

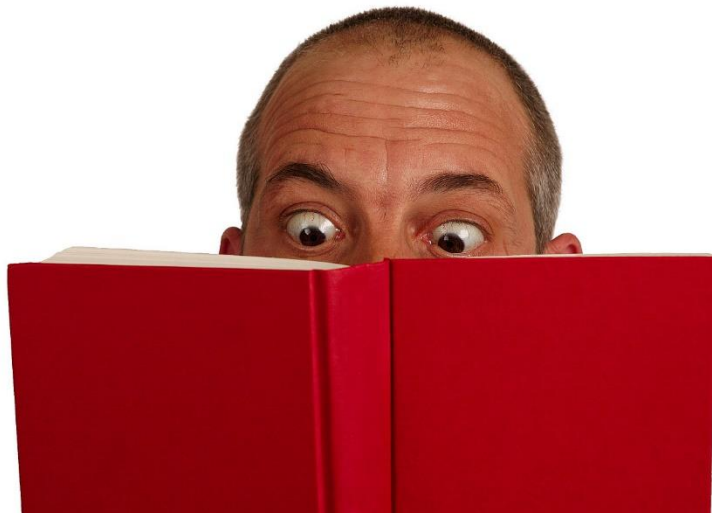


# 8. prednáška (15.11.2021)

## Čítanie zo súborov

a

pár vecí navyše



**Maven**<sup>TM</sup>





# Už vieme...

## ● Výnimky

- **špeciálne objekty**
- vznikajú vo **výnimočných stavoch**, keď nejaké metódy nemôžu prebehnúť štandardným spôsobom alebo nevedia vrátiť očakávanú hodnotu
- sú inštanciami rôznych „výnimkových“ tried
  - ArithmeticException
  - NullPointerException
  - NumberFormatException
  - FileNotFoundException
  - ...



# Odchytávame výnimky (1)

```
try {  
    // blok príkazov, kde môžu vzniknúť výnimky,  
    // ktoré si trúfame odchytiť  
} catch (TypVýnimky1 e) {  
    // vysporiadanie sa s daným typom výnimky  
} catch (TypVýnimky2 e) {  
    // vysporiadanie sa s daným typom výnimky  
} finally {  
    // kód, ktorý sa vykoná VŽDY!  
}
```



# Odchytávame výnimky (2)

Riešenie pre prípad rovnakej reakcie na rôzne triedy výnimiek.

```
try {  
    // blok príkazov, kde môžu vzniknúť výnimky,  
    // ktoré si trúfame odchytiť  
} catch (TypVýnimky1|TypVýnimky2 e) {  
    // vysporiadanie sa s daným 2 typmi výnimiek  
} finally {  
    // kód, ktorý sa vykoná VŽDY!  
}
```

Objekty akej triedy referencuje e?



# Trieda File

- Objekty triedy File uchovávajú **cestu** k súboru alebo adresáru
  - tento súbor alebo adresár **nemusí reálne existovať!**
- Analógia:



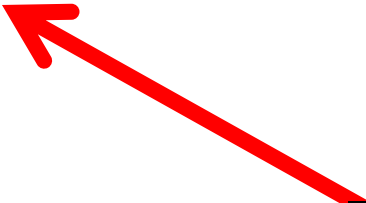
Taká adresa neexistuje,  
ale možno časom bude a  
možno na nej bude aj bývať  
Jožko Turtlák





# Práca s textovými súbormi

- Práca s **obsahom (nielen) súborov** sa vždy skladá z 3 krokov:
  - **otvorenie** súboru, ktoré sa udeje pri vzniku nejakého čítača alebo zapisovača
  - **práca s obsahom** súboru (teda čítanie alebo zapisovanie)
  - **zatvorenie** súboru



Čo sme otvorili,  
musíme **VŽDY** zatvoriť.

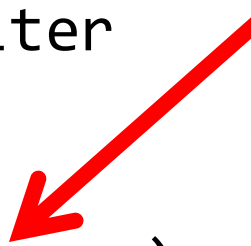


# Zápis do textového súboru

## ● Kto?

- objekty triedy `PrintWriter`

## ● Ako ho vytvoriť?

- `new` `PrintWriter(`  `);`
  - ak súbor neexistuje, vytvorí sa
  - ak súbor existuje, jeho obsah sa zmaže

Objekt triedy `File` s cestou k súboru, kam zapisujeme.

## ● Ako písať?

- cez metódy `println` a `print` - presne ako ich má `System.out`

## ● Ako zatvoriť súbor otvorený na zapisovanie?

- metóda `close`



# Schéma práce s PrintWriter-om

```
File subor = new File("C:\\adresare\\subor.txt");
PrintWriter pw = null;
try {
    pw = new PrintWriter(subor);

    // píšeme do pw

} catch (FileNotFoundException e) {
    System.err.println("Súbor " +
        subor.getName() + " som nenašiel");
} finally {
    if (pw != null)
        pw.close();
}
```





# Schéma práce s PrintWriter-om

```
File subor = new File("C:\\adresare\\subor.txt");  
try (PrintWriter pw = new PrintWriter(subor)) {  
  
    // píšeme do pw  
  
} catch (FileNotFoundException e) {  
    System.err.println("Súbor " +  
        subor.getName() + " som nenašiel");  
}
```

