Informatika 1

Objekt trieda

doc. Ing. Ján Janech, PhD.
27. 11. 2023

ŽILINSKÁ UNIVERZITA V ŽILINE
Fakulta riadenia
a informatiky

Pojmy zavedené v 8. prednáške (1)

- kontajnery s pevným počtom prvkov

- pole
  - definícia
  - vytvorenie a inicializácia
  - práca s poľom ako celkom
  - práca s prvkami poľa

  - dĺžka poľa – length

Pojmy zavedené v 8. prednáške (2)

- N-rozmerné polia
  - Dvojrozmerné polia – matica
  - definícia – typ[][] premenna
  - inicializácia – new typ[pocetRiadkov][pocetStlpcov]
  - práca s prvkami – premenna[riadok][stlpec]
- pole polí
  - inicializácia – new typ[pocetRiadkov][]
  - inicializácia prvkov
    – pole[riadok] = new typ[pocetStlpcov]

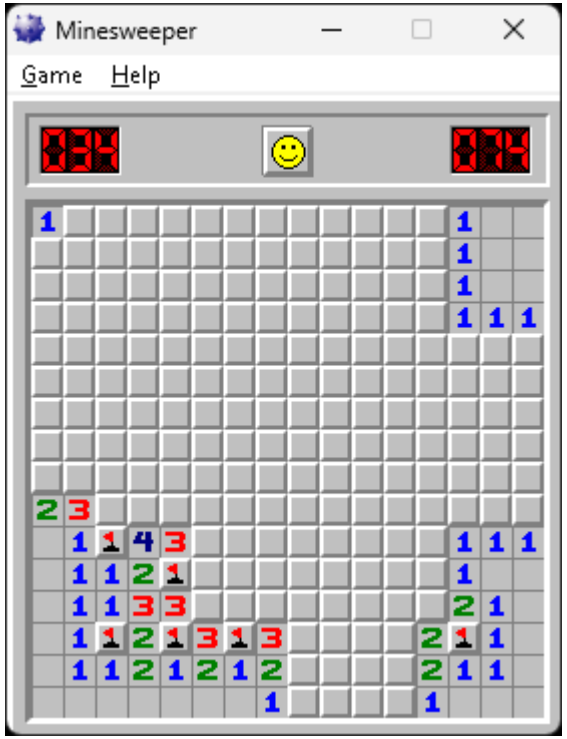
Cieľ prednášky

- trieda ako objekt
- trieda ako množina svojich inštancií
  - enum

