



Polymorfizmus III

Pojmy zavedené v 4. prednáške

- význam interface pre polymorfizmus
 - statický typ
- testovanie tried pre priamu komunikáciu s hráčom
 - polymorfizmus terminálu – interface
- pretypovanie

Ciel' prednášky

- interface
 - implementácia viac interface v jednej triede
 - rozširovanie jedného interface v inom interface
- pretypovanie – pokračovanie
- balíčky – štruktúra knižníc
- príklad: World of FRI

„World of FRI“ – úloha 1

- nový prvak hry – vrecková čierna diera
 - hráč ju môže zobrať do inventára
 - hráč ju môže položiť do miestnosti
 - hráč môže vojsť do čiernej diery

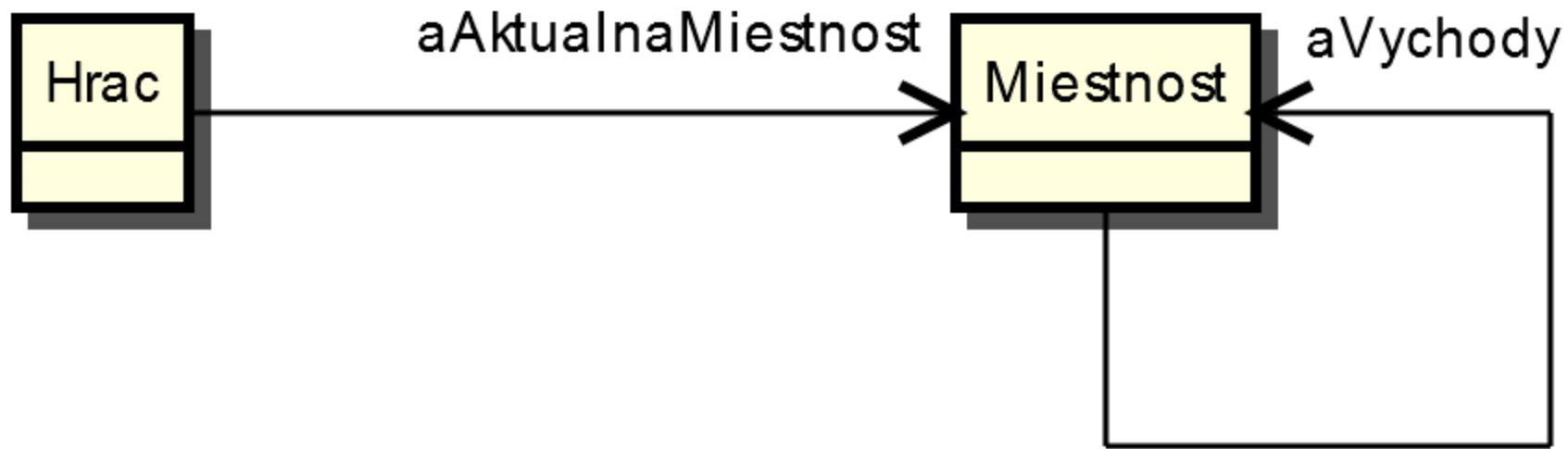
Vrecková čierna diera

- je to miestnosť – dá sa do nej vojsť?
- je to predmet – dá sa zdvihnúť, položiť?
- oboje je možné