Od Javy verzie 7

Zatvorenie sa zrealizuje automaticky



# Zapisujeme do súboru

```
public void saveNumbersToFile(File subor, int[] pole) {
    PrintWriter pw = null;
    try {
        pw = new PrintWriter(subor);
    }
}
```



náš program

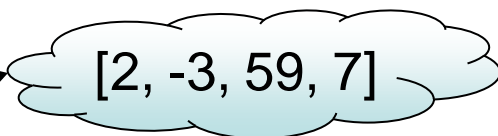
objekty

reálny svet

pole



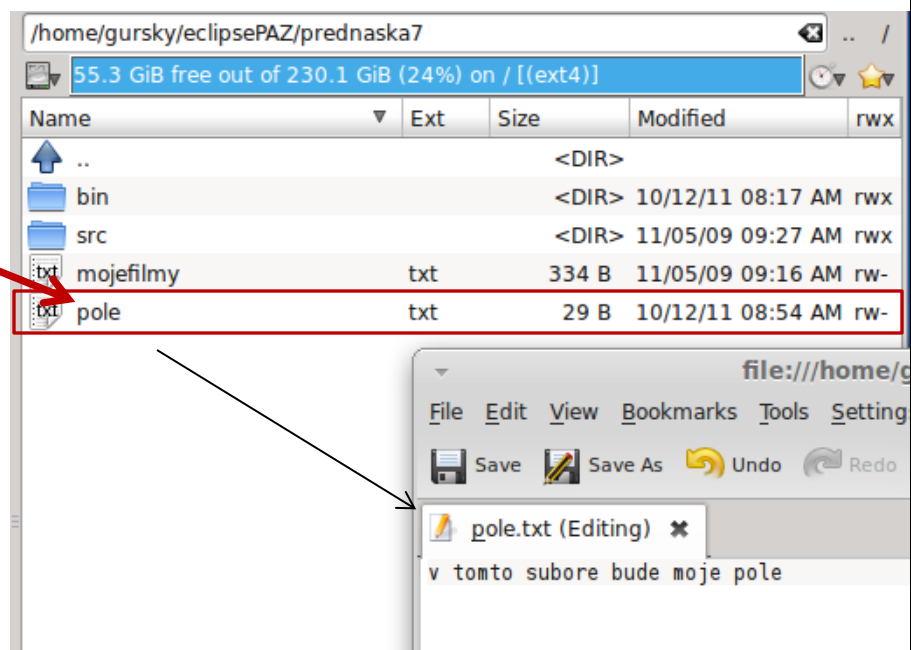
[2, -3, 59, 7]



subor



pw





# Zapisujeme do súboru

```
public void saveNumbersToFile(File subor, int[] pole) {
    PrintWriter pw = null;
    try {
        pw = new PrintWriter(subor);
```



náš program

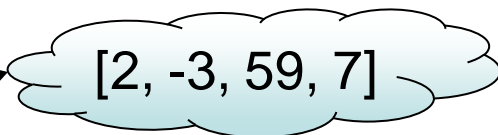
objekty

reálny svet

pole



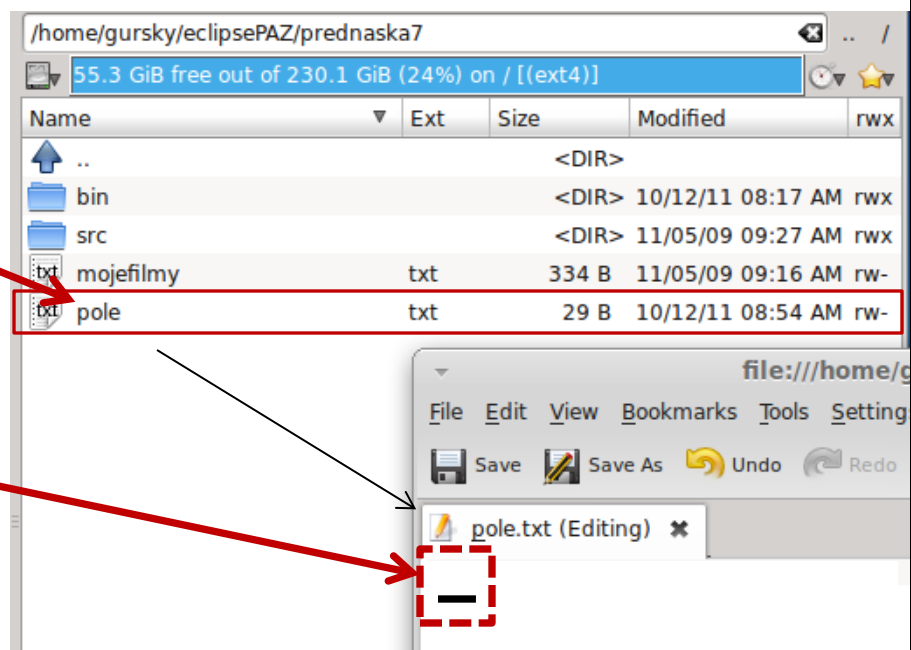
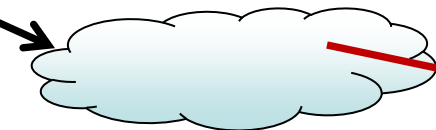
[2, -3, 59, 7]



subor



pw





# Zapisujeme do súboru

```
public void saveNumbersToFile(File subor, int[] pole) {
    PrintWriter pw = null;
    try {
        pw = new PrintWriter(subor);
        pw.println(pole.length);
        for (int i = 0; i < pole.length; i++) {
            pw.print(pole[i]+" ");
        }
    }
}
```



náš program

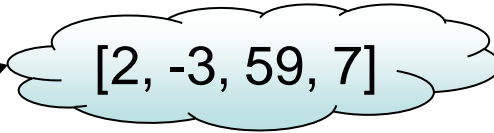
objekty

reálny svet

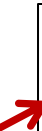
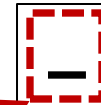
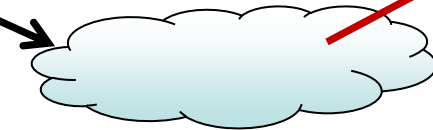
pole