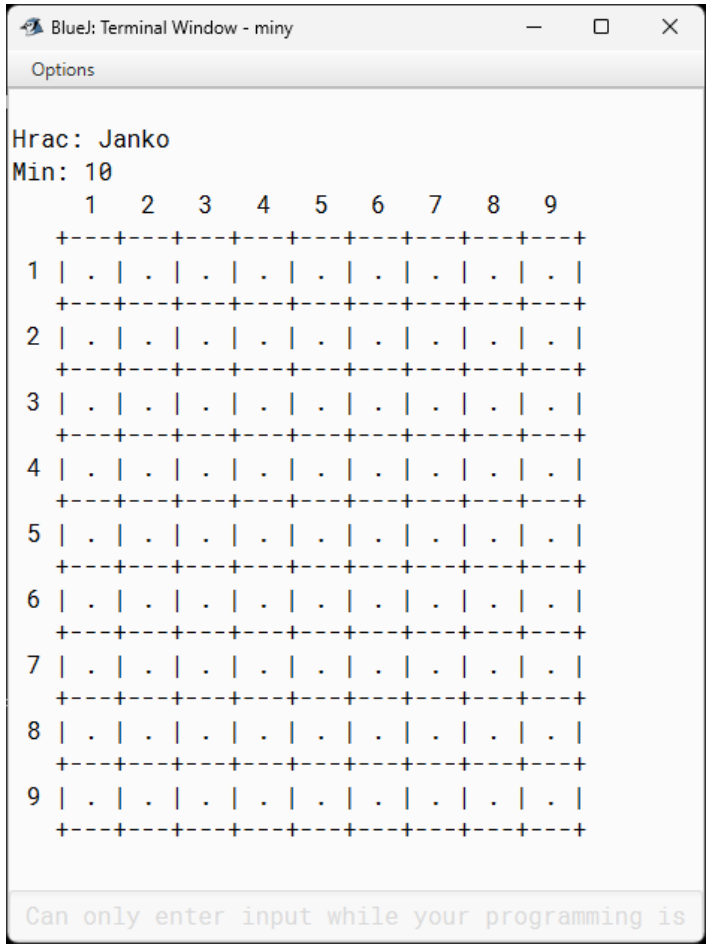
- príklad: míny

Projekt miny – zadanie

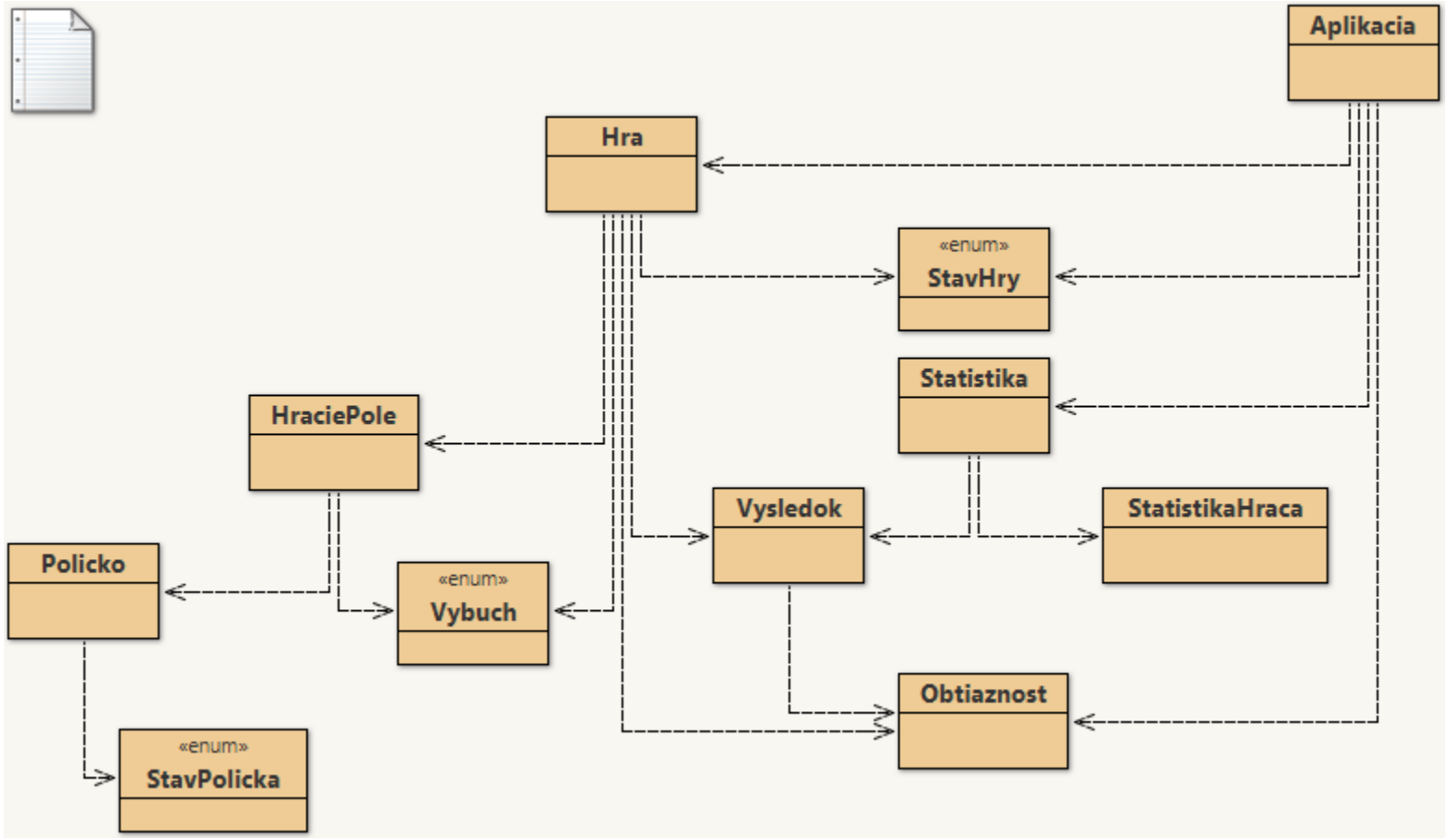
- známa logická hra míny
- cieľom hry Míny je určiť umiestnenia všetkých mín tak, aby ste pritom žiadnu z nich neodkryli
- ak odkryjete mínu, prehrávate



Projekt miny – grafická reprezentácia



BlueJ – diagram tried



Trieda Aplikacia – zadanie

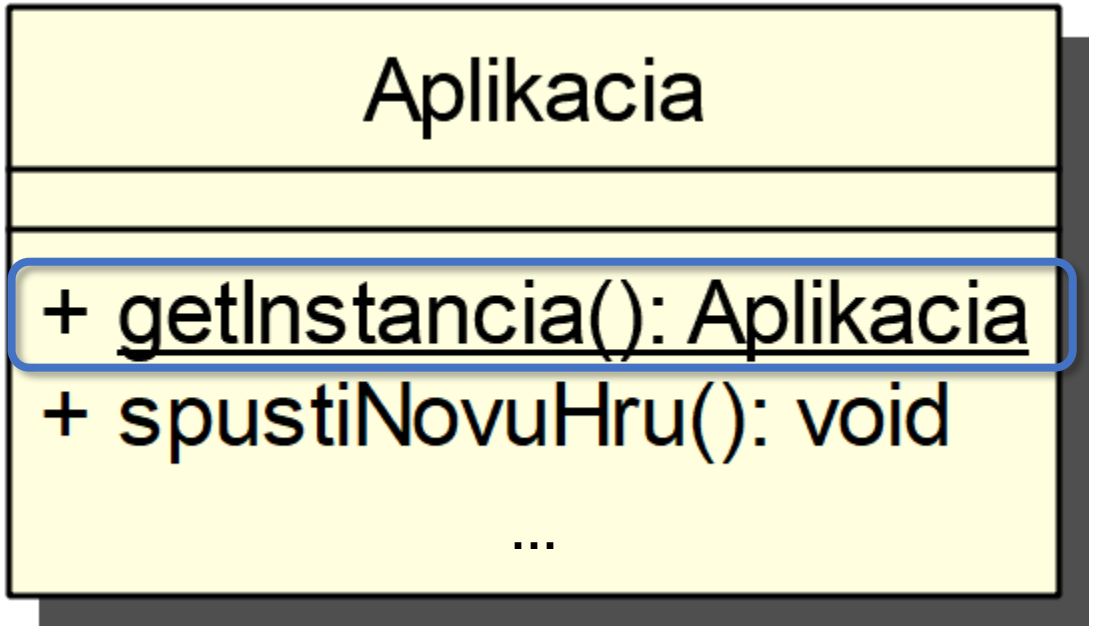
- prístup ku hre míny
- len jedna inštancia
- zmena mena hráča
- zmena obtiažnosti
- spustenie novej hry
- odkrytie políčka a označenie míny
- poskytovanie informácií o stave hry (výhra/prehra)
- zobrazenie štatistiky
- ...

Trieda Aplikacia

- jediná inštancia
- správa new – ľubovoľný počet inštancií
- nová správa triede – getInstancia
- správa new – nesmie byť vo verejnom rozhraní

