Академија струковних
студија Шумадија
одсек у Крагујевцу

Увод у програмирање

Презентација 3

Академија струковних студија Шумадија

Одсек у Крагујевцу

Студијски програм Информатика

Крагујевац, 2020. година

Сложене наредбе

- Наредбе у Јави могу бити просте и сложене.
- У основне наредбе спадају:
  - дефинисање променљивих;
  - наредбе доделе и
  - наредбе за позивање метода.
- Сложене наредбе се граде од простијих програмских елемената на три начина, чиме се добијају посебне категорије управљачких наредби:
  - блок наредби;
  - наредбе гранања и
  - наредбе понављања.
- О овој подели детаљније (опширније) говоримо у наредним слајдовима.

Блок (1)

- Блок наредби (или краће само блок) је најпростији начин комбиновања наредби којим се само низ наредби групише у једну целину.
- Општи облик:

```
{
    наредба_1;
    наредба_2;
    …
    наредба_n;
}
```

- Блок се обично налази унутар других сложених наредби када је потребно да се више наредби групише у једну (сложену) наредбу.

Блок (2)

- Блок наредби се може писати на сваком месту у програму где се може користити обична наредба.
- Када се наиђе на `{` све наредбе у низу које следе иза тога се извршавају једна за другом, док се не наиђе на `}`.
- До сада смо блок наредби користили главним класама као и у `main()` методи сваког програма.
- Област важења (домет) неке променљиве дефинисане у блоку је од тачке дефиниције те променљиве до краја блока.
- Локалне променљиве имају домет само у оквиру свог блока и не могу се користити у другим блоковима.
- Пример:

Блок (3)

```
{
        int x, y;
        {
                int i = 5;
                x = (i++) - 3;
                y = i + 4;
        }
        i = 0; // greška
}
```

1 2

- 1 – први блок који у овом примеру можемо сматрати за спољашњи блок.
- 2 - други блок који у овом примеру можемо сматрати за унутрашњи блок.
- Променљива `int i` се налази у блоку 2, не може јој се задати вредност у блоку 1. Приметите да за променљиве **x** и **y** ово не важи! Зашто?

Блок (4)

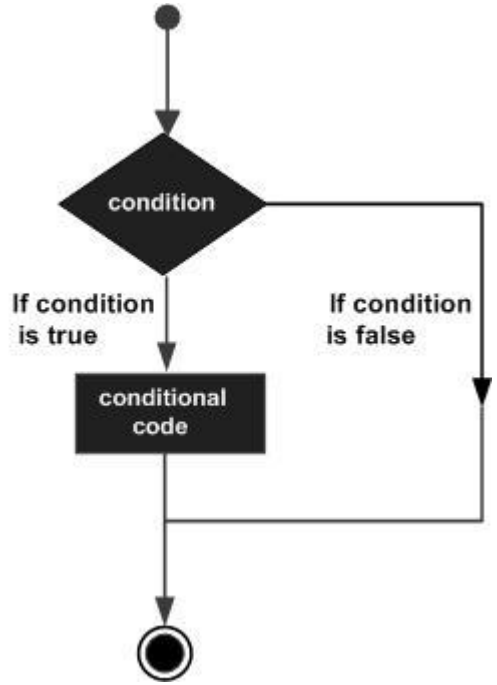
- Променљива која је дефинисана унутар блока је недоступна ван тог блока.
  - у нашем примеру је то `int i`
- Променљива се "уништава" (*garbage collection*) се након извршења блока.
- Употребом блока се спречавају се озбиљнији проблеми ненамерне употребе исте променљиве за друге сврхе.
- Због тога се, у нашем примеру конкретно, `int i` назива локална променљива за тај блок, односно тај блок је њена област важења.
- Блокове користимо свуда у Јави, почев од класа, метода, у нарадбама за контролу тока програма (гранања), наредбама понављања (петљама), интерфејсима, ...

Управљачке наредбе

- Управљачке наредбе се у програмским језицима користе за управљање током извршавања програма.
- Јавине управљачке наредбе сврставају се у следеће категорије:
  - условне наредбе,
  - итерације и
  - скокови.
- Условне наредбе (наредбе гранања) омогућавају да програм "изабере" различит ток извршавања.
- Наредбе за итерације (наредбе понављања) омогућавају програму да једну или више наредби понови више пута.
- Наредбе за скокове омогућавају нелинеарно извршавање програма.

Условне наредбе

- Јава подржава две условне наредбе:
  - наредба **`if`**
  - наредба **`switch`**
- За условно извршење наредбе или за избор између извршења две наредбе обично се користи наредба **`if`**.
- Алгоритамски представљено:
  - уколико је услов (*condition*) тачан извршиће се наредба (*conditional code*),
  - уколико је услов нетачан програм ће прескочити наредбу (*conditional code*) и наставити даље да се извршава.