[2, -3, 59, 7]



pw





# Zapisujeme do súboru

```
public void saveNumbersToFile(File subor, int[] pole) {
    PrintWriter pw = null;
    try {
        pw = new PrintWriter(subor);
        pw.println(pole.length);
        for (int i = 0; i < pole.length; i++) {
            pw.print(pole[i]+" ");
        }
    }
}
```



náš program

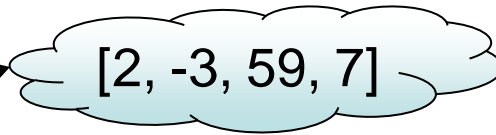
objekty

reálny svet

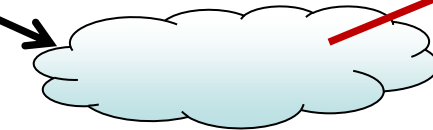
pole



[2, -3, 59, 7]



pw



4





# Zapisujeme do súboru

```
public void saveNumbersToFile(File subor, int[] pole) {
    PrintWriter pw = null;
    try {
        pw = new PrintWriter(subor);
        pw.println(pole.length);
        for (int i = 0; i < pole.length; i++) {
            pw.print(pole[i]+" ");
        }
    }
}
```



náš program

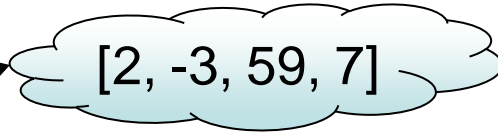
objekty

reálny svet

pole



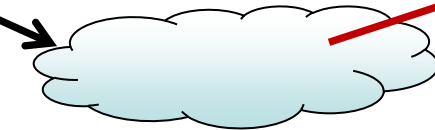
[2, -3, 59, 7]



pw

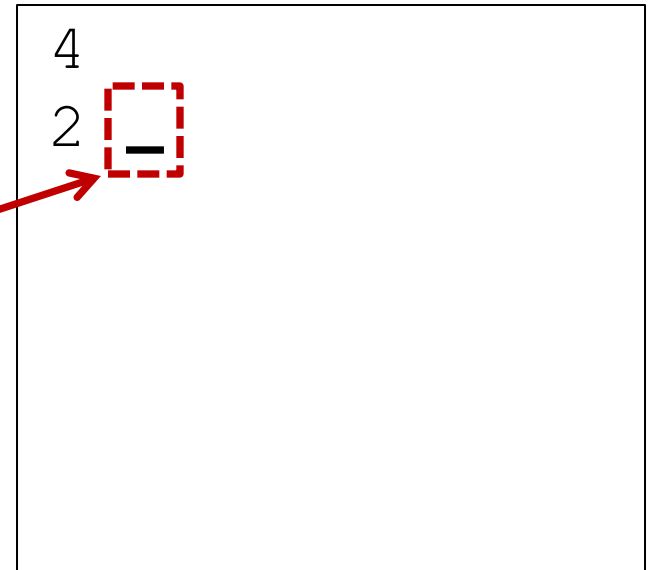


i



4

2





# Zapisujeme do súboru

```
public void saveNumbersToFile(File subor, int[] pole) {
    PrintWriter pw = null;
    try {
        pw = new PrintWriter(subor);
        pw.println(pole.length);
        for (int i = 0; i < pole.length; i++) {
            pw.print(pole[i]+" ");
        }
    }
}
```



náš program

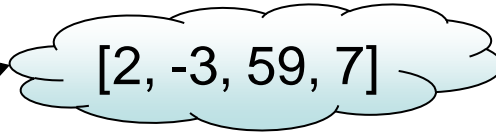
objekty

reálny svet

pole



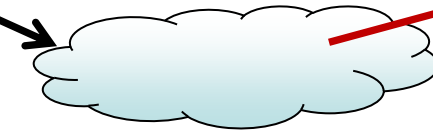
[2, -3, 59, 7]



pw



i



4

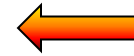
2 -3





# Zapisujeme do súboru

```
public void saveNumbersToFile(File subor, int[] pole) {
    PrintWriter pw = null;
    try {
        pw = new PrintWriter(subor);
        pw.println(pole.length);
        for (int i = 0; i < pole.length; i++) {
            pw.print(pole[i]+" ");
        }
    }
}
```



náš program

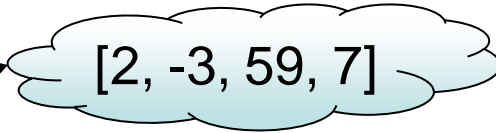
objekty

reálny svet

pole



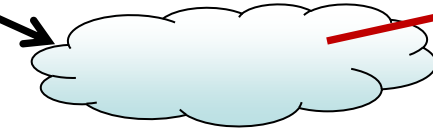
[2, -3, 59, 7]



pw



i



4

2 -3 59 7








# Zapisujeme do súboru

```
public void saveNumbersToFile(File subor, int[] pole) {  
    try (PrintWriter pw = new PrintWriter(subor)) {  
        pw.println(pole.length);  
        for (int i = 0; i < pole.length; i++) {  
            pw.print(pole[i]+" ");  
        }  
    } catch (FileNotFoundException e) {  
        System.err.println("Súbor " + subor.getName()  
            + " sa nenašiel");  
    }  
}
```



# Čítanie z textového súboru

- Kto?
  - objekty triedy Scanner
- Ako ho vytvoriť?
  - `new Scanner(                    );`
- Ako čítať?
  - `nextXYZ` a `hasNextXYZ` metódy
- Ako zatvoriť súbor otvorený na čítanie?
  - metóda `close`



Objekt triedy `File` s cestou k súboru, z ktorého čítame.



## Schéma práce so Scanner-om

```
File subor = new File("C:/adresare/subor.txt");
Scanner scanner = null;
try {
    scanner = new Scanner(subor);
    // čítame zo scannera
} catch (FileNotFoundException e) {
    System.out.println("Súbor " +
        subor.getName() + " som nenašiel");
} finally {
    if (scanner != null) {
        scanner.close();
    }
}
```



## Schéma práce so Scanner-om

```
File subor = new File("C:/adresare/subor.txt");
try (Scanner scanner = new Scanner(subor)) {

    // čítame zo scannera

} catch (FileNotFoundException e) {
    System.out.println("Súbor " +
        subor.getName() + " som nenašiel");
}
```



# Svet očami Scanner-u

- Scanner pozerá len dopredu
  - čítací kurzor sa posúva len smerom ku koncu súboru
- **oddelovače** vs. tokeny



A Scanner breaks its input into tokens using a delimiter pattern, which by default matches whitespace.

Default: oddeľovač (delimiter) je ľubovoľná postupnosť za sebou idúcich „bielych“ / „prázdnych“ znakov (whitespace).



