

Naredbe

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)

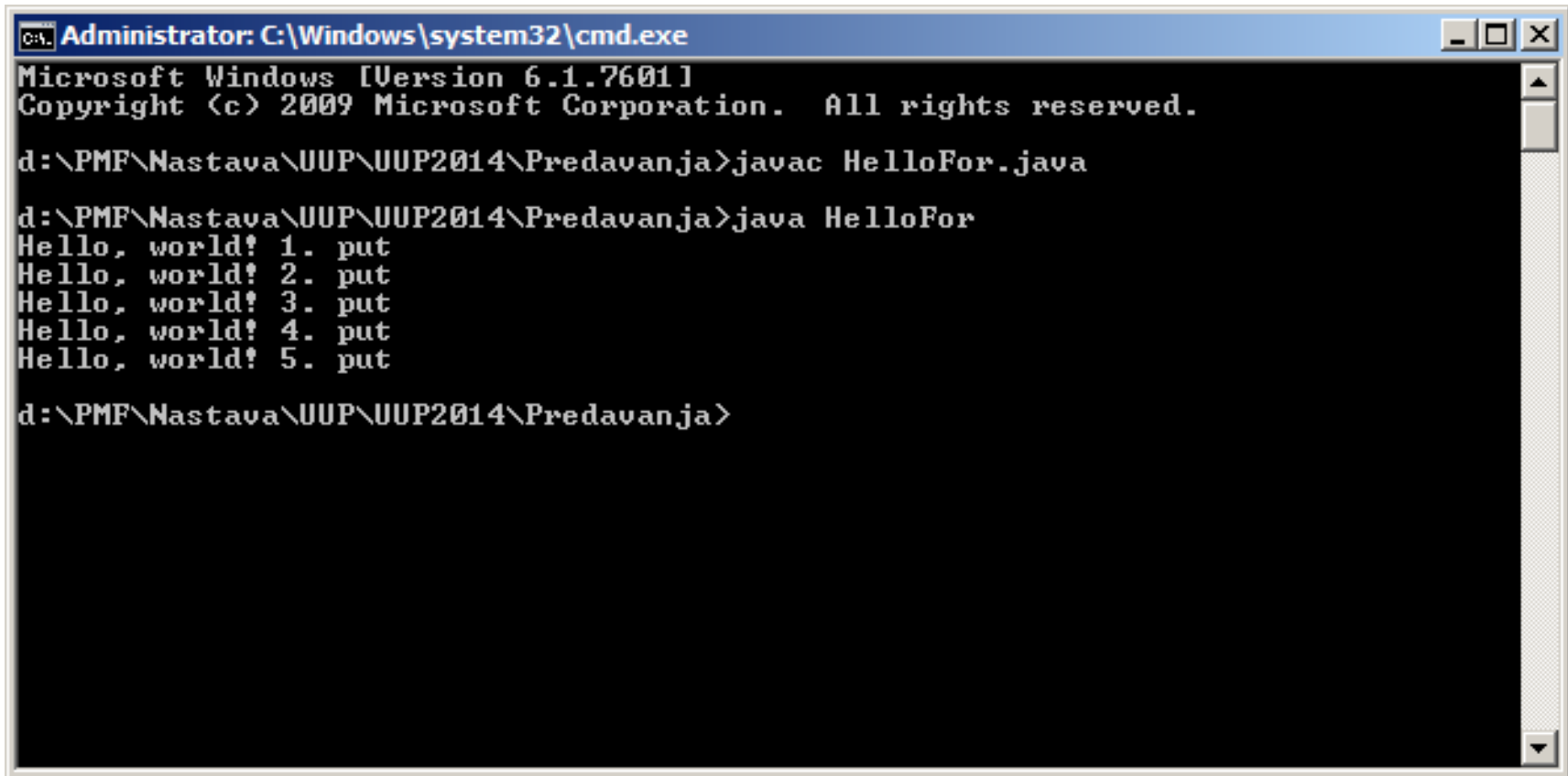
Naredbe ponavljanja: **for**

- 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++;  
