Informatika 1

11

Ďalšie vlastnosti triedy

KST Katedra softvérových technológií
Fakulta riadenia a informatiky, Žilinská univerzita v Žiline

Pojmy zavedené v 10. prednáške(1)

- zapuzdrenie
  - vonkajší vs. vnútorný pohľad
  - modifikátory prístupu
  - ukrývanie informácií
- bodková notácia na prístup ku atribútom

Pojmy zavedené v 10. prednáške(2)

- dokumentácia objektu
  - forma rozhrania
  - dokumentačné komentáre
  - javadoc – jazyk Java
    - tagy @author, @version, @param, @return

Pojmy zavedené v 10. prednáške(3)

- trieda ako objekt
  - atribúty triedy
  - metódy triedy
  - kľúčové slovo static
- návrhový vzor Singleton
  - súkromný konštruktor

Pojmy zavedené v 10. prednáške(3)

- konštantné atribúty
- kľúčové slovo final
  - nemeniteľné objekty
- preťažovanie správ a metód

Cieľ prednášky

- trieda ako objekt – dokončenie
- trieda ako množina svojich inštancií
  - enum
- algoritmizácia
  - switch

- príklad: míny – pokračovanie

Atribúty triedy

- definícia – kľúčové slovo static
- inicializácia – spojená s definíciou

```
private static final Obtiaznost[] aObtiaznosti = {
   new Obtiaznost("Lahka", 10, 9, 9),
   new Obtiaznost("Stredna", 40, 16, 16),
   new Obtiaznost("Tazka", 99, 16, 30)
};

private static Aplikacia aInstancia;
private static Aplikacia aInstancia = null;
```

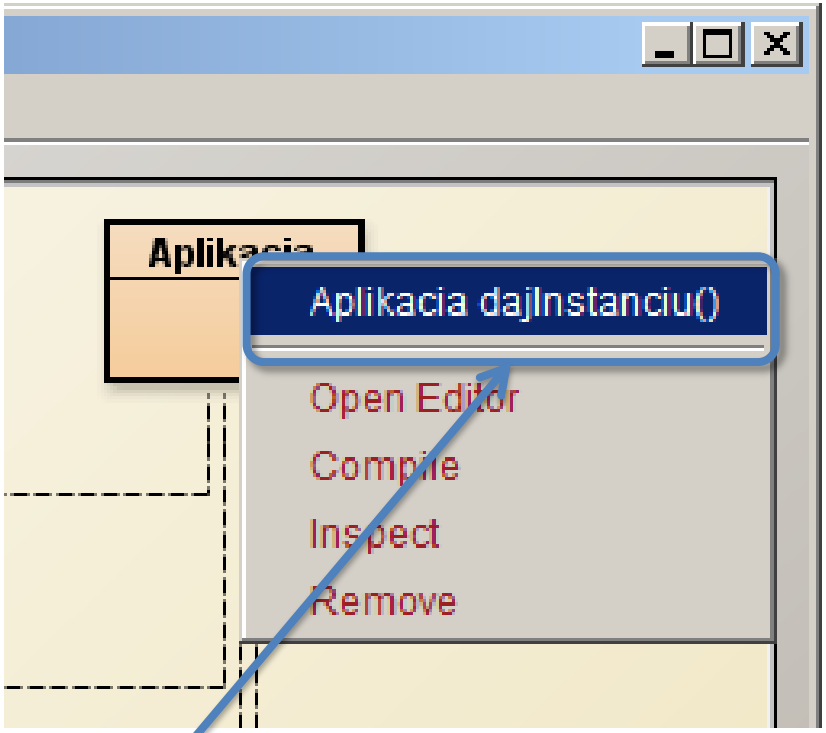
Metódy triedy$_{(1)}$

- definícia – kľúčové slovo static

```
public static Aplikacia dajInstanciu()
{
    ...
}
```

Metódy triedy(2)

- použitie – poslanie správy triede
  - BlueJ



  - Java – zdrojový text

```
Aplikacia miny = Aplikacia.dajInstanciu();
```

Míny – ďalšia požiadavka

- poskytovanie informácií o stave hry
  - výhra
  - prehra
- metóda dajStavHry()

Vyjadrenie stavu hry(1)

- nový atribút aVyhral
  - true – hráč vyhral
  - false – hráč prehral

- problémy:
  1. aký stav je v priebehu hry?
     - true – nie, hráč ešte nevyhral
     - false – nie, hráč ešte neprehral

Vyjadrenie stavu hry v priebehu hry

- ďalší nový atribút aHraSkoncila
  - false – hra ešte neskončila
  - true – hra už skončila, výsledok je v aVyhral

- problémy:
  1. ak hra ešte neskončila, dotazom na aVyhral dostaneme vždy nesprávnu odpoveď
  2. programátorská pýcha nedovolí také primitívne riešenie

Vyjadrenie stavu hry(2)

- možné stavy:
  - nerozhodnuta – hra ešte neskončila
  - vyhra – hráč vyhral
  - prehra – hráč stupil na mínu

- záver: dve hodnoty nestačia, treba tri

Vyjadrenie stavu hry(3)

- možné stavy – číslovanie stavov:
  - hodnota 0 – hra ešte neskončila
  - hodnota 1 – hráč vyhral
  - hodnota 2 – hráč stupil na mínu

- problémy:
  1. neprehľadné, pri pohľade na zdrojové kódy nie je jasný význam čísla
  2. pri preklade neoznámi prekladač nesprávnu hodnotu (-1, 3, ...)
  3. programátorská hrdosť nedovolí také primitívne riešenie

Vyjadrenie stavu hry(4)

- možné stavy – označenie reťazcami:
  - hodnota "nerozhodnuta" – hra ešte neskončila
  - hodnota "vyhral" – hráč vyhral
  - hodnota "prehral" – hráč stupil na mínu

- problémy:
  1. pri preklade neoznámi prekladač nesprávnu hodnotu (preklepy)
  2. programátorská hrdosť nedovolí také primitívne riešenie

Vyjadrenie stavu hry(5)

- riešenie – vymenovaný typ – enum

- extenzia triedy – množina všetkých inštancií triedy.

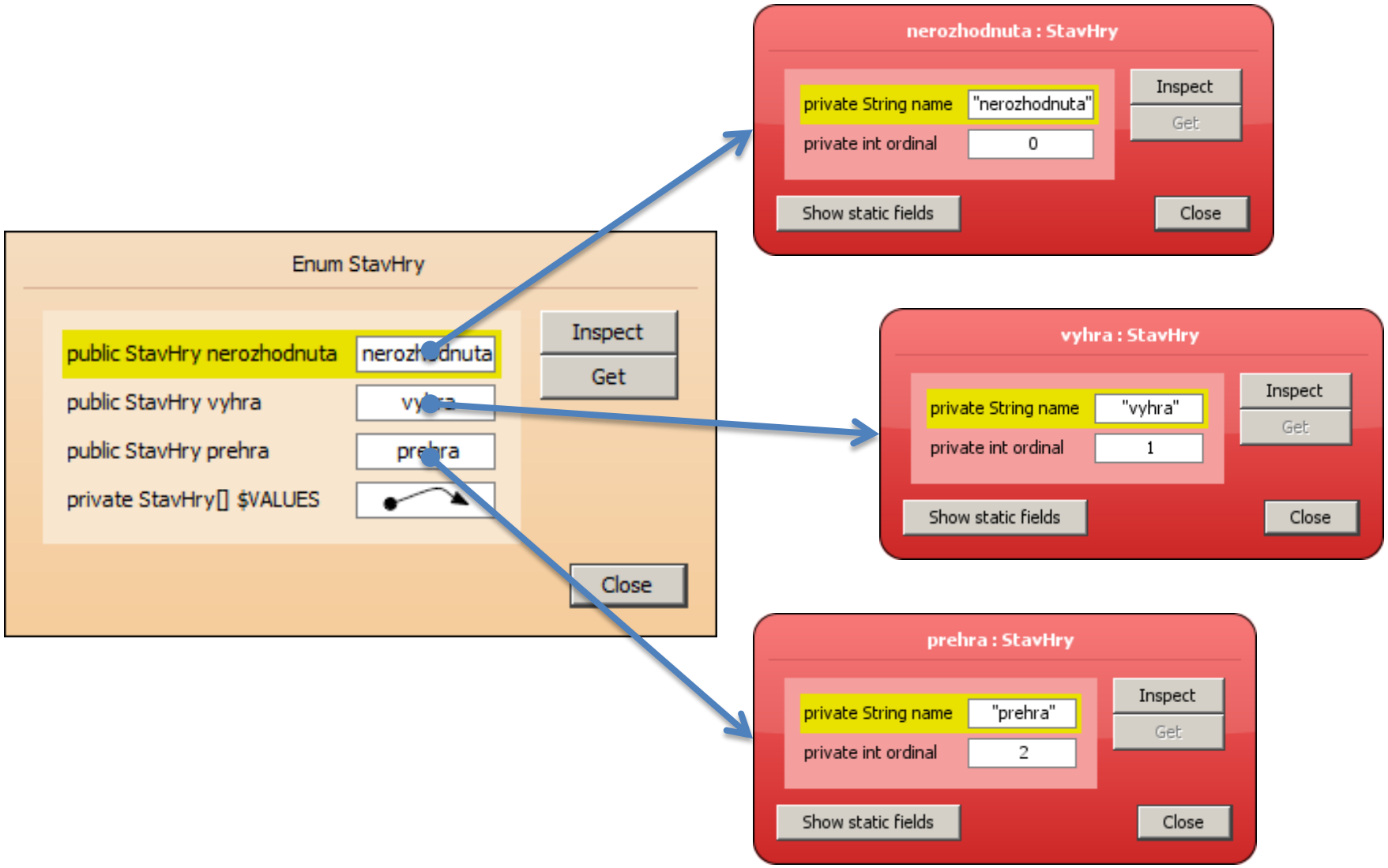
- enum – trieda s konštantnou extenziou.
- enum – trieda s pevne určenými inštanciami.
- enum – konštantná množina objektov
  - obsahuje svoje inštancie ako nemenné objekty

