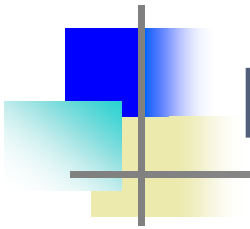
Prednáška 2: Objektovo-orientované programovanie v jazyku Java. Integrované vývojové prostredie Eclipse pre Javu. Organizácia programových súborov a zdrojových súborov

Ján Lang

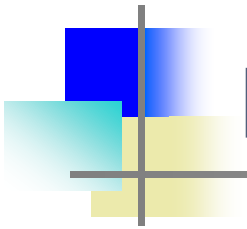
kanc. 4.34, lang@fiit.stuba.sk, http://www2.fiit.stuba.sk/~lang/zoop/

Ústav informatiky, informačných systémov a softvérového inžinierstva
Fakulta informatiky a informačných technológií
Slovenská technická univerzita v Bratislave
26. Septembra 2023



Rámcové zadanie

Produkty: tovary či služby



Hry, herné umenie a gamifikácia

Výroba produktov je značne zložitá činnosť, ktorá vyžaduje veľa plánovania a riadenia. To je prakticky nepredstaviteľné bez zodpovedajúcej softvérovej podpory. Plánovanie a riadenie výrobných procesov je špecifický druh plánovania podnikových zdrojov. Zahŕňa smerovanie (materiálov), rozvrhovanie, dispečing a následné zhodnotenie. Mnohí výrobcovia sa usilujú byť úsporní (lean manufacturing), čo znamená nevyrábať to, čo nie je potrebné a vyrábať práve včas. Priame mapovanie entít reálneho sveta do objektov v modeli podporovanom počítačom aplikujte vo svojom programe - v jazyku Java.

Vypracujte konkretizáciu rámcovej témy do podoby zámeru projektu súvisaceho s produktami a to tovarmi, alebo službami.

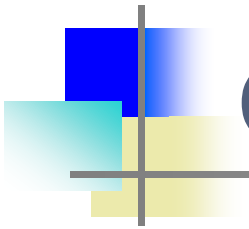
Objekt

- Pojem **objekt** a **inštancia** sú synonymá.
- Pojmu objekt sa dáva väčšia prednosť ak hovoríme všeobecne o objektoch
- Pojmu inštancia dávame prednosť vtedy ak chceme zdôrazniť do akej triedy objektov patrí (najmä ak ju vtedy spomíname)
- Objekty vytvárame pomocou kľúčového slova *new*

```
new Student();
```

- Takto vznikne inštancia bez toho aby sme na ňu mali dosah. Potrebujeme k nej mať prístup. Preto:

```
Student myFirstStudent = new Student();
```



Objekt

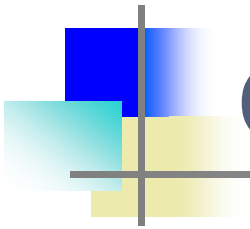
```
Student myFirstStudent = new Student();     //OK
Student myFirstStudent = new Student;       //Zle
```

Metóda `main()`

- Hlavný program resp. metóda, ktorá je volaná po spustení programu ako prvá sa musí volať `main() {}`
- Je jediná a musí byť uvedená (existujú výnimky...)
- Je súčasťou triedy, ktorej názov sa musí zhodovať s názvom súboru, v ktorom je uvedená. Java je case sensitive...
- Mala by byť uvedená v triede ktorá je označená ako `public`
- Má tvar:

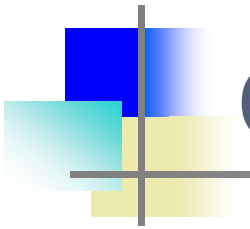
```
public static void main(String[] args) {
    // telo metódy main
}
```

- Je to statická metóda a preto **je ju možné volať aj bez existencie inštancie triedy**, ktorej je súčasťou



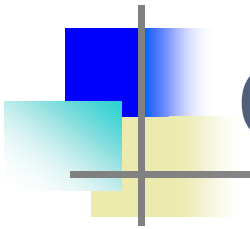
Objekt

```
Student jano = new Student();
```



Objekt

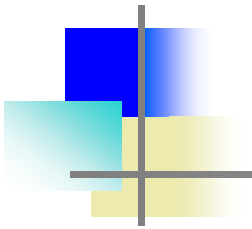
```
public static void main(String [] args)  {
    Student jano = new Student();
    System.out.println(jano.firstName);
}
```



Objekt

...čo vypíše nasledovné volanie metódy println???

```
System.out.println(jano.firstName);
```



???

a) nevypíše nič
b) zahlási chybu
c) null
d) ???
e) syntax error
f) jano
g) Student@hashcodenumber
h) class
i) Object
j) Jano
k) undefined