# Oddelovače a tokeny

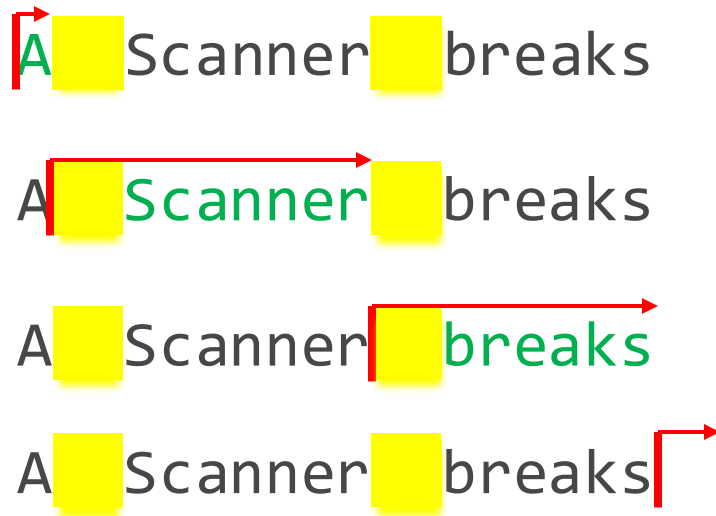
Dobrý deň**STOP**Takto vyzerá telegram**STOP**.

- Oddelovač: **STOP**
- Tokeny:
  - Dobrý|deň
  - Takto|vyzerá|ntelegram
  - .

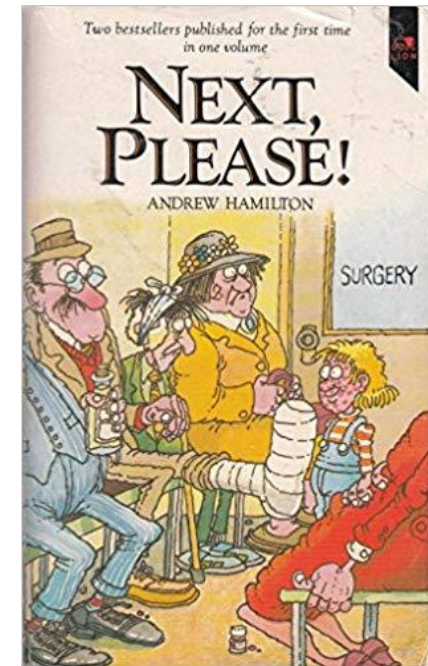


# Scanner

- **hasNext()** - vráti, či sa niekde pred čítacím kurzorom nachádza **nejaký token**
- **next()** - vráti najbližší token pred čítacím kurzorom a kurzor sa posunie na prvý znak za prečítaným tokenom



Demo





# VisualScanner vs. Maven

Maven repozitár: <https://search.maven.org/>

pom.xml:

```
<dependency>
```

```
    <groupId>sk.upjs</groupId>
```

```
    <artifactId>visual-scanner</artifactId>
```

```
    <version>1.0.0</version>
```

```
</dependency>
```

Maven Update





# Scanner a jeho filozofia

- Filozofia práce:
  - **boolean** hasNextXXX();
    - vráti **true** práve vtedy, ak sa dá prečítať token a tento token je konvertovateľný na hodnotu typu XXX
  - XXX nextXXX();
    - prečíta token a vráti ho konvertovaný na hodnotu typu XXX



# hasNext... a next...

Overenie výskytu	Prečítanie výskytu	Konvertujeme na
hasNext()	next()	String (slovo)
hasNextLine()	nextLine()	String (riadok)
hasNextInt()	nextInt()	int
hasNextDouble()	nextDouble()	double
hasNextBoolean()	nextBoolean()	boolean

Vráti true, ak máme token a ten je konvertovateľný do príslušného typu.

Ak token je konvertovateľný do príslušného typu, vráti hodnotu.

Inak vyhodí: **InputMismatchException**

Dobrá rada: **nemiešajte** rôzne dvojice!, napr. hasNextLine() a next()



# Čítame zo súboru

- Vytvorme si metódu, ktorá načíta zo zadaného súboru v prvom riadku veľkosť poľa čísiel a v druhom riadku obsah poľa čísiel.



# Čítame zo súboru

```
public int[] nacistajZoSuboruPole(File subor) {
    int[] pole = null;
    Scanner citac = null;
    try {
        citac = new Scanner(subor);
        int pocet = citac.nextInt();
    }
}
```



náš program

objekty

reálny svet

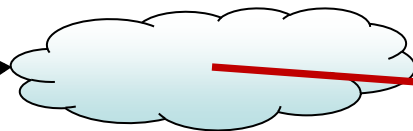
pole

null

subor

citac

null



Name	Ext	Size	Modified	rwx
..	<DIR>			
bin	<DIR>	10/12/11 08:17 AM	rw-	
src	<DIR>	11/05/09 09:27 AM	rw-	
mojefilmy	txt	334 B	11/05/09 09:16 AM	rw-
pole	txt	29 B	10/12/11 08:54 AM	rw-

file:///home/g...  
File Edit View Bookmarks Tools Setting  
Save Save As Undo Redo  
pole.txt (Editing) ✕  
4  
2 -3 59 7



# Čítame zo súboru

```
public int[] nacistajZoSuboruPole(File subor) {
    int[] pole = null;
    Scanner citac = null;
    try {
        citac = new Scanner(subor);
        int pocet = citac.nextInt();
```



náš program

objekty

reálny svet

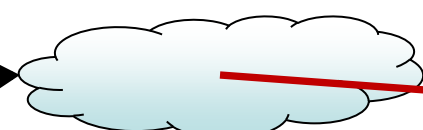
pole



subor



citac



Name	Ext	Size	Modified	rwx
..	<DIR>			
bin	<DIR>	10/12/11 08:17 AM	rw-	
src	<DIR>	11/05/09 09:27 AM	rw-	
mojefilmy	txt	334 B	11/05/09 09:16 AM	rw-
pole	txt	29 B	10/12/11 08:54 AM	rw-

file:///home/g...  
File Edit View Bookmarks Tools Setting  
Save Save As Undo Redo  
pole.txt (Editing) \*  
4  
2 -3 59 7