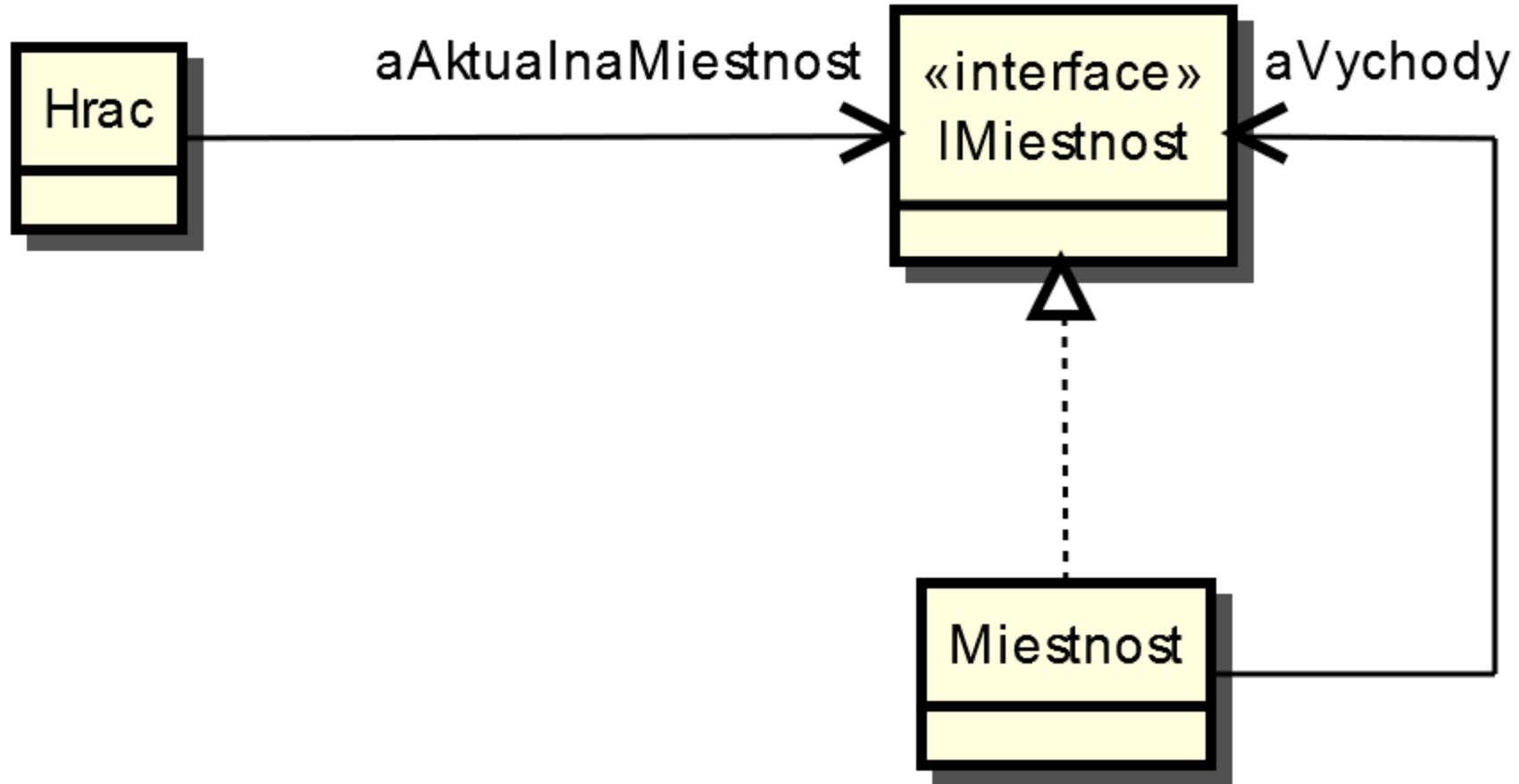
Riešenie

- čierna diera je
 - predmet – implementácia interface IPredmet
 - miestnosť – implementácia interface?
- interface pre miestnosti