Objekt

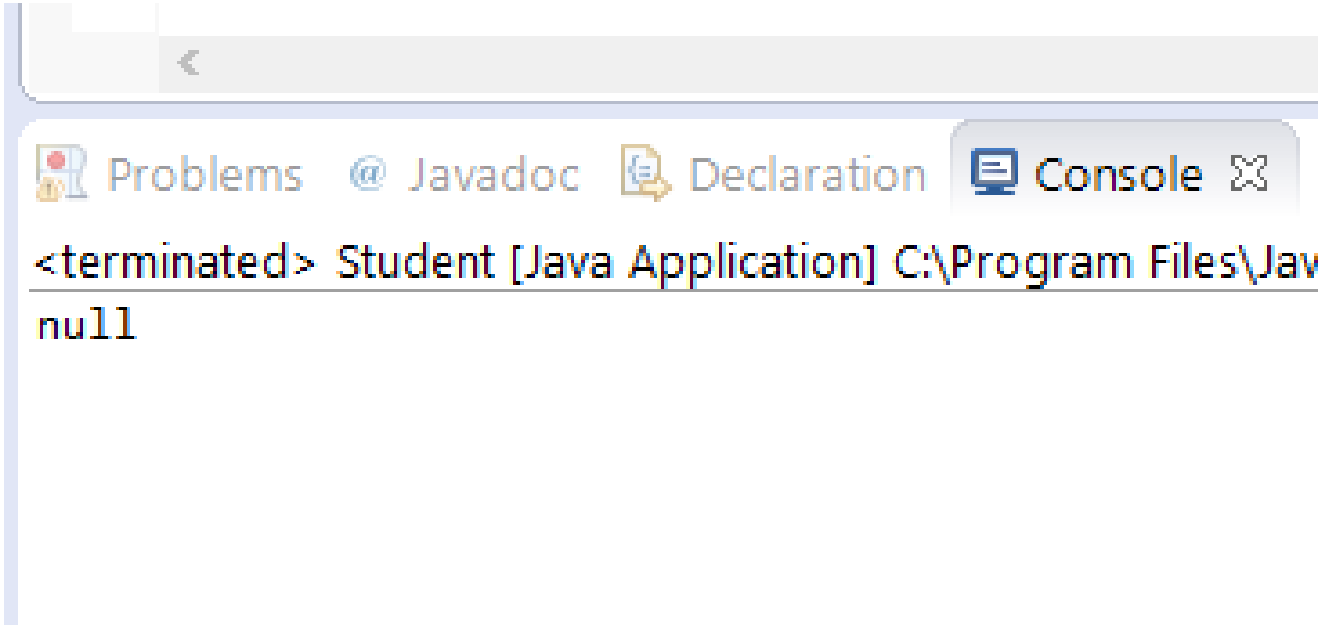
- Referencia typu Student vs. premenné triedy Student. Hoci sa premenná/referencia typu Student nazýva `jano`, neznamená to, že premenná `firstName` musí byť inicializovaná na `jano`

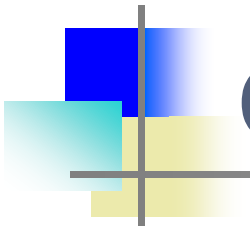
```
public static void main(String [] args)  {
    Student jano = new Student();
    System.out.println(jano.firstName);
}
```

Objekt

- Referencia typu Student vs. premenné triedy Student. Hoci sa premenná/referencia typu Student nazýva `jano`, neznamená to, že premenná `firstName` musí byť inicializovaná na `jano`

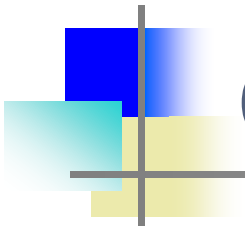
```
public static void main(String [] args)  {
    Student jano = new Student();
    System.out.println(jano.firstName);
}
```





Objekt

...prečo `null`?

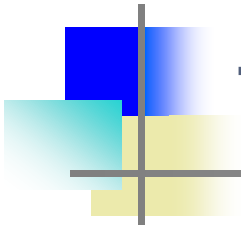


Objekt

- Vytvorme dvoch študentov
- Inicializujme ich atribúty
- Prostredníctvom (.) operátora bodky pristupujme k jeho atribútom zatiaľ dostupným na čítanie a zápis
- Po spustení programu Java zavolá triednu metódu main, ktorá zabezpečí vytvorenie študentov, vytvorí ich premenné, a vypíše ak je treba niečo do konzoly Eclips-u
- Zatiaľ si kľúčové slová - public, static nevšímajte, ozrejmime si ich neskôr. Metóda main však musí byť verejná, statická, nemá nič vracať a má jeden parameter

- A parameter is the variable which is part of the method's signature (method declaration). An argument is an expression used when calling the method[1] (sk.stuba.fiit.argumenty_2)

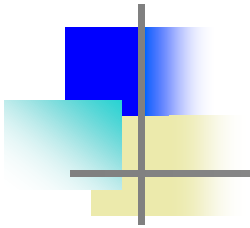
http://stackoverflow.com/questions/1788923/parameter-vs-argument



Trieda

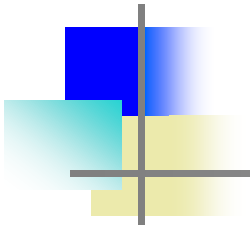
- Štandardná štruktúra triedy:

```
class CLASSNAME {
    // VARIABLE DEFINITION
    // METHOD DEFINITION
}
```



??

Je študent zodpovedný za vytvorenie iných študentov?

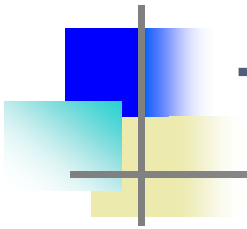


??

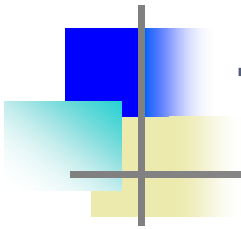
Je študent zodpovedný za vytvorenie iných študentov?

Tak prečo je metóda main v definícii triedy Student?

Trieda



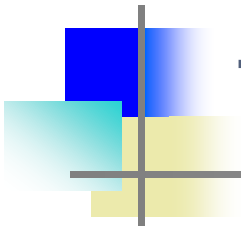
- Oddeľme túto logiku od definície triedy Student
- Ktorá trieda by mala byť zodpovedná za vytvorenie študentov?
- Napr. **ManazerStudentov**
- Preto vytvorme triedu s názvom ManazerStudentov
- Správanie týkajúce sa vytvárania študentov je takto oddelené od samotnej definície študent
- V našom prípade `jano` – dátová entita pomerne „nie veľmi múdrej triedy“



Trieda

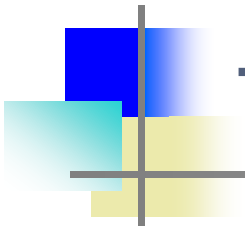
- Štandardná štruktúra:

```
class CLASSNAME {
    // VARIABLE DEFINITION
    // METHOD DEFINITION
}
```