Условне наредбе

- Пример:

1

```
public class Test {
    public static void main(String[] args){
        int broj = 6;
        if(broj > 5){
            System.out.println("Положили сте испит!");
        }
    }
}
```

2

```
public static void main(String[] args){
    boolean ispit = false;
    if(ispit == true){
        System.out.println("Положили сте испит!");//neće se izvršiti
    }
    System.out.println("Уколико читате ово услов није тачан, а то значи да
                        сте пали сте на испиту.");//ne pripada if bloku
}
```

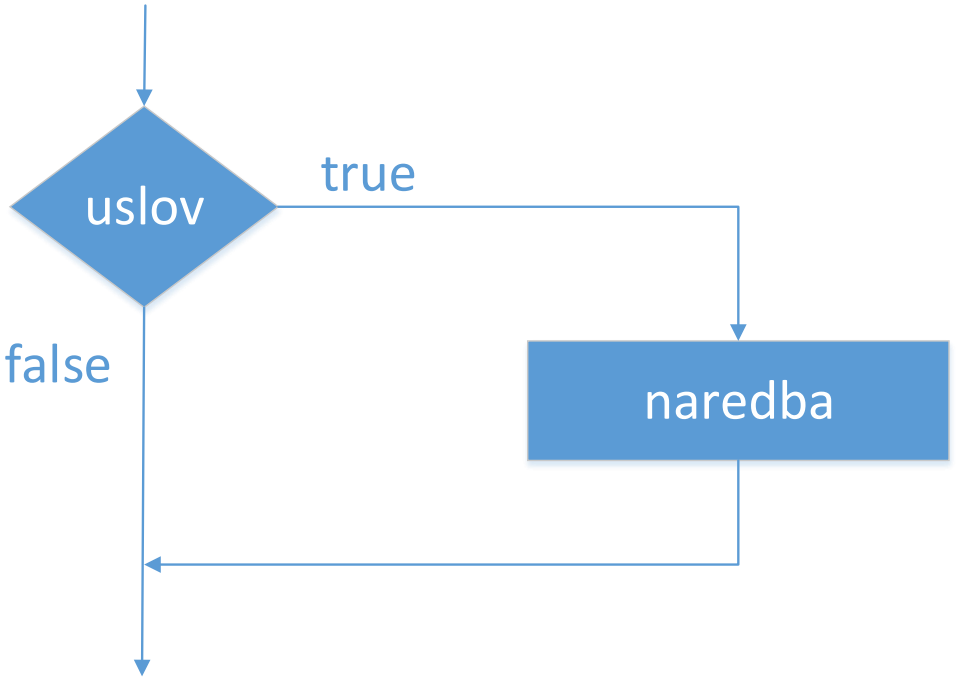
Условне наредбе

- Када иза наредбе **if** постоји само једна наредба не морате да стављате блок **{ }**.
- Стављање блока је добра пракса, студенти који уче програмирање се неће збунити, јер се јасно одваја шта припада или не припада наредби **if**.
- Претходни пример (пример 2):

```
public class Test {
    public static void main(String[] args){
        boolean ispit = false;
        if(ispit == true)
            System.out.println("Положили сте испит!");
        System.out.println("Уколико читате ово услов није тачан, а то
                           значи да сте пали сте на испиту.");
    }
}
```

Условне наредбе

- У општем случају, извршавање `if` наредбе се изводи у две фазе.
- Прво се израчунава вредност логичког израза у загради.
- Уколико је та вредност тачна (*true*), извршава се наредба (блок) у продужетку.
- Ако је вредност нетачна (*false*), ништа се додатно не извршава, наредба у продужетку се прескаче.
- Да поновимо још једном, веома је важно да ово научите:



Условне наредбе

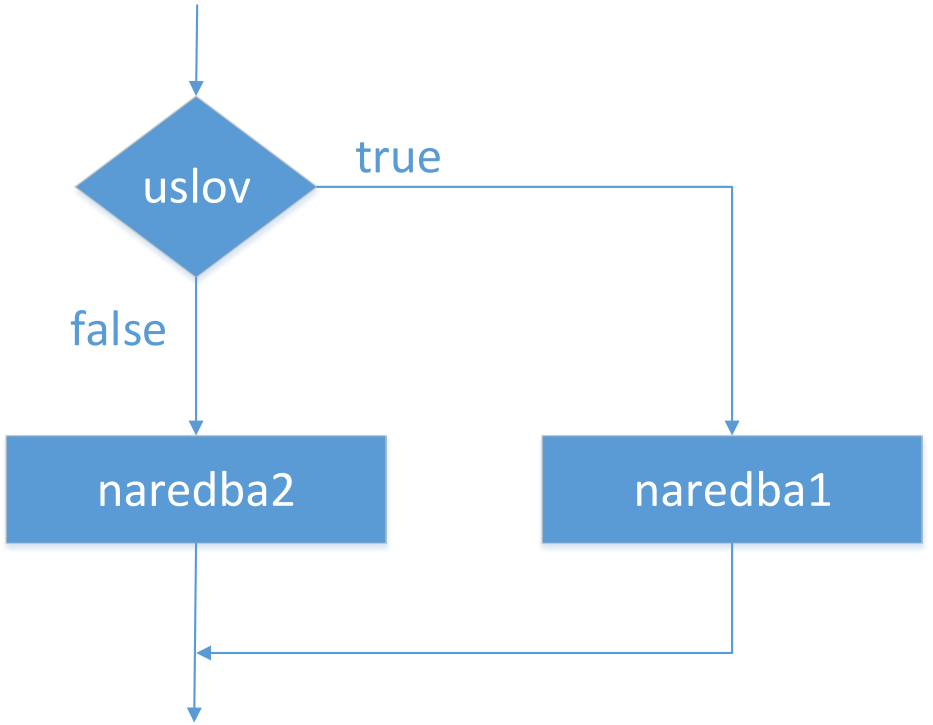
- Уколико се због нетачног услова наредба која се налази на грани алгоритма "*true*" прескаче, логика налаже да се уради нешто друго.
- Због тога се наредба **`if`** проширује са **`else`** у чијем блоку (не мора да буде блок ако постоји само једна наредба) налази наредба која се извршава уколико је услов "*false*". Ово се сада зове **`if-else`** наредба.

```
public static void main(String[] args) {
    int r = 3;
    if (r != 0) { //ovde se pojavljuje logička greška
        System.out.println("Обим круга је: " + 2 * r * Math.PI);
    } else {
        System.out.println("Полупречник не сме бити 0 или негативан број.");
    }
}
```

- Задајте променљивој **`x`** вредност -3. Који део кода се извршава?

Условне наредбе

- Извршавање наредбе **`if-else`** је слично наредби **`if`**.
- Уколико је вредност логичког израза *true*, извршава се *naredba1*, а прескаче *naredba2*.
- Уколико је вредност логичког израза *false*, прескаче се *naredba1* и извршава *naredba2*.
- Увек се извршава само једна наредба.
- *naredba1* и *naredba2* могу бити блокови наредби.

