

Naredbe

Naredbe

- Izraz, izraz-naredba, blok
- Prazna naredba
- Naredbe grananja: `if`, `if-else`, `switch`
 - Omogućuju grananje izvršavanja, odnosno izbor dela koda koji će da se izvrši
- Naredbe ponavljanja (petlje): `while`, `do-while`, `for`
 - Omogućuju da se deo koda izvrši više puta
- Naredbe skoka: `break`, `continue`, `return`
 - Omogućuju prekid izvršavanja koda i nastavak izvršavanja na drugom mestu

Izraz

- Izraz u Javi je:
 - Literal
 - Lokalna promenljiva
 - Element niza*
 - Poziv metoda*
 - Pristup polju klase ili objekta*
 - Kreiranje novog objekta pomoću `new` operatora*
 - Primena ključne reči `this`*
 - Izraz u običnim zagradama
 - Primena nekog od operatora (osim operatora `new`) na operande koji su izrazi odgovarajućeg tipa
 - Pristup objektu tipa `Class` kojim se predstavlja neki tip**

*: radićemo kasnije **: nećemo raditi na ovom kursu

Izraz-naredba

- Neke vrste izraza u Javi mogu da se nalaze u programu na mestu gde se očekuje naredba (npr. kao jedna od naredbi u telu metoda `main`)
- Tako korišćene izraze nazivamo izraz-naredbe
- Izraz-naredbe u Javi su:
 - Naredba dodele
 - Primena operatora dodele: osnovni `=`, i sa prethodnom primenom drugih operatora `+=`, `*=`, ...
 - Primena operatora `++` i `--` prefiksno i postfiksno
 - Poziv metoda
 - Kreiranje novog objekta pomoću `new` operatora
- Delovi izraz-naredbi (npr. deo posle operatora dodele) mogu biti i proizvoljni izrazi, ali osnova mora biti neka od navedenih vrsta izraza
- Izračunata vrednost izraza koji se koristi kao naredba se gubi – izraz se izračunava zbog efekta koji izračunavanje ima na promenljive, itd.

Izraz-naredba

■ Primeri:

```
x = 1; // naredba dodele
++i; // prefiksna inkrementacija promenljive i
--a[3]; // prefiksna dekrementacija elementa niza a
a /= b %= c -= d *= 5; // a = a / (b = b % (c = c - (d = d*5)))
// polju ime tekućeg objekta dodeljuje rezultat poziva metoda
this.ime = spisak.uzmiImeSaRednimBrojem(slucajniBrojevi.getInt(10));
kola.rezervoar.kapacitet++; // sufiksna inkrementacija polja objekta
tacka.x--; // sufiksna dekrementacija polja objekta
figura.iscrtajSe(); // poziv metoda iscrtajSe() objekta figura
new MojaKlasa(1, 2, 3); // kreira objekat
new Thread(mojaNit).start(); // kreira objekat i poziva metod
```

Prazna naredba

- Ne proizvodi nikakvu akciju niti efekat
- Oblik:
;
- U nekim situacijama može biti korisna kod složenih naredbi (`for`, `while`...) kada moramo da navedemo naredbu, a ne želimo
- Primere ispravnog korišćenja navešćemo kasnije, ali je u principu treba izbegavati zbog mogućnosti pravljenja suptilnih grešaka

Prazna naredba

- Primer:

```
class EmptyBad {  
    public static void main(String[] args) {  
        int i;  
        for (i = 0; i < 5; i++);  
        System.out.println("i = " + i);  
    }  
}
```

- Izlaz:

i = 5

Prazna naredba

- Primer:

```
class EmptyBadFixed {  
    public static void main(String[] args) {  
        int i;  
        for (i = 0; i < 5; i++)  
            System.out.println("i = " + i);  
    }  
}
```

- Izlaz:

```
i = 0  
i = 1  
i = 2  
i = 3  
i = 4
```


Blok

- Blok se sastoji od sekvence naredbi i deklaracija promenljivih (i deklaracija lokalnih klasa) navedenih između vitičastih zagrada { }
- Blok se u programu može naći na bilo kom mestu gde se očekuje (jedna) naredba (i otuda oblici bez i sa { } naredbi `if`, `for`, itd.)
- Dve osnovne namene bloka:
 - Grupisanje naredbi
 - Ograničenje oblasti važenja lokalnih promenljivih

Blok: deklaracija lokalne promenljive

- U Javi, svaka promenljiva se pre korišćenja mora deklarirati
- Lokalna promenljiva se deklarira u okviru nekog bloka
- Takve promenljive smo do sad i koristili:

```
int br;                                double x;  
String s;                             boolean nasao;
```

- Moguće je promenljivama dodeliti početnu vrednost prilikom deklaracije, gde se sa desne strane operatora = može nalaziti izraz:

```
int br = 45 - 3;                       double x = 5.5 + 2.7;  
String s = "Neki tekst";              boolean nasao = true;
```

- A moguće je i deklarirati/inicijalizovati više promenljivih odjednom:

```
int br, n, i, j;  
double x = 1.0, y = 2.0, z = 3.0;
```

- Lokalne promenljive se moraju inicijalizovati pre prvog korišćenja njihovih vrednosti

Blok: deklaracija lokalne promenljive

- Pre deklaracije promenljive može se staviti ključna reč `final`
- Takva promenljiva se može inicijalizovati samo jednom (ne obavezno pri deklaraciji, mada je preporučljivo)
- Tako se u stvari dobija **konstanta**, odnosno promenljiva čija se vrednost ne menja
- Primeri:

```
final double PI = 3.14;  
final double DVA_PI = 2 * PI;  
final int THE_ANSWER = 42;  
final String JA = "Taj i taj";
```

Blok: vidljivost lokalne promenljive

- Lokalna promenljiva (promenljiva deklarirana u nekom bloku) se može koristiti (vidljiva je) od mesta na kojem se deklarira, uključujući i deo eventualnog odeljka za inicijalizaciju, pa sve do kraja bloka
- Primer:

```
class Vazenje {  
    public static void main(String[] args) {  
        int x = 10; // vidljiva u celom main metodu  
        if (x == 10) { // nova oblast  
            int y = 20; // vidljiva u ovoj unutrašnjoj oblasti  
            System.out.println("x i y: " + x + " " + y);  
            // x i y su vidljivi u unutrašnjem bloku  
            x = y * 2;  
            // double x = 5.5; // ovo bi bila greska, x vec postoji  
        }  
        // y = 100; // ovo bi bila greska, y vise nije vidljivo  
        System.out.println("x = " + x); // x se vidi u ovom bloku  
    }  
}
```

Naredbe grananja

- Kontrola toka programa se grana u zavisnosti od nekih uslova, tj. rezultata izračunavanja nekog izraza
- Naredbe grananja u Javi:
 - `if` naredba
 - `if-else` naredba
 - `switch` naredba

Naredbe grananja: **if**

- Specificira da će se data naredba (ili blok naredbi) izvršiti ako i samo ako je vrednost datog logičkog izraza jednaka `true`
- Efekat je da tok izvršavanja programa može da “krene” nekom “granom” ili ne
- Oblici:

```
if (izraz)
    naredba;
```

```
if (izraz) {
    naredba;
    ...
    naredba;
}
```