- Návrhový vzor Singleton – Jedináčik
- Projekt „tvary“ – trieda Platno

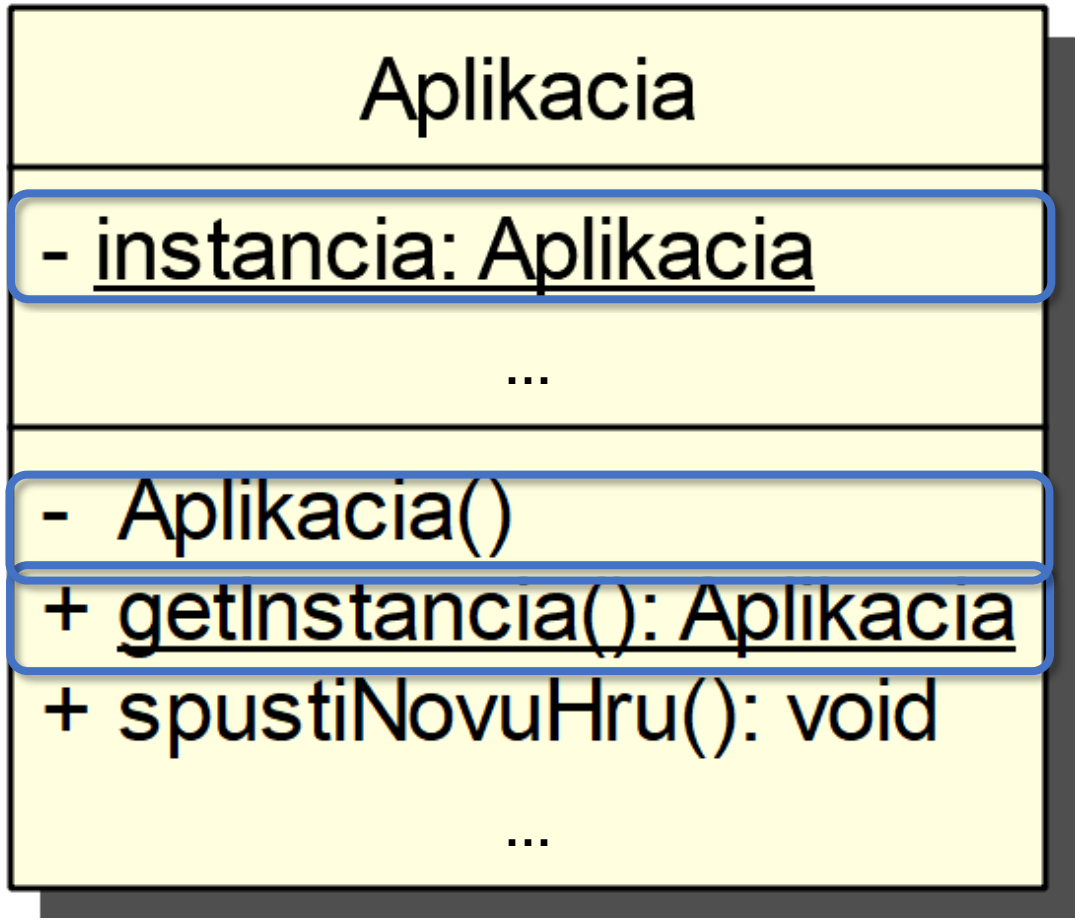
Trieda Aplikacia – rozhranie



Trieda Aplikacia – vnútorný pohľad (1)

- nová metóda triedy
  - reakcia na správu triede Aplikacia.getInstancia()
- nový atribút triedy
  - instancia – jediná inštancia triedy
- konštruktor označený ako private
  - označenie, že trieda nemá verejnú správu new

Trieda Aplikacia – vnútorný pohľad (2)



Aplikacia – trieda

```
public class Aplikacia {
    private static Aplikacia instancia;

    private Aplikacia() {
        ...
    }
```

Aplikacia – metóda getInstancia

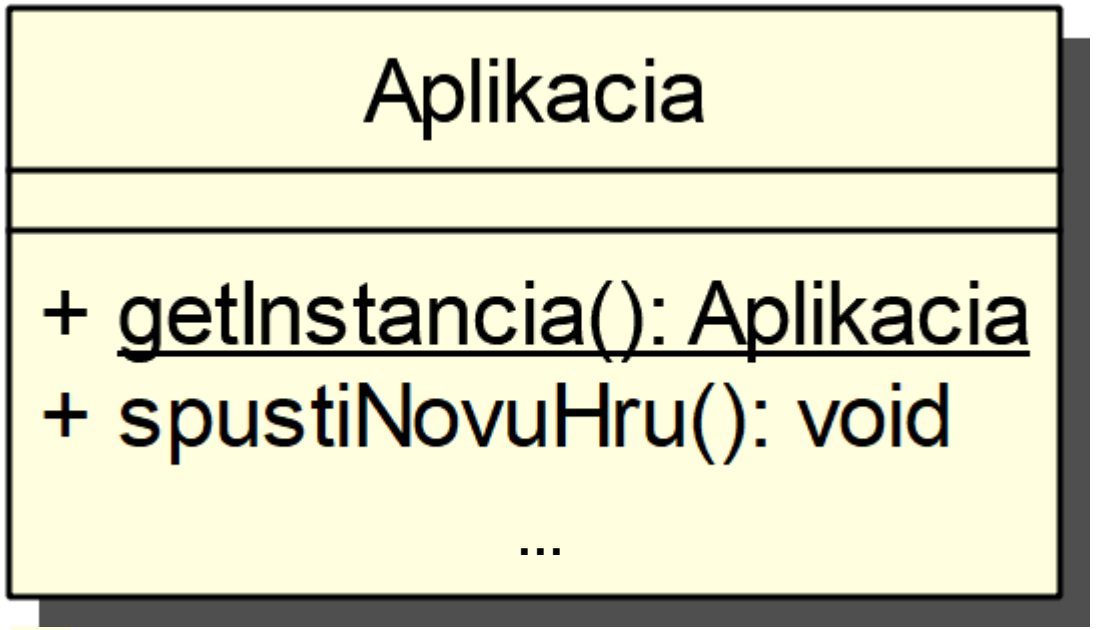
```
public static Aplikacia getInstancia() {
    if (Aplikacia.instancia == null) {
        Aplikacia.instancia = new Aplikacia();
    }

    return Aplikacia.instancia;
}
```

Trieda ako objekt

- vonkajší pohľad
  - rozhranie triedy – správy triede
    - doteraz len správa new
- vnútorný pohľad
  - atribúty triedy
  - metódy triedy