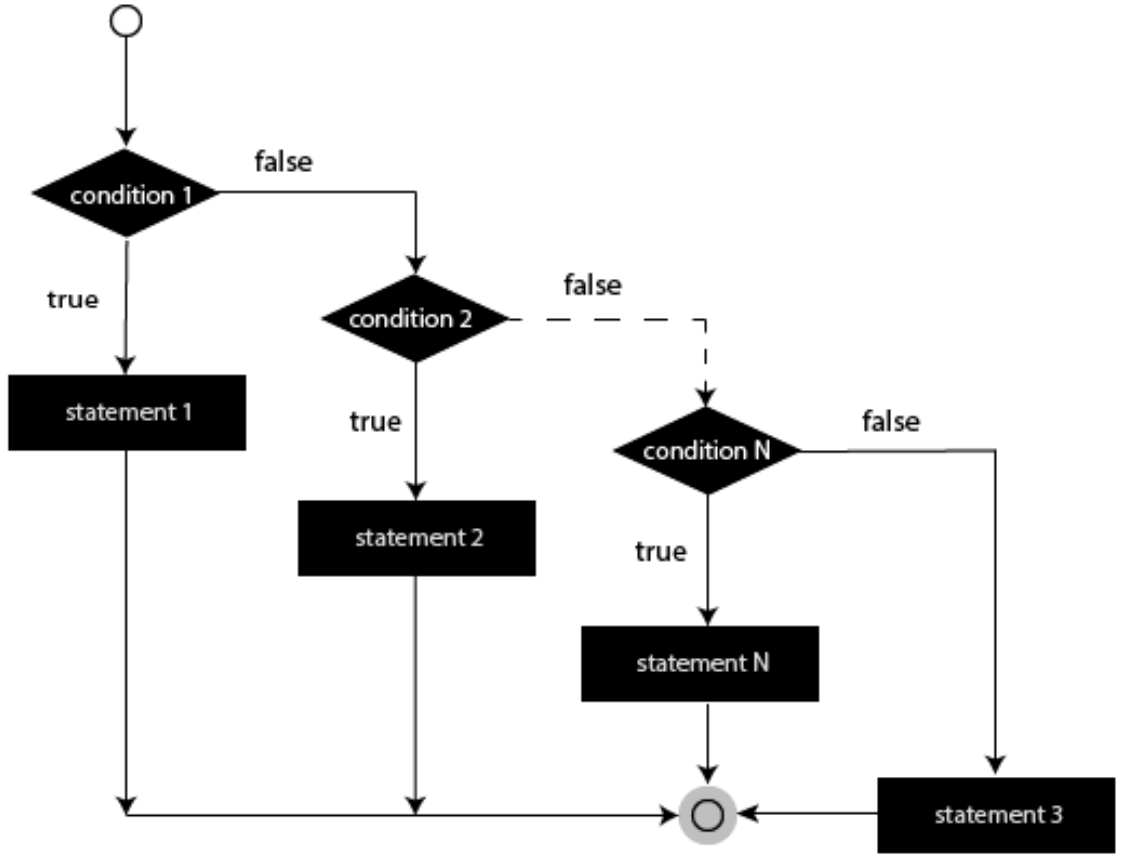


Условне наредбе

- Наредбе у оквиру **`if`** и **`if-else`** могу бити било које наредбе.
- То могу бити и друге наредбе **`if`** или **`if-else`**.
  - Уколико је то случај онда се то звове угњежђавање наредби.

```
if (uslov1)
   naredba1
else if (uslov2)
   naredba2
else if (uslov3)
   naredba3
...
else if (uslov(n))
   naredba(n)
else
   naredba(n+1)
```

≠



Условне наредбе

```
public static void main(String[] args) {
    int bodovi;
    System.out.println("Унеси бр. освојених бодова:");
    Scanner ulaz = new Scanner(System.in);
    bodovi = ulaz.nextByte(); // u principu se koristi nextInt()
    if (bodovi > 50 & bodovi <= 60) {
        System.out.println("Оцена 6");
    } else if (bodovi > 60 & bodovi <= 70) {
        System.out.println("Оцена 7");
    } else if (bodovi > 70 & bodovi <= 80) {
        System.out.println("Оцена 8");
    } else if (bodovi > 80 & bodovi <= 90) {
        System.out.println("Оцена 9");
    } else if (bodovi > 90) {
        System.out.println("Оцена 10");
    } else {
        System.out.println("Оцена 5");
    }
}
```

Условне наредбе

- Логички изрази у заградама се извршавају редом одозго надоле.
- Користили смо и логички оператора *AND - &*.
  - Он условљава да у исто време лева и десна страна у логичком изразу морају да врате вредност *true*, како би услов био тачан.
- Кад се нађе први услов чија је вредност *true*, извршава се његова наредба и прескаче све остало.
- Ако су вредности свих логичких израза *false*, извршава се наредба у последњем `else` делу.

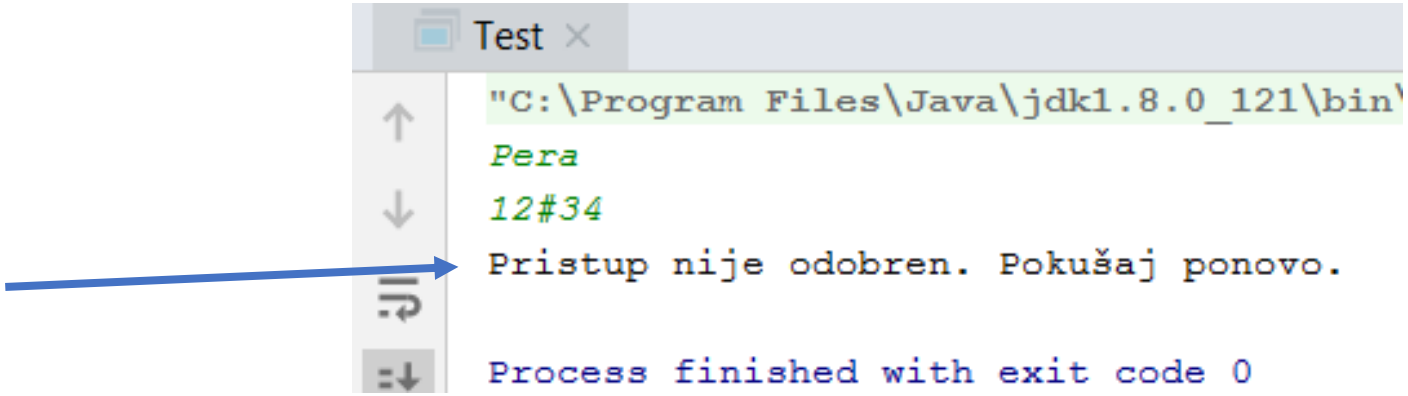
Уколико се до сада нисте "упознали" са краткоспојним логичким операторима *AND* (**&&**) и *OR* (**||**), онда је сада прави тренутак да то урадите.