Naredbe grananja: `if`

- Primeri:

```
if (x == 2)
    System.out.println("x je dva");
```

```
if ('a' <= c && c <= 'z') {
    System.out.println("malo slovo");
    brSlova++;
}
```

Naredbe grananja: **if-else**

- Specificira koja će se od dve date naredbe (ili blokova naredbi) izvršiti u zavisnosti od vrednosti datog logičkog izraza
- Efekat je da tok izvršavanja programa “bira” jednu od ponuđenih grana na osnovu logičkog izraza
- Oblici:

```
if (izraz)          if (izraz) {
    naredba;         naredba;
else                ...
    naredba;         naredba;
                    }
                    else {
                        naredba;
                        ...
                        naredba;
                    }
                    }
```

- Moguće je i kombinovati pojedinačnu naredbu u jednoj grani i blok u drugoj

Naredbe grananja: if-else

- Primeri:

```
if (x < 0)
    absX = -x;
else
    absX = x;
```

```
if (i > 10 || i < -10) {
    System.out.println("van opsega");
    i = 0;
    j++;
}
else {
    System.out.println("unutar opsega");
    j--;
}
```

Naredbe grananja: preporuke

- Naredbe navedene u granama treba uvući u odnosu na samu `if` / `if-else` naredbu (2, 4 znaka razmaka, tabulator... važna je konzistentnost)
- Da bi se izbegla konfuzija, preporučljivo je uvek koristiti blokove (tj. navoditi naredbe u granama u vitičastim zagradama `{ }`), čak i kad u grani želimo da navedemo samo jednu naredbu

- Primer:

```
if (x < 0)
    absX = -x;
else
    absX = x;
```

```
if (x < 0) {
    absX = -x;
}
else {
    absX = x;
}
```

Naredbe grananja: ugneždavanje

- *Jedna od naredbi u prvoj grani* `if (-else)` naredbe može opet da bude `if (-else)` naredba, čime se postiže ugneždavanje

- Primer:

```
if (delilac != 0) {  
    if (deljenik > 1000) {  
        System.out.println("Deljenik je van opsega");  
    }  
}  
else {  
    System.out.println("Delilac ne sme biti 0");  
}
```

Naredbe grananja: if-else-if

- *Jedina* naredba u drugoj grani if-else naredbe može da bude nova if(-else) naredba, čime se na koncizan način postiže kaskadno vezivanje if-else naredbi (lestvica)

- **Primer:**

```
if (bodovi >= 85) {  
    System.out.println("Odlicno!!!");  
}  
else if (bodovi >= 75) {  
    System.out.println("Vrlo dobro!");  
}  
else if (bodovi >= 65) {  
    System.out.println("Dobro.");  
}  
else if (bodovi >= 55) {  
    System.out.println("Dovoljno.");  
}  
else {  
    System.out.println("Zao mi je...");  
}
```

Naredbe grananja: `switch`

- Specificira koje će se naredbe izvršiti na osnovu vrednosti celobrojnog izraza
- Efekat je da tok izvršavanja programa direktno “skoči” na deo koda koji odgovara zadatoj vrednosti izraza
- Osnovni oblik:

```
switch (izraz) {  
    case vrednost1:  
        naredba;  
        ...  
        naredba;  
        break;  
    ...  
    case vrednostN:  
        naredba;  
        ...  
        naredba;  
        break;  
}
```
- Vrednosti se navode u okviru “slučajeva”, odnosno labela (ključna reč `case`) kao konstantni izrazi, tj. izrazi koje kompajler može da izračuna u toku prevođenja programa
- Naredba `break` označava izlazak iz `switch` naredbe, odnosno skok na prvu naredbu posle }
- Nakon poslednjeg niza naredbi `break` nije neophodan
- Ako izraz nije jednak ni jednoj od zadatih vrednosti , neće se izvršiti ni jedna od datih naredbi

Naredbe grananja: `switch`

- Oblik sa `default` granom:

```
switch (izraz) {  
    case vrednost1:  
        naredba;  
        ...  
        naredba;  
        break;  
    ...  
    case vrednostN:  
        naredba;  
        ...  
        naredba;  
        break;  
    default:  
        naredba;  
        ...  
        naredba;  
}
```

- Ako izraz nije jednak ni jednoj od zadatih vrednosti, izvršiće se naredbe iz `default` grane

Naredbe grananja: switch

- Primer switch naredbe i if naredbe sa istim efektom:

```
switch (ocena) {  
    case 5:  
        System.out.println("Odlican");  
        break;  
    case 4:  
        System.out.println("Vrlo dobar");  
        break;  
    case 3:  
        System.out.println("Dobar");  
        break;  
    case 2:  
        System.out.println("Dovoljan");  
        break;  
    default:  
        System.out.println("Nedovoljan");  
}
```

```
if (ocena == 5) {  
    System.out.println("Odlican");  
}  
else if (ocena == 4) {  
    System.out.println("Vrlo dobar");  
}  
else if (ocena == 3) {  
    System.out.println("Dobar");  
}  
else if (ocena == 2) {  
    System.out.println("Dovoljan");  
}  
else {  
    System.out.println("Nedovoljan");  
}
```

Naredbe grananja: `switch`

- Glavni izraz `switch` naredbe mora biti tipa `char`, `byte`, `short` ili `int`
 - Glavni izraz takođe može biti nabrojivog tipa i tipa `String` (o čemu će više reći biti kasnije)
- U pravljenju konstantnih izraza moguće je primeniti samo sledeće operatore:
 - eksplicitne konverzije u tip `String` ili u neki prost tip
 - unarne operatore `+`, `-`, `~` i `!`
 - binarne operatore `*`, `/`, `%`, `+`, `-`, `<<`, `>>`, `>>>`, `<`, `<=`, `>`, `>=`, `==`, `!=`, `&`, `^`, `|`, `&&` i `||`
 - ternarni operator `?:`
- `switch` naredba je efikasnija od niza ugnježđenih `if` naredbi

Naredbe grananja: `switch`

- Nije obavezno da se niz naredbi uz svaku labelu završi naredbom `break`
 - Efekat je da izvršavanje naredbi “propadne” u sledeći slučaj
 - Nenamerno izostavljanje `break` naredbe je česta greška
- Niz naredbi uz labelu može da bude prazan
 - Omogućava da se jedna ista grupa naredbi izvrši u različitim slučajevima bez dupliranja koda

Naredbe grananja: switch

- Primer switch naredbe sa pogrešno izostavljenim break naredbama:

```
class SwitchNoBreak {  
    public static void main(String[] args) {  
        System.out.print("Unesite ceo broj: ");  
        int broj = Svetovid.in.readInt();  
        switch (broj) {  
            case 0:  
                System.out.println("nula");  
            case 1:  
                System.out.println("jedan");  
            case 2:  
                System.out.println("dva");  
            case 3:  
                System.out.println("tri");  
            default:  
                System.out.println("manji od nula ili veci od tri");  
        }  
    }  
}
```