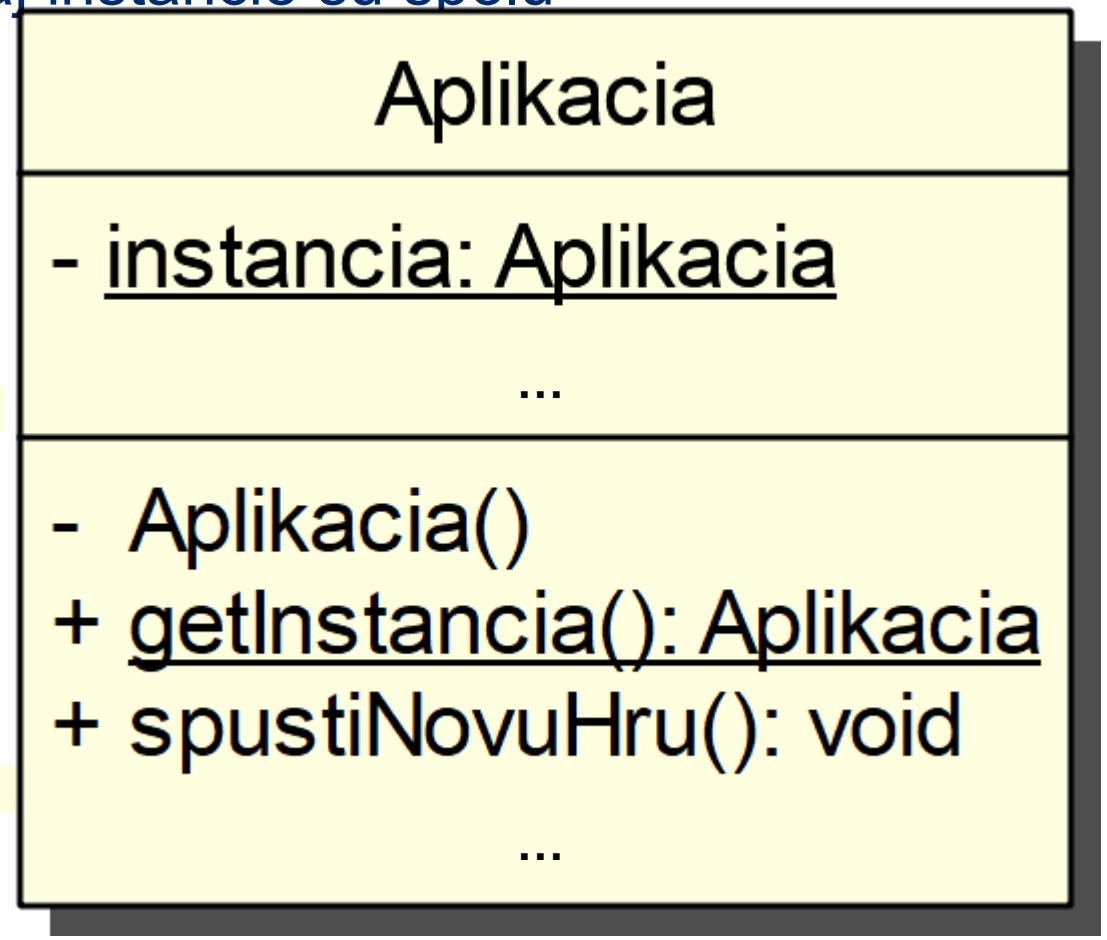
Trieda ako objekt v UML (1)

- správy triede a správy inštancii – spoločné rozhranie
- správy triede – podčiarknuté



Trieda ako objekt v UML (2)

- vnútorný pohľad – atribúty a metódy triedy aj inštancie sú spolu
- atribúty a metódy triedy – podčiarknuté



Trieda ako objekt v jazyku Java

- rovnako ako v UML – atribúty a metódy triedy aj inštancie sú spolu v definícii triedy
- rozlíšenie – kľúčové slovo static

```
private static Aplikacia instancia;
```

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

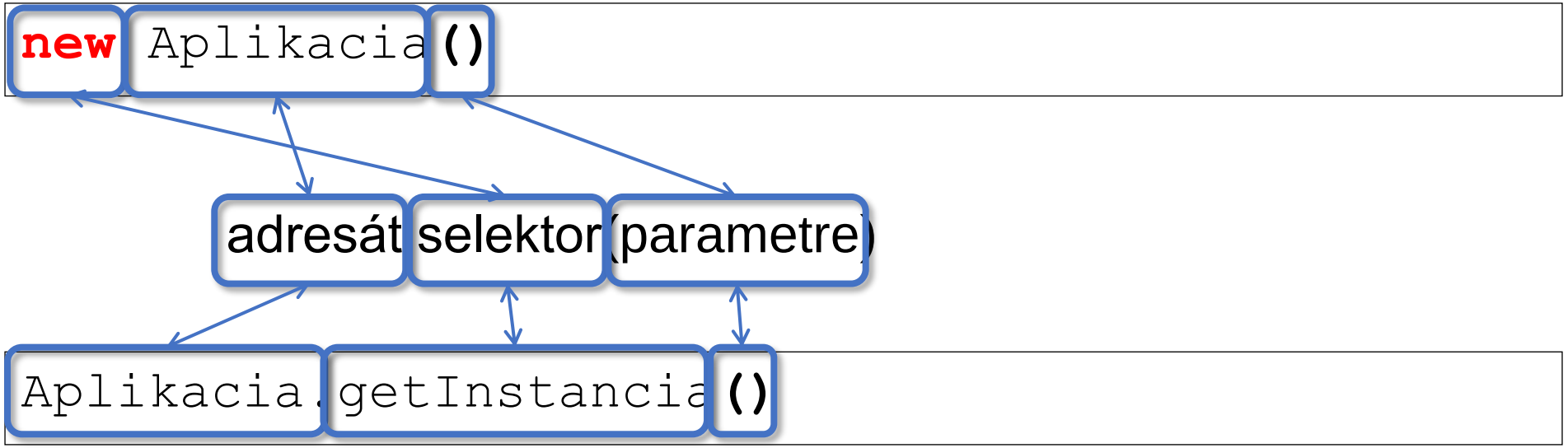
Poradie definícií v triede – konvencia

- konvencia zavedená firmou Sun
  1. atribúty triedy
  2. atribúty inštancie
  3. konštruktory
  4. metódy triedy
  5. metódy inštancie

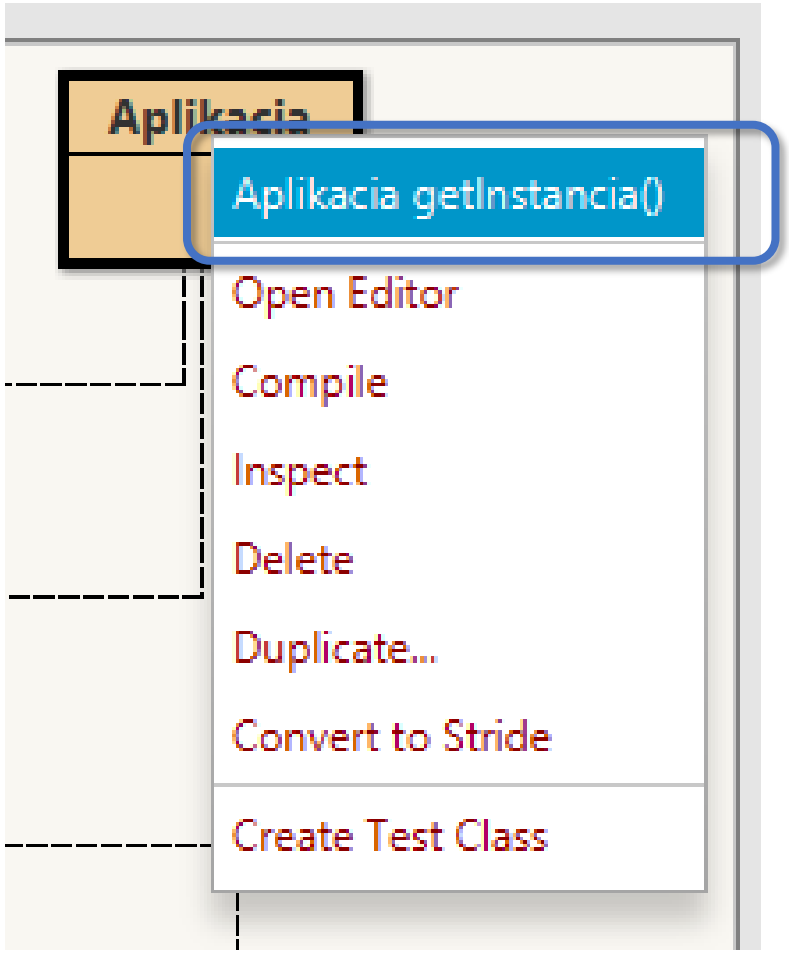
     a) verejné zložky – public
     b) neverejné zložky – private