Условне наредбе

- Оператори за поређење вредности се могу користити **искључиво са простим типовима података**.
- Објекти типа **`String`** се пореде на други начин, о чему ћемо накнадно опширније говорити (градиво за ООП), јер ако поредите два објекта преко оператора "==" ви заправо **поредите садржај показивача на те објекте (адресе у меморији), а не садржај атрибута објеката**.
- Ви ћете добити "исправан" (синтакса је исправна) резултат ако упоредите два "стринга" помоћу оператора "==", међутим **то није исправан начин** за поређење "стрингова".
- Уколико је потребно да упоредите два "стринга", пре него што детаљно обрадимо (у настави) класу **`String`** користите метод **`equals()`**.
- Пример:

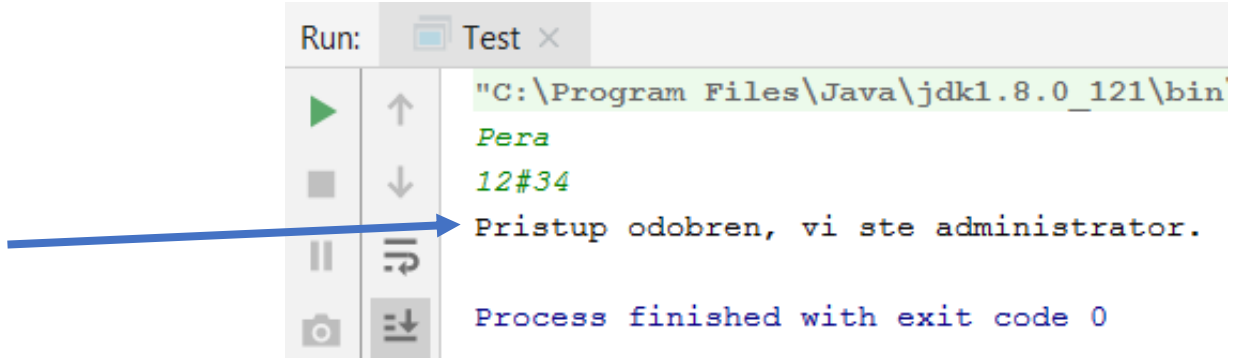
Условне наредбе

```
public static void main(String[] args) {
    String admin, password, proveraAdmin, proveraPassword;
    admin = "Pera"; password = "12#34";
    Scanner ulaz = new Scanner(System.in);
    proveraAdmin = ulaz.nextLine();
    proveraPassword = ulaz.nextLine();
    if(admin == proveraAdmin && password == proveraPassword){
        System.out.println("Pristup odobren, vi ste administrator.");
    }else {
        System.out.println("Pristup nije odobren. Pokušaj ponovo.");
    }
```



Условне наредбе

```
public static void main(String[] args) {
    String admin, password, proveraAdmin, proveraPassword;
    admin = "Pera"; password = "12#34";
    Scanner ulaz = new Scanner(System.in);
    proveraAdmin = ulaz.nextLine();
    proveraPassword = ulaz.nextLine();
    if (admin.equals(proveraAdmin) && password.equals(proveraPassword)) {
        System.out.println("Pristup odobren, vi ste administrator.");
    } else {
        System.out.println("Pristup nije odobren. Pokušaj ponovo.");
    }
}
```