} UUP: Naredbe

Naredbe grananja: switch

- Izlaz:

```
Administrator: C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

d:\PMF\Nastava\UUP\UUP2014\Predavanja\04>javac SwitchNoBreak.java

d:\PMF\Nastava\UUP\UUP2014\Predavanja\04>java SwitchNoBreak
Unesite ceo broj: 0
nula
jedan
dva
tri
manji od nula ili veci od tri

d:\PMF\Nastava\UUP\UUP2014\Predavanja\04>java SwitchNoBreak
Unesite ceo broj: 2
dva
tri
manji od nula ili veci od tri

d:\PMF\Nastava\UUP\UUP2014\Predavanja\04>java SwitchNoBreak
Unesite ceo broj: 77
manji od nula ili veci od tri

d:\PMF\Nastava\UUP\UUP2014\Predavanja\04>_
```

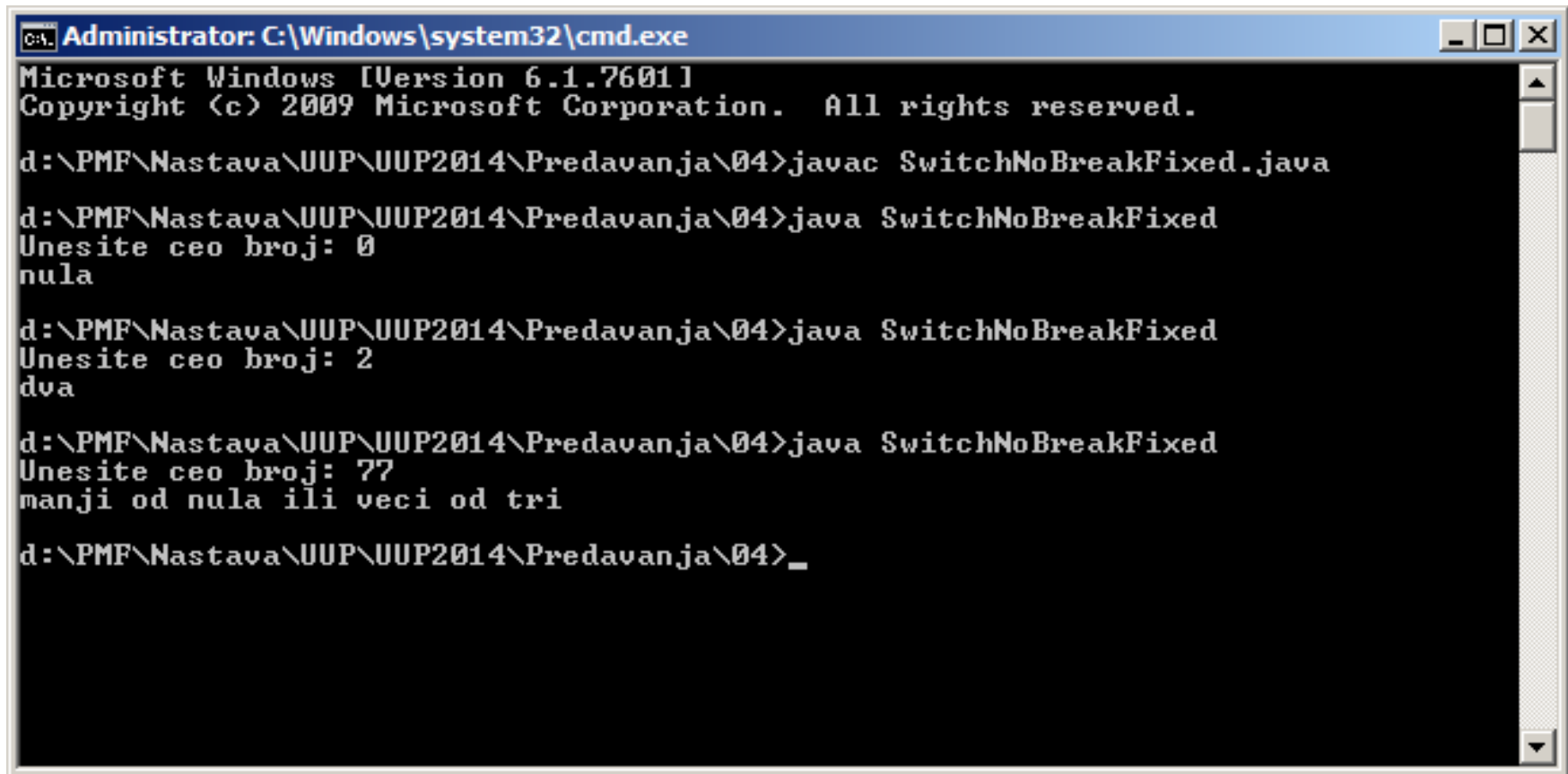
Naredbe grananja: switch

- Prethodni primer sa ispravno navedenim `break` naredbama:

```
class SwitchNoBreakFixed {  
    public static void main(String[] args) {  
        System.out.print("Unesite ceo broj: ");  
        int broj = Svetovid.in.readInt();  
        switch (broj) {  
            case 0:  
                System.out.println("nula");  
                break;  
            case 1:  
                System.out.println("jedan");  
                break;  
            case 2:  
                System.out.println("dva");  
                break;  
            case 3:  
                System.out.println("tri");  
                break;  
            default:  
                System.out.println("manji od nula ili veci od tri");  
        }  
    }  
}  
} UUP: Naredbe
```

Naredbe grananja: switch

- Izlaz:

A screenshot of a Windows command prompt window titled "Administrator: C:\Windows\system32\cmd.exe". The window shows the execution of a Java program named "SwitchNoBreakFixed.java". The program takes a command-line argument "Unesite ceo broj: 0" and outputs "nula". The user then enters "2" and "77", which result in "dva" and "manji od nula ili veci od tri" respectively. The prompt ends with a trailing underscore.

```
Administrator: C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

d:\PMF\Nastava\UUP\UUP2014\Predavanja\04>javac SwitchNoBreakFixed.java

d:\PMF\Nastava\UUP\UUP2014\Predavanja\04>java SwitchNoBreakFixed
Unesite ceo broj: 0
nula

d:\PMF\Nastava\UUP\UUP2014\Predavanja\04>java SwitchNoBreakFixed
Unesite ceo broj: 2
dva

d:\PMF\Nastava\UUP\UUP2014\Predavanja\04>java SwitchNoBreakFixed
Unesite ceo broj: 77
manji od nula ili veci od tri

d:\PMF\Nastava\UUP\UUP2014\Predavanja\04>_
```

Naredbe ponavljanja

- Omogućavaju ponavljanje izvršavanja date naredbe ili bloka naredbi
- Još se nazivaju i “petlje” (eng. *loop*)
- U Javi postoje tri naredbe ponavljanja:
 - `while` naredba
 - `do-while` naredba
 - `for` naredba