# Čítame zo súboru

```
public int[] nacistajZoSuboruPole(File subor) {
    int[] pole = null;
    Scanner citac = null;
    try {
        citac = new Scanner(subor);
        int pocet = citac.nextInt();
        int i = 0;
        pole = new int[pocet];
        while (citac.hasNextInt()) {
            pole[i] = citac.nextInt();
            i++;
        }
    }
}
```



náš program

objekty

reálny svet

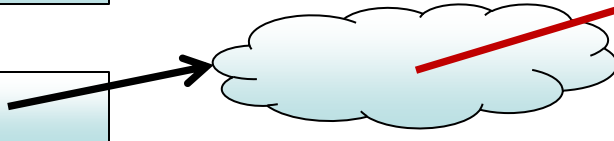
pole

null

citac

4

2 -3 59 7





# Čítame zo súboru

```
public int[] nacistajZoSuboruPole(File subor) {
    int[] pole = null;
    Scanner citac = null;
    try {
        citac = new Scanner(subor);
        int pocet = citac.nextInt();
        int i = 0;
        pole = new int[pocet];
        while (citac.hasNextInt()) {
            pole[i] = citac.nextInt();
            i++;
        }
    }
}
```



naš program

pole

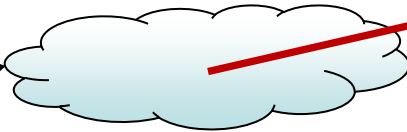
null

citac

pocet

4

objekty



reálny svet

4	-		
2	-3	59	7



# Čítame zo súboru

```
public int[] nacistajZoSuboruPole(File subor) {
    int[] pole = null;
    Scanner citac = null;
    try {
        citac = new Scanner(subor);
        int pocet = citac.nextInt();
        int i = 0;
        pole = new int[pocet];
        while (citac.hasNextInt()) {
            pole[i] = citac.nextInt();
            i++;
        }
    }
}
```



naš program

pole

null

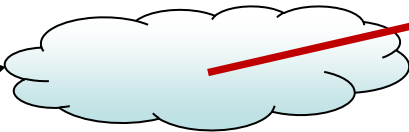
citac

pocet i

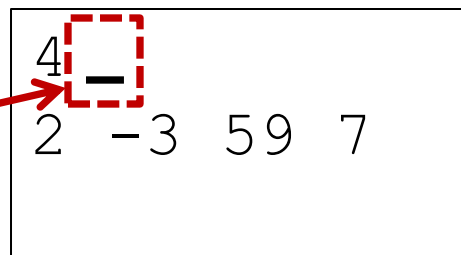
4

0

objekty



reálny svet







# Čítame zo súboru

```
public int[] nacistajZoSuboruPole(File subor) {
    int[] pole = null;
    Scanner citac = null;
    try {
        citac = new Scanner(subor);
        int pocet = citac.nextInt();
        int i = 0;
        pole = new int[pocet];
        while (citac.hasNextInt()) {
            pole[i] = citac.nextInt();
            i++;
        }
    }
}
```



naš program

pole



citac

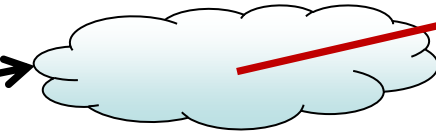


pocet i

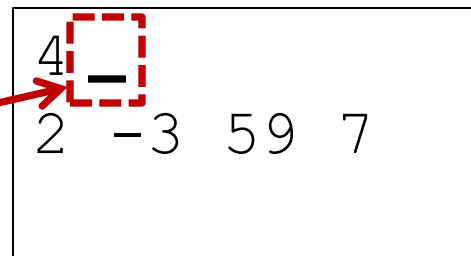


objekty

[ 0, 0, 0, 0 ]



reálny svet





# Čítame zo súboru

```
public int[] nacistajZoSuboruPole(File subor) {
    int[] pole = null;
    Scanner citac = null;
    try {
        citac = new Scanner(subor);
        int pocet = citac.nextInt();
        int i = 0;
        pole = new int[pocet];
        while (citac.hasNextInt()) {
            pole[i] = citac.nextInt();
            i++;
        }
    }
}
```



naš program

pole



citac



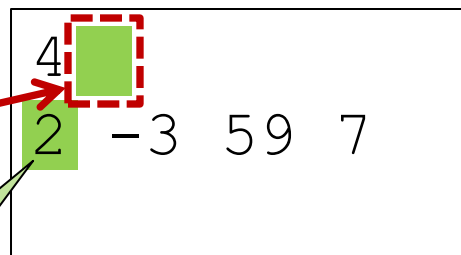
pocet i



objekty

[ 0, 0, 0, 0 ]

reálny svet



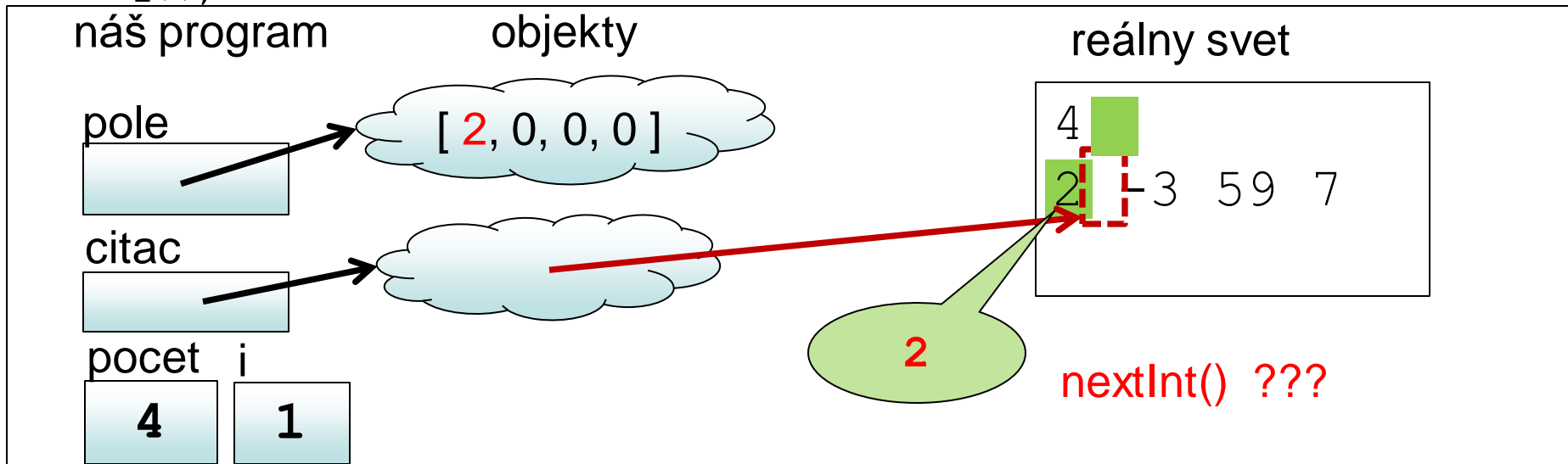
true

hasNextInt() ???