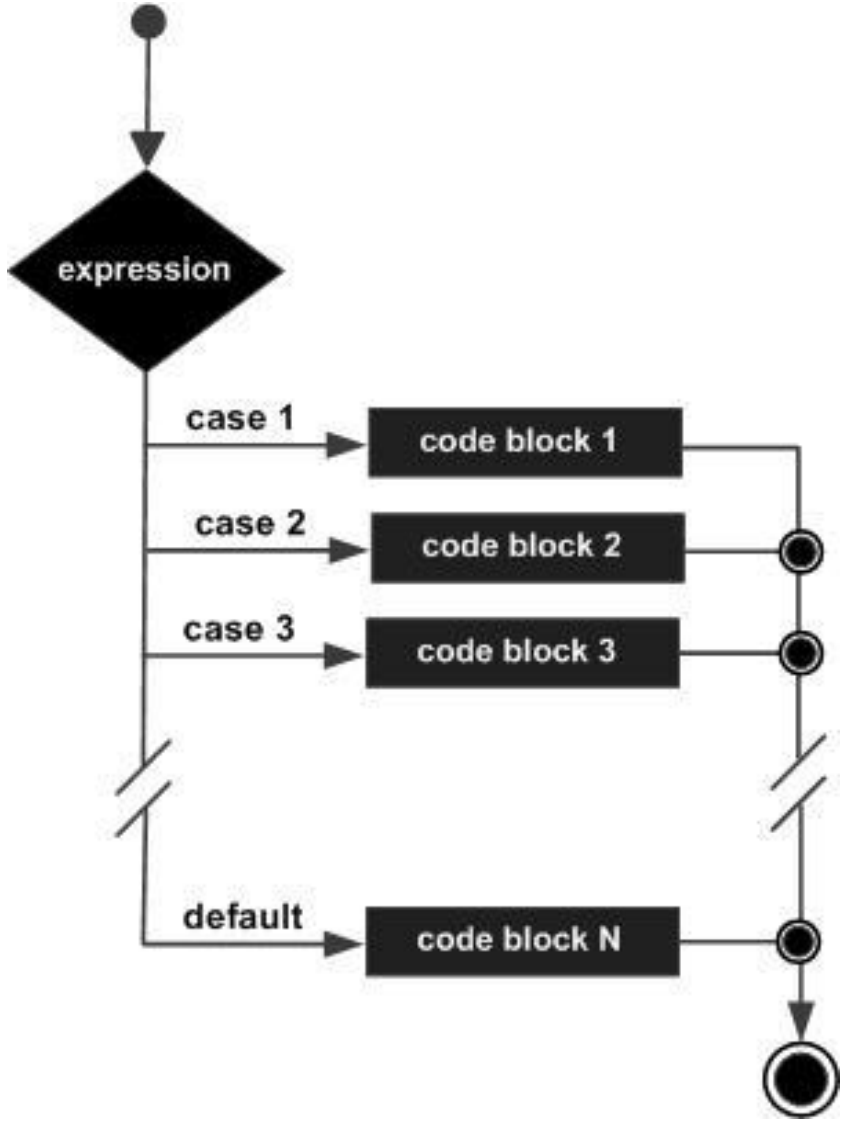
Условне наредбе

- Наредба **`switch`** омогућава гранање програма у више смерова.
- Заправо, наредба **`switch`** служи за избор једне наредбе из скупа од неколико могућих, а на основу вредности неког израза.
- Ако избор више алтернативних блокова за извршавање зависи од вредности једног израза употребите наредбу **`switch`**.
- За вредност израза (вредност коју задајемо) који се налази у **`()`** уз кључну реч **`switch`**, од верзије **JDK 7**, можете да користите и тип **`String`**.
- Тип свака излазне вредности мора бити компатибилан са типом податка који се налази у изразу.
- Општи облик наредбе **`switch`**:

Условне наредбе

```
switch (izraz){
case vrednost_1:
    grupa_naredbi; break;
case vrednost_2:
    grupa_naredbi; break;
...
case vrednost_n:
    grupa_naredbi; break;
default:
    grupa_podrazumevanih_naredbi;
}
```

≠



Условне наредбе

Наредба **`switch`** функционише на следећи начин:

- Вредност израза се пореди са сваким литералом у наредби **`case`**.
- Ако у једној наредби **`case`** постоји поклапање, извршава се низ наредби који следу ту наредбу **`case`**.
- Уколико нема ниједног поклапања извршава се наредба **`default`**.
- Наредба није **`default`** обавезан део наредбе **`switch`** али је добра пракса, макар на почетку док учите програмирање, да се користи.
- Наредба **`break`** се користи унутар наредбе **`switch`** као завршетак групе наредби.
- Наредба **`break`** прекида извршавање наредбе **`switch`** и преноси даље извршавање програма на прву наредбу иза - **`}`** (крај блока **`switch`**).

Условне наредбе

```
int mesec;
Scanner ulaz = new Scanner(System.in);
System.out.print("Unesite broj od 1 - 12: ");
mesec = ulaz.nextInt();
switch (mesec) {
    case 1: System.out.println("Januar"); break;
    case 2: System.out.println("Februar"); break;
    case 3: System.out.println("Mart"); break;
    case 4: System.out.println("April"); break;
    case 5: System.out.println("Maj"); break;
    case 6: System.out.println("Jun"); break;
    case 7: System.out.println("Jul"); break;
    case 8: System.out.println("Avgust"); break;
    case 9: System.out.println("Septembar"); break;
    case 10: System.out.println("Oktobar"); break;
    case 11: System.out.println("Novembar"); break;
    case 12: System.out.println("Decembar"); break;
    default: System.out.println("Neispravan broj!");
}
```