Súčasný stav



Interface IMiestnost



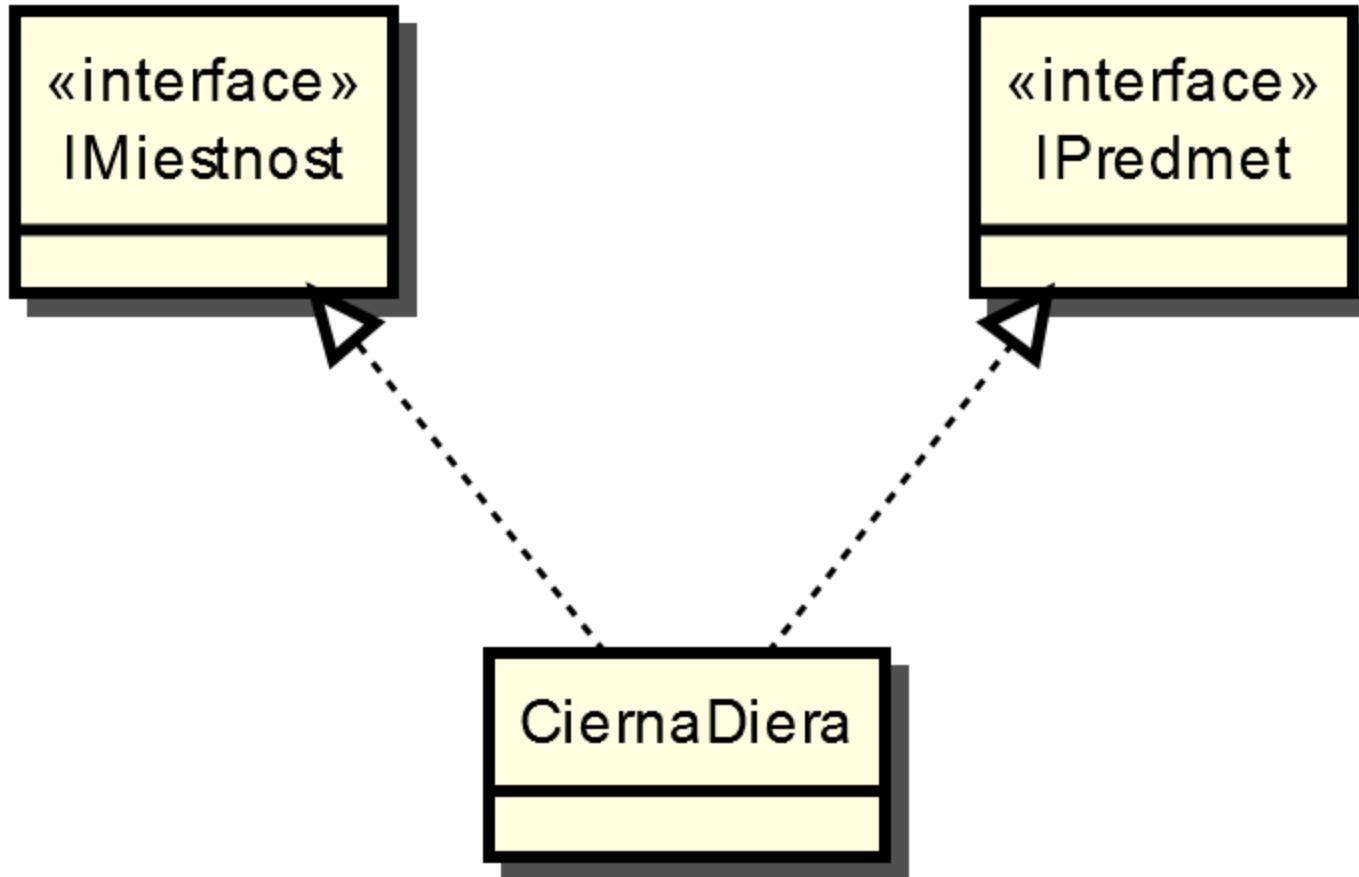
Interface IMiestnost

```
public interface IMiestnost
{
    IMiestnost dajVychod(String paSmer);
    String dajUplnyPopis();
    IPredmet zoberPredmet(String paNazov);
    void polozPredmet(IPredmet paPredmet);
}
```

Dôsledky zavedenia IMiestnost

- trieda Hrac
 - aktuálna miestnosť je typu IMiestnost
- trieda Miestnost
 - východy sú typu IMiestnost
- trieda MapaHry
 - východzia miestnosť je typu IMiestnost