Posielanie správy triede v jazyku Java

- ako adresát správy sa uvádza trieda
- špeciálna správa new – špeciálny zápis
- ostatné správy – štandardný zápis



Posielanie správy triede v nástroji BlueJ



Inštancie a trieda – vzájomný prístup

- trieda môže svojim inštanciám posielať správy zo súkromného rozhrania
- inštancia môže svojej triede posielať správy zo súkromného rozhrania

- trieda môže priamo pristupovať ku atribútom svojich inštancií
- inštancia môže priamo pristupovať ku atribútom svojej triedy

Súkromný konštruktor

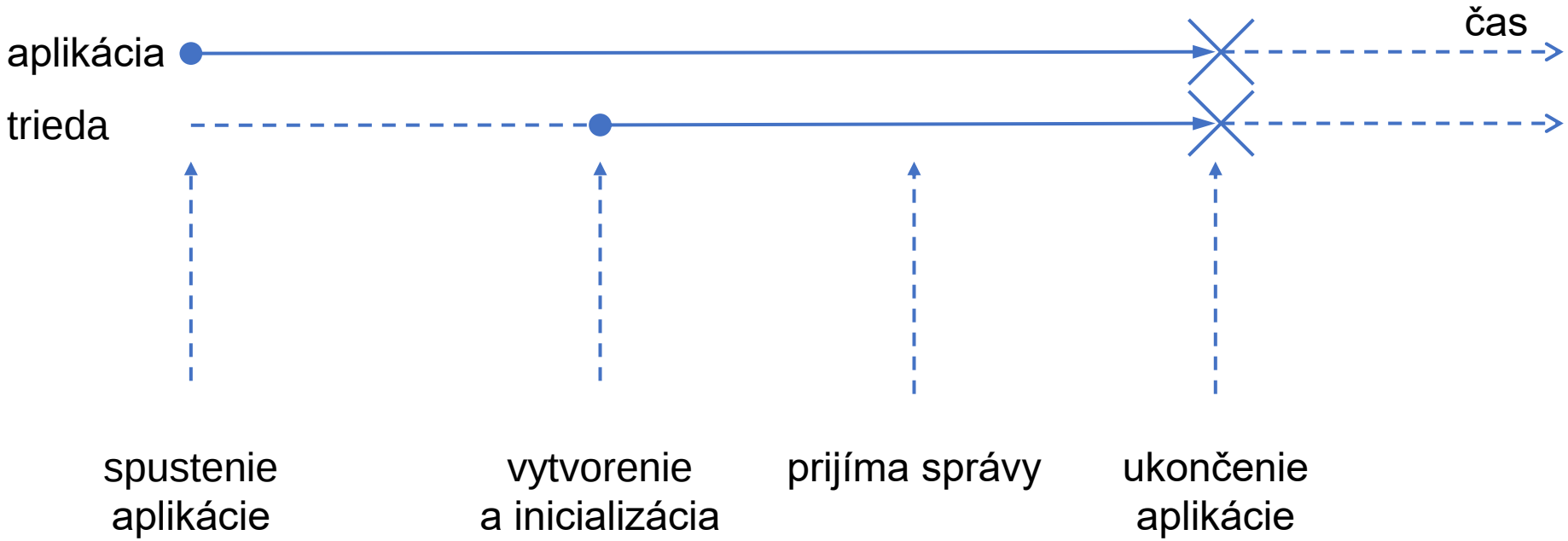
- private konštruktor – správa new v súkromnom rozhraní
- ak má trieda len súkromný konštruktor, musí mať metódu triedy, ktorá sa stará o vytvorenie

Jedináčik

```
public class Jedinacik {
    private static Jedinacik instancia;
    private Jedinacik() {
    }
    public static Jedinacik getInstancia() {
        if (Jedinacik.instancia == null) {
            Jedinacik.instancia = new Jedinacik();
        }
        return Jedinacik.instancia;
    }
}
```

Životný cyklus triedy v jazyku Java

- trieda je priamo definovaný objekt
- vzniká pri prvom použití (prvá správa triede)
- zaniká spolu s ukončením aplikácie



Jedináčik – iná implementácia

```
public class Jedinacik {
    private static Jedinacik instancia = new Jedinacik();

    private Jedinacik() {
    }

    public static Jedinacik getInstancia() {
        return Jedinacik.instancia;
    }
}
```

Míny – stav hry

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

Vyjadrenie stavu hry (1)

- nový atribút „vyhral“
  - true – hráč vyhral
  - false – hráč prehral

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

Vyjadrenie stavu hry (2)

- ďalší nový atribút „hraSkoncila“
  - false – hra ešte neskončila
  - true – hra už skončila, výsledok je vo „vyhral“

- problémy:
  1. ak hra ešte neskončila, dotazom na „vyhral“ dostaneme vždy nesprávnu odpoveď

Analýza možných stavov hry

- možné stavy:
  - nerozhodnutá – hra ešte neskončila
  - výhra – hráč vyhral
  - prehra – hráč stupil na mínu

- záver: dve hodnoty nestačia, treba tri

Vyjadrenie stavu hry typom int

- 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 typom String

- 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. pozor na porovnávanie – equals
  3. programátorská hrdosť nedovolí také primitívne riešenie

Vyjadrenie stavu hry enumom

- riešenie – vymenovaný typ – enum

- enum – trieda s konštantnou extenziou.

- extenzia triedy – množina všetkých inštancií triedy.