Trieda - VARIABLE DEFINITION

- Uchovávanie dát reprezentované premennými
  - Premenné inštancií - budú uchovávať charakteristiky jednotlivých inštancií/objektov
  - Premenné tried - premenné, ktoré nie sú viazané na objekt ale triedu
  - Finálne premenné - po prvotnej inicializácii nemôžu byť zmenené
- Platnosť premenných
  - Lokálne premenné - deklarované v metóde či v bloku
  - Parametre - argumenty metód, premenné v slučkách
  - Premenné inštancií - deklarované pri definícii triedy



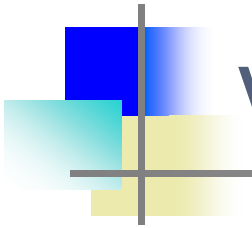
Trieda

- Štandardná štruktúra:

```
class CLASSNAME {
    // VARIABLE DEFINITION
    // METHOD DEFINITION
}
```

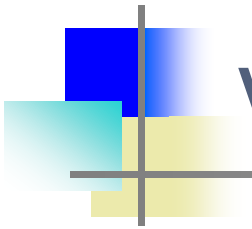
Trieda - METHOD DEFINITION

- Klasifikácia metód
  - Metódy inštancií - metódy dostupné objektom
  - Triedne metódy - premenné, ktoré nie sú viazané na objekt ale triedu
  - Konštruktory - špeciálny prípad, inicializácia inštancii/objektov
  - Metóda main - špeciálna metóda pre spustenie aplikácie
  - Default metóda – v rozhraniach



Vytváranie typov

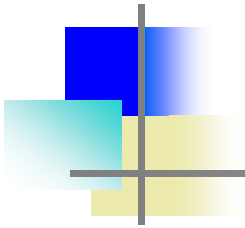
Rozšírme náš courseAdministrtation projekt o nové typy...



Vytváranie typov

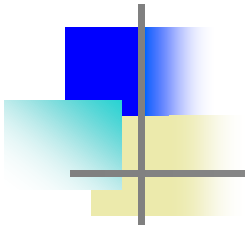
Rozšírme náš courseAdministrtation projekt o nové triedy...

napr. Course, ClassRoom, Book...



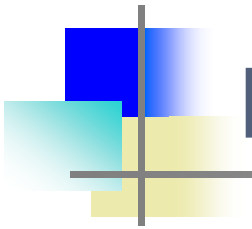
Prístup k OOP

- Alan Key, päť základných charakteristík prvého úspešného objektovo-orientovaného jazyka
  - **Všetko je objekt**
  - **Každý objekt má svoju časť pamäte**
  - **Každý objekt má svoj typ**
  - **Všetky objekty určitého typu, môžu prijímať rovnaké správy**
  - **Program je zväzok objektov navzájom komunikujúcich prostredníctvom posielania správ**



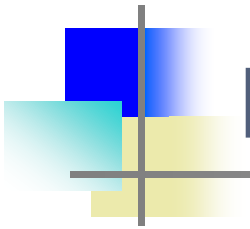
Prístup k OOP

- Alan Key, päť základných charakteristík prvého úspešného objektovo-orientovaného jazyka
  - **Všetko je objekt**
  - **Každý objekt má svoju časť pamäte**
  - **Každý objekt má svoj typ**
  - **Všetky objekty určitého typu, môžu prijímať rovnaké správy**
  - **Program je zväzok objektov navzájom komunikujúcich prostredníctvom posielania správ**



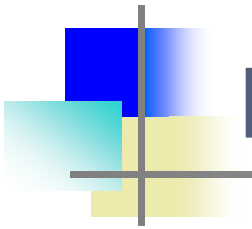
Prístup k OOP

...aké správy môže prijímať náš študent?



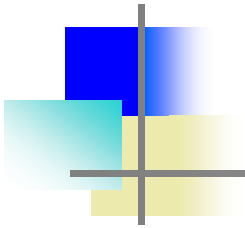
Prístup k OOP

...napr. move(Poloha p)



Prístup k OOP

...komunikácia objektov pokročilá – TCPClient, TCPServer

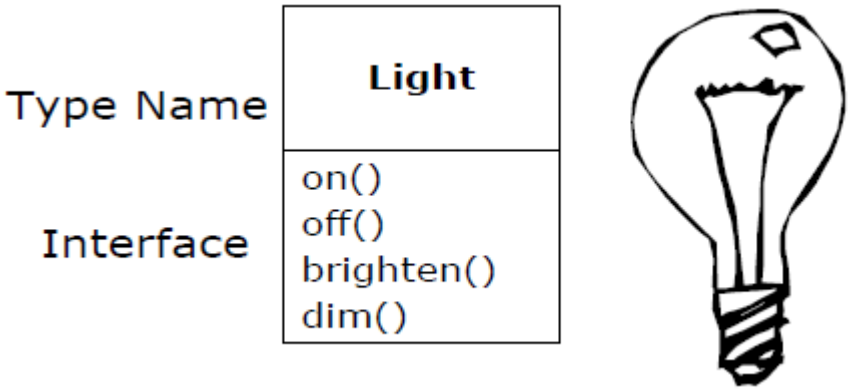


Prístup k OOP

- **Vytváranie dátových typov** `tried (classes)` **abstrahovaním** je fundamentálnym konceptom objektovo-orientovaného programovania
- To čo skutočne robíme, keď objektovo-orientovaným spôsobom programujeme je, že **vytvárame nové dátové typy**. V tomto kontexte sú **typ a trieda synonymá**
- Typy disponujú súborom atribútov – abstrahovaných vlastností reálnych inštancií (Ovocie – chuť, farba, hmotnosť a pod., Študent – meno, priezvisko, ročník, odbor štúdia a pod.)