Naredbe ponavljanja: `while`

- Omogućava ponavljanje izvršavanja date naredbe ili bloka naredbi dokle god je vrednost datog logičkog izraza jednaka `true`
- Vrednost logičkog izraza se izračunava pre prvog ponavljanja, što znači da se date naredbe ne moraju izvršiti ni jednom
- Važno je da naredbe koje se ponavljaju (ili samo izračunavanje vrednosti logičkog izraza) utiču na sledeće izračunavanje logičkog izraza, u protivnom izvršavanje programa može da uđe u “beskonačnu petlju”
- Oblici:

```
while (izraz)
    naredba;
```

```
while (izraz) {
    naredba;
    ...
    naredba;
}
```

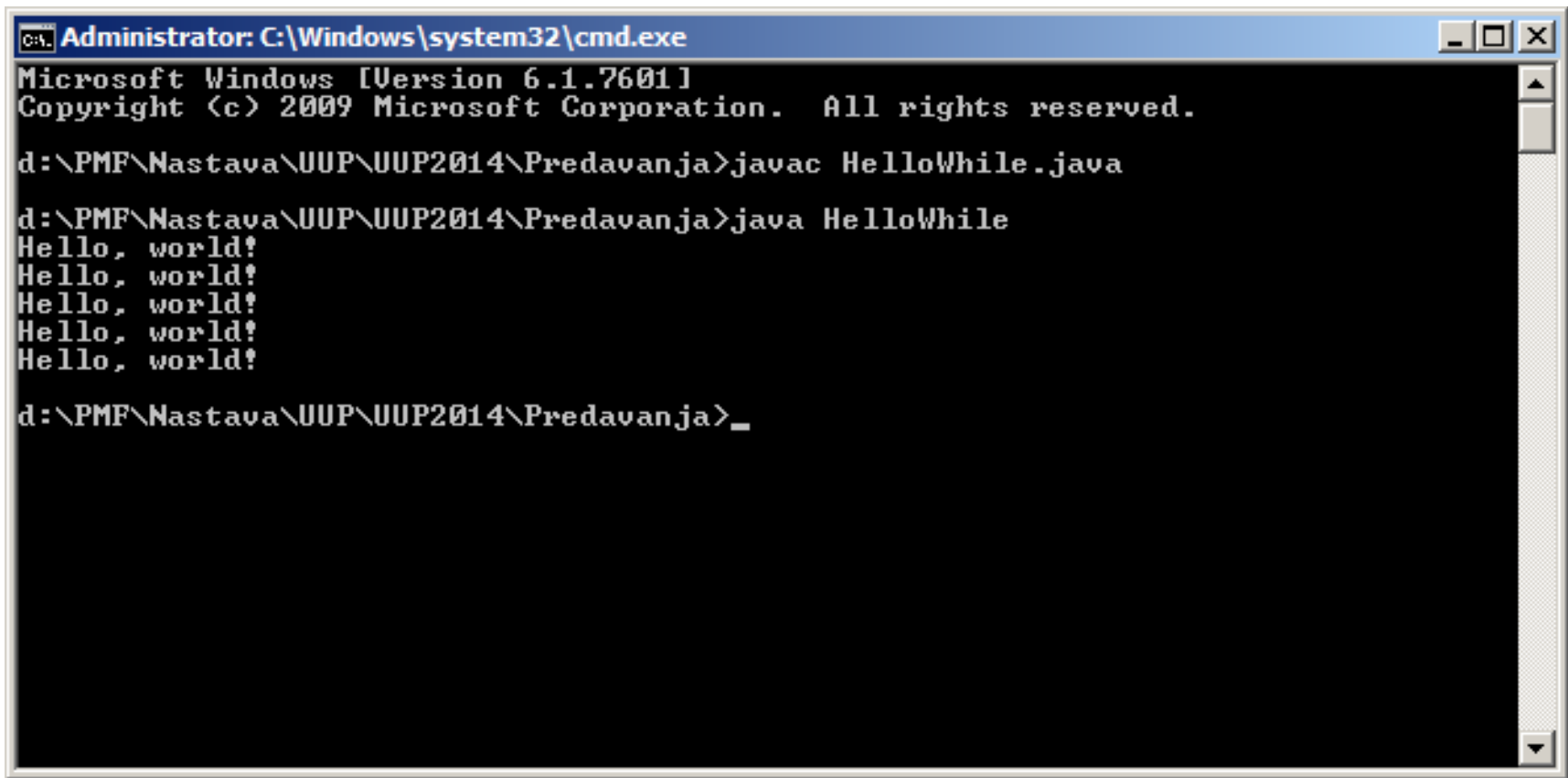
Naredbe ponavljanja: while

- Primer:

```
class HelloWorld {  
    public static void main(String[] arguments) {  
        int i = 1;  
        while (i <= 5) {  
            System.out.println("Hello, world!");  
            i = i + 1;  
        }  
    }  
}
```


Naredbe ponavljanja: while

- Izlaz:

A screenshot of a Windows command prompt window titled "Administrator: C:\Windows\system32\cmd.exe". The window shows the following text:

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

d:\PMF\Nastava\UUP\UUP2014\Predavanja>javac HelloWorld.java

d:\PMF\Nastava\UUP\UUP2014\Predavanja>java HelloWorld
Hello, world!
Hello, world!
Hello, world!
Hello, world!
Hello, world!