- 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

Konvencia pre názvy inštancií

- veľká podčiarkovníková notácia
  - všetky písmená veľké
  - slová oddelené podčiarkovníkom

```
STANICNA_KOLAJ
TRATOVA_KOLAJ
MANIPULACNA_KOLAJ
VYHYBKA
NAVESTIDLO
```

Prístup k inštanciám enumu

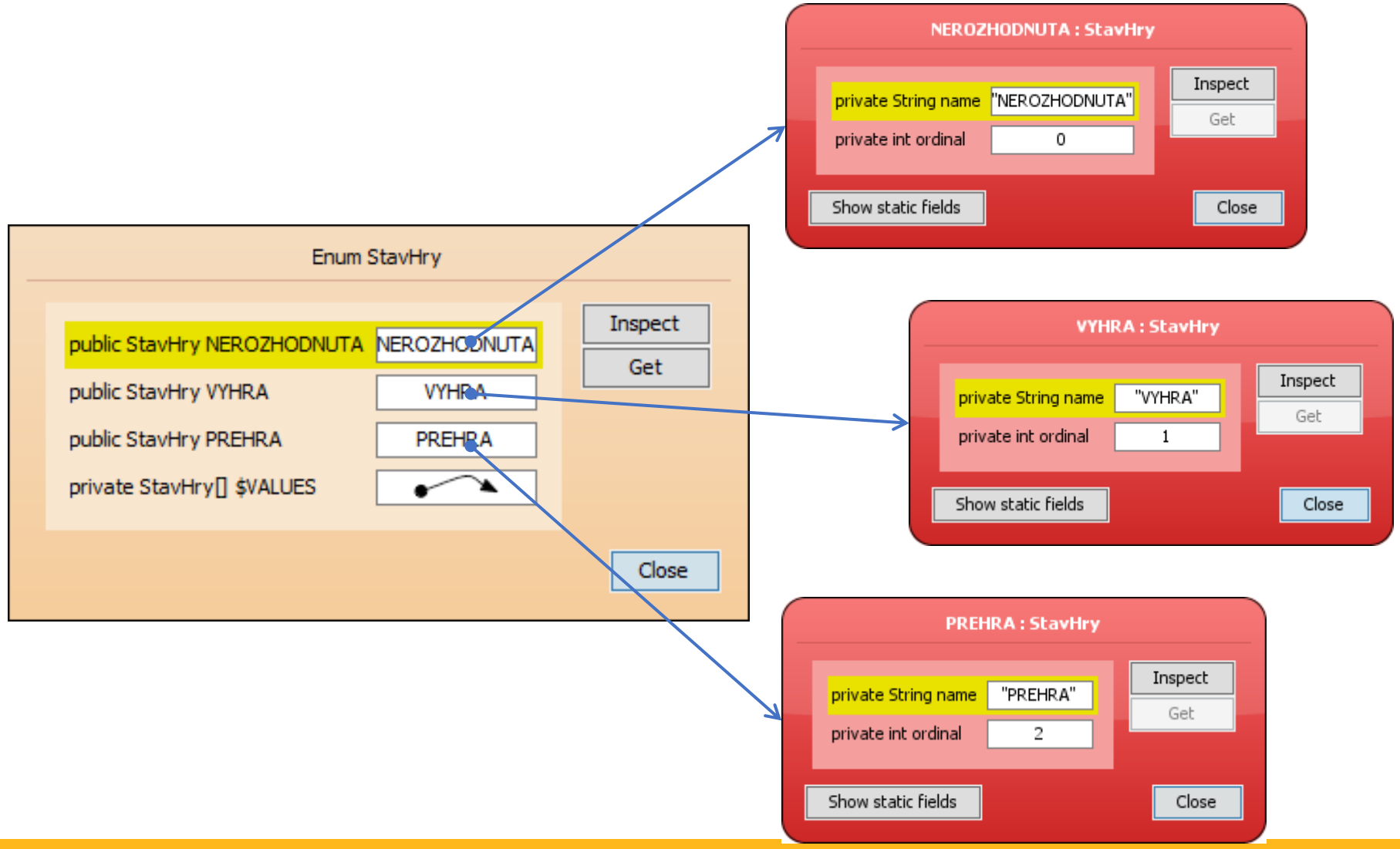
```
NazovEnumu.NAZOV_INSTANCIE
```

- vráti referenciu na inštanciu enumu
- vždy vráti tú istú referenciu (konštantná extenzia)

```
NazovEnumu.values()
```

- správa triede
- vráti zoznam všetkých inštancií (extenziu) vo forme poľa

Enum StavHry v BlueJ



Použitie enum StavHry – trieda Hra (1)

```
public class Hra {
    private StavHry stav;

    public Hra(...) {
        this.stav = StavHry.NEROZHODNUTA;
        ...
```

Použitie enum StavHry – trieda Hra (2)

```
if (this.stav == StavHry.VYHRA) {
    ...
}
...
System.out.println(this.stav);
```

Vetvenie pomocou enum

- porovnanie inštancií enum na rovnosť referencií

- podmienené vykonanie:

```
if (this.stav == StavHry.VYHRA) {
     ...
}
```

Vetvenie pomocou enum – switch

```
switch (this.stav) {
    case VYHRA:
        ...
        break;
    case PREHRA:
        ...
        break;
    case NEROZHODNUTA:
        ...
        break;
}
```

Míny – stav políčka

- informácie o stave políčka
  - enum
  - getReprezentacia() vracia reťazec, ktorý sa má vypísať do terminálu