Условне наредбе

Издвојићемо неке важне особине наредбе **`switch`**:

- Наредба **`switch`** се разликује од наредбе **`if`** по томе што само испитује једнакост, док наредбом **`if`** испитујемо сваки логички израз.
  - Другим речима, наредба **`switch`** утврђује да ли постоји поклапање између вредности израза и литерала који је наведен иза наредбе **`case`**.
- Наредба **`switch`** је углавном ефикаснија од великог низа угнежђених наредби **`if`**.
- Када компајлира наредбу **`switch`**, Јавин компајлер испитује сваки литерал у наредбама **`case`** и саставља тзв. "табелу скокова" коју користи за бирање путање извршавања програма у зависности од вредности израза. Због тога се наредба **`switch`** извршава много брже него еквивалентна логика која је имплементирана помоћу наредби **`if-else`**.

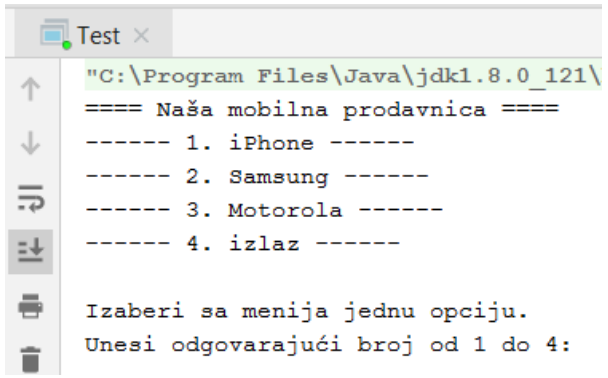
Прављење "менија"

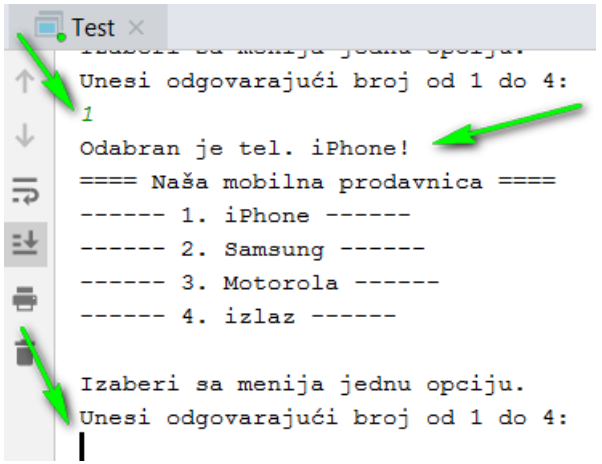
- Често се наредба **`switch`** користи за креирање "менија" (опција са више могућности одабира) али се уз такву имплементацију мора користити и наредба понављања или једноставно речено петља.
- Како су наредбе понављања тема коју обрађујемо на наредном предавању, у наставку следи један пример употребе наредбе switch која се користи у петљи **`do-while`**, без дубљег залажења у начин рада саме петље.
- Кориснику овог програма се нуди да изабере један од четири понуђена избора са "менија".
- Докле год корисник не одабере избор 4, програм наставља да се извршава нудећи кориснику поновни избор.
- Програм има и недостатак који ми за сада не можемо да разрешимо. Корисним мора да укуца податак типа **`int`** или програм "пуца".

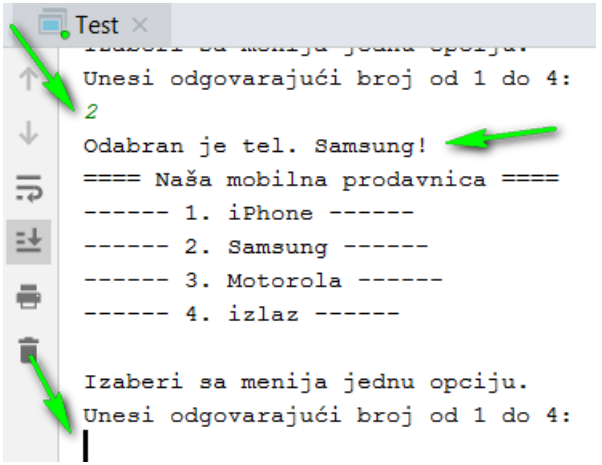
Прављење "менија"

