



Академија струковних
студија Шумадија
одсек у Крагујевцу

Увод у програмирање

Презентација 2

Академија струковних студија Шумадија

Одсек у Крагујевцу

Студијски програм Информатика

Крагујевац, 2020. година



Подсетник

- **Case Sensitivity** – у Јави постоји разлика између малих и великих слова.
- Имена пакета - прво слово у имену пакета је мало, свака следећа реч почиње великим словом. Препорука је да имена пакета носе обрнуто име домена: **rs.edu.asss.uup**
- Имена променљивих – прво слово у имену променљиве је мало, свака следећа реч почиње великим словом: **indeks, brojIndeksa**
- Имена класа – прво слово у имену класе великим почетним словом. Ако користимо више речи свака наредна реч почиње великим словом: **class Automobil, class MojAutomobil, ...**
- Имена метода – прво слово у имену методе је мало, свака следећа реч почиње великим словом – **površina(), izračunajPovršinu()**



Први програм (1)

- Извршавање Јава програма почиње позивањем методе **main**, која је неизоставни део сваког Јава програма.
- Скраћеница у IntelliJ IDEA за ауто-комплетирање методе **main** је **psvm**.

```
package zadatak_1;  
public class Main {  
    public static void main(String[] args) {  
        System.out.println("Pozdrav");  
    }  
}
```

- Овај програмски код илуструје базичне елементе Јава програм.
- Пажња у даљем излагању ће бити усмерена само ка битним елементима кода, без икакве дубље анализе истог.



Први програм (2)

- О Јава пакетима ће бити речи касније.
- **public class** - ово је основна и полазна тачка нашег кода. Сваки Јава програм мора имати најмање једну класу.
- За декларацију класе користи се кључна реч **class**, а након тага се задаје име класе.
- Класа може имати било које име (дозвољена имена у Јави смо већ помињали), док се тело класе ограничава блоком великих заграда **{...}**.

```
public class Main {  
    ...  
}
```

- Метод **main** чини класу главном класом у програму и компајлер прво чита оно што пише у методи **main**.



Први програм (3)

- Значи, запамтите веома је важно, извршавање Јава програма почиње позивом **main** методе класе.
- У нашем првом програму декларација методе **main** садржи:
 - **public** - означава да је метода глобално доступна у оквиру целог пројекта.
 - **static** - означава да је метода класног типа, не морамо да имамо инстанцу тј. објекат да би смо користили овакву методу.
 - **void** - означава да метода нема повретну вредност, методе које морају да врате повратну вредност обавезно садрже наредбу **return**.
 - *main* - представља име методе.
 - *String[]* - тип улазног аргумента методе, у овом случају је то низ „стрингова“.
 - *args* - име за улазни аргумент методе, у нашем случају низ „стрингова“.
- О наведеним појмовима ће бити накнадно речи у наредним часовима.



Први програм (4)

- Прва наредба са којом сте се упознали је:

```
System.out.println("Pozdrav");
```

- Она садржи:
 - `System` – стандардна класа која се налази у пакету ***java.lang***, који је иницијално укључен у наш пројекат.
 - ***out*** - објекат класе ***PrintStream*** која се такође налази у пакету ***java.lang***, који је иницијално укључен у наш пројекат.
 - `println()` - представља методу класе ***PrintStream*** која има „задатак“ да испише аргументе који се налазе у ***()***. У нашем примеру је то један „string“, тј. низ карактера „Поздрав“.
- О наведеним појмовима ће такође бити накнадно речи у наредним часовима.



Типови података

- Јава је строго типизиран језик, помињали смо ово али треба поновити.
- Свака променљива има свој тип, а сваки тип је строго дефинисан.
- У Јави није дозвољено аутоматско наметање типова, нити претварање некомпатибилних типова.
- Јавин компајлер проверава све изразе и параметре чиме се утврђује да ли су сви типови међусобно сагласни.
- Свако неслагање типова представља грешку која се мора исправити да би компајлер завршио превођење.
- Нпр. грешке у иницијализацији променљиве:
 - `int napon = 12.5;`
 - `double koeficijent = 100; // компајлер ће ово да „прогута“`