Enum StavHry

```
public enum StavHry
{
    nerozhodnuta,
    vyhra,
    prehra;
}
```

zoznam inštancií
– extenzia triedy StavHry

Enum – StavHry



Použitie enum StavHry – trieda Hra(1)

```
public class Hra
{
    private StavHry aStav;

    public Hra(...)
    {
        aStav = StavHry.nerozhodnuta;
        ...
```

Použitie enum StavHry – trieda Hra(2)

```
if (aStav == StavHry.vyhra) {
    ...
}
...
System.out.println(aStav);
```

Enum – špeciálna trieda

- zjednodušená syntax
- inštancie môžu mať atribúty
- inštancie môžu mať metódy

Míny – ďalšia požiadavka

- informácie o stave políčka
  - enum
  - toString vracia reťazec, ktorý sa má vypísať do terminálu

Enum - StavPolicka(1)

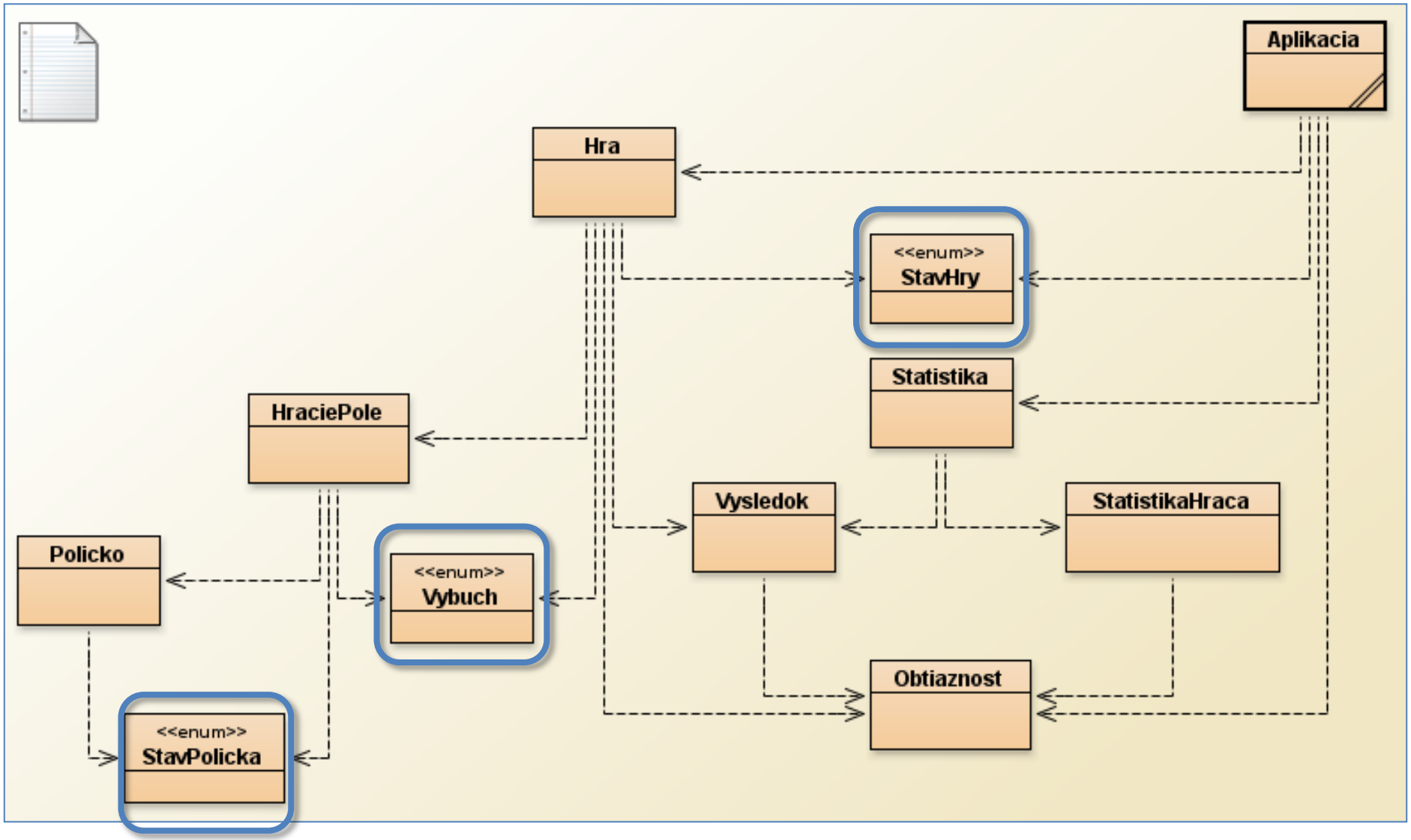
```
    private final String aReprezentacia;

    StavPolicka(String paReprezentacia)
    {
        aReprezentacia = paReprezentacia;
    }
    public String toString()
    {
        return aReprezentacia;
    }
}
```

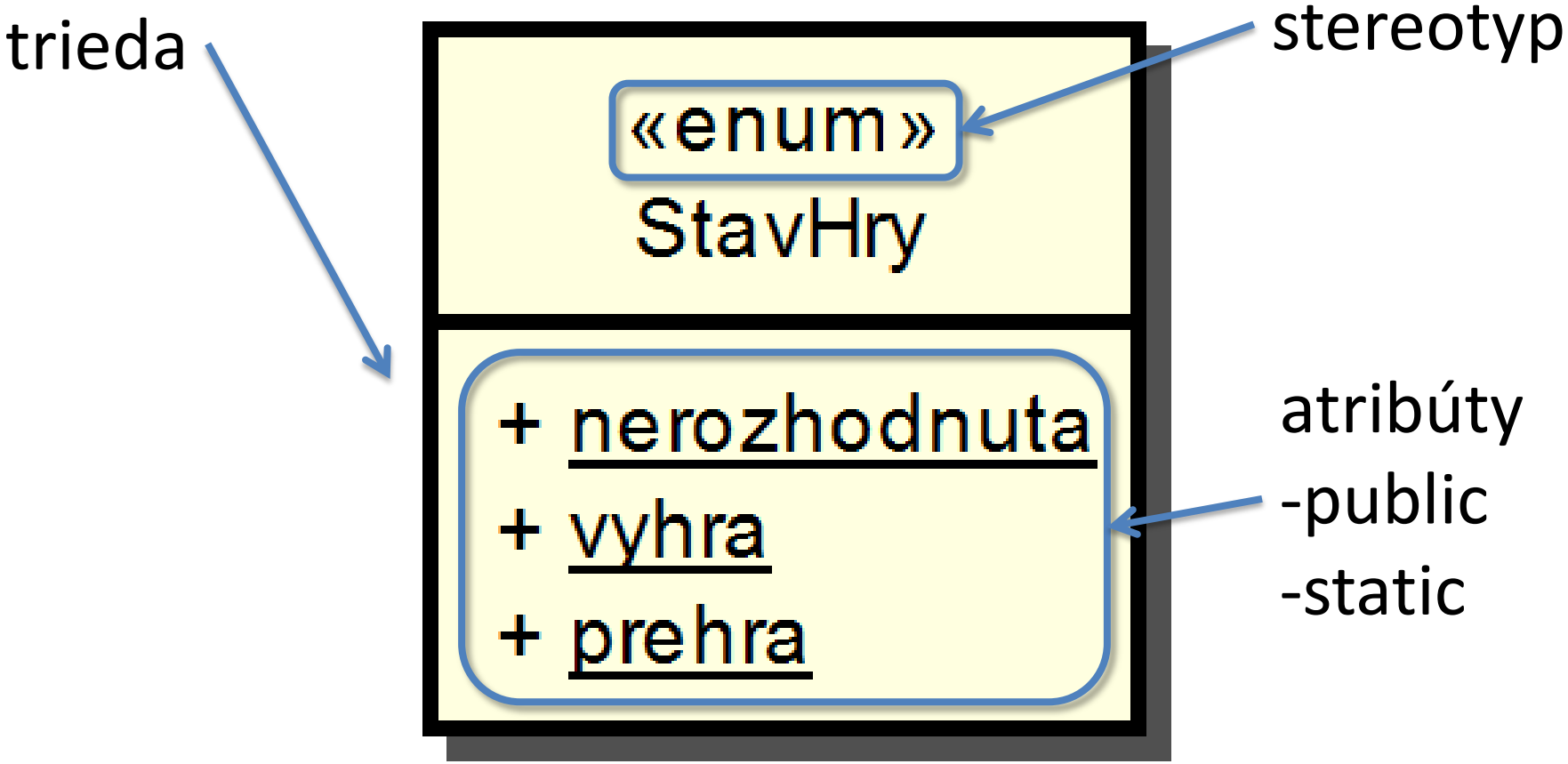
Enum - StavPolicka(2)

```
public enum StavPolicka
{
    zakryte("."),
    oznacene("F"),
    ukazanaMina("+"),
    prazdne(" "),
    odkryte("n"),
    vybuchnute("*");
    ...
```