Vrecková čierna diera



Implementácia viac interface

- trieda môže implementovať ľubovoľný počet interface (0...n)
- kľúčové slovo **implements** len jeden krát
- oddelovač – čiarka

```
public class VreckovaCiernaDiera  
    implements IPredmet, IMiestnost
```

Trieda CiernaDiera₍₁₎

```
public class CiernaDiera
    implements IPredmet, IMiestnost
{
    private String aNazov;

    public CiernaDiera(String paNazov)
    {
        aNazov = paNazov;
    }
    ...
}
```

Trieda CiernaDiera₍₂₎

```
public IMiestnost dajVychod(String paSmer)
{
    return null;
}

public String dajUplnyPopis()
{
    return "Cierna diera. Rutis sa rychlostou svetla
           ku horizontu udalosti.";
}
```

Trieda CiernaDiera₍₃₎

```
public IPredmet zoberPredmet(String paNazov)
{
    return null;
}

public void polozPredmet(IPredmet paPredmet)
{
    System.out.println("Flunk... Cierna diera zhltla
                        predmet " + paPredmet.dajNazov());
}
```

Trieda CiernaDiera₍₄₎

```
public String dajNazov()
{
    return aNazov;
}

public String dajPopis()
{
    return "Nieco cierne... derave... waaa, cierna diera";
}

public String pouzi(Hrac paHrac)
{
    return "Ako by si ju asi tak chcel pouzit?";
}
```

Priebeh hry

BlueJ: Terminal Window - zuul-12-ciernaDiera02

Options

```
terasa - hlavny vstup na fakultu
Vychody: juh sever vychod
Predmety: kamen

> chod juh
pocitacove laboratorium
Vychody: sever vychod
Predmety: vreckovaCiernaDiera

> chod vreckovaCiernaDiera
Tam nie je vychod!
>
```

Výsledok implementácie IMiestnost

- čierna diera môže byť používaná aj ako predmet aj ako miestnosť
- do čiernej diery sa však nedá vojsť
 - trieda Miestnost nevie, že predmet typu CiernaDiera je tiež východ
 - treba zmeniť implementáciu dajVychod

dajVychod v triede Miestnost

- stará implementácia

```
public IMiestnost djayVychod(String paSmer)
{
    return aVychody.get(paSmer);
}
```

- vráti východ daným smerom
- ak neexistuje, vráti null

Nová implementácia dajVychod₍₁₎

- zistí, či existuje predmet s daným názvom
- ak áno:
 - pretypuje ho na IMiestnosť a vráti
- ak nie:
 - vráti východ daným smerom
 - ak neexistuje, vráti null

Nová implementácia dajVychod₍₂₎

```
public IMiestnost dajVychod(String paSmer)
{
    if (aPredmety.containsKey(paSmer)) {
        IPredmet predmet = aPredmety.get(paSmer);
        return (IMiestnost)predmet;
    }
    return aVychody.get(paSmer);
}
```

Priebeh hry

```
BlueJ: Terminal Window - zuul-12-ciernaDiera03b
Options

terasa - hlavny vstup na fakultu
Vychody: juh sever vychod
Predmety: kamen

> chod juh
pocitacove laboratorium
Vychody: sever vychod
Predmety: vreckovaCiernaDiera

> chod vreckovaCiernaDiera
Cierna diera. Rutis sa rychlosťou svetla ku horizontu
>
```

Vojdenie do kameňa

The screenshot shows a BlueJ IDE interface with a terminal window titled "BlueJ: Terminal Window - zuul-12-ciernaDiera03b". The terminal window displays the following text:

```
terasa - hlavny vstup na fakultu
Vychody: juh sever vychod
Predmety: kamen
> chod kamen
```

A blue rounded rectangle highlights the line "Predmety: kamen". Below the terminal window, the Java code editor shows:

```
java.lang.Object
    ↗ java.lang.ClassCastException:
        at Main.main()
        at Main.main()
        at Main.main()
```

A blue rounded rectangle highlights the error message "java.lang.ClassCastException: Predmet cannot be cast to IMiestnost".