Променљиве

- Променљива је резервирано место за чување вредности одређеног типа: "stringa", броја или неког другог податка.
- Свака променљива има своје име (не сме да користи резервисану реч) које је познато и под називом идентификатор.
- Пре него што почнете да користите променљиву, морате је декларисати – задати јој тип и име.
- Да поновимо још једном, у Јави је дефинисано осам простих типова података:
 - целобројни тип: **byte**, **short**, **int** и **long**
 - реални тип или тип са покретним зарезом: **float** и **double**
 - знаковни тип: **char**, и
 - логички тип: **boolean**



Прости типови променљивих

Тип податка	Величина (бит)	Опсег вредности
long	64 бита	-9223372036854775808 до 9223372036854775807
int	32 бита	-2147483648 до 2147483647
short	16 бита	-32768 до 32767
byte	8 бита	-128 до 127
double	64 бита	чува вредности до 15 децималних цифара
float	32 бита	чува вредности 6 до 7 децималних цифара
char	16 бита	од 0 до 65536, нема негативних вредности
boolean	-	true или false (тачно или нетачно)



Класни типови променљивих

- За разлику од примитивних типова података који су подразумевано "уграђени" у Јава програмски језик, класне типове мора корисник сам да дефинише.
- Специјални случајеви су класа **String** као и омотачи (*Wrapper*) класе примитивних типова података (сваки примитивни тип има омотач класу).
- Јавин тип за знаковне податке, **String**, није прост тип, међутим он није ни обичан низ знакова.
- Тип **String** дефинише објекат и да би смо га у потпуности разумели и описали морамо да разумемо концепте објектно оријентисаног програмирања.
- За сада ћемо класу **String** користити подразумевано за знаковне типове података.



Променљиве – пример 1

Зашто је вредност у испису на конзоли негативан број?

Зашто се јавила грешка (**Exception**) у извршавању овог програма?

```
package rs.edu.asss.uup;
public class Main {
    public static void main(String[] args) {
        int i = 1000000;
        System.out.println(i * i);
        long l = i;
        System.out.println(l * l);
        System.out.println(20296 / (l - i));
    }
}
```



Променљиве – пример 1

```
Run: Main (1) x
"C:\Program Files\Java\jdk1.8.0_161\bin\java.exe" ...
-727379968
10000000000000
Exception in thread "main" java.lang.ArithmeticException: / by zero
    at rs.edu.asss.uup.Main.main(Main.java:8)

Process finished with exit code 1
```

- Негативан број се исписао јер смо премашили опсег дозвољених вредности за променљиву типа `int`. У методи `println()` смо вредност коју носи променљива типа `int` квадрирали (`i * i`) чиме смо добили број који не може да „стане“ у дозвољени опсег променљиве `int`.
- Грешка која се појавила је последица недозвољене операције дељење броја са нулом. **Како смо добили вредност нула?**



Променљиве – пример 2

Како смо помоћу броја 88 добили латинични карактер X?

```
package rs.edu.asss.uup;
public class Main {
    public static void main(String[] args) {
        char znak1, znak2;
        znak1 = 88; // код за слово X
        znak2 = 'Y';
        System.out.print("Ispis slova na konzoli: ");
        System.out.println(znak1 + " " + znak2);
    }
}
```



Променљиве – пример 2

```
Run: Main (1) x
"C:\Program Files\Java\jdk1.8.0_161\bin\java.exe" ...
Ispis slova na konzoli: X Y
Process finished with exit code 0
```

- Стандардни скуп знакова, познат као **ASCII** (помињали смо га на предавањима из ОИТ) заузима опсег вредности од 0 до 127, а проширени скуп 8-битних карактера од 0 до 255.
- Ако погледамо табелу: <http://www.asciitable.com/> видећемо да децимална вредност 88 одговара знаку латиничном знаку X.
- Да ли то значи да се на карактере из табеле могу применити аритметичке операције (нпр. сабирање)? Одговор је ДА.