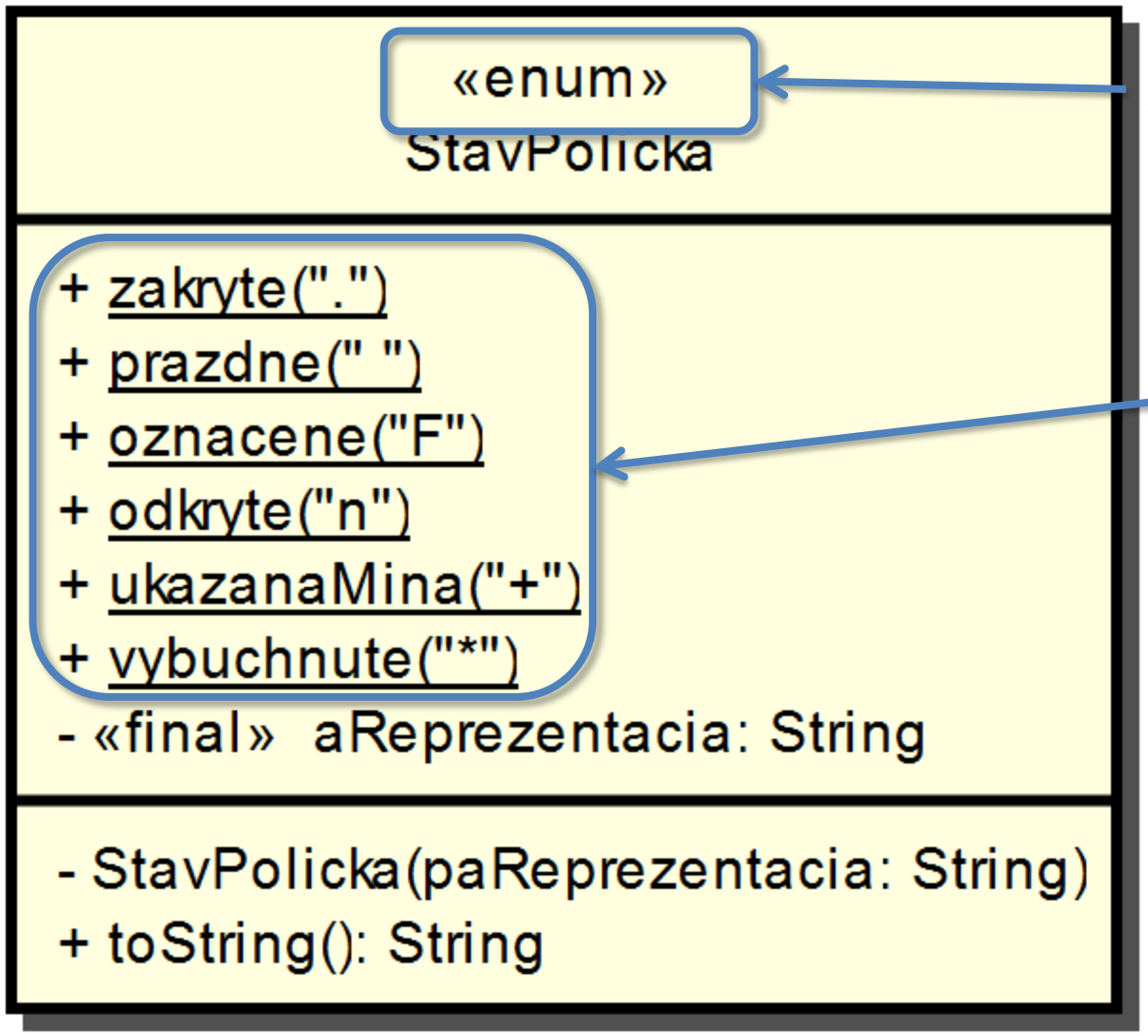
BlueJ – diagram tried



Enum v UML(1)


trieda
stereotyp
«enum»
StavHry
+ nerozhodnuta
+ vyhra
+ prehra
atribúty
-public
-static

Enum v UML(2)



Enum ako nemeniteľný objekt

- Java umožnuje aj zmenové metódy pre inštancie enum
- Inštancie enum majú byť nemeniteľné
- Zmena stavu inštancii enum je považovaná za nesprávny postup

Trieda ako množina

- trieda má extenziu – množinu inštancií
- ak zanedbáme ostatné vlastnosti – trieda reprezentuje množinu inštancií daného typu
- enum – konštantná množina

Pohľady na triedu – zhrnutie(1)

1. trieda ako objekt
   - vonkajší pohľad
     - rozhranie – správy triede
   - vnútorný pohľad
     - atribúty triedy
     - metódy triedy

Pohľady na triedu – zhrnutie(2)

2. trieda ako továreň
   – hlavná úloha triedy – vytvárať inštancie
   – špeciálna správa new – žiadosť o novú inštanciu

3. trieda ako šablóna
   – potreba poznať štruktúru inštancie pri vytváraní
   – definícia vnútorného pohľadu na inštancie
     - atribúty inštancie
     - metódy inštancie

Pohľady na triedu – zhrnutie(3)

4. trieda ako typ
   - predstavuje typ inštancie
   - definícia premenných (atribúty, parametre, lokálne premenné)
   - definícia typu návratovej hodnoty

5. trieda ako množina
   - extenzia triedy – množina všetkých inštancií danej triedy
   - špecifický prípad – enum

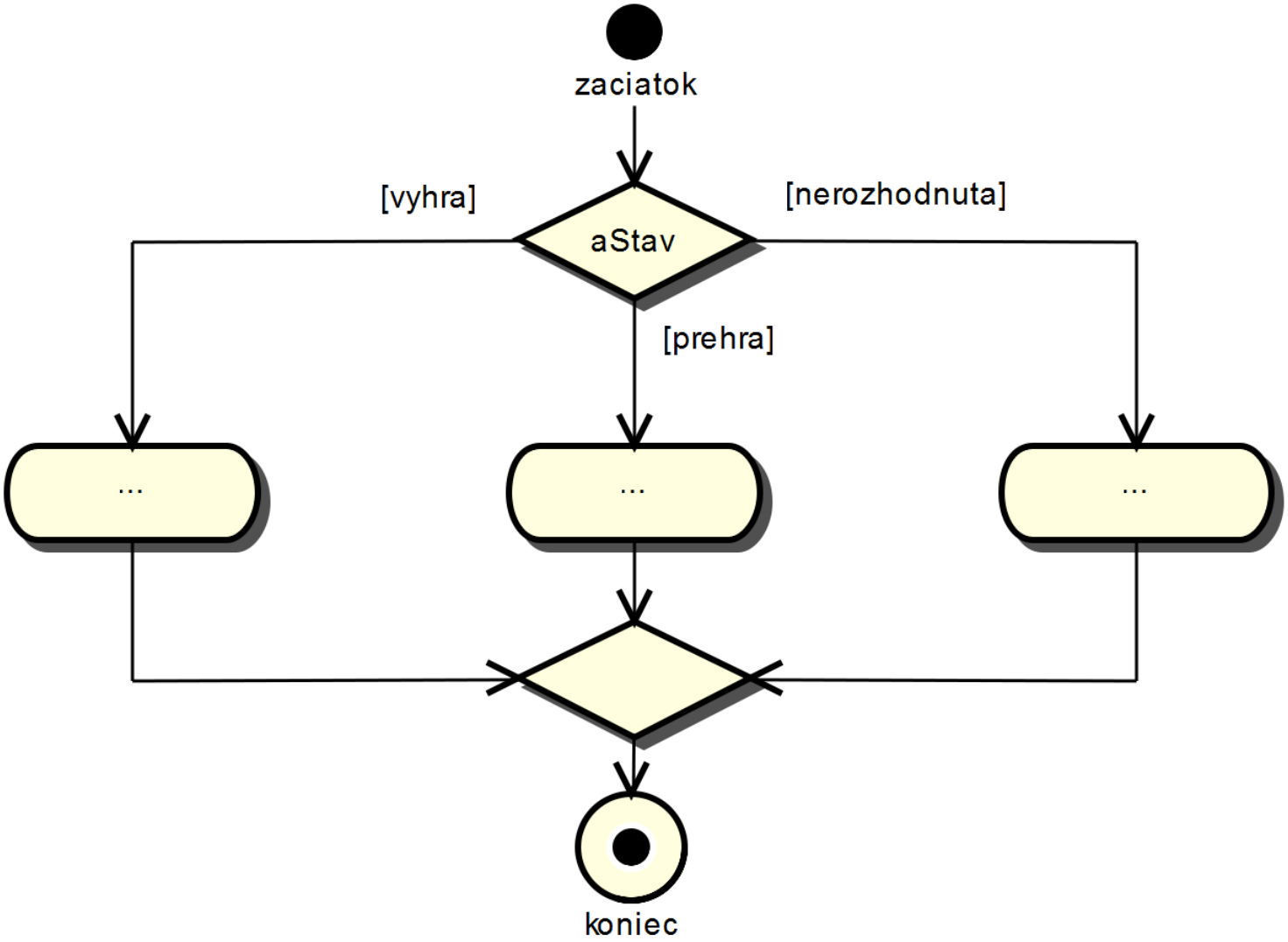
Vetvenie pomocou enum

- porovnanie inštancií enum na rovnosť referencií

- podmienené vykonanie:

```
if (aStav == StavHry.vyhra) {
    ...
}
```

Viaccestné vetvenie – UML



Vetvenie pomocou enum – else if

```
if (aStav == StavHry.vyhra) {
    ...
} else if (aStav == StavHry.prehra) {
    ...
} else if (aStav == StavHry.nerozhodnuta) {
    ...
}
```