Oprava implementácie dajVychod

```
public IMiestnost dajVychod(String paSmer)
{
    if (aPredmety.containsKey(paSmer)) {
        IPredmet predmet = aPredmety.get(paSmer);
        if (predmet instanceof IMiestnost) {
            return (IMiestnost)predmet;
        }
    }
    return aVychody.get(paSmer);
}
```

Operátor instanceof₍₁₎

prvyOperand **instanceof** druhýOperand

- prvyOperand – hodnota objektového typu
- druhýOperand – typ (trieda, enum, interface)

Operátor instanceof₍₂₎

- vracia true ak
 - je prvyOperand inštanciou danej triedy
 - je prvyOperand inštanciou daného enumu
 - je prvyOperand inštanciou triedy implementujúcej daný interface
- kontroluje typovú kompatibilitu dynamického typu so zadaným typom

Bezpečné pretypovanie₍₁₎

- operácia pretypovania funguje iba ak
 - je hodnota inštanciou danej triedy
 - je hodnota inštanciou daného enumu
 - je hodnota inštanciou triedy implementujúcej daný interface
- v opačnom prípade zlyhá a vyhodí behovú chybu
- => nutnosť testovania pomocou operátora `instanceof`

Bezpečné pretypovanie₍₂₎

```
if (premenna instanceof Typ) {  
    Typ pretypovana = (Typ)premenna;  
    // prikazy po bezpecnom pretypovani  
}
```

- príkazy sa vykonajú len ak sa dá premenná pretypovať na daný typ

Interface – štandardná knižnica Javy₍₁₎

- Java knižnica – triedy a interface
- mnohé triedy implementujú interface
- ArrayList – usporiadaný zoznam prvkov
- HashSet – (neusporiadana) množina prvkov

Interface – štandardná knižnica Javy₍₂₎

ArrayList<Predmet> aPredmety

alebo

HashSet<Predmet> aPredmety

- v metódach triedy – parametre a lokálne premenné rovnakého typu
- čo treba zmeniť, ak zmeníme typ z ArrayList na HashSet (alebo naopak)?
 - všetky definície premenných
- je to možné zjednodušiť?

Interface – štandardná knižnica Javy₍₃₎

- čo majú ArrayList aj HashSet spoločné?
- implementujú interface Collection
 - pridávanie a vyberanie prvkov z kontajnera
 - prechádzanie kontajnera pomocou cyklu for each
 - ...
- definície obsahujú statický typ
- Collection – statický typ
- ArrayList alebo HashSet – dynamický typ
- => využitie polymorfizmu

Interface Collection

«interface»
Collection<TPrvok>

- + size(): int
- + isEmpty(): boolean
- + clear(): void
- + add(paPrvok: TPrvok): boolean
- + remove(paPrvok: TPrvok): boolean

Priklad – HashSet₍₁₎

```
public HashSet<Integer> zlozityAlgoritmus()
{
    HashSet<Integer> vysledok =
        new HashSet<Integer>();
    napln(vysledok);
    vypocet(vysledok);
    vyhodnot(vysledok);

    return vysledok;
}
```

Priklad – HashSet₍₂₎

```
public void napln(HashSet<Integer> paVysledok)
{
    ...
}

public void vypocet(HashSet<Integer> paVysledok)
{
    ...
}

public void vyhodnot(HashSet<Integer> paVysledok)
{
    ...
}
```

Priklad – Collection₍₁₎

```
public Collection<Integer> zlozityAlgoritmus()
{
    Collection<Integer> vysledok =
        new ArrayList<Integer>();
    napln(vysledok);
    vypocet(vysledok);
    vyhodnot(vysledok);

    return vysledok;
}
```