Objekt a jeho rozhranie

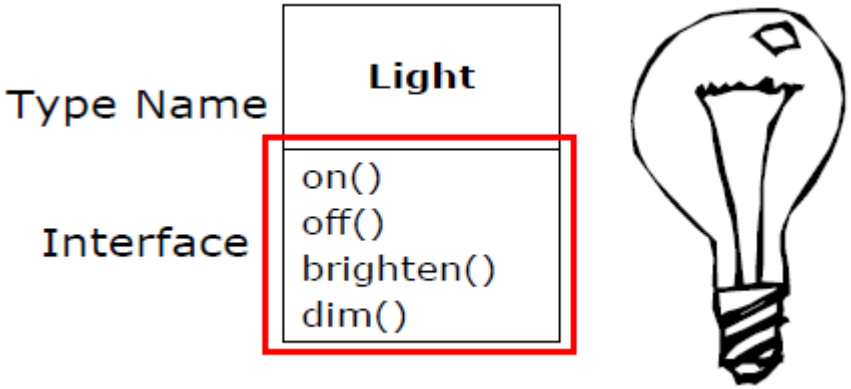
- Jednou z výziev OOP je 1:1 mapovanie medzi prvkami z domény problému a objektmi v doméne riešenia
- Ako možno objekty primäť k tomu aby konali užitočné?
- Každý objekt môže uspokojiť iba určité požiadavky. Požiadavky na objekt sú definované jeho rozhraním a to je to čo vlastne určuje jeho typ (trieda)



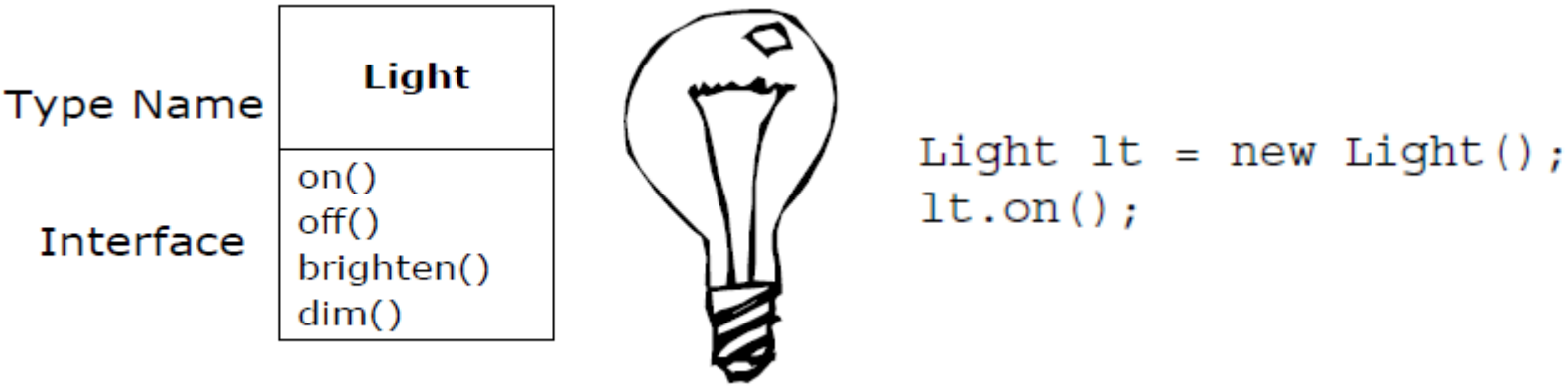
- Jednoduchý príklad pomenovaného typu a predpísaného rozhrania (Bruce Eckel, TIJ) - light bulb

Objekt má svoje rozhranie

- Rozhranie poskytované objektom vymedzuje požiadavky, ktoré naň môžu byť kladené
- Uspokojenie tejto požiadavky je definované v kóde. Implementačná **záležitosť, ukrytá spolu s dátami**
- Typ má asociovanú metódu s možnou požiadavkou. V momente realizácie požiadavky smerovanej na objekt **je príslušná metóda volaná**. Proces nazývaný **posielanie správ**



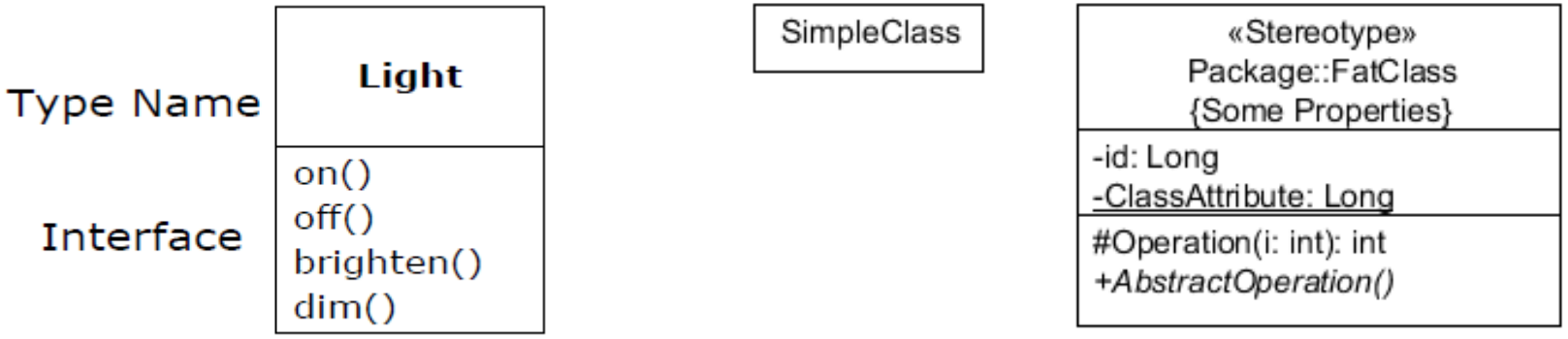
Objekt má svoje rozhranie

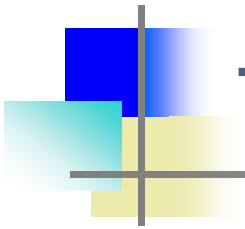


- V našom príklade máme typ s menom **Light**
- Možné požiadavky na objekty tejto triedy (tohto typu) sú: zapni (rozsvieť), vypni (?), pridaj jas a uber jas
- Objekty tejto triedy budeme vytvárať referenciou a kľúčovým slovom `new`
- Preto poslanie správy objektu (resp. volanie metódy príslušného objektu) zabezpečíme zápisom pozostávajúcim z mena objektu a príslušnej metódy oddelenej bodkou napr. `lt.on();`