Vetvenie pomocou enum – switch

```
switch (aStav) {
    case vyhra:
        ...
        break;
    case prehra:
        ...
        break;
    case nerozhodnuta:
        ...
        break;
}
```

Príkaz switch

- príkaz switch – viaccestné vetvenie

```
switch (výraz) {
   // možnosti a vetvy
}
```

- výraz
  - musí byť buď celočíselný výraz, alebo výraz typu enum
  - porovnáva s možnosťami

Príkaz swich – case (návestie)

- možnosti musia byť konštantné celočíselné výrazy, alebo hodnoty enum
  - typovo kompatibilné s výrazom v príkaze switch
  - Java 7 – aj reťazce
- pri enum sa nedáva identifikátor triedy

```
case moznost1:
case moznost2:
...
   // príkazy vetvy
```

Ukončenie vetvy

- každá vetva má byť ukončená
  - príkaz return – ukončenie vykonávania metódy
  - príkaz break – ukončenie vykonávania príkazu switch
  - chýbajúce ukončenie
    - prekladač jazyka Java neupozorní
    - vykonávanie pokračuje ďalšou vetvou

Príkaz switch – default (návestie)

- default – vetva, ktorá sa uplatní, ak sa nenájde príslušná možnosť
- obdoba else v príkaze if

```
default:
   // príkazy vetvy
```

Java mimo BlueJ

KST Katedra softvérových technológií
Fakulta riadenia a informatiky, Žilinská univerzita v Žiline

Práca v prostredí BlueJ

- priama komunikácia s objektmi
  - posielanie správ triedam – vytváranie nových inštancií
  - posielanie správ inštanciám – práca s objektmi
- nástroje na zjednodušenie vývoja, testovanie aplikácií

Aplikácia mimo BlueJ

- dve formy programu
  - zdrojový kód – číta/píše programátor
  - cieľový kód – vykonáva počítač
- šíriť sa dajú obe formy
- používateľ – cieľový kód určený na vykonanie

Cieľový kód

- obsahuje priamo počítaču známe príkazy – inštrukcie
- sada inštrukcií môže byť rôzna

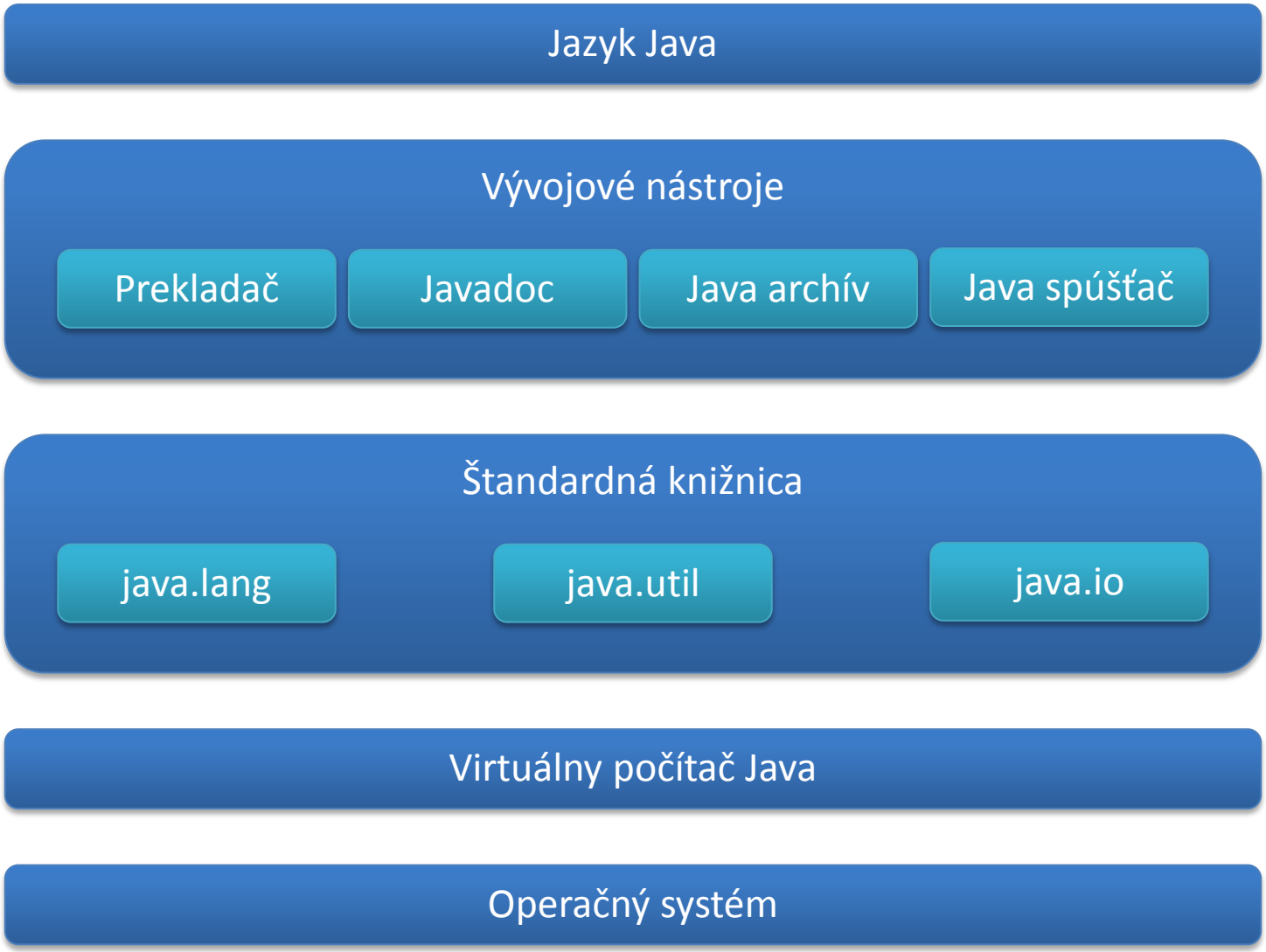
Výpočtová platforma

- hardvér + softvér nutný na spustenie programu
- typicky
  - počítačová architektúra
  - operačný systém (MS Windows, Linux, BSD, Unix, MacOS, Symbian, Android, iOS...)
  - programovací jazyk (Java...)
  - knižnice
  - ...

Výpočtová platforma Java(1)

- programy v jazyku Java nevykonáva priamo procesor počítača
- sú vykonávané virtuálnym počítačom Java (jvm – Java Virtual Machine)
- platformou pre vývoj a spustenie programu v jazyku Java
- nezávislosť od operačného systému a hardvéru skutočného počítača
  - stačí mať nainštalovanú platformu Java

Výpočtová platforma Java(2)



Edície platformy Java

- Java Card
  - Java pre čipové karty
- Java ME (Mobile Edition)
  - Java pre mobilné telefóny
- Java SE (Standard Edition)
  - Java pre štandardné použitie
- Java EE (Enterprise Edition)
  - Java pre komplexné informačné systémy

Dva typy inštalácie platformy Java(1)

- JRE – Java runtime environment
  - spúštacie prostredie Java – spúštanie programov v jazyku Java
- JDK – Java developement kit
  - vývojové prostredie Java – vývoj programov v jazyku Java (obsahuje aj JRE)
- Zadarmo na
  - http://www.oracle.com/technetwork/java/javase/downloads/index.html

Dva typy inštalácie platformy Java(2)

Spustenie programu bez BlueJ

- treba mať nainštalované JRE
- treba mať preložený program vo forme
  - class súborov,
  - alebo jar súboru.
- aspoň jedna trieda musí vedieť prijať správu main.