Enum – špeciálna trieda

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

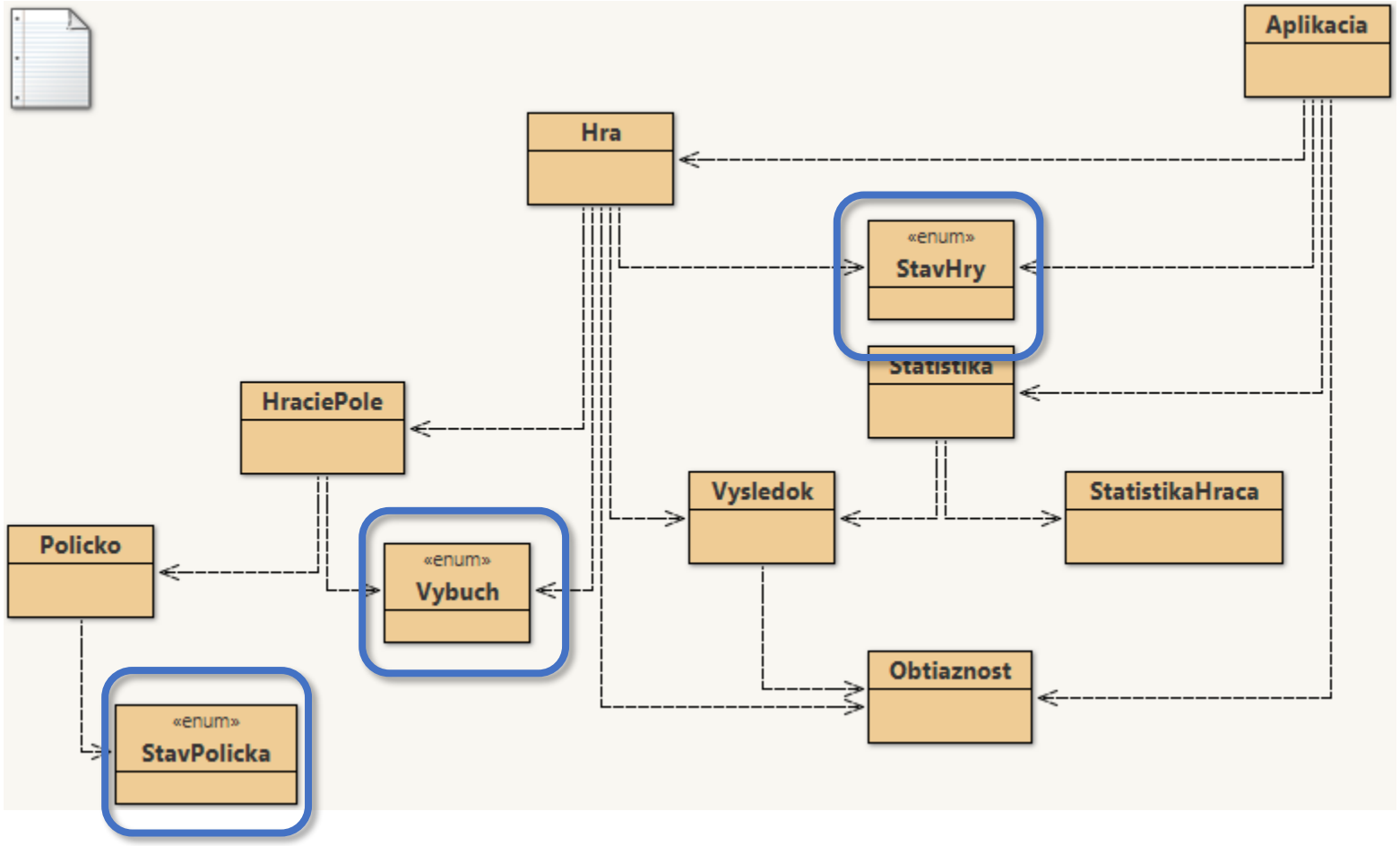
Enum – StavPolicka (1)

```
private char reprezentacia;

StavPolicka(char reprezentacia) {
    this.reprezentacia = reprezentacia;
}

public char getReprezentacia() {
    return this.reprezentacia;
}
```

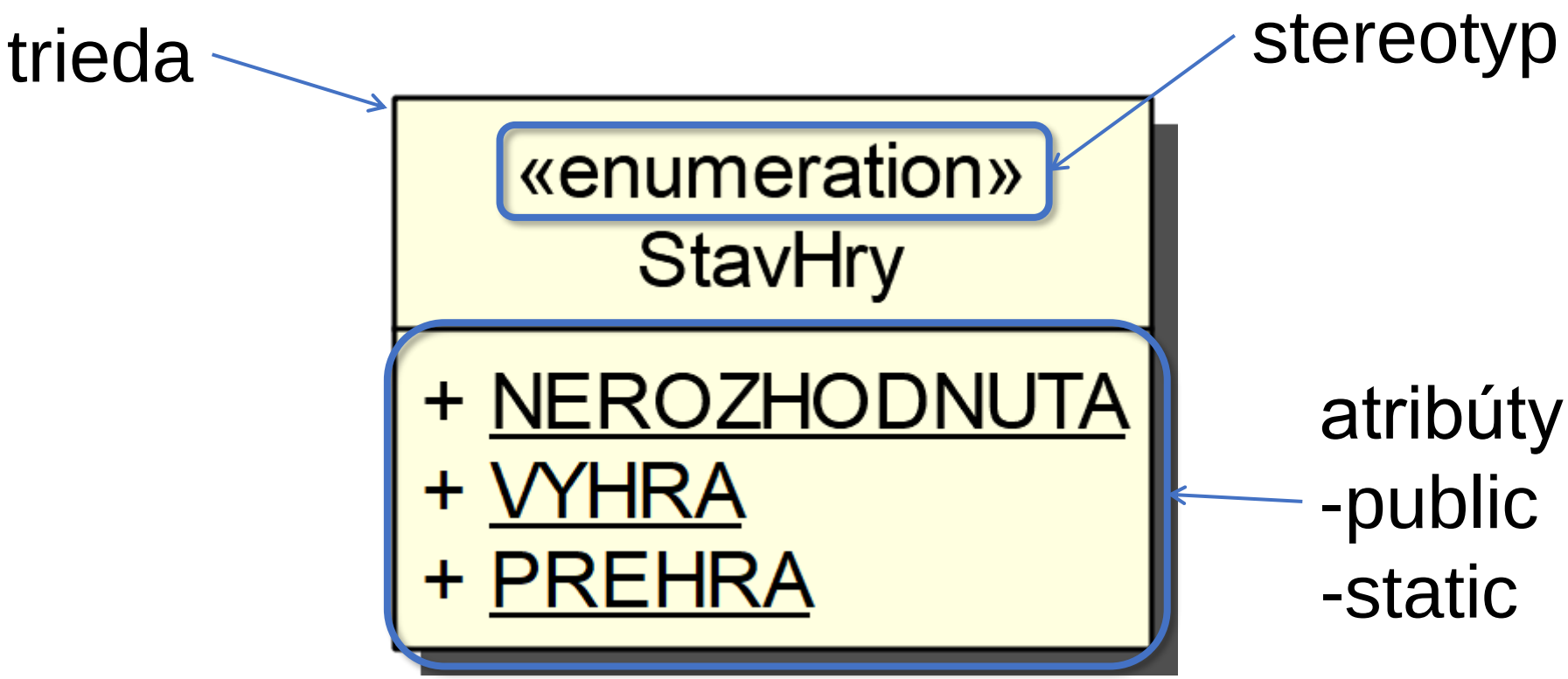
Enum – StavPolicka (2)

```
public enum StavPolicka {
     ZAKRYTE('.'),
     PRAZDNE(' '),
     OZNACENE('F'),
     ODKRYTE(' '),
     UKAZANA_MINA('+'),
     VYBUCHNUTE('*');

     ...
}
```