Priklad – Collection₍₂₎

```
public void napln(Collection<Integer> paVys)
{
    ...
}

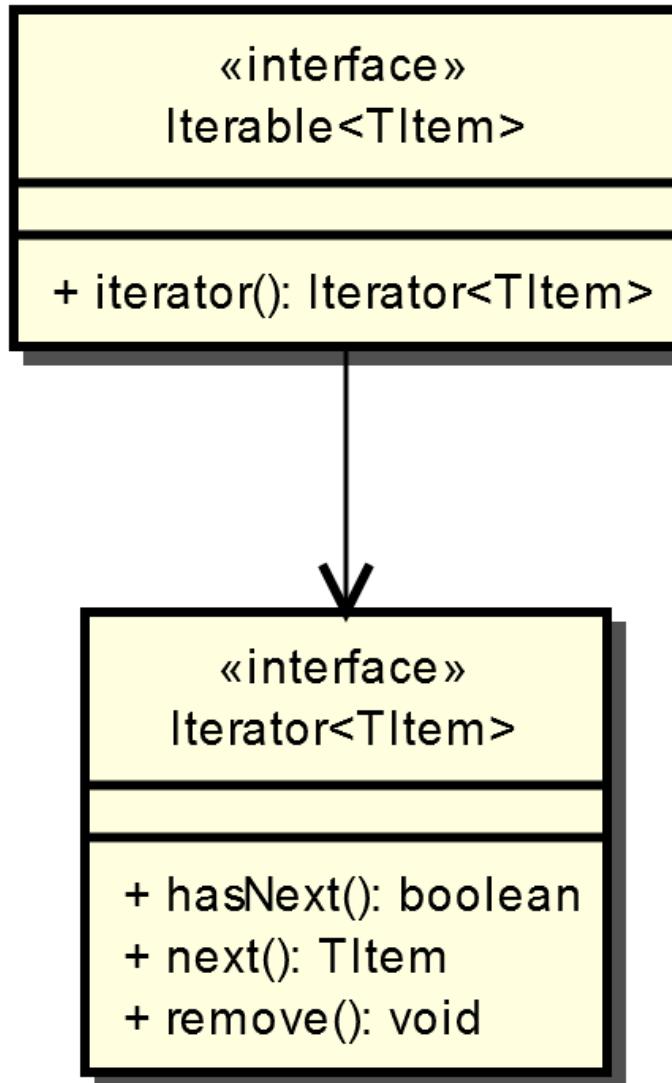
public void vypocet(Collection<Integer> paVys)
{
    ...
}

public void vyhodnot(Collection<Integer> paVys)
{
    ...
}
```

Iterátor

- návrhový vzor
- sekvenčné prechádzanie kontajnera po prvkoch
- dvojica interface
 - Iterable
 - Iterator
- implementované väčšinou kontajnerov
 - napr. ArrayList, HashSet, ...
- využíva ho foreach cyklus

Iterable a Iterator



Iterátor – foreach cyklus

```
for (TypPrvkov prvok : kontajner) {  
    // telo cyklu  
}
```

- je to isté ako

```
Iterator<TypPrvkov> prst = kontajner.iterator();  
while (prst.hasNext()) {  
    TypPrvkov prvok = prst.next();  
    // telo cyklu  
}
```

Collection implementujú

- `ArrayList<TPrvok>`
- `HashSet<TPrvok>`
- ...
- vkladanie a vyberanie prvkov z kontajnera
- a iné