Správa main

- vstupný bod programu
- špeciálna verejná správa triede
- Trieda.main(parametre)
  - parametre – zoznam parametrov zadaných používateľom
- spúšťač Java ju posiela zadanej triede ihneď po spustení programu

! po ukončení vykonávania priradenej metódy program končí

Metóda main

- priradená správe main

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

- príkazy vykonané v tele sa vykonajú ihneď po spustení programu
- po ukončení vykonávania metódy main program končí

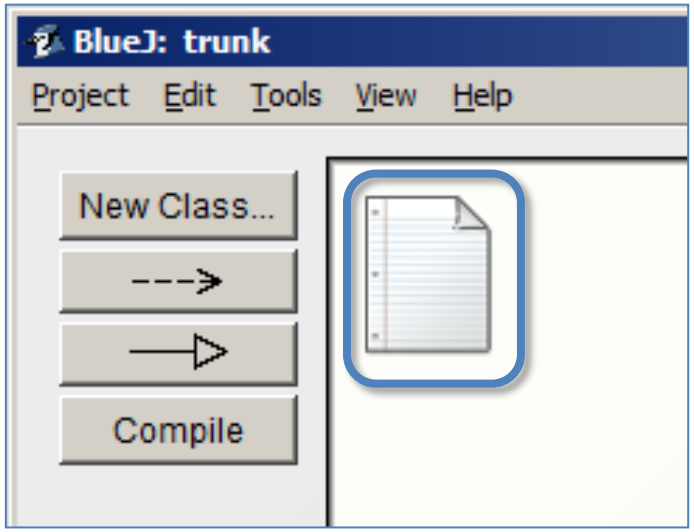
Program HelloWorld

- najznámejší program implementovaný asi v každom jazyku

```
public class HelloWorldApplication
{
   public static void main(String[] paParametre)
   {
      System.out.println("Hello world!");
   }
}
```

Súbory v adresári s projektom

- package.bluej
  - dáta ku projektu (pozície tried v diagrame...) používané programom BlueJ
- README.txt – popis projektu napísaný v programe BlueJ



Súbory v adresári s projektom

- Pre každú triedu:
  - HelloWorldApplication.java
    - zdrojový kód triedy – jazyk Java
  - HelloWorldApplication.class
    - cieľový kód triedy
  - HelloWorldApplication.ctxt
    - Info o triede používané programom BlueJ

Preložený program

- triedy – súbory s príponou class
- jedna z tried musí obsahovať metódu main

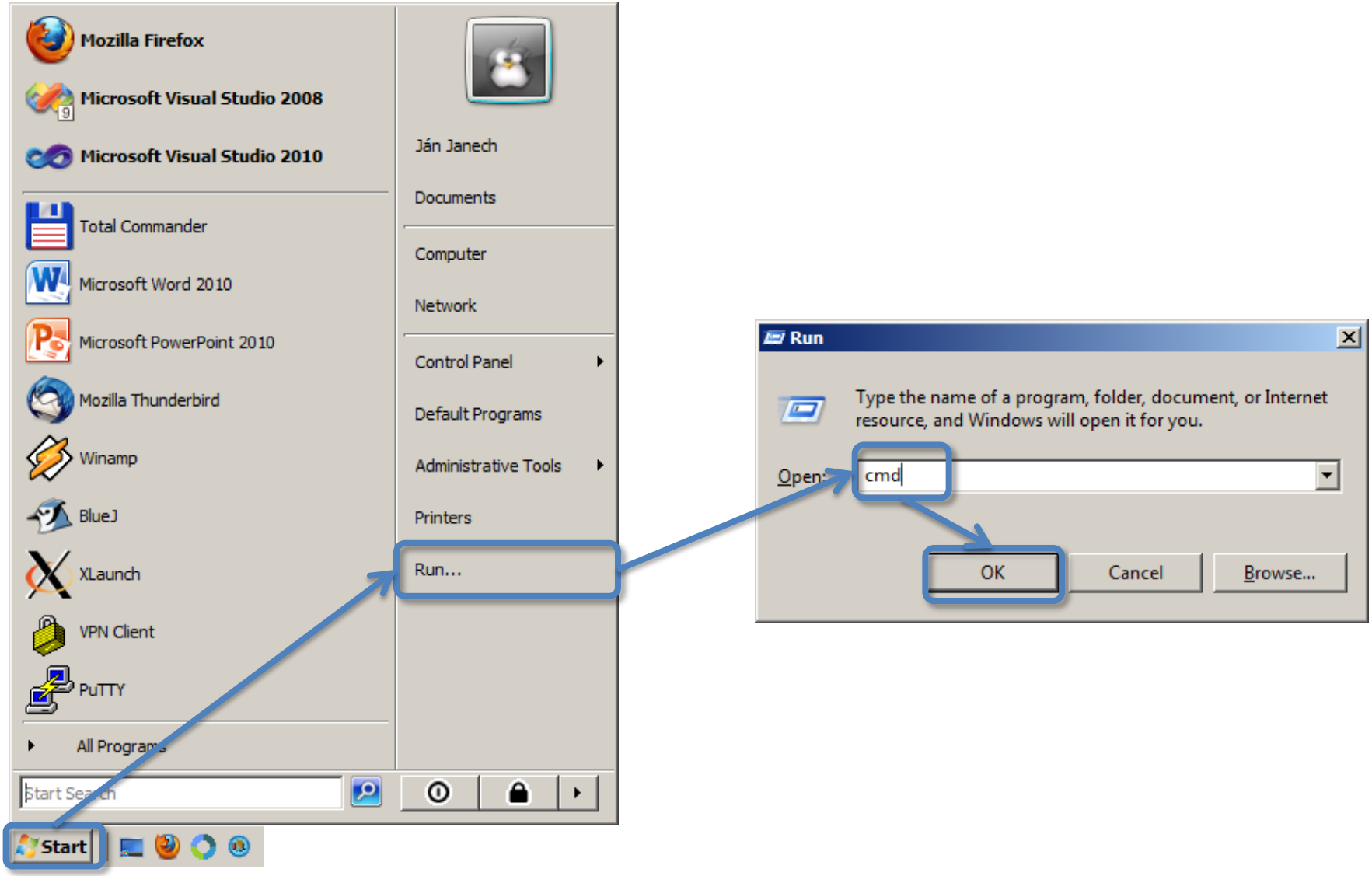
- spustenie programu
  - príkaz (v adresári s projektom):

```
java TriedaSMetodouMain
```

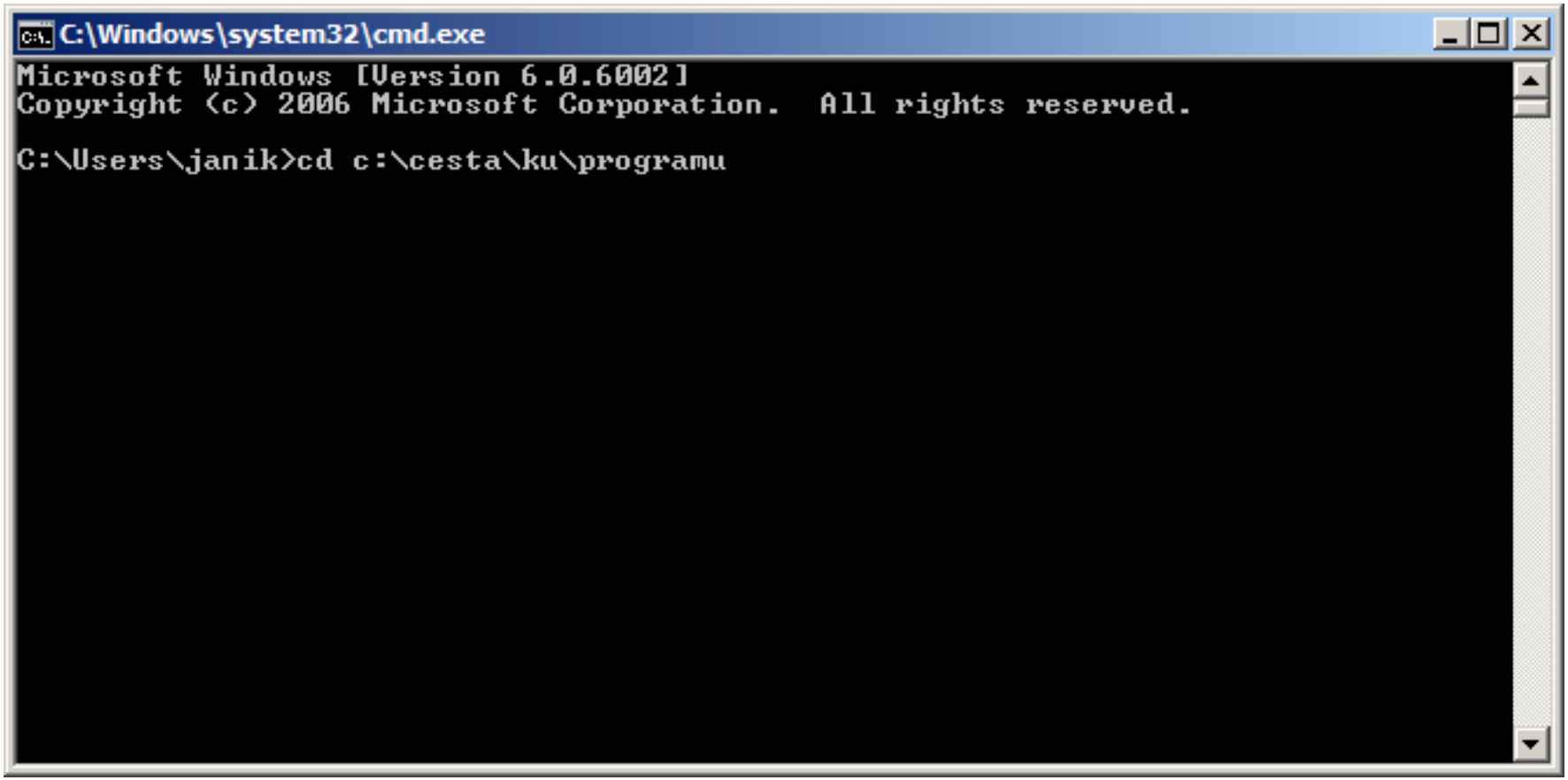
Java spúšťač

Názov triedy obsahujúcej metódu main

Ako spustiť preložený program(1)