Променљиве – пример 2

Ако у седмом реду додамо једну наредбу, добијамо следећи резултат:

```
package rs.edu.asss.uup;
public class Main {
    public static void main(String[] args) {
        char znak1, znak2;
        znak1 = 88; // КОД ЗА СЛОВО X
        znak2 = 'Y';
        znak1 +=1;
        System.out.print("Ispis slova na konzoli: ");
        System.out.println(znak1 + " " + znak2);
    }
}
```

```
Run: Main (1) x
"C:\Program Files\Java\jdk1.8.0_161\bin\java.exe" ...
Ispis slova na konzoli: Y Y
Process finished with exit code 0
```



Методе `println()` И `print()`

- Стандардни излаз, конзола, је пријемник на који програм може да шаље информације (текст).
- Подржавају га сви уобичајени оперативни системи.
- Јава нуди посебан објект `System.out` за рад са стандардним излазом.
- Често ћемо га користити за штампање нечега помоћу метода `println()` и `print()`.
- Разлика између ова два метода је у томе што `println()` метод штампа информације на конзоли и премешта курсор у нови ред, док метод `print()` ради исту ту операцију али курсор остаје на крају одштампане информације.
- Ово је илустровано следећим примерима:



Методе `println()` И `print()`

- Није практично писати ћирицом приликом програмирања али је лепо знати да у Јави и то може.

```
System.out.println("Ја ");  
System.out.println("знам ");  
System.out.println("Јаву ");  
System.out.println("добро.");
```



```
Run: Main (1) x  
"C:\Program Files\Java\jdk1.8.0_161\bin\java.exe" ...  
Ја  
знам  
Јаву  
добро.
```

- Можемо исписивати и празан ред када нам је то потребно (ради читког приказа текста у конзоли):

```
System.out.println("Јава је популаран програмски  
језик.");  
System.out.println(); // ова наредба штампа празну линију  
System.out.println("Користи се широм света.");
```

```
Run: Main (1) x  
"C:\Program Files\Java\jdk1.8.0_161\bin\java.exe" ...  
Јава је популаран програмски језик.  
  
Користи се широм света.
```



Методе `println()` И `print()`

- Ако први пример за штампање испишемо помоћу методе `print()` добићемо следећи резултат:

```
System.out.print("Ја ");  
System.out.print("знам ");  
System.out.print("Јаву ");  
System.out.print("добро.");
```

The screenshot shows a Java IDE window titled "Main (1) x". The console output displays the text "Ја знам Јаву добро." with spaces between the words, matching the output of the `print()` method calls. Below the output, it says "Process finished with exit code 0".

- Приметите да је текст у конзоли „лепо“ исписан, да смо добили размаке између речи. То морамо сами да уређујемо помоћу ове две наредбе тако што ћемо између знака наводника да остављамо простор испред или иза карактера које желимо да прикажемо у конзоли.
- Како ћемо ове две наредбе користити у свим програмима док учимо Јаву, најчешће комбинујући обе, требало би да запамтите и разлике између њих.



Оператори

- Јавини оператори се углавном могу поделити у четири групе:
 - аритметички оператори,
 - оператори над битовима,
 - оператори поређења и
 - логички оператори.
- Постоје и додатни оператори посебне намене нпр. ->
- Аритметички оператори се користе у математичким изразима на исти начин као што се користе и у алгебри.
- Операнди аритметичких оператора морају да буду нумеричког типа.
- Аритметичке операторе не можемо да применимо на податке типа **boolean** али можемо на податке типа **char**, као што смо то видели и у примеру.



Аритметички оператори

- Књига: Јава комплетан приручник (JDK 9)