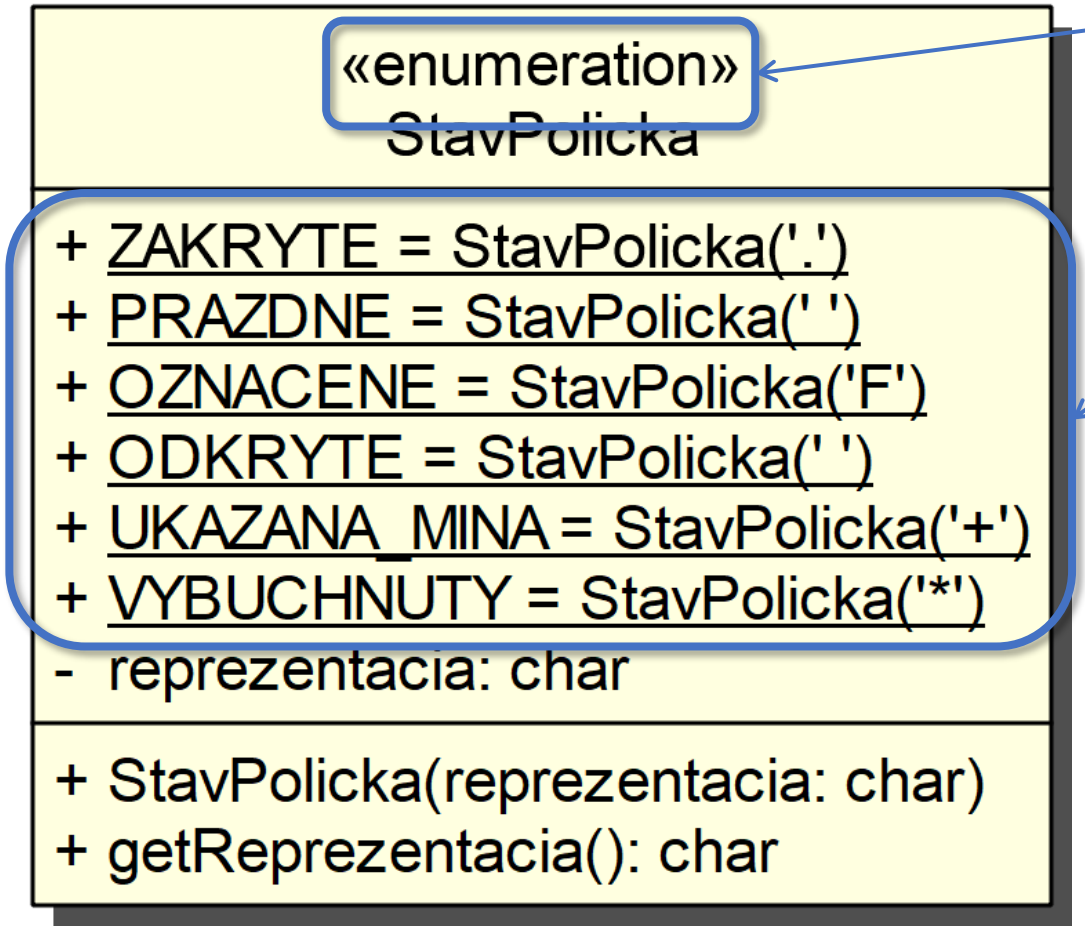
BlueJ – diagram tried



Jednoduchý enum v UML



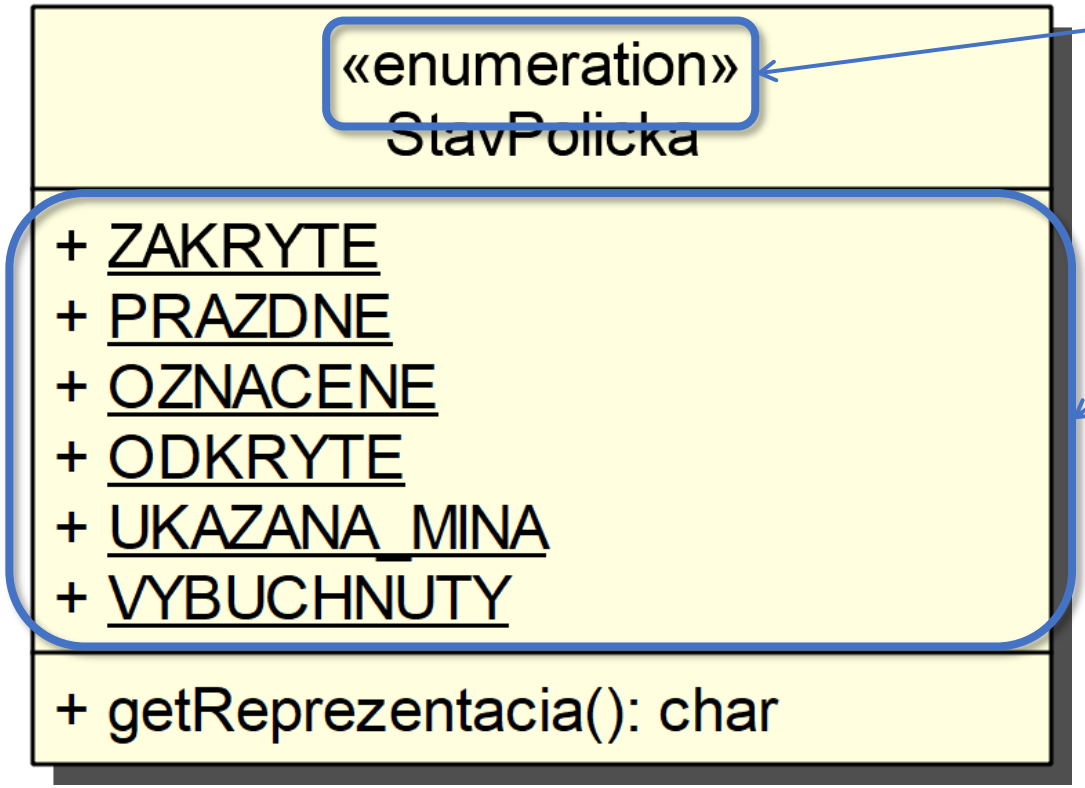
Zložitejší enum v UML – vnútorný pohľad



stereotyp

atribúty
-public
-static

Zložitejší enum v UML – vonkajší pohľad



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
   - životný cyklus

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