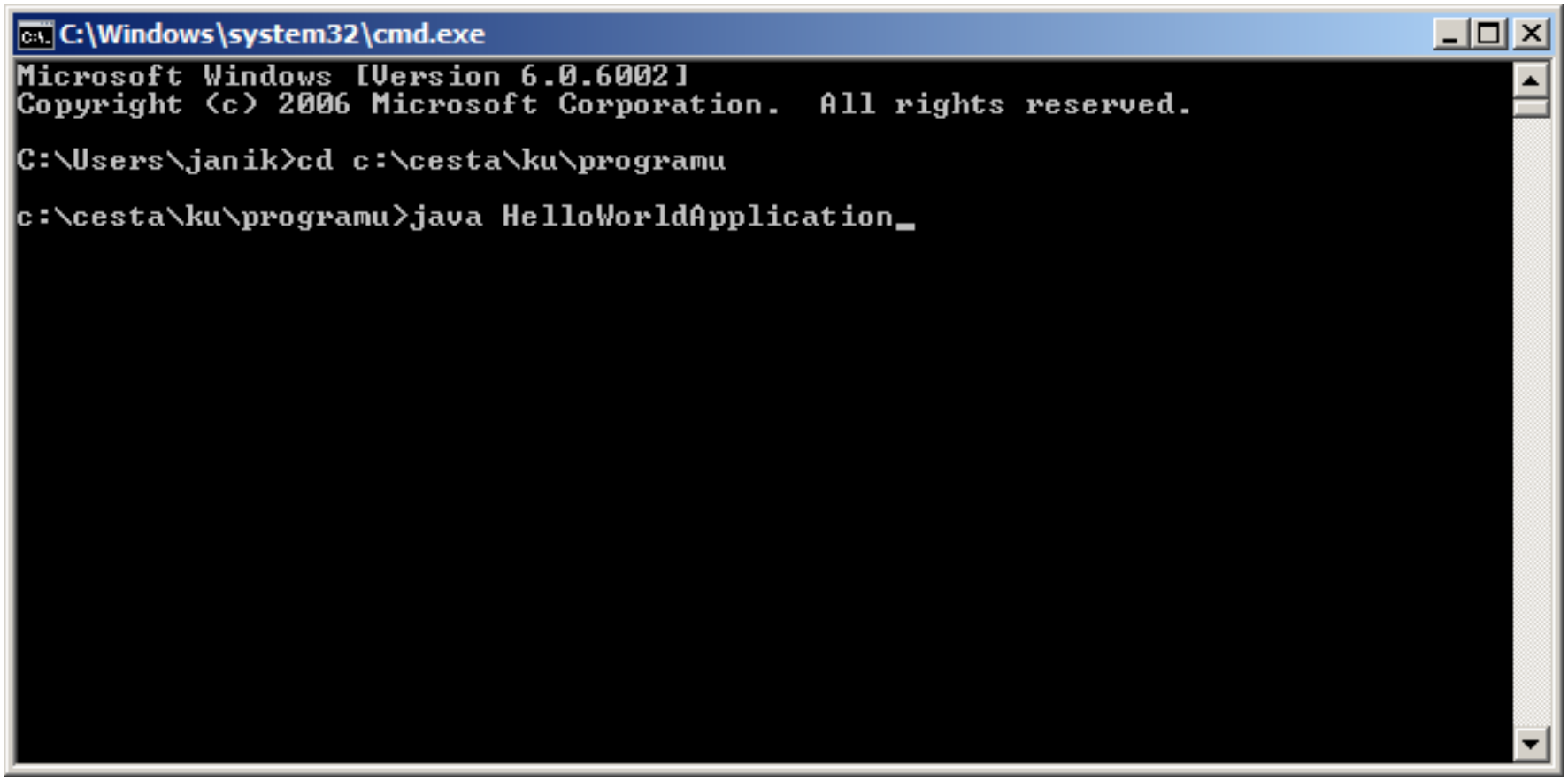
Ako spustiť preložený program(2)



príkaz `cd c:\cesta\ku\programu`

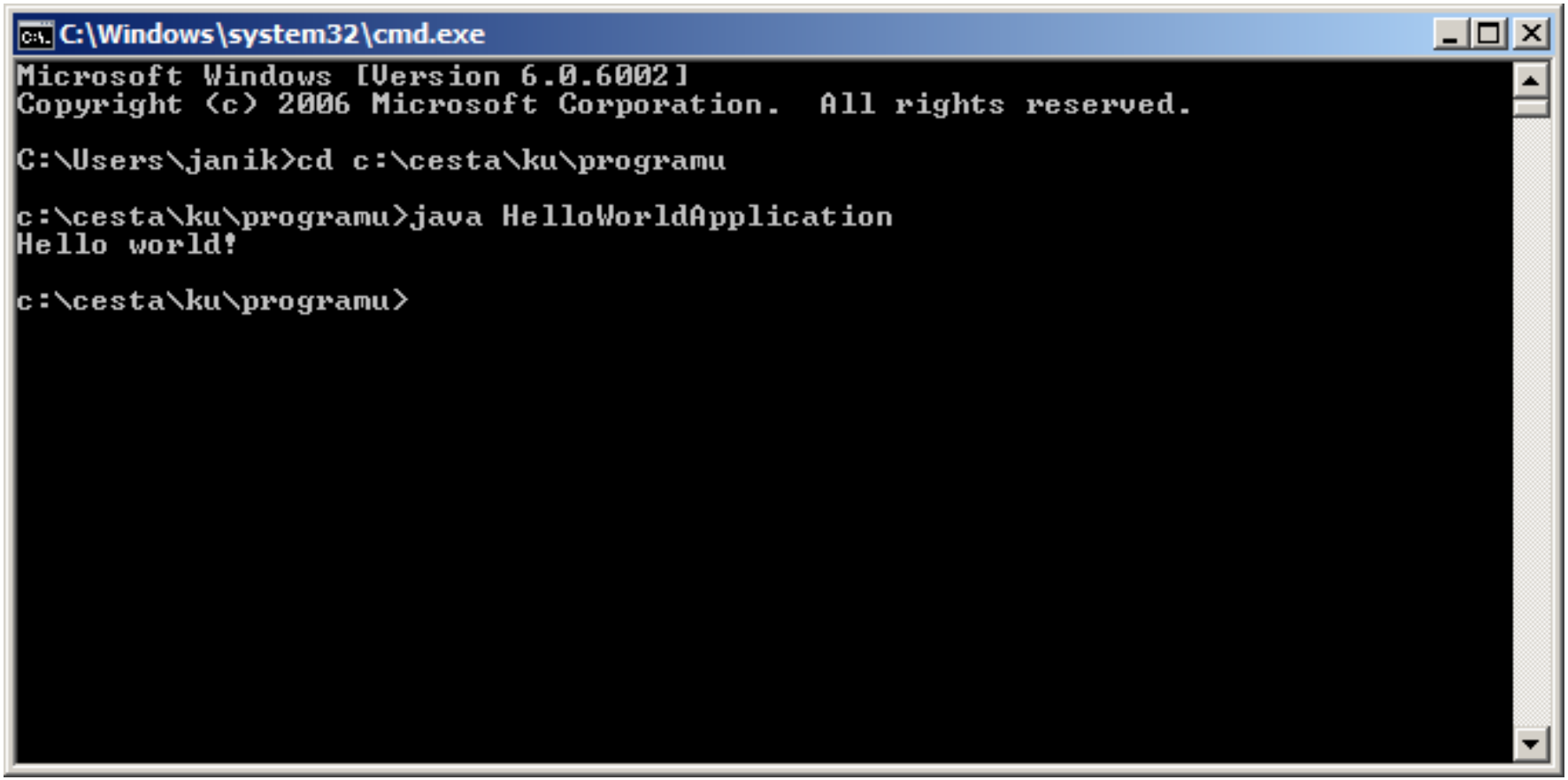
cesta vyjadruje cestu ku projektu v BlueJ

Ako spustiť preložený program(3)



príkaz `java -cp . HelloWorldApplication`
pošle správu main triede HelloWorldApplication

Ako spustiť preložený program(4)



Spustenie – jednoduchšie

- v adresári s projektom:
- windows – dávkový súbor
  - HelloWorldApplication.bat

```
java HelloWorldApplication
```

- ostatné OS – shell skript
  - HelloWorldApplication.sh
  - nastaviť práva na spustiteľný

```
#!/bin/sh
java -cp . HelloWorldApplication
```