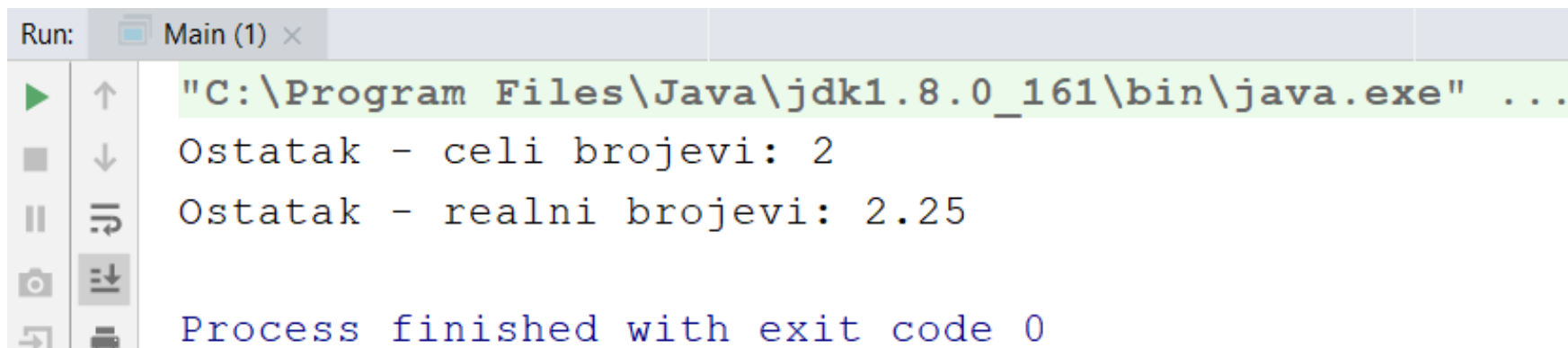
Operator	Rezultat
+	Sabiranje
-	Oduzimanje (takođe i unarni minus)
*	Množenje
/	Deljenje
%	Modulo – ostatak pri deljenju
++	Inkrementiranje
+=	Dodeljivanje uz sabiranje
- =	Dodeljivanje uz oduzimanje
*=	Dodeljivanje uz množenje
/=	Dodeljivanje uz deljenje
%=	Dodeljivanje uz računanje ostatka pri deljenju
--	Dekrementiranje



Аритметички оператори

- Операторе сабирања, одузимања, множења и дељења користимо на исти начин као што би смо их користили у аритметици, тако се и понашају.
- Остатак при дељењу, оператор % може да се примени на целе и реалне бројеве.

```
int x = 42;  
double y = 42.25;  
System.out.println("Ostatak - celi brojevi: " + x%10);  
System.out.println("Ostatak - realni brojevi: " + y%10);
```



```
Run: Main (1) x  
"C:\Program Files\Java\jdk1.8.0_161\bin\java.exe" ...  
Ostatak - celi brojevi: 2  
Ostatak - realni brojevi: 2.25  
Process finished with exit code 0
```



Аритметички оператори

- Операторе доделе представљају комбинују аритметичку операцију са операцијом доделе:

```
int x = 4; // додељивање вредности  
x = x + 6; // вредност за x је сада 10  
x += 2; // вредност за x је сада 12 (била је 10)
```

- Инкрементирање (оператор ++) и декрементирање (оператор --) повећавају или смањују за 1 (у зависности који користите).
- Обратите пажњу да није исто ако ови оператори стоје испред или иза променљиве. Они заправо раде оно за шта су и намењени, повећавају или смањују вредност за 1, међутим, разликују се начини извођења у зависности где ови оператори стоје (испред или иза променљиве).

`x++;` или `++x;` (ово ће бити важно нпр. код петљи)



Оператори над битовима

- За разумевање како рачунар ради са битовима, како се они померају у леву или десну страну и испуњавају нулама потребно је издвојити прилично времена. За сада је довољно да знате да ови оператори постоје.

Operator	Rezultat
+	Sabiranje
-	Oduzimanje (takođe i unarni minus)
*	Množenje
/	Deljenje
%	Modulo – ostatak pri deljenju
++	Inkrementiranje
+=	Dodeljivanje uz sabiranje
- =	Dodeljivanje uz oduzimanje
*=	Dodeljivanje uz množenje
/=	Dodeljivanje uz deljenje
%=	Dodeljivanje uz računanje ostatka pri deljenju
--	Dekrementiranje

Оператори за поређење

- Оператори за поређење утврђују однос једног операнда према другом, тј. утврђују једнакост или релативан редослед операнада.

Operator	Rezultat
==	Jednako
!=	Različito
>	Veće od
<	Manje od
>=	Veće od ili jednako
<=	Manje od ili jednako

- Резултат ових операција је логичка вредност (тип **boolean**). Најчешће их користимо у наредбама за контролу тока програма и наредбама за понављање (петље).

Логички оператори

- Ови оператори делују искључиво на операнде типа **boolean**.