Trieda

- Obrázok znázorňuje triedu v jazyku UML (Unified Modeling Language)
- Trieda môže byť v jazyku UML reprezentovaná:
  - Obdĺžnikom so svojim názvom uvedenom vo vnútri obdĺžnika
  - Obdĺžnikom s uvedením názvu v hornej časti pod ktorou bývajú uvedené jej vlastnosti (atribúty a metódy) minimálne svojim menom. Atribúty ako aj metódy môžu byť uvedené širšie a to doplnené o typ, návratovú hodnotu a parametre





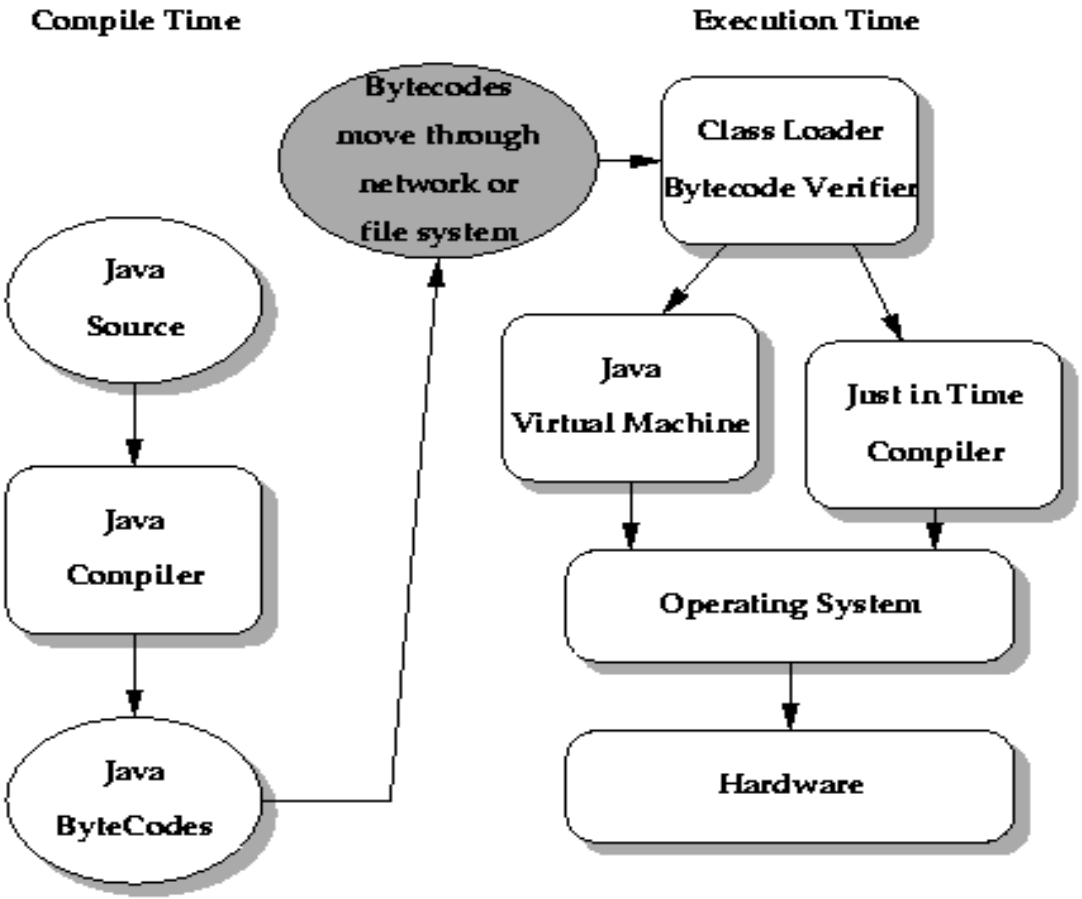
Trieda

- Trieda popisuje množinu objektov, ktoré majú rovnaké vlastnosti (atribúty, dátové prvky) a správanie (metódy, funkcie). Stav jednotlivých objektov však môže byť odlišný. **Stav inštancii triedy je definovaný hodnotami ich atribútov.**

  Uveďte príklad pre stav inštancií: Ovocie, Zelenina,...

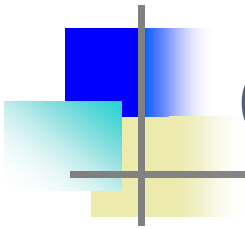
- Vytvorením triedy môžeme následne vytvárať jej inštancie v množstve podľa potreby a hardvérových limitov
- Program v Jave obsahuje vždy aspoň jednu triedu
- Deklarácia triedy kľúčovým slovom `class`
- V tele triedy ohraničenom `{}` sú deklarované atribúty a metódy

Ako to vôbec funguje...