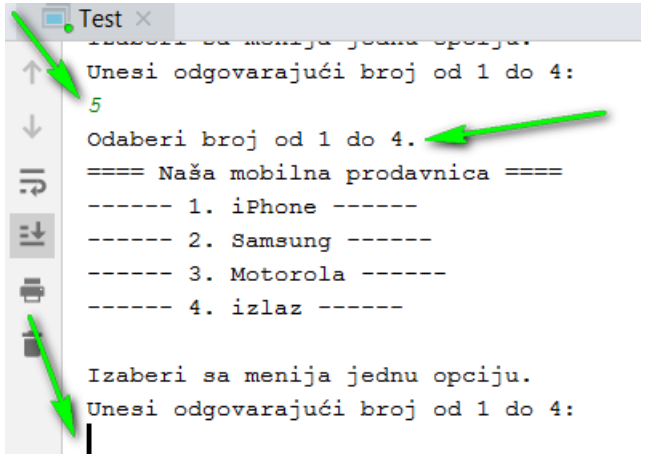
```
static Scanner ulaz = new Scanner(System.in);
static int odabir;
public static void main(String[] args) {
    do {
        System.out.println("==== Naša mobilna prodavnica ====");
        System.out.println("------ 1. iPhone ------");
        System.out.println("------ 2. Samsung ------");
        System.out.println("------ 3. Motorola ------");
        System.out.println("------ 4. izlaz ------");
        System.out.println();
        System.out.println("Izaberi sa menija jednu opciju.");
        System.out.println("Unesi odgovarajući broj od 1 do 4:");
        odabir = ulaz.nextInt();
        switch (odabir){
            case 1: System.out.println("Odabran je tel. iPhone!"); break;
            case 2: System.out.println("Odabran je tel. Samsung!"); break;
            case 3: System.out.println("Odabran je tel Motorola!"); break;
            case 4: System.out.println("Hvala što kupujete kod nas."); break;
            default: System.out.println("Odaberi broj od 1 do 4.");
        }
    }while (odabir != 4);
}
```

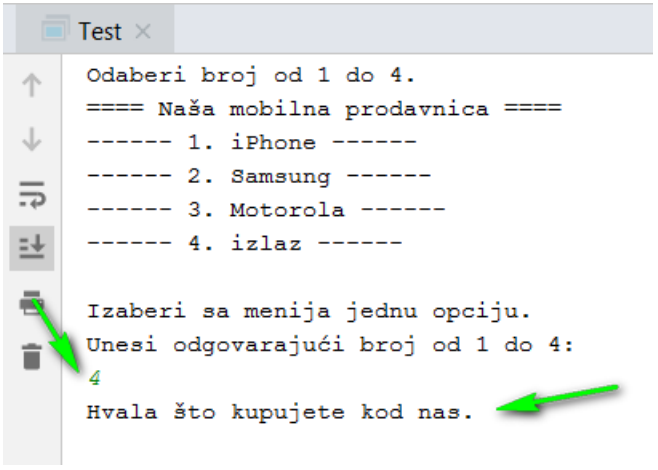
Прављење "менија"

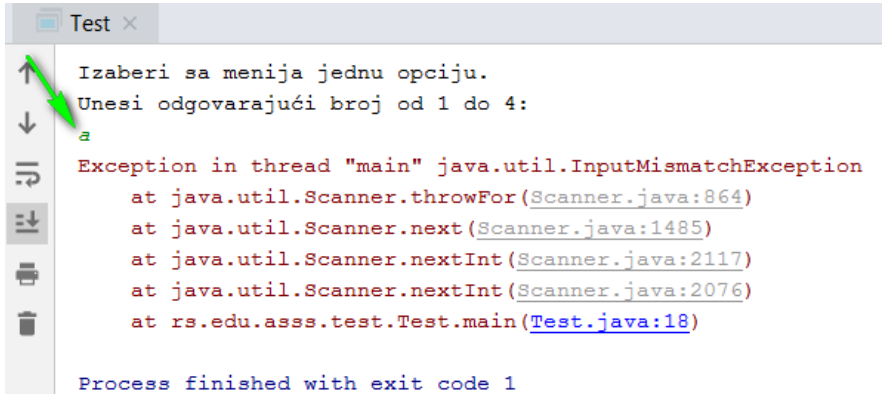












Литература

- Градиво изложено у другом и трећем предавању "покрива" следећа поглавља из препоручене литературе (дате у уводном предавању):
- Поглавље 3
- Поглавље 4
- Поглавље 5 – део 5.1 и 5.2

https://singipedia.singidunum.ac.rs/izdanje/40716-osnove-java-programiranja

- Ако пратите препоручене видео лекције онда градиво које смо до сада обрадили "покривају" првих 5 видео предавања.

https://www.youtube.com/playlist?list=PL-UTrxF0y8kK49N01V5ttb2Xaua7JfyXu