d:\PMF\Nastava\UUP\UUP2014\Predavanja>_
```

Naredbe ponavljanja: **while**

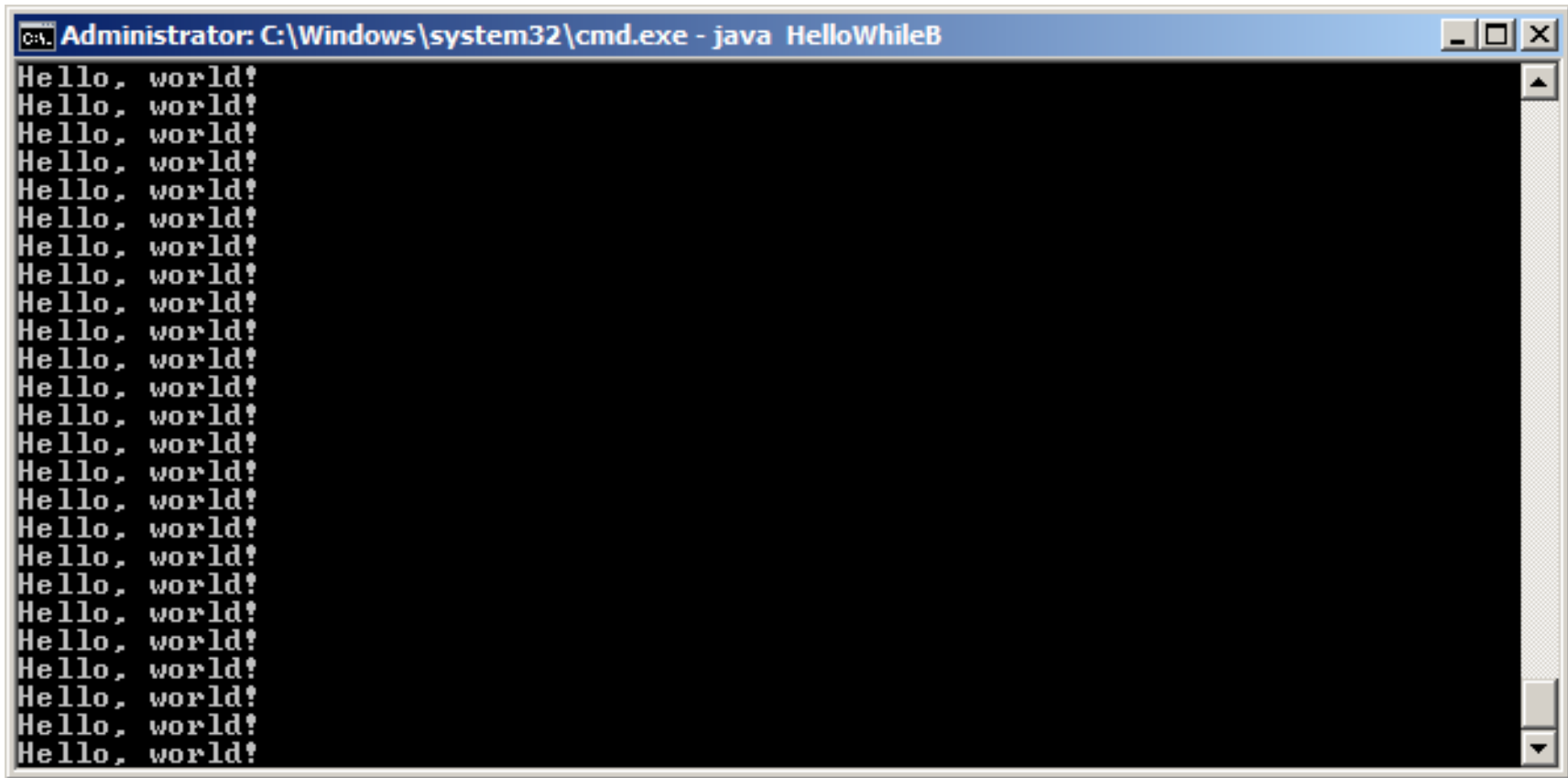
- Pitanje: kako bi izgledao izlaz sledećeg programa?

```
class HelloWorld {  
    public static void main(String[] arguments) {  
        int i = 1;  
        while (i <= 5) {  
            System.out.println("Hello, world!");  
        }  
    }  
}
```

Naredbe ponavljanja: `while`

- Izgledao bi ovako:

(Ctrl+C prekida izvršavanje)



```
Administrator: C:\Windows\system32\cmd.exe - java HelloWhileB
Hello, world!
Hello, world!
Hello, world!
Hello, world!
Hello, world!
Hello, world!
Hello, world!
Hello, world!
Hello, world!
Hello, world!
Hello, world!
Hello, world!
Hello, world!
Hello, world!
Hello, world!
Hello, world!
Hello, world!
Hello, world!
Hello, world!
Hello, world!
Hello, world!
Hello, world!
```

Naredbe ponavljanja: `while`

- Alternativni način da se inkrementira promenljiva `i`:

```
class HelloWorldFixed {  
    public static void main(String[] arguments) {  
        int i = 1;  
        while (i++ <= 5) {  
            System.out.println("Hello, world!");  
        }  
    }  
}
```

Naredbe ponavljanja: while

- Sav “posao” petlje može da se obavi prilikom izračunavanja logičkog izraza, pri čemu se kao telo petlje stavlja prazna naredba (;). Sledeći primer pronalazi sredinu intervala između i i j :

```
class Sredina {  
    public static void main(String[] args) {  
        int i = 100;  
        int j = 200;  
        // sredina intervala izmedju i i j, ako je u startu i < j  
        while (++i < --j);  
        System.out.println("Sredina je: " + i);  
    }  
}
```

Naredbe ponavljanja: `do-while`

- Kao i `while` naredba, omogućava ponavljanje izvršavanja date naredbe ili bloka naredbi dokle god je vrednost datog logičkog izraza jednaka `true`
- Vrednost logičkog izraza se izračunava *posle* prvog ponavljanja, što znači da će se date naredbe izvršiti bar jednom
- Kao i kod `while` naredbe, važno je uticati na sledeća izračunavanja logičkog izraza
- Oblici:

```
do  
    naredba;  
while (izraz);
```

```
do {  
    naredba;  
    ...  
    naredba;  
} while (izraz);
```

Naredbe ponavljanja: do-while

- Primer:

```
class HelloDoWhile {  
    public static void main(String[] arguments) {  
        int i = 1;  
        do {  
            System.out.println("Hello, world!");  
            i++;  
        } while (i <= 5);  
    }  
}
```

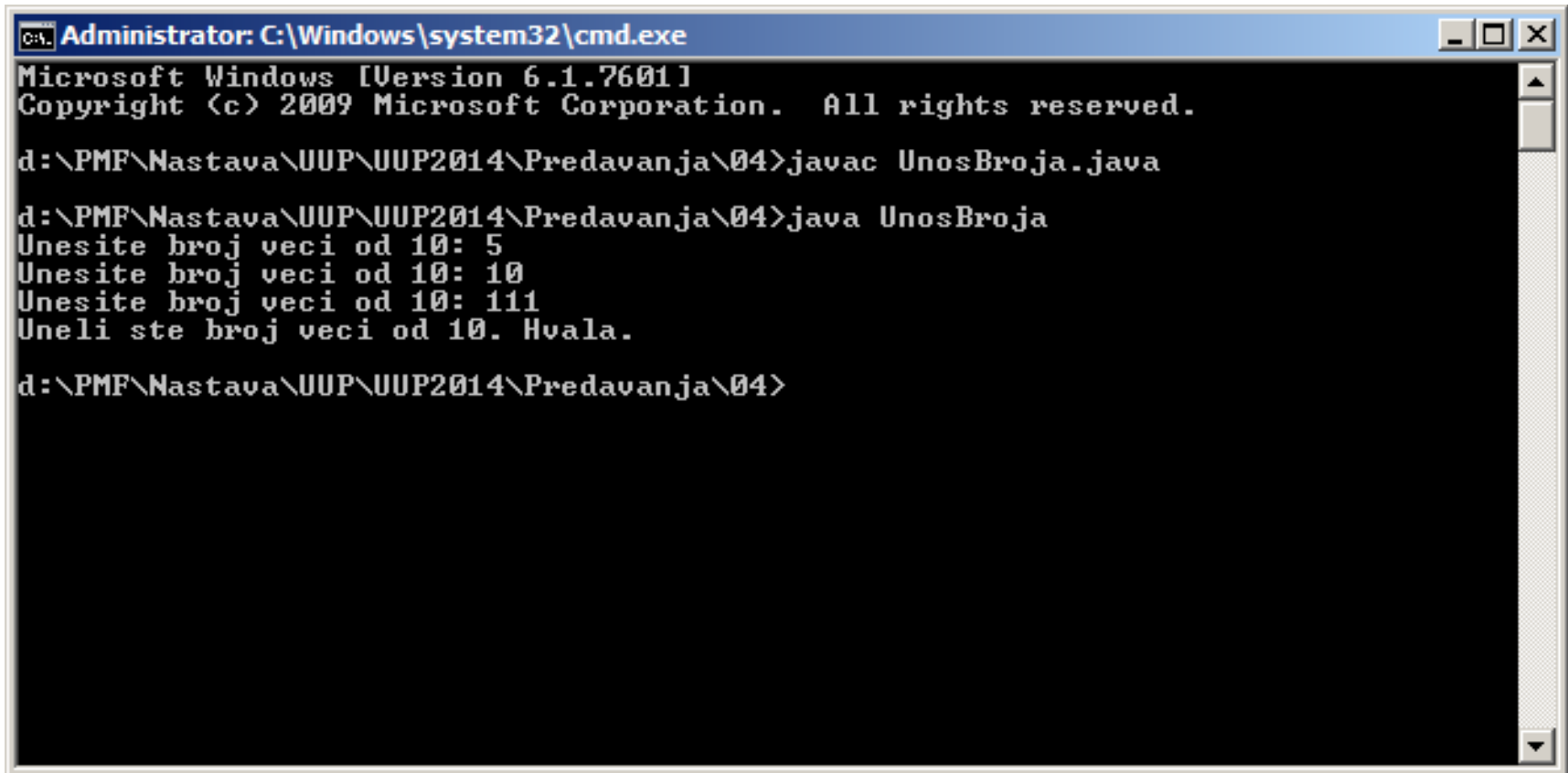
Naredbe ponavljanja: do-while

- Primer:

```
class UnosBroja {  
    public static void main(String[] arguments) {  
        int broj;  
        do {  
            System.out.print("Unesite broj veci od 10: ");  
            broj = Svetovid.in.readInt();  
        } while (broj <= 10);  
        System.out.println("Uneli ste broj veci od 10. Hvala.");  
    }  
}
```


Naredbe ponavljanja: do-while

- Izlaz:

A screenshot of a Windows command prompt window titled "Administrator: C:\Windows\system32\cmd.exe". The window shows the execution of a Java program named "UnosBroja.java". The output of the program is displayed in the command prompt, showing the user entering numbers and the program responding with the count of digits. The command prompt window has a standard Windows interface with a title bar, menu bar, and scrollbars.

```
Administrator: C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

d:\PMF\Nastava\UUP\UUP2014\Predavanja\04>javac UnosBroja.java

d:\PMF\Nastava\UUP\UUP2014\Predavanja\04>java UnosBroja
Unesite broj veci od 10: 5
Unesite broj veci od 10: 10
Unesite broj veci od 10: 111
Uneli ste broj veci od 10. Hvala.

d:\PMF\Nastava\UUP\UUP2014\Predavanja\04>
```

Naredbe ponavljanja: `for`

- Naredba `for` prvenstveno služi da ponavlja naredbu (ili blok naredbi) za određene vrednosti brojačke (kontrolne) promenljive
 - Na primer, za vrednosti celobrojne promenljive `i` redom 1, 2, 3, 4, 5
- Naredba `for` dozvoljava i šire primene od gore opisane
- Oblik:

```
for (pocetak; izraz; korak) {  
    naredba;  
    ...  
    naredba;  
}
```
- `pocetak`: inicijalizacija brojačke promenljive ili promenljivih (može i deklaracija sa inicijalizacijom), u slučaju više promenljivih inicijalizacije se odvajaju sa `,`
- `izraz`: ako ovaj logički izraz ima vrednost `false`, izlazi se iz petlje (obično proverava da li je brojačka promenljiva dostigla neku vrednost)
- `korak`: naredba (ili naredbe razdvojene sa `,`) koja se izvršava u svakoj iteraciji (obično ažuriranje brojačke promenljive)

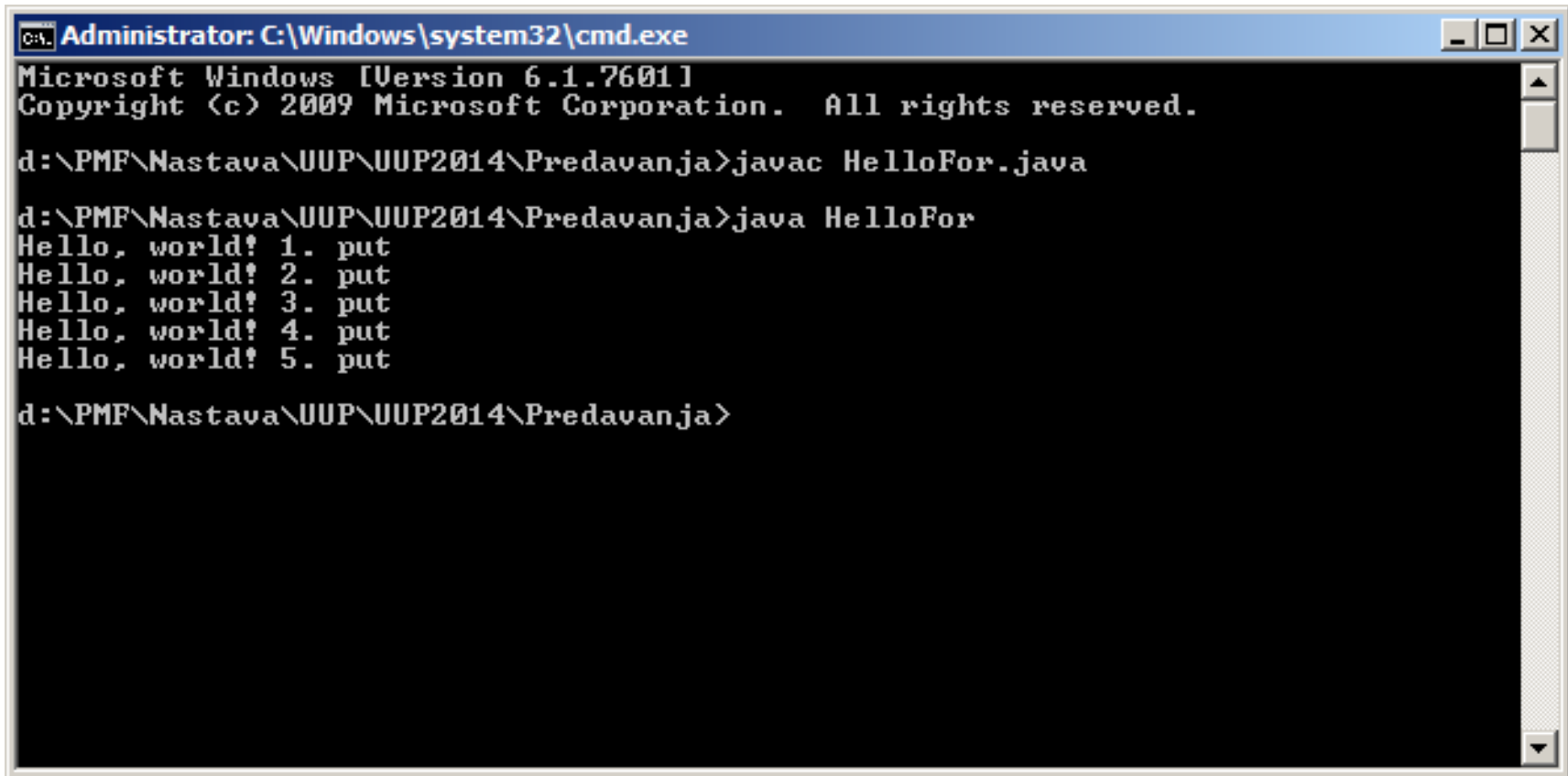
Naredbe ponavljanja: for

- Primer:

```
class HelloFor {  
    public static void main(String[] arguments) {  
        for (int i = 1; i <= 5; i = i + 1) {  
            System.out.println("Hello, world! " + i + ". put");  
        }  
    }  
}
```

Naredbe ponavljanja: for

- Izlaz:

A screenshot of a Windows command prompt window titled "Administrator: C:\Windows\system32\cmd.exe". The window shows the execution of a Java program. The prompt is at the directory "d:\PMF\Nastava\UUP\UUP2014\Predavanja". The user enters "javac HelloFor.java" and then "java HelloFor". The output of the program is five lines of "Hello, world!" followed by "1. put" through "5. put".

```
Administrator: C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

d:\PMF\Nastava\UUP\UUP2014\Predavanja>javac HelloFor.java

d:\PMF\Nastava\UUP\UUP2014\Predavanja>java HelloFor
Hello, world! 1. put
Hello, world! 2. put
Hello, world! 3. put
Hello, world! 4. put
Hello, world! 5. put

d:\PMF\Nastava\UUP\UUP2014\Predavanja>
```

Naredbe ponavljanja: **for**

- Svaki od tri dela (pocetak, izraz, korak) se može izostaviti
- Primer ekvivalentnih `for` naredbi:

```
for (int i = 1; i <= 5; i++) {  
    System.out.println("Hello, world! " + i + ". put");  
}
```

```
int i = 1;  
for (; i <= 5; i++) {  
    System.out.println("Hello, world! " + i + ". put");  
}
```

```
for (int i = 1; i <= 5;) {  
    System.out.println("Hello, world! " + i + ". put");  
    i++;  
}
```

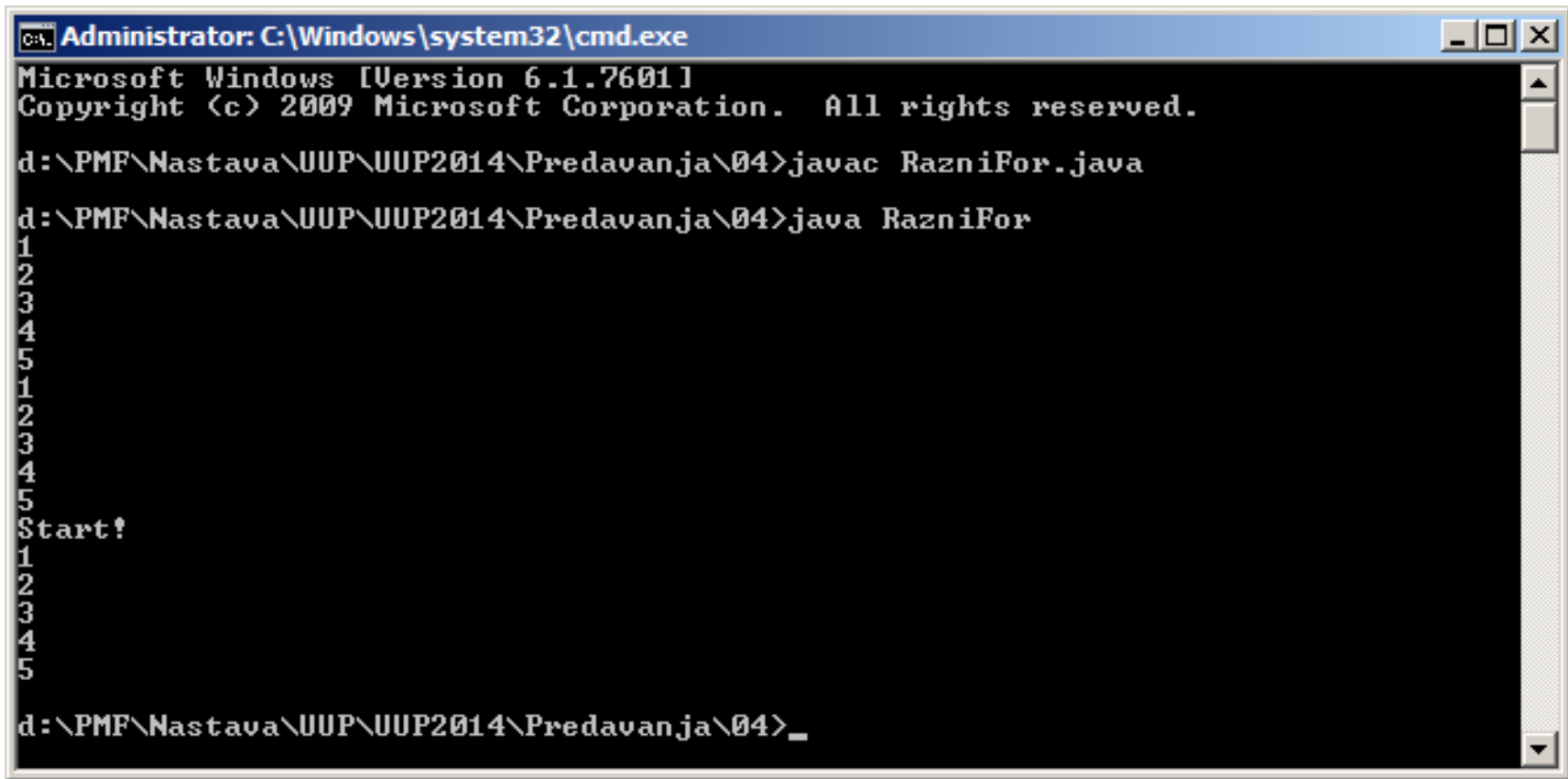
Naredbe ponavljanja: **for**

- Delovi pocetak i korak mogu sadržati više naredbi odvojenih zarezima, ne obavezno posvećenih inicijalizaciji, odnosno ažuriranju vrednosti
- Primer:

```
class RazniFor {  
    public static void main(String[] arguments) {  
        for (int i = 1; i <= 5; i++) {  
            System.out.println(i);  
        }  
        for (int i = 1; i <= 5; System.out.println(i), i++);  
        int i = 1;  
        for (System.out.println("Start!"), i = 1;  
            i <= 5;  
            System.out.println(i++));  
    }  
}
```

Naredbe ponavljanja: for

- Izlaz:

A screenshot of a Windows command prompt window titled "Administrator: C:\Windows\system32\cmd.exe". The window shows the output of a Java program. The first two lines are the standard Windows copyright notice. The third line shows the command "javac RazniFor.java" being executed. The fourth line shows the command "java RazniFor" being executed. The output of the program is a list of numbers 1 through 5, followed by the text "Start!", and then another list of numbers 1 through 5. The prompt "d:\PMF\Nastava\UUP\UUP2014\Predavanja\04>" is visible at the bottom.