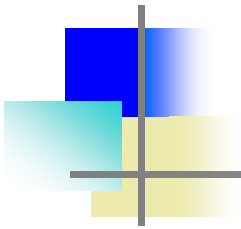
- JITC – dynamický preklad počas vykonávania programu

http://www.cab.u-szeged.hu/WWW/java/whitepaper/java-whitepaper-8.html



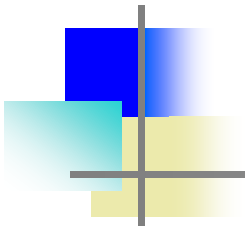
Compile time

- Implementácia možná od jednoduchého textového editora (vi, vim, WordPad, notepad++,...) po komplexné IDE (Eclipse, NetBeans, IntelliJ IDEA,...)
- Po implemenácii kompilácia/preklad. Výsledkom je bytecode.
- Compile time (doba prekladu, niektoré nástroje informujú o tejto skutočnosti, zaujímavá z titulu možných limitov)



Doba prekladu

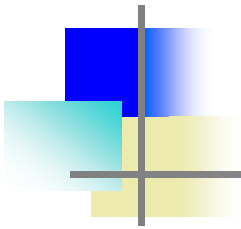
- **Čo sa môže pokaziť počas doby prekladu?**, aké chyby môžu nastať:
  - Syntaktické chyby, preklepy
  - Chyby typovej inkonzistencie
  - (Zriedka) problém/pád počas prekladu
- Ak je kompilácia úspešná, čo to znamená, čo z toho plynie?
  - Program je korektný
  - Program je možné spustiť. (Program môže vzápätí padnúť, ale ako pokus je to možné.)
- What are the inputs and outputs?
  - Input was the program being compiled, plus any header files, interfaces, libraries, or other voodoo that it needed to *import* in order to get compiled.
  - Output is hopefully assembly code or relocatable object code or even an executable program. Or if something goes wrong, output is a bunch of error messages.



Doba vykonávania

- run time, run-time, runtime
- A class loader is an object that is responsible for loading classes
- The bytecode verifier traverses the bytecodes, constructs the type state information, and verifies the types of the parameters to all the bytecode instructions[1]
- JVM, interpreter a samotný run time.

1 http://stackoverflow.com/questions/846103/runtime-vs-compile-time



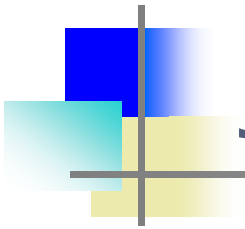
Run time

- **What can go wrong?** - *run-time errors*:
  - Division by zero
  - Deferencing a null pointer
  - Running out of memory

  Also there can be errors that are detected by the program itself:
  - Trying to open a file that isn't there
  - Trying find a web page and discovering that an alleged URL is not well formed
- If run-time succeeds, the program finishes (or keeps going) without crashing.
- Inputs and outputs are entirely up to the programmer. Files, windows on the screen, network packets, jobs sent to the printer, you name it. If the program launches missiles, that's an output, and it happens only at run time

Java virtual machine JVM

- Java virtual machine je softvérovou implementáciou počítača, ktorá dokáže spúšťať/vykonávať programy ako skutočný počítač
- JVM je napísaná pre špecifickú platformu (Windows, Linux, Android,...
- Java programy sú kompilované Java kompilátorom výsledkom čoho je Java bytecode. Java virtual machine interpretuje tento bytecode a vykonáva Java program



JRE vs. JDK

- Java Runtime Environment vs. Java Development Kit
- Java distribúcia dve verzie:
  - *Java Runtime Environment* (JRE)
  - *Java Development Kit* (JDK)
- Java runtime environment (JRE) pozostáva z JVM, knižníc tried a obsahuje nevyhnutnú funkcionalitu na spúšťanie Java programov
- Java Development Kit (JDK) obsahuje navyše vývojové nástroje na vytváranie Java programov. Pozostáva z Java prekladača, JVM a knižníc Java tried

Metóda `main()`

- Hlavný program resp. metóda, ktorá je volaná po spustení programu ako prvá sa musí volať `main() {}`
- Je jediná a musí byť uvedená (existujú výnimky...)
- Je súčasťou triedy, ktorej názov sa musí zhodovať s názvom súboru, v ktorom je uvedená. Java je case sensitive...
- Mala by byť uvedená v triede ktorá je označená ako `public`
- Má tvar:

```
public static void main(String[] args) {
    // telo metódy main
}
```

- Je to statická metóda a preto **je ju možné volať aj bez existencie inštancie triedy**, ktorej je súčasťou

