- *operator +* se koristi za spajanje (konkatenaciju) stringova
  - postoji automatska konverzija primitivnog tipa u tip String (npr. *int* → *String*)

```
String s  = "Rezultat je: " + rez;   // rez je promenljiva tipa int
```

Argumenti komandne linije

public static void main(String[] args)

↑ args je niz stringova, *args.length* je dužina niza

Argumenti komandne linije: *args[0], args[1], args[2], …*
Zadavanje argumenata komandne linije u konzolnom pokretanju programa:
C:\> java Zdravo a b c (navode se posle imena Java klase, ovde: a, b, c)

10 Klasa java.util.Scanner

Klasa kojom se implementira jednostavni čitač (skener) koji izdvaja (parsira) primitivne tipove i stringove koristeći regularne izraze.
Ulaz se deli na tokene, a kao delimiter se podrazumevano koriste beline (' ', '\n', '\t', …).
Dobijeni tokeni se mogu konvertovati u vrednosti različitih tipova koristeći razne *next...* metode:

Metodi next*() i hasNext*()

next()* i *hasNext*()* metodi postoje za većinu primitivnih tipova (int, double, long, boolean,...).
Metodi *next*()* preskaču karaktere sa ulaza koji odgovaraju delimiterima i pokušavaju da vrate naredni token koji odgovara tipu.

```
public boolean hasNext();
```
vraća true ako postoji naredni token na ulazu

```
public String next();
```
pronalazi i vraća ceo naredni token; ”ceo” token je okružen delimiterima (i ispred i iza njega su delimiteri)

```
public boolean hasNextLine();
```
vraća true ako postoji naredna linija na ulazu

```
public String nextLine();
```
vraća ostatak tekuće linije isključujući oznaku za kraj reda sa njenog kraja

Skup metoda *next*()* i *hasNext*()* gde se umesto * nalazi (*Int*, *Double*, *Float*, *Short*, *Long*, *Byte*, *Boolean*) - jasno je na osnovu imena šta rade.

Dokumentacija
https://docs.oracle.com/javase/8/docs/api/java/util/Scanner.html

Primeri:
Čitanje celog broja sa standardnog ulaza (*System.in*):
```
Scanner sc = new Scanner(System.in);
int ceo = sc.nextInt();
```

Može se čitati i iz stringa:

```
Scanner sc = new Scanner("1 8 java jdk");
System.out.println(sc.nextInt()); // 1
System.out.println(sc.nextInt()); // 8
System.out.println(sc.next()); // java
System.out.println(sc.next()); // jdk
```

Osim belina, mogu se koristiti i drugi delimiteri koji se postavljaju metodom *useDelimiter():*

```
Scanner sc = new Scanner("1, 2, 3, 4, 5, 6, 7, 8, 9, 10").useDelimiter(",");
while(sc.hasNextInt()) {
  int broj = sc.nextInt();
  if (broj % 2 == 0) System.out.println(broj);
}
```

Neodgovarajući token
Kada neki od metoda *next*()* izbaci izuzetak tipa *InputMismatchException*, ne prosleđuje se token koji je izazvao izuzetak. Token se može dobiti ili preskočiti primenom nekog drugog metoda.
Napomena: Izbacivanje izuzetka tipa *IOException* predstavlja kraj ulaza.

Metodom close() zatvara se skener. Kada se skener zatvori, obično se zatvara i ulazni tok.

Zadaci za vežbu

- Napisati program koji izračunava obim i površinu kruga datog poluprečnika r = 12.45 i ispisuje dobijene vrednosti na standardni izlaz.

- Napisati program koji sabira prirodne brojeve od 1 do n.

- Napisati program koji učitava dva broja sa standardnog ulaza i ispisuje njihov zbir na standardni izlaz.

11 Eclipse razvojno okruženje

Java programi se mogu pokrenuti kako iz komandne linije, tako i pomoću velikog broja Java razvojnih alata, kao što su:
- Borland JBuilder, Net Beans, Sun ONE Studio (komercijalna verzija Net Beans-a)
- Eclipse
- Web Gain Visual Cafe i drugi

Oni podržavaju tzv. *IDE (Integrated Development Environment)* za brz razvoj Java programa. Editovanje, kompajliranje, debagovanje, online pomoć su integrisani u grafičko korisničko okruženje ovih alata.