```
Administrator: C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

d:\PMF\Nastava\UUP\UUP2014\Predavanja\04>javac RazniFor.java
d:\PMF\Nastava\UUP\UUP2014\Predavanja\04>java RazniFor
1
2
3
4
5
1
2
3
4
5
Start!
1
2
3
4
5
d:\PMF\Nastava\UUP\UUP2014\Predavanja\04>_
```

Naredbe ponavljanja: `for`

- Naredba `for` može istovremeno da radi sa više brojačkih promenljivih
- Primer:

```
class ForViseBrojaca {  
    public static void main(String[] args) {  
        int a, b;  
        for(a = 1, b = 4; a < b; a++, b--) {  
            System.out.println("a = " + a);  
            System.out.println("b = " + b);  
        }  
    }  
}
```

- Izlaz:

```
a = 1  
b = 4  
a = 2  
b = 3
```


Naredbe ponavljanja: `for`

- Naredba `for` ima i unapređenu (eng. *enhanced*) varijantu, koja se još naziva i `for-each` naredba
- Koristi se kod nizova i kolekcija da bi se na jednostavan način iteriralo (prošlo) kroz sve elemente niza/kolekcije
- Oblik:

```
for (elem : nizIliKolekcija) {  
    naredba;  
    ...  
    naredba;  
}
```
- `elem`: ime ili deklaracija promenljive
- `nizIliKolekcija`: ime objekta koji predstavlja niz ili kolekciju čiji su elementi istog tipa kao `elem`
- Efekat je da će promenljiva `elem` redom dobijati vrednosti elemenata iz `nizIliKolekcija`, i za svaku tu dodelu biće izvršene navedene naredbe
- Primere `for-each` naredbe daćemo kad budemo obrađivali nizove

Naredbe sa labelom

- Labele (oznake) se koriste da bi se pomoću naredbi `break` i `continue` uticalo na tok izvršavanja programa
- Unutar označene naredbe, naredbama `break` i `continue` postiže se skok toka programa na željeno mesto
- Oblik navođenja labele:
`identifikator: naredba;`

Naredba `break`

- Naredba `break` ima efekat “skoka” toka izvršavanja programa, i ima tri namene:
 - Izlazak iz `switch` naredbe (gde je praktično neizbežna)
 - Izlazak iz naredbi ponavljanja (gde se ređe koristi)
 - Kao “civilizovana” varijanta `goto` naredbe (gde se najređe koristi)

- Naredba `break` ima dva oblika:
 - Bez labele
 - Sa labelom

Naredba `break`

- Naredba `break` bez labele:
 - Može da se nalazi samo unutar naredbi `switch`, `while`, `do-while` i `for`
 - Prebacuje tok izvršavanja programa posle prve (najunutrašnije) naredbe `switch`, `while`, `do-while` ili `for` koja sadrži `break`
- Naredba `break` sa labelom:
 - Ne mora da se nalazi unutar naredbi `switch`, `while`, `do-while` i `for`
 - Naredba koja sadrži `break` mora biti označena labelom navedenom uz `break` naredbu
 - Prebacuje tok izvršavanja programa posle naredbe označene labelom
 - Najčešće služi za izlazak iz više ugneždenih petlji

Naredba break

- Primer:

```
class BreakFor {  
    public static void main(String[] arguments) {  
        for (int i = 1; i <= 5; i++) {  
            System.out.println("i = " + i);  
            if (i == 3) break;  
        }  
    }  
}
```

- Izlaz:

```
i = 1  
i = 2  
i = 3
```

Naredba break

- Primer:

```
class BreakForNested {  
    public static void main(String[] arguments) {  
        prva:  
        for (int i = 1; i <= 5; i++) {  
            for (int j = 1; j <= 4; j++) {  
                System.out.println("i = " + i + ", j = " + j);  
                if (i == 3) break prva;  
            }  
        }  
    }  
}
```

- Izlaz:

```
i = 1, j = 1  
i = 1, j = 2  
i = 1, j = 3  
i = 1, j = 4  
i = 2, j = 1  
i = 2, j = 2  
i = 2, j = 3  
i = 2, j = 4  
i = 3, j = 1
```

Naredba break

- Primer:

```
class Breaks {  
    public static void main(String[] arguments) {  
        boolean t = true;  
        prvi: {  
            drugi: {  
                treci: {  
                    System.out.println("pre break");  
                    if (t) break drugi;  
                    System.out.println("ovo se ne izvrsava");  
                }  
                System.out.println("ovo se ne izvrsava");  
            }  
            System.out.println("posle bloka drugi");  
        }  
    }  
}
```

- Izlaz:
pre break
posle bloka drugi

Naredba `continue`

- Slična naredbi `break`
- Može da se koristi samo u okviru naredbi ponavljanja (`while`, `do-while` i `for`)
- Efekat je da se tekuća iteracija petlje preskače, i tok izvršavanja programa nastavlja sa sledećom iteracijom
- Kao i naredba `break`, naredba `continue` ima dva oblika:
 - Bez labele
 - Sa labelom

Naredba `continue`

- Naredba `continue` bez labele:
 - Prebacuje tok izvršavanja programa na početak sledeće iteracije prve (najunutrašnije) naredbe `while`, `do-while` ili `for` koja sadrži `continue`

- Naredba `continue` sa labelom:
 - Naredba koja sadrži `continue` mora biti označena labelom navedenom uz `continue` naredbu
 - Prebacuje tok izvršavanja programa na sledeću iteraciju petlje koja je označena tom labelom, i ne mora biti najunutrašnjija (koristi se kod više ugneždenih petlji)

Naredba continue

- Primer:

```
class Prosti {  
    public static void main(String[] arguments) {  
        System.out.print("Prosti brojevi od 1 do 100 su: ");  
        System.out.print(2);  
        prvi:  
        for (int i = 3; i <= 100; i++) {  
            if (i % 2 == 0) continue;  
            for (int j = 3; j <= (int)Math.sqrt(i); j++)  
                if (i % j == 0) continue prvi;  
            System.out.print(", " + i);  
        }  
        System.out.println();  
    }  
}
```

- Izlaz:

Prosti brojevi od 1 do 100 su: 2, 3, 5, 7, 11, ..., 89, 97

Naredba `return`

- Koristi se samo u telu metoda (i konstruktora klase)
- Efekat je da se izvršavanje tela metoda prekida, i tok programa nastavlja nakon mesta gde je metod pozvan
- Naredba `return` ima dva oblika:
 - Bez izraza: `return;`
 - Povratak iz metoda koji nema povratnu vrednost (vraća tip `void`)
 - Sa izrazom: `return izraz;`
 - Povratak iz metoda koji ima povratnu vrednost (istog tipa kao `izraz`)
- O naredbi `return` biće još reči kada budemo obrađivali metode

Naredba return

- Primer:

```
class PrekidMetoda {  
    public static void main(String[] args) {  
        boolean t = true;  
        System.out.println("pre return-a");  
        if (t) return;  
        System.out.println("nece se izvrsiti");  
    }  
}
```

- Izlaz:

pre return-a