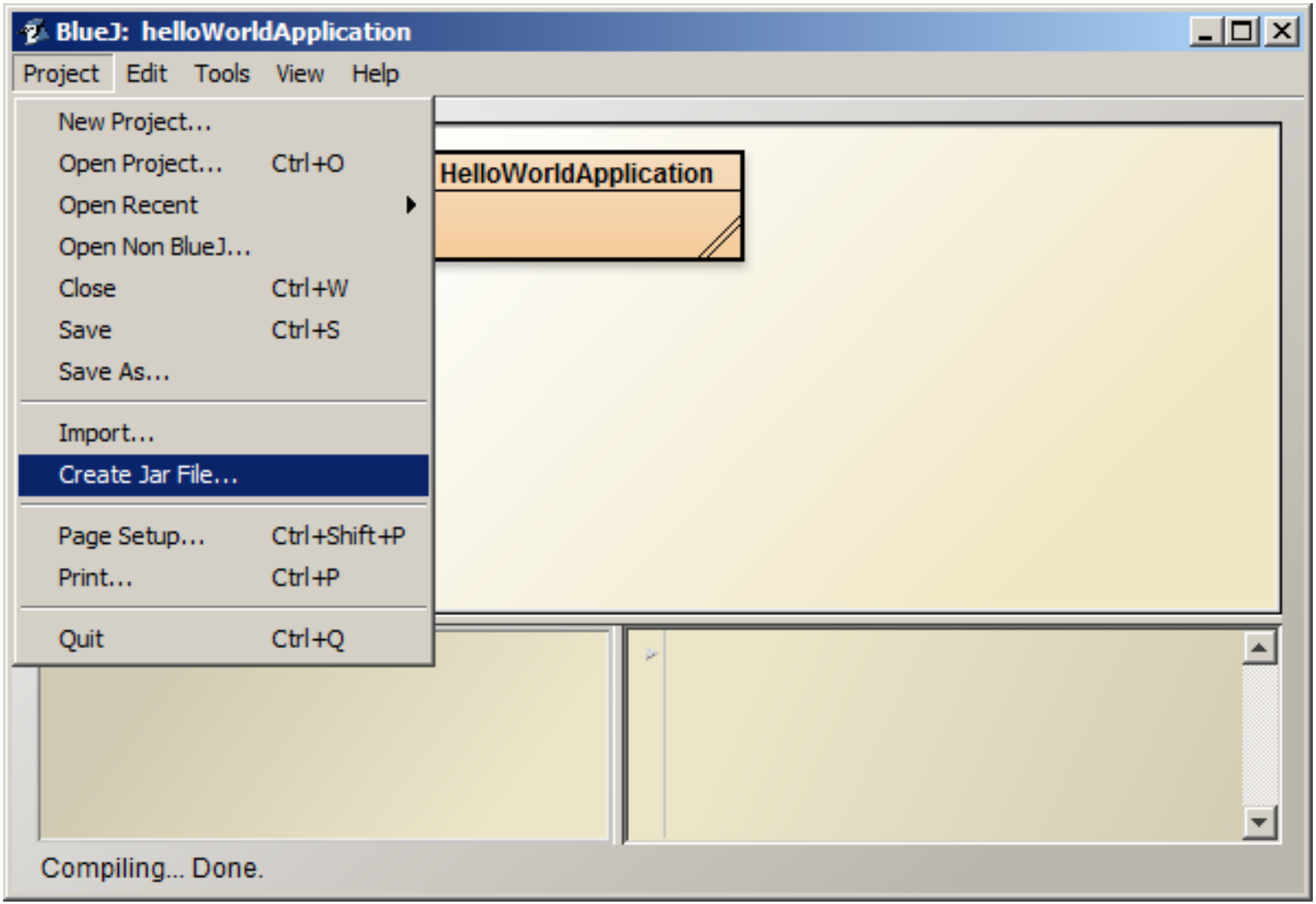
Distribúcia programu

- na iný počítač treba nakopírovať
- súbory s príponou class – triedy preložené do cieľového kódu
- dávkový súbor/shell skript – na jednoduché spúšťanie programu

Súbory jar

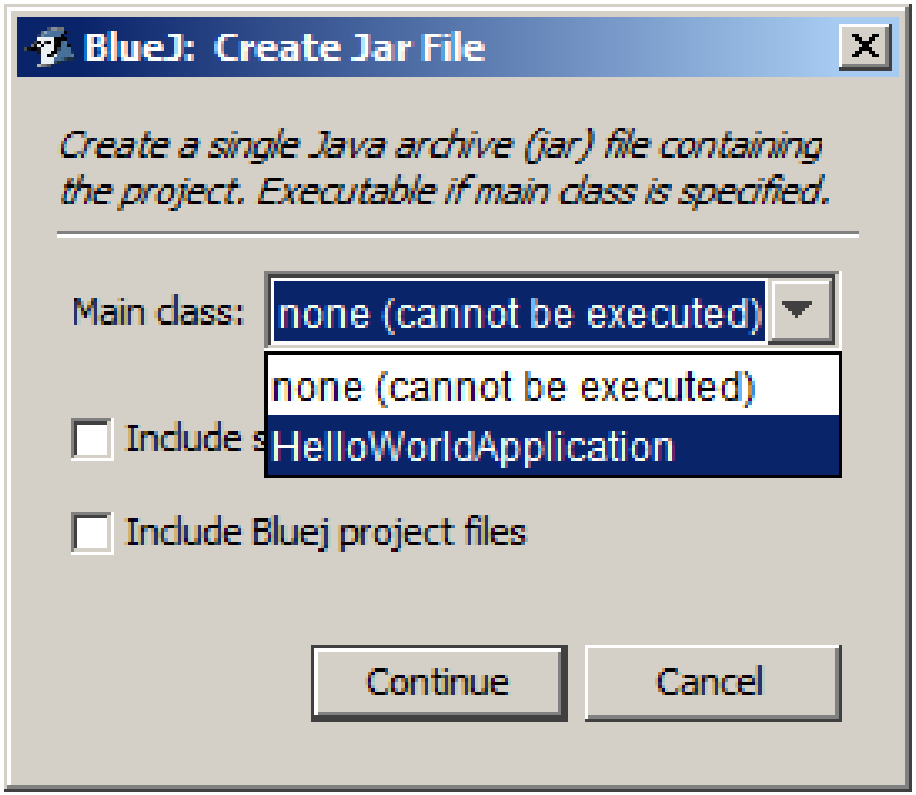
- jar – Java Archive – archív obsahujúci všetky súbory nutné na spustenie programu
  - hlavne súbory class
- jednoduchšia distribúcia

Vytvorenie jar v nástroji BlueJ(1)



Vytvorenie jar v nástroji BlueJ(2)

- dôležité: vybrať triedu, ktorá obsahuje metódu main



Programy vo formáte jar

- vo Windows sa dajú spúšťať dvojkliknutím na súbor
- neotvárajú konzolu
- riešenie – dávkový súbor/shell skript s príkazom:

```
java -jar helloWorldApplication
```

Názov jar súboru

Komunikácia s používateľom

- prostredníctvom terminálu
  - vstup z klávesnice
  - výstup do okna terminálu

- grafické prostredie
  - budúci semester

Používateľský vstup

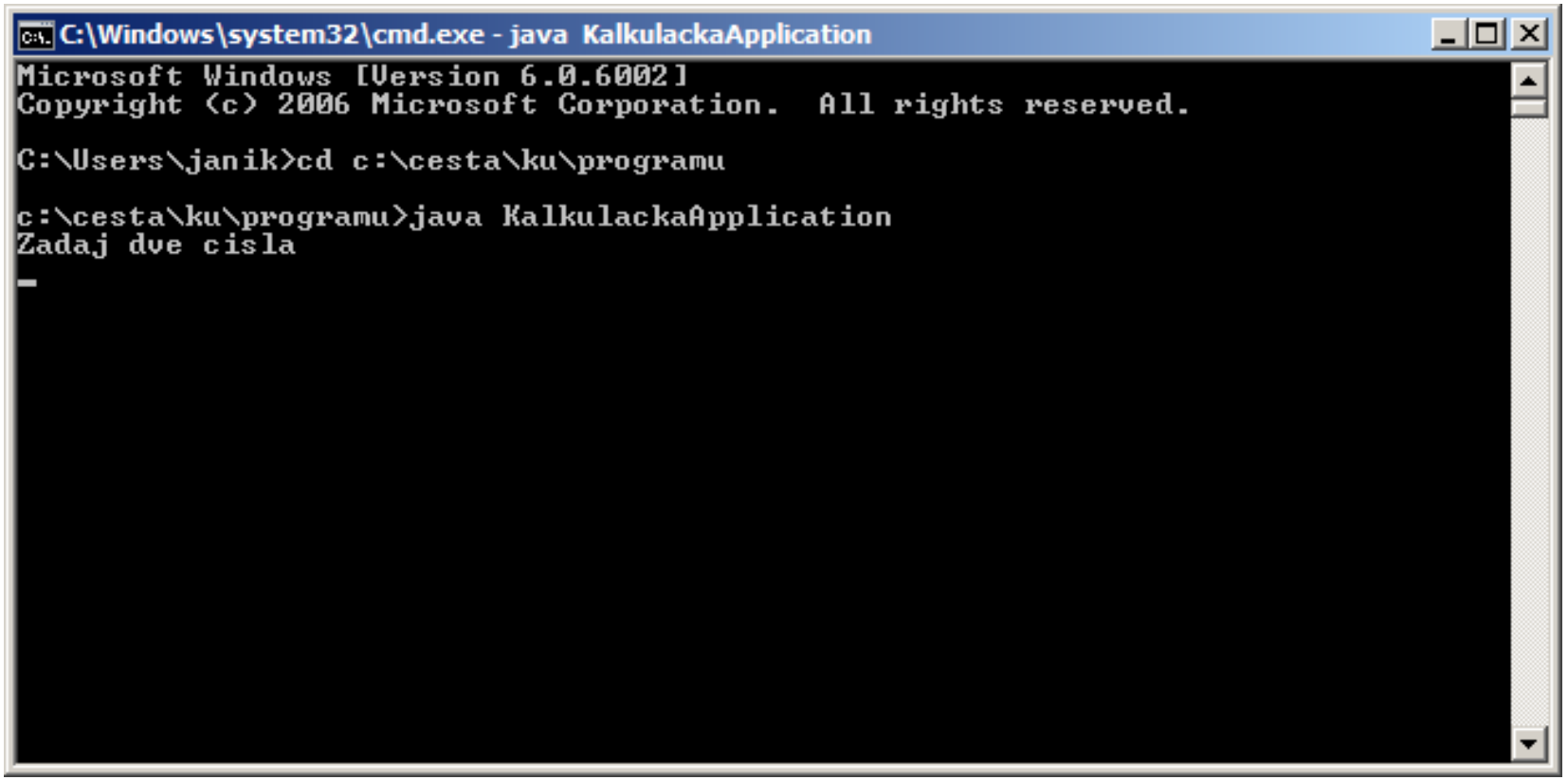
- trieda Scanner
  - rovnako ako čítanie zo súboru
- objekt System.in ako parameter konštruktora

```
Scanner vstup = new Scanner(System.in);
```