Eclipse tutorial

Eclipse može da se pokreće na bilo kojoj platformi koja ima podršku za JVM.

Neophodno je najpre instalirati odgoravajuću verziju JDK-a (Java Development Kit).

Eclipse se ne instalira, već se samo raspakuje odgovarajuća datoteka. Pokreće se dvoklikom na ikonu *eclipse.exe* u odgovarajućem direktorijumu ili dvoklikom na odgovarajuću prečicu na radnoj površini.
Najpre se pojavljuje prozor *Workspace Launcher* koji nudi izbor direktorijuma u kome će se raditi. Potom se otvara glavni prozor sa inicijalno otvorenim pozdravnim prozorom.

Izbor perspective

Perspektiva predstavlja inicijalni skup i raspored komponenti u prozoru. Perspektive kontrolišu šta se pojavljuje u određenim menijima i linijama sa alatkama (toolbars).

Java perspektiva se koristi za editovanje Java izvornog koda.
Debug perspektiva omogućava pregled debagovanja programa.
Moguće je prelaziti iz jedne u drugu perspektivu.

Za pisanje Java programa opcijom
Window → Open Perspective → Java iz glavnog menija otvara se Java perspektiva.
U gornjem desnom uglu (u liniji sa alatkama) nalazi se ikona koja označava izabranu perspektivu. Klikom na ikonu *Open Perspective* (levo od nje) otvara se prozor gde se iz ponuđene liste može izabrati druga perspektiva (npr. Debug, za debagovanje programa).

Pravljenje projekta

Projekat se pravi izborom opcije *File → New → Java Project* ili klikom na ikonu *New* u okviru linije sa alatkama i izborom opcije *Java Project*, čime se prikazuje prozor *New Java Project.*
U polje *Project name* uneti ime projekta.
U delu JRE iz padajućeg menija izabrati odgovarajuću verziju JDK-a. Podrazumevano je izabrana JavaSE-1.8.
U delu Project Layout izabrati opciju *Use project folder as root for sources and class files.*
Klikom na dugme *Finish* projekat biva napravljen.

Pravljenje paketa i klasa

Pravljenje paketa u postojećem projektu: *File → New → Package* ili selektovanje projekta i klik na ikonu *New Java Package* iz linije sa alatkama.
Pravljenje klase u postojećem paketu: *File → New → Class*, čime se otvara *New Java Class* prozor (ili selektovanje paketa i klik na ikonu *New Java Class* iz linije sa altkama).
U polje *Name* unosi se ime klase (počinje velikim slovom) bez ekstenzije .java.
U delu *Modifiers* moguće je čekirati neku od opcija – svaka predstavlja neku vrstu modifikatora klase.
U polju *Superclass* navodi se ime klase koja se nasleđuje (ostavlja se *java.lang.Object* ukoliko klasa ne nasleđuje neku drugu klasu).
U polju *Interfaces* navode se imena interfejsa koje klasa implementira (klasa u Javi može da implementira VIŠE interfejsa, ali da nasledi može samo JEDNU klasu. O nasleđivanju klasa i implementiranju interfejsa biće reči kasnije).
Poslednji deo se odnosi na izbor metoda koji su sastavni deo klase:
- public static void main(String[] args)
- konstruktori natklase
- nasleđeni apstraktni metodi

Prevođenje i pokretanje programa

Izvorni Java kod se **dinamički kompajlira** za vreme editovanja.
Na primer, ako se ne otkuca ‘; ‘ za kraj naredbe, prikazuje se crvena krivudava linija koja ukazuje na grešku.
Prilikom kucanja koda, kada se, na primer, otkuca tačka (”.”), nakon kratke pauze prikazuje se pomoćni (padajući) meni sa mogućnostima za kompletiranje koda.

Izvršavanje programa: *Run → Run As → Java Application* iz glavnog menija ili kliknuti na odgovarajuću ikonu *Run* iz linije sa alatkama.

Izlaz je prikazan u panelu *Console* u dnu prozora.
Ukoliko konzolni panel nije vidljiv, iz menija *Window* izabrati *Show View → Console* (skraćenica Shift+Alt+Q C).