# Čítame zo súboru

```
public int[] nacistajZoSuboruPole(File subor) {
    int[] pole = null;
    Scanner citac = null;
    try {
        citac = new Scanner(subor);
        int pocet = citac.nextInt();
        int i = 0;
        pole = new int[pocet];
        while (citac.hasNextInt()) {
            pole[i] = citac.nextInt();
            i++;
        }
    }
}
```





# Čítame zo súboru

```
public int[] nacistajZoSuboruPole(File subor) {
    int[] pole = null;
    Scanner citac = null;
    try {
        citac = new Scanner(subor);
        int pocet = citac.nextInt();
        int i = 0;
        pole = new int[pocet];
        while (citac.hasNextInt()) {
            pole[i] = citac.nextInt();
            i++;
        }
    }
}
```



naš program

pole



citac



pocet i



objekty

[ 2, 0, 0, 0 ]

reálny svet

4  
2 -3 59 7

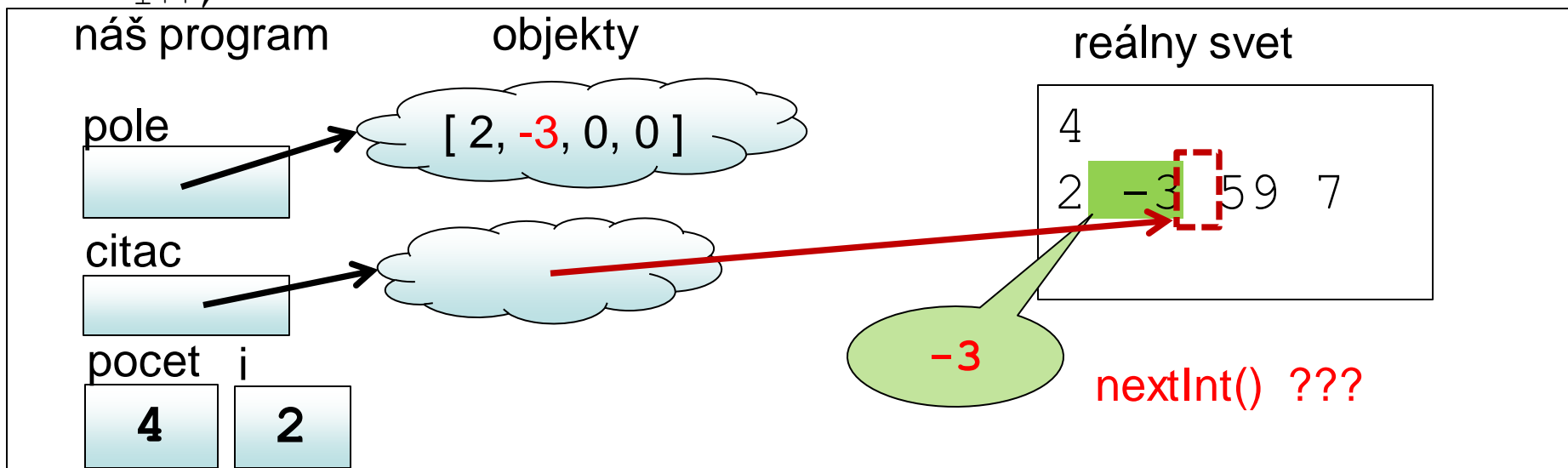
true

hasNextInt() ???



# Čítame zo súboru

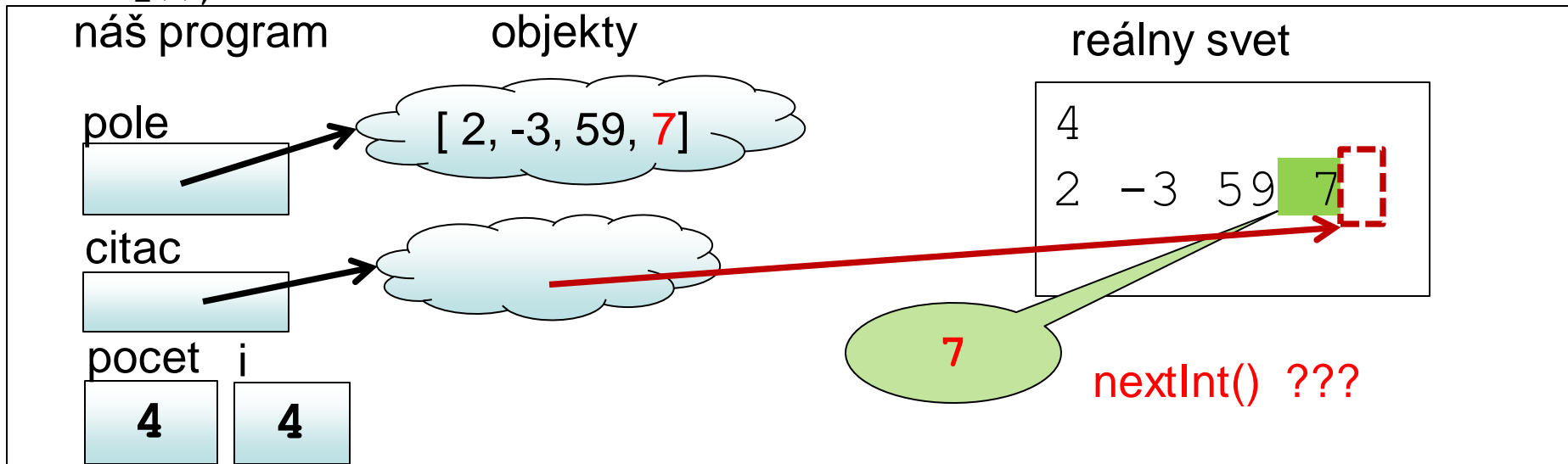
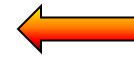
```
public int[] nacistajZoSuboruPole(File subor) {
    int[] pole = null;
    Scanner citac = null;
    try {
        citac = new Scanner(subor);
        int pocet = citac.nextInt();
        int i = 0;
        pole = new int[pocet];
        while (citac.hasNextInt()) {
            pole[i] = citac.nextInt();
            i++;
        }
    }
}
```