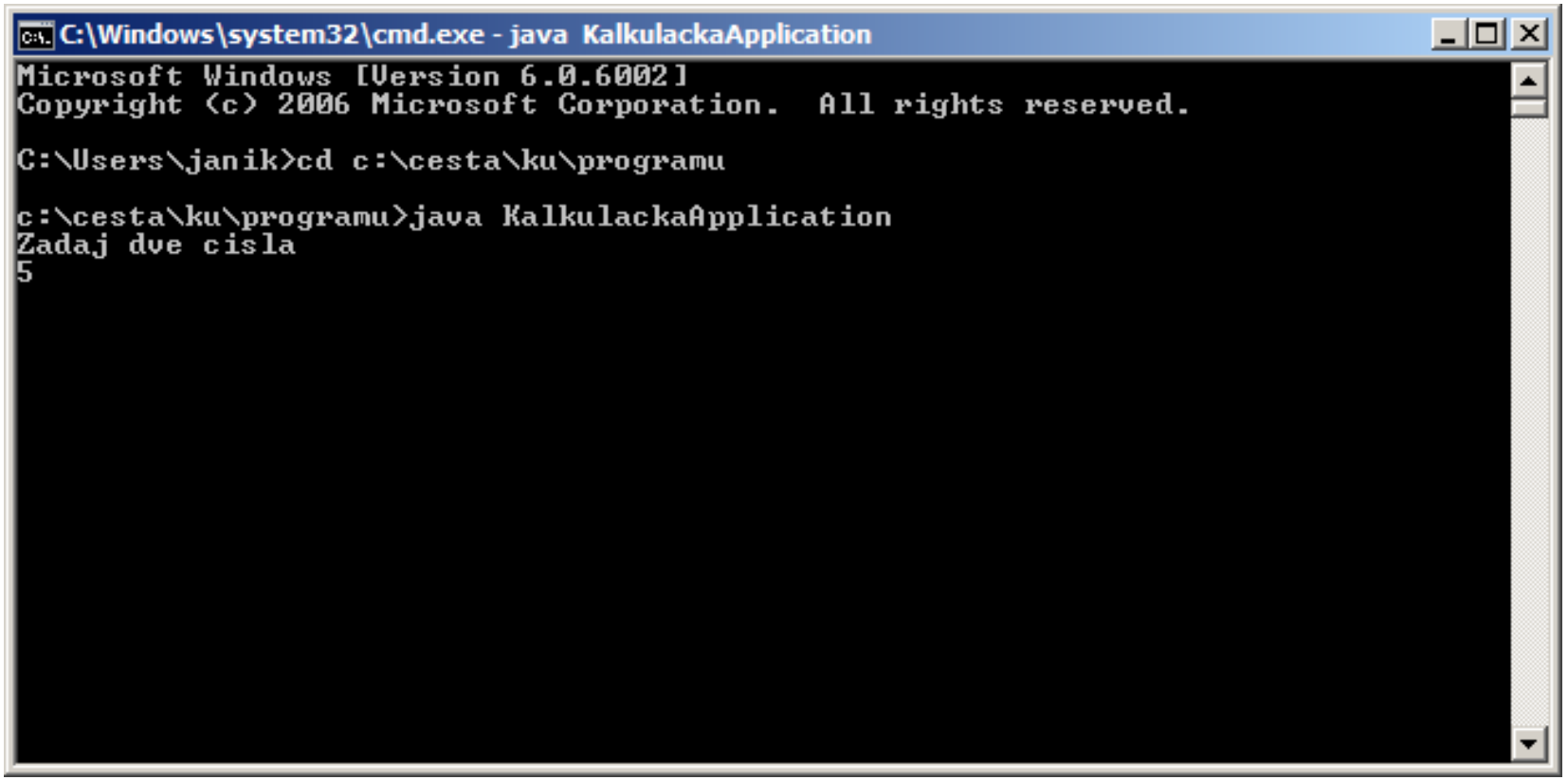
Kalkulačka

```
import java.util.Scanner;
public class KalkulackaApplication
{
    public static void main(String[] paParametre)
    {
        Scanner vstup = new Scanner(System.in);
        System.out.println("Zadaj dve cisla");
        int prve = vstup.nextInt();
        int druhe = vstup.nextInt();
        System.out.println("Sucet je " + (prve + druhe));
    }
}
```

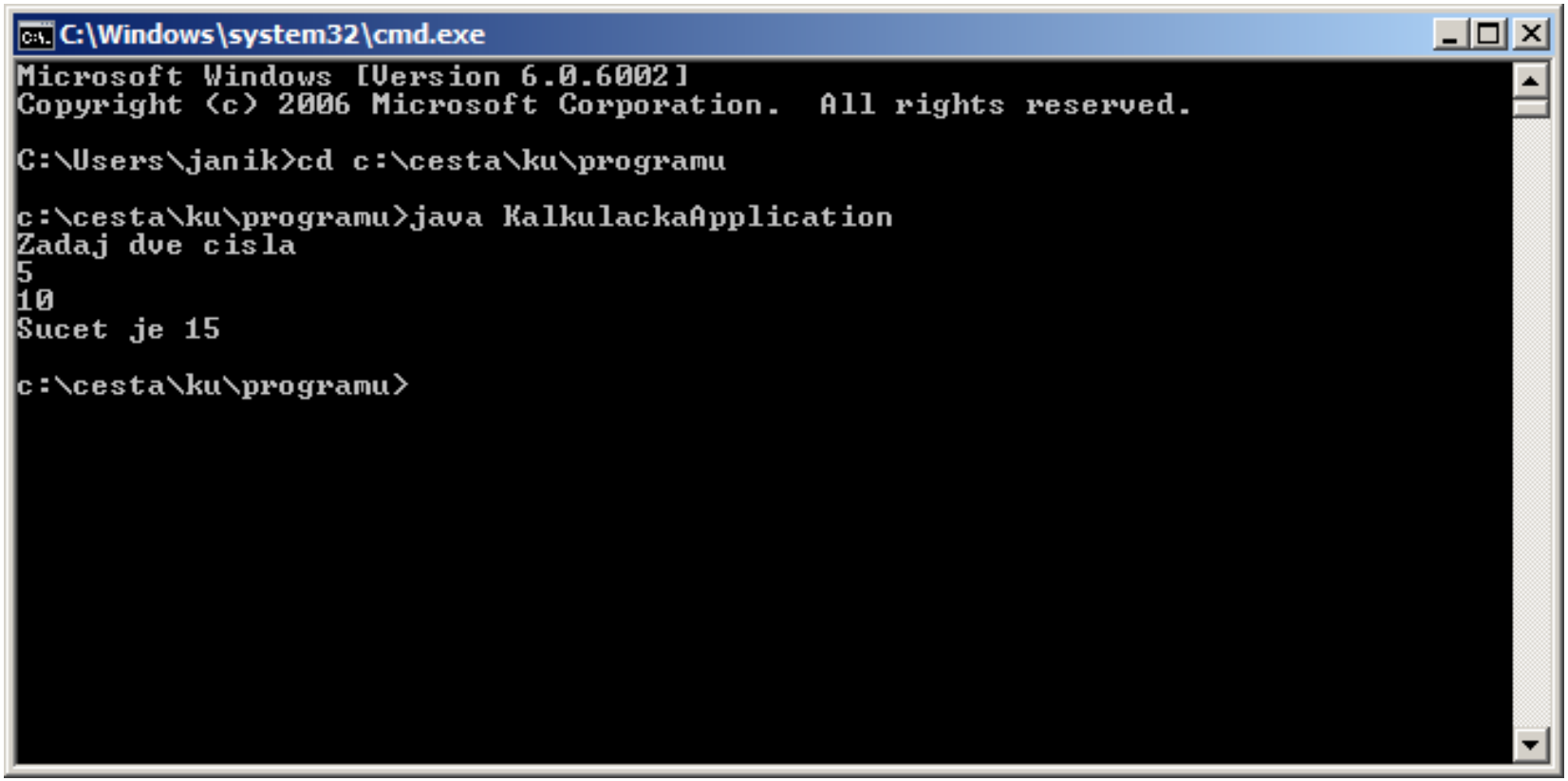
Kalkulačka výstup(1)



Kalkulačka výstup(2)



Kalkulačka výstup(3)



Vďaka za pozornosť