}
```

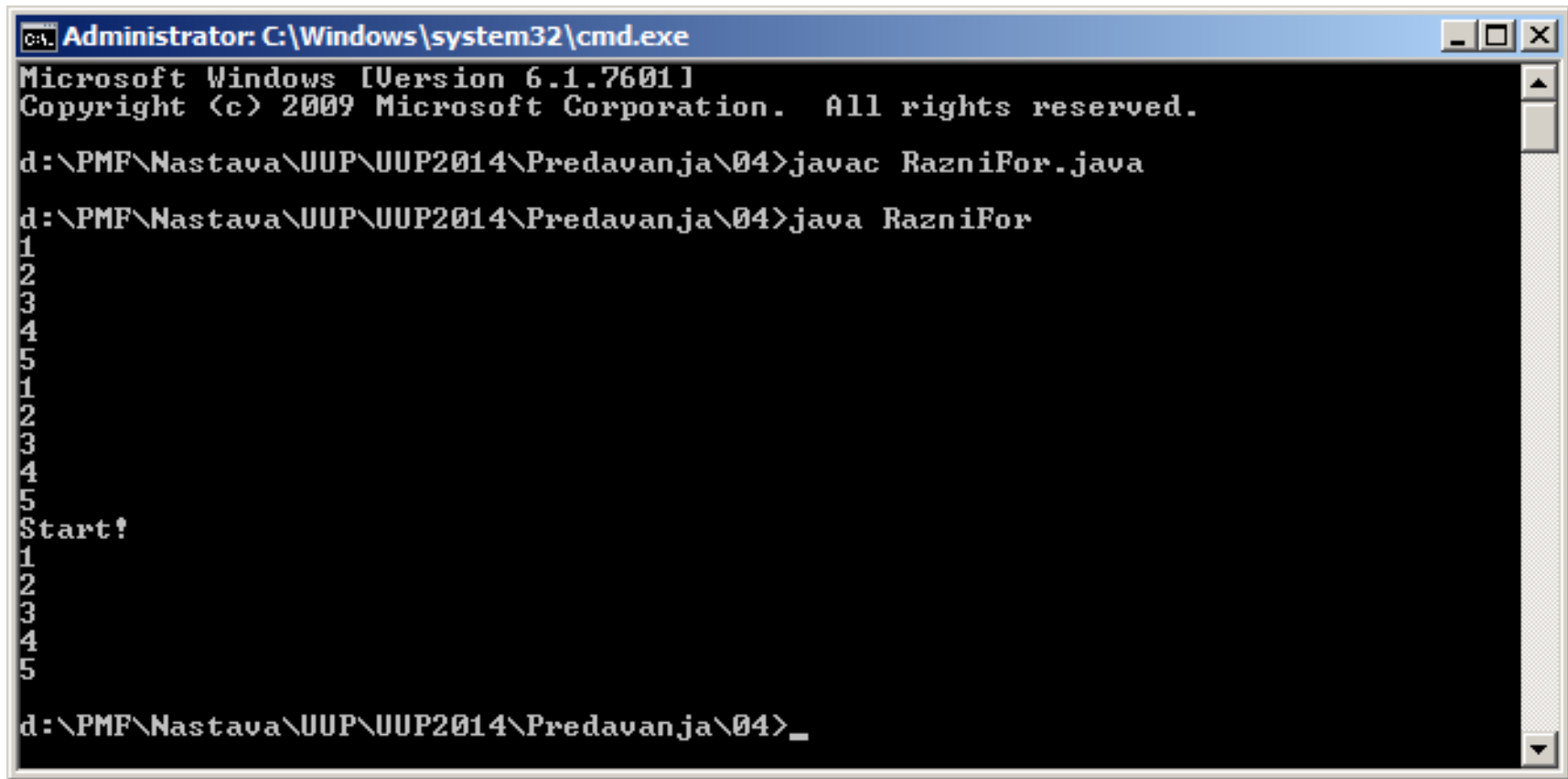
Naredbe ponavljanja: **for**

- 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:



```
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