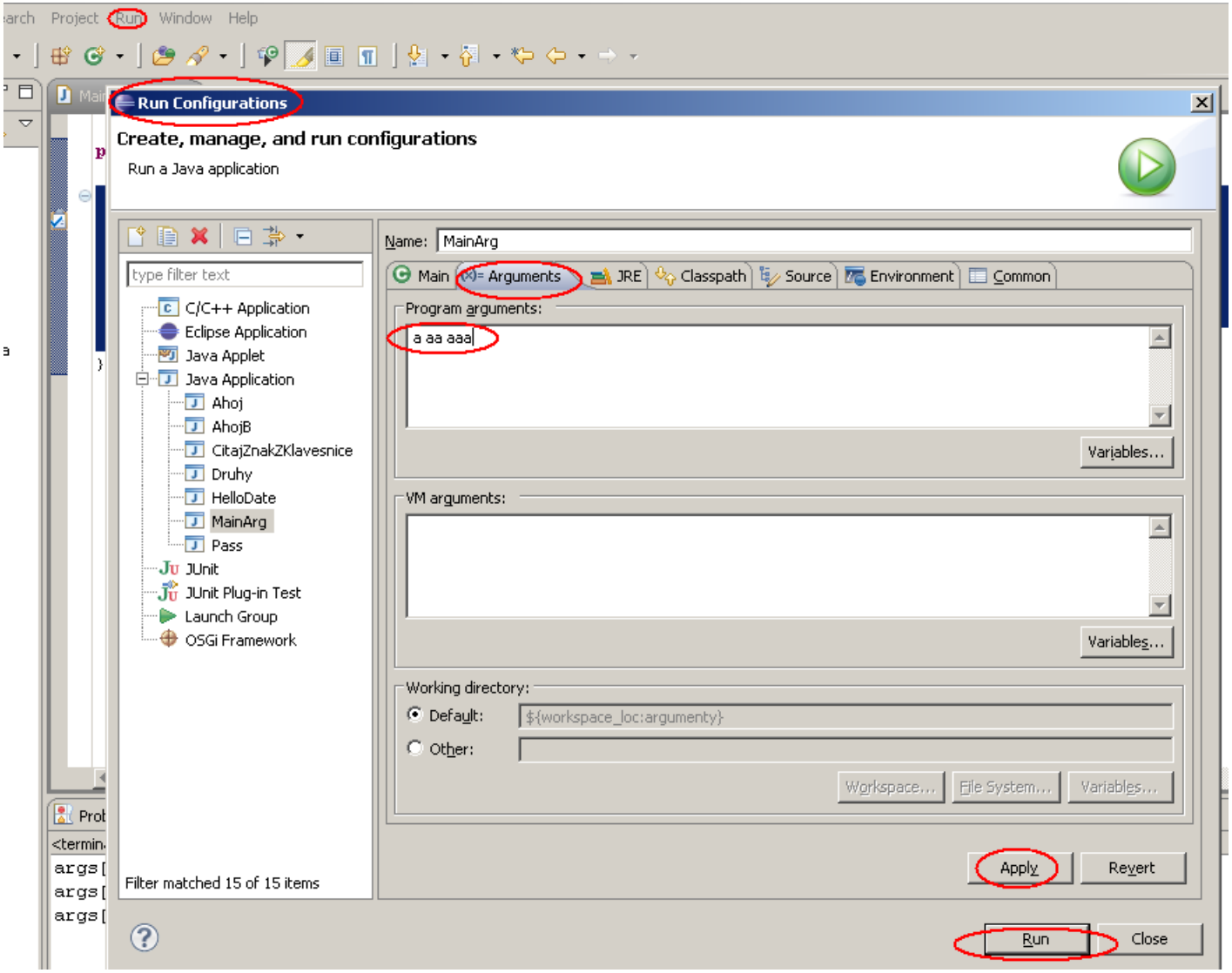
Argumenty príkazového riadku

- Spôsob ako možno inicializovať okamžite po spustení programu je odovzdať programu data ako parametre/argumenty pred jeho spustením. Priamo z príkazového riadku resp. v konfigurácii vývojového prostredia

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

- Pole `args` je reťazcové pole argumentov `args[0], args[1], args[2], ...`
- Pri zadávaní ich oddeľujeme medzerou

Zadanie argumentov

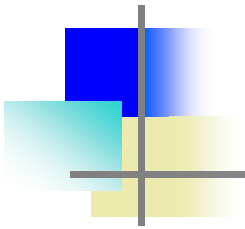


Argumenty príkazového riadku

- Celočíselné argumenty
- Nakoľko každý argument príkazového riadku je reťazec (String), presnejšie povedané objekt triedy String, môžeme ho chcieť interpretovať ako číslo

```
public static void main(String[] args) {
    for (int i = 0; i < args.length; i++) {
        int n = Integer.parseInt(args[i]);
        System.out.println("args["+ i+ "]=" + n);
    }
}
```

- Konverzia String na int. Môže skončiť neúspešne!

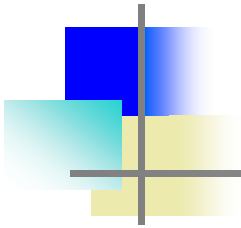


Dátové typy v Jave

- Triedy (obaľujúce triedy, wrapper class pre každý základný dátový typ resp. primitívny dátový typ)
- Základné dátové typy, primitívne dátové typy – celočíselné, znakové, logické, reálne a prázdny dátový typ `void`, ktorý sa používa ako návratový typ pri metódach. Metódami sa v OO jazykoch nazývajú funkcie známe napr. z jazyka C
- Metódy vracajúce prázdny dátový typ `void` asociujú princíp procedúry inak ide o funkcie známe aj z jazyka C

Dátové typy v Jave

Názov	bitov	Bytov	rozsah
`bool`	1	1/8	`true` and `false` are defined constants of the language and are not the same as `True` and `False`, `TRUE` and `FALSE`, `zero` and `nonzero`, `1` and `0`
`byte`	8	1	-128 až 127
`short`	16	2	-32 768 až 32767
`int`	32	4	-2 147 483 648 až 2 147 483 647
`long`	64	8	-9,223,372,036,854,775,808 až 9,223,372,036,854,775,807
`float`	32	4	1.40129846432481707e-45 až 3.40282346638528860e+38
`double`	64	8	4.94065645841246544e-324d až 1.79769313486231570e+308d
`char`	16	2	Unicode, 0 to 65,535



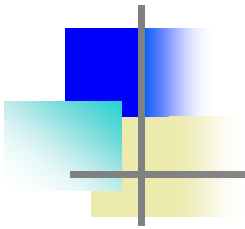
Príklad

- Napíšte program, ktorý pre všetky celočíselné základné dátové typy vypíše ich rozsah

```
public class RozsahyCelych {
  public static void main(String[] args) {
    System.out.println("min byte: " + Byte.MIN_VALUE);
    System.out.println("max byte: +" + Byte.MAX_VALUE);
    System.out.println("min short: " + Short.MIN_VALUE);
    System.out.println("max short: +" + Short.MAX_VALUE);
    System.out.println("min int: " + Integer.MIN_VALUE);
    System.out.println("max int: +" + Integer.MAX_VALUE);
    System.out.println("min long: " + Long.MIN_VALUE);
    System.out.println("max long: +" + Long.MAX_VALUE);
  }
}
```

Formátovanie

Escape Sekvencia	Opis
\t	Insert a tab in the text at this point.
\b	Insert a backspace in the text at this point.
\n	Insert a newline in the text at this point.
\r	Insert a carriage return in the text at this point.
\f	Insert a formfeed in the text at this point.
\'	Insert a single quote character in the text at this point.
\"	Insert a double quote character in the text at this point.
\\	Insert a backslash character in the text at this point.
\uXXXX	Character from the Unicode character set (XXXX is four hex digits)