Iterable implementujú

- `ArrayList<TPrvok>`
- `HashSet<TPrvok>`
- ...
- možno použiť cyklus `for each`

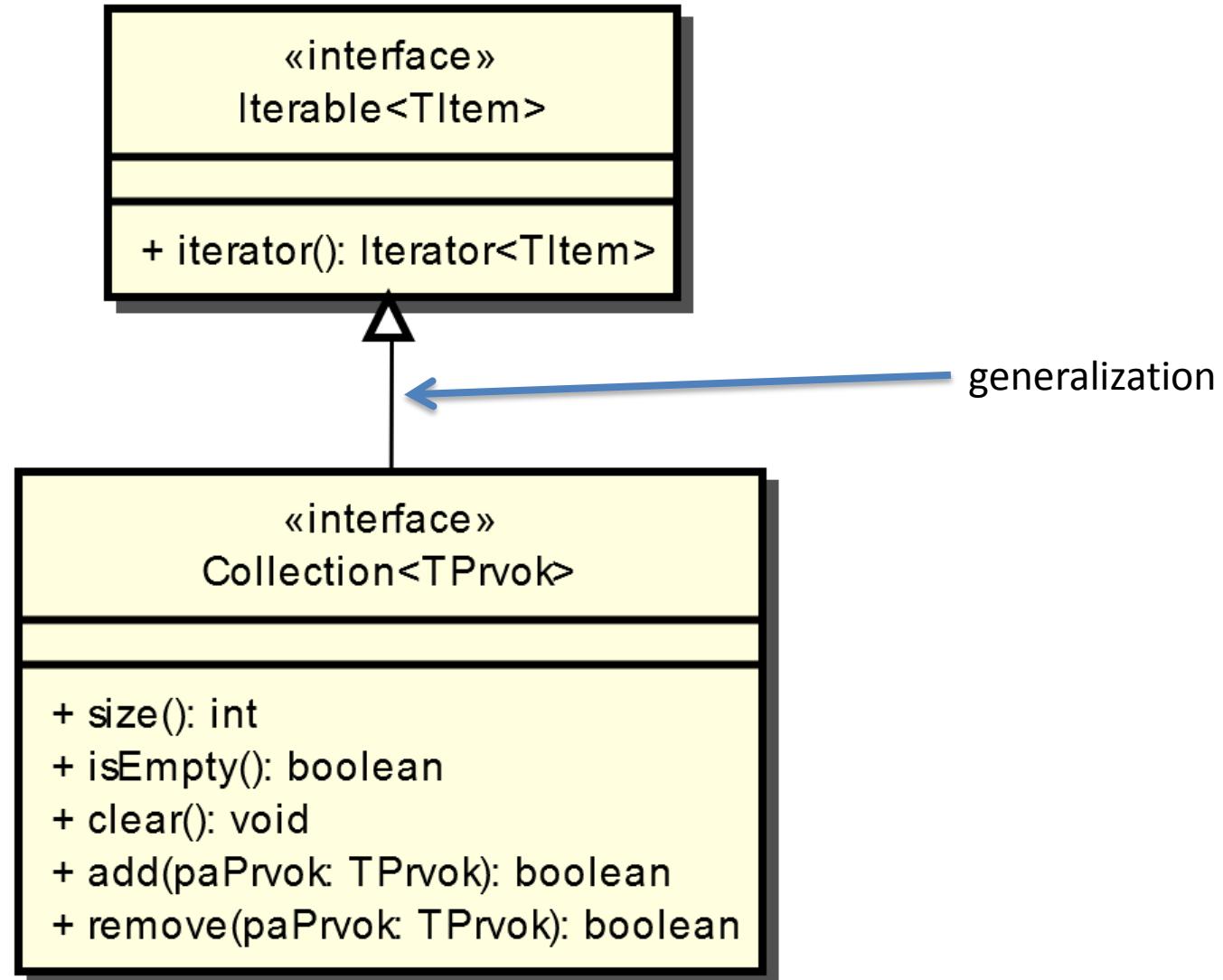
Rozširovanie interface (extends)

public interface Collection<TPrvok>

extends Iterable<TPrvok>

- interface Collection rozširuje Iterable
- => ku správam z Iterable doplní nové
 - zjednotenie množín správ
 - pozor: preťažovanie metód
- trieda implementujúca Collection implementuje aj Iterable
 - netreba uvádzat explicitne