# Čítame zo súboru

```
public int[] nacistajZoSuboruPole(File subor) {
    int[] pole = null;
    Scanner citac = null;
    try {
        citac = new Scanner(subor);
        int pocet = citac.nextInt();
        int i = 0;
        pole = new int[pocet];
        while (citac.hasNextInt()) {
            pole[i] = citac.nextInt();
            i++;
        }
    }
}
```





# Čítame zo súboru

```
public int[] nacistajZoSuboruPole(File subor) {
    int[] pole = null;
    Scanner citac = null;
    try {
        citac = new Scanner(subor);
        int pocet = citac.nextInt();
        int i = 0;
        pole = new int[pocet];
        while (citac.hasNextInt()) {
            pole[i] = citac.nextInt();
            i++;
        }
    }
}
```



naš program

pole



citac

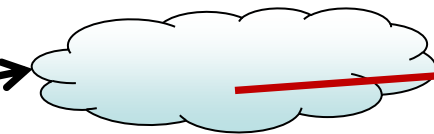
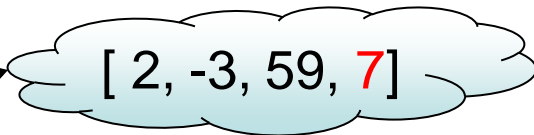


pocet i



objekty

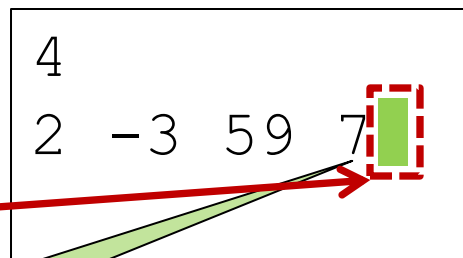
[ 2, -3, 59, 7 ]



reálny svet

4

2 -3 59 7



false

hasNextInt() ???





# Čítame zo súboru

```
public int[] nacistajZoSuboruPole(File subor) {
    int[] pole = null;
    Scanner citac = null;
    try {
        citac = new Scanner(subor);
        int pocet = citac.nextInt();
        int i = 0;
        pole = new int[pocet];
        while (citac.hasNextInt()) {
            pole[i] = citac.nextInt();
            i++;
        }
    } catch (FileNotFoundException e) {
        System.err.println("Súbor " + subor.getName() + " sa nenašiel");
    } finally {
        if (citac != null)
            citac.close();
    }
    return pole;
}
```





# Čítame zo súboru so zdrojmi

```
public int[] nacistajZoSuboruPole(File subor) {  
    int[] pole = null;  
    try (Scanner citac = new Scanner(subor)) {  
        int pocet = citac.nextInt();  
        int i = 0;  
        pole = new int[pocet];  
        while (citac.hasNextInt()) {  
            pole[i] = citac.nextInt();  
            i++;  
        }  
    } catch (FileNotFoundException e) {  
        System.err.println("Súbor " + subor.getName() + " sa nenašiel");  
    }  
    return pole;  
}
```



# Delimiter - oddeľovač

- Default: ľubovoľná postupnosť *whitespace* znakov:
  - napr. `'\t'`, `' '` a `'\n'`
- Dá sa zmeniť metódou `useDelimiter()`
  - parametrom je regulárny výraz (pre fajšmekrov)
- Pre dvojicu `hasNextLine()` a `nextLine()` je oddeľovačom vždy `'\n'`



# Scanner dokáže aj...

- Dokáže čítať textové vstupy:
  - z textového súboru
    - `File subor = new File("D:/vstup.txt");`  
`Scanner scanZoSuboru = new Scanner(subor);`
    - `File subor = new File("D:/vstup.txt");`  
`Scanner scanZoSuboru = new Scanner(subor, "UTF-8");`
  - z konzoly
    - `Scanner scanZKonzoly = new Scanner(System.in);`
  - z reťazca
    - `Scanner scanZRetazca1 = new Scanner("Ahoj Java");`
    - `Scanner scanZRetazca2 = new Scanner("D:\\x.txt");`



# *System.* [in/out/err]

- **System.in** - vstup z klávesnice v konzole
  - reálne využitie: shellové dialógy
- **System.out** - výpis na konzolu
  - už poznáme cez jeho metódy `print()` a `println()`
- **System.err** - chybový výpis na konzolu
  - pracuje sa s ním rovnako ako so `System.out`
  - výpis v Eclipse červeným písmom



# Zapisujeme a čítame double

- PrintWriter - **nezohľadňuje** regionálne nastavenia (Locale)
  - 10.5
- Scanner - **zohľadňuje** regionálne nastavenia
  - 10.5 vs 10,5
- Trik pre Scanner:
  - `scanner.useLocale(Locale.US);`



# try so zdrojmi

```
try (Scanner sc = new Scanner(new File(fn));  
    PrintWriter pw = new PrintWriter(new File(fn))) {  
  
    // práca so Scannerom a PrintWriterom  
  
} catch (FileNotFoundException e) {  
    System.err.println("Nepodarilo sa otvoriť súbor.");  
}
```

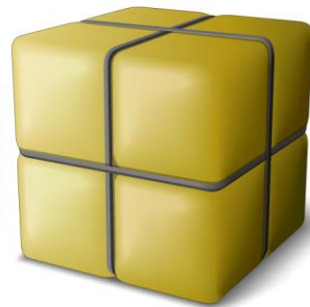
Zatvorenie sa zrealizuje automaticky

- v poradí od posledného vytvoreného objektu Scanner/PrintWriter.



# Balíčky

```
package paz1a.lectures.lecture8;  
  
import java.io.File;  
import java.io.FileNotFoundException;  
import java.io.PrintWriter;  
import java.util.Locale;  
import java.util.Scanner;
```





# Konflikty v názvoch tried

Package Explorer

- lecture1
- lecture2
- lecture3
- lecture4
- lecture5
- lecture6
- lecture7
  - src/main/java
  - src/main/resources
  - JRE System Library [JavaSE-1.8]
  - Maven Dependencies
  - src
  - target
  - pokus.txt
  - pom.xml

Dependencies

Filter:

Dependencies	Dependency Management
<ul style="list-style-type: none"> <li>jpaz2 : 1.1.1</li> <li>visual-scanner : 1.0.0</li> </ul>	
<ul style="list-style-type: none"> <li>Add...</li> <li>Remove</li> <li>Properties...</li> <li>Manage...</li> </ul>	<ul style="list-style-type: none"> <li>Add...</li> <li>Remove</li> <li>Properties...</li> </ul>

To manage your transitive dependency exclusions, please use the [Dependency Hierarchy](#) page.

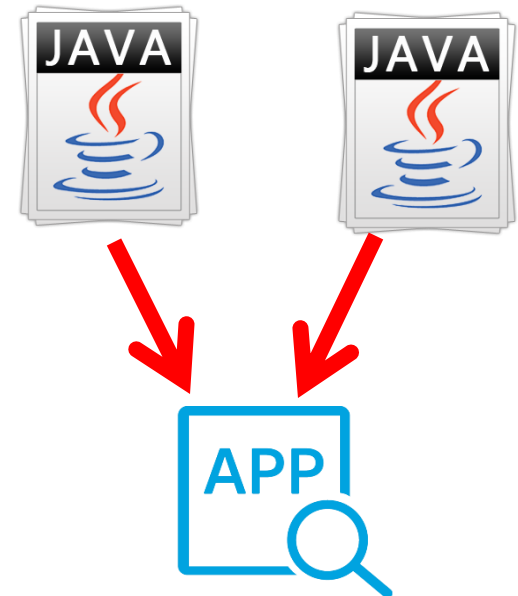
Overview | Dependencies | Dependency Hierarchy | Effective POM | pom.xml





# Konflikty v názvoch tried

- Čo ak v projekte použijeme knižnice (maven artefakty), ktoré obsahujú triedy s rovnakým názvom?
  - Date - na reprezentáciu dátumu a času v Java
  - Date - na reprezentáciu dátumu v databáze
  - Date - na reprezentáciu nejakého rande v zoznamke
  - Menu - ponuka tlačidiel a akcií v aplikácii (File, Edit, View, Help...)
  - Menu - vysúvateľné menu v mobilnej aplikácii
  - Menu - zoznam jedál v reštaurácii





# Riešenie konfliktov



Veľa ľudí ma rovnaké krstné meno (názov triedy).

Na rozlíšenie môžeme použiť priezvisko (package).



# Riešenie konfliktov

PAZ1a2017 - lecture6/src/main/java/paz1a/lectures/lecture6/Farma.java - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer

- lecture1
- lecture2
- lecture3
- lecture4
- lecture5
- lecture6
  - src/main/java
    - paz1a.lectures.lecture6
      - Farma.java
      - Ihrisko.java
      - Launcher.java
      - Lopta.java
      - Piskvorky.java
      - SmartTurtle.java
  - src/main/resources
  - JRE System Library [JavaSE-1.8]
  - Maven Dependencies
  - src
  - target
  - pom.xml
- lecture7
  - src/main/java

```

1 package paz1a.lectures.lecture6;
2
3 import java.awt.Color;
9
10 public class Farma extends WinPane {
11
12     private int[] zahony;
13
14     public Farma() {
15         this.resize(500, 300);
16         this.setBackground(new Color(
17             this.zahony = new int[7];
18         this.kresliZahony(this.zahony.le
19     }
20
21     public void kresliZahony(int pocetZahony) {

```

A red arrow points from the Package Explorer to the code editor, highlighting the package path.



# Čo je balíček (package)

- **Skupina tried**, ktoré patria k sebe
  - adresár
  - group ID v Mavene
  - priezvisko
  
- **Minimalizácia konfliktov v pomenovaní balíčkov:**
  - reverzne napísaná doména tvorca/projektu
  - príklady:
    - `sk.upjs.jpaz2`
    - `org.htmlparser`



# Čo je balíček (package)

- Zvyčajne (99.9%) balíček triedy **určuje** umiestnenie triedy v adresárovej štruktúre

- Plné mená tried:

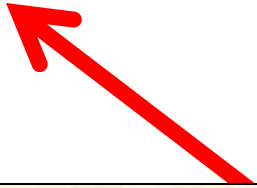
```
sk.upjs.jpaz2.WinPane plocha = new sk.upjs.jpaz2.WinPane();
```





# Import

```
package paz1a.lectures.lecture8;  
  
import java.io.File;  
import java.io.FileNotFoundException;  
import java.io.PrintWriter;  
import java.util.Locale;  
import java.util.Scanner;
```



Chcem používať skrátene meno  
triedy: Scanner

Ak napíšem Scanner, myslím tým  
java.util.Scanner.




# Import

```
package paz1a.lectures.lecture8;
```

```
import java.io.File;  
import java.io.FileNotFoundException;  
import java.io.PrintWriter;  
import java.util.Locale;  
import java.util.Scanner;
```

```
package paz1a.lectures.lecture8;
```

```
import java.io.*;  
import java.util.*;
```



Chcem importovať  
všetky triedy v balíčku  
java.util



# Balíčky

- V každom importe môže byť iba jedna hviezdička na konci
- Hviezdička neimportuje podbalíky

```
import sk.upjs.paz.*;
```

- Nevieme ani použiť:

```
import sk.upjs.paz.*.*;
```

- Netreba importovať:
  - triedy v rovnakom balíčku
  - triedy z balíčka `java.lang`





**Ďakujem za pozornosť !**