Príklad

- Napíšte program, ktorý vypíše formátovaný text?
  She said "Hello!" to me.

```
public static void main(String[] args) {
    // TODO Auto-generated method stub
    System.out.println("She said \"Hello!\" to
      me.");
}
```

Komentáre

- Niekto hovorí: Zvyk je železná košeľa. Ja pridávam: Zlozvyk oceľová. (Max Kašparů) PRETO jednoznačne používať...
- Tri typy komentárov
  - Jednoriadkový, // - štýl známy z jazyka C++

```
public class CitajZnakZKlavesnice { // jednoriadkový komentár
```

  - Komentárový blok /* ... */ - štýl známy z jazyka C

```
/*
 *...toto je komentárový blok,
 * každý riadok začína zankom *
 */
```

  - Dokumentačný komentár

```
...nástroj na extrahovanie dokumentácie - javadoc, špeciálne
   komentárové tagy, ktoré vkladáme priamo do kódu. Výsledok je HTML
   súbor.
/** ... Začiatok. Nasleduje napr. @author author-information
 */ ... a koniec. TIJ: Comments and embedded documentation...
```

- Vnorený komentár prekladač ohlási ako chybu

Príklad použitia

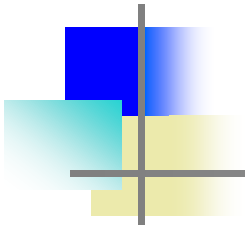
```
//: c02:HelloDate.java
import java.util.*;
/** The first Thinking in Java example program.
* Displays a string and today's date.
* @author Bruce Eckel
* @author www.BruceEckel.com
* @version 2.0
*/
public class HelloDate {
    /** Sole entry point to class & application
    * @param args array of string arguments
    * @return No return value
    * @exception exceptions No exceptions thrown
    */
        public static void main(String[] args) {
            System.out.println("Hello, it's: ");
            System.out.println(new Date());
    }
} ///:~
```

Príklad

- Napíšte program, ktorý vypíše iba celú časť premennej deklarovanej ako: double d = 3.14;

```
public class CelaCast {
    public static void main(String[] args) {
        double d = 3.14;
        int i = (int) d;

        System.out.println("Cela cast z " + d + " je " +
          i);
        System.out.println("Desetinna cast z " + d + " je "
          + (d - i));
    }
}
```



Príklad

- Napíšte program, ktorý vydelí dve reálne čísla. Ak bude výsledok NaN (Not a Number) alebo nekonečno, program vypíše: „Delitel je prilis male cislo!“. V ostatných prípadoch vypíše podiel.

Príklad

```
public class Podiel {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Zadaj 1. cislo: ");
        double d1 = sc.nextDouble();
        System.out.print("Zadaj 2. cislo: ");
        double d2 = sc.nextDouble();
        double vys = d1 / d2;
        if (Double.isInfinite(vys) == true)
          System.out.println("Delitel je prilis male
          cislo!");
        else if (Double.isNaN(vys) == true)
          System.out.println("Delitel je prilis male
          cislo!");
        else
          System.out.println("Podiel je: " + vys);
    }
}
```

Príklad

- Napíšte program, ktorý prečíta znak a vypíše jeho ASCII hodnotu v desiatkovej sústave.

  Vstup: A

  Výstup: A = 65

```
public class KodZnaku {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        System.out.print("Napis znak: ");
        char c = sc.nextLine().charAt(0);
        int i = (int) c;
        System.out.println(c + " = " + i);
    }
}
```

Príklad

- Napíšte program, ktorý pripočíta 19% daň.
  Vstup: Zadajte cenu bez dane: 100
  Výstup: Predajna cena s danou (19%): 119

```
public class VypocetDane {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    sc.useLocale(Locale.US);
    int percenta = 19;
    System.out.print("Zadajte cenu bez dane: ");
    double cena = sc.nextDouble();
    System.out.println("Predajna cena s danou (" +
  percenta + "%): "
                       + (cena * (1.0 + (double)
  percenta / 100.0)));
  }
```

Príklad

- Napíšte program, ktorý načíta tri celé čísla a vypíše ich súčet, súčin, priemer, najväčšie a najmenšie číslo.

```
import java.util.*;
public class CelaCisla {
  public static void main(String[] args) {
    int min, max;
    Scanner sc = new Scanner(System.in);
    System.out.print("Zadajte postupne tri cele cisla: ");
    int i1 = sc.nextInt();
    int i2 = sc.nextInt();
    int i3 = sc.nextInt();
    System.out.println("Sucet nieje: " + i1 + i2 + i3);
    System.out.println("Sucet je: " + (i1 + i2 + i3));
    System.out.println("Sucin je: " + i1 * i2 * i3);
    double priemerCely = (i1 + i2 + i3) / 3;
    double priemerDesetinny = (i1 + i2 + i3) / 3.0;
    System.out.println("Priemer je: " + priemerCely);
    System.out.println("Priemer je: " + priemerDesetinny);
```

Príklad - pokračovanie

```
max = i1;
    if (i2 > max)
      max = i2;
    if (i3 > max)
      max = i3;
    System.out.println("Najvacie
  je: " + max);
    if (i1 < i2) {
      if (i1 < i3) {
        min = i1;
      }
      else {
        min = i3;
      }
    }
```

```
else {
      if (i2 < i3) {
        min = i2;
      }
      else {
        min = i3;
      }
    }

System.out.println("Najmens
ie je: " + min);
  }
}
```

```
System.out.println("Najvacie prostrednictvom dostupnej funkcionality
z Java API 10 je: " + Math.max(i1, Math.max(i2, i3)));
System.out.println("Najmensie prostrednictvom dostupnej funkcionality
z Java API 10 je: " + Math.min(i1, Math.min(i2, i3)));
```

TODO

- Aj vy môžete pomôcť vylepšiť tento predmet študentom pre nasledujúci akademický rok. Vaše odporúčanie, komentár či otázka.