Operator	Rezultat
&	Logička konjunkcija (AND – i)
	Logička disjunkcija (OR – ili)
^	Logička isključiva disjunkcija (isključivo OR ili XOR – isključivo ili)
	Kratkospojena disjunkcija
&&	Kratkospojena konjunkcija
!	Logička unarna negacija (NOT – ne)
&=	Dodeljivanje uz logičku konjunkciju
=	Dodeljivanje uz logičku disjunkciju
^=	Dodeljivanje uz logičku isključivu disjunkciju
==	Jednako
!=	Različito
?:	Ternarni uslovni operator



Оператор за доделу вредности

- Оператор за доделу вредности користимо још од првих примера (не буквално, за исписивање поруке "Поздрав" га нисмо користили).
- Оператор за доделу вредности је знак једнакости, **=**.
- Општи облик његове примене изгледа овако:

тип имеПроменљиве = вредност (израз) ;

int x = 4; // додељивање вредности

int x = 4 * 6; // додељивање вредности изразом

int x, y, z; // декларација променљивих

x = y = z = 4; // додељивање вредности



Конверзија типова

- Често се вредност променљиве једног типа додељује променљивој другог типа уколико су типови међусобно компатибилни.
- У Јави постоје два типа конверзије: **аутоматска** и **експлицитна**.
- Ако су типови међусобно компатибилни Јава ће назначену конверзију одредити аутоматски.
- Нпр. увек је могуће вредност типа **int** доделити типу **long**.
- Међутим, пошто нису сви типови променљивих компатибилни, није свака конверзија једног типа у други дозвољена.
- Нпр. не постоји аутоматска конверзија типа **double** у тип **byte**.
- То је ипак могуће применити али морамо да употребимо експлицитну конверзију, уз могућ (не и обавезан) губитак одређених информација.



Експлицитна конверзија

- Експлицитна конверзија је операција која конвертује вредност једног типа податка у вредност податка другог типа.
- Конверзија типа са мањим опсегом вредности у тип који има већи опсег вредности, је позната као **проширење типа**.
- Конверзија типа са већим опсегом вредности са типом који има мањи опсег вредности, је позната као **сужавање типа**.
- Јава ће аутоматски проширити неки тип, али **сужавање типа морате ви експлицитно да извршите**.
- Примена експлицитне конверзије се врши уз помоћу кастовања (енгл. casting) што је приказано следећим примером.



Експлицитна конверзија - пример

```
public class EksplicitnaKonverzija {  
  
    public static void main(String[] args) {  
  
        byte b;  
        int i = 257;  
        double d = 323.142;  
  
        System.out.println("Konverzija tipa int u byte: ");  
        b = (byte) i;  
        System.out.print("Promenljiva i: " + i);  
        System.out.println(". Promenljiva b: " + b + ".");  
  
        System.out.println("Konverzija tipa double u int: ");  
        i = (int) d;  
        System.out.print("Promenljiva d: " + d);  
        System.out.println(". Promenljiva i: " + i + ".");  
  
        System.out.println("Konverzija tipa double u byte: ");  
        b = (byte) d;  
        System.out.print("Promenljiva d: " + d);  
        System.out.println(". Promenljiva b: " + b + ".");  
    }  
}
```

Када се вредност 257 експлицитно конвертује у променљиву типа **byte**, резултат је $i \% 256$ (опсег вредности за тип **byte**, од -128 до 127 укључујући и нулу), **тј. број 1**. Код конверзије типа **double** у тип **int** **децимални део се губи**. И на крају, када се тип **double** конвертује у тип **byte**, **његов децимални део се губи** (јер је **byte** целобројна вредност) и као резултат конверзије исписује се $d \% 256$, **тј. број 67**.



Литература

- Градиво изложено у прва два предавања "покрива" следећа поглавља из препоручене литературе (дате у уводном предавању):
- Поглавље 1
- Поглавље 2 (прескочити, није обавезно читати главу 2.2)
- Поглавље 3

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

- Ако пратите препоручене видео лекције онда градиво које смо до сада обрадили "покривају" прва 4 видео предавања.

<https://www.youtube.com/playlist?list=PL-UTrxFOy8kK49N01V5ttb2Xaua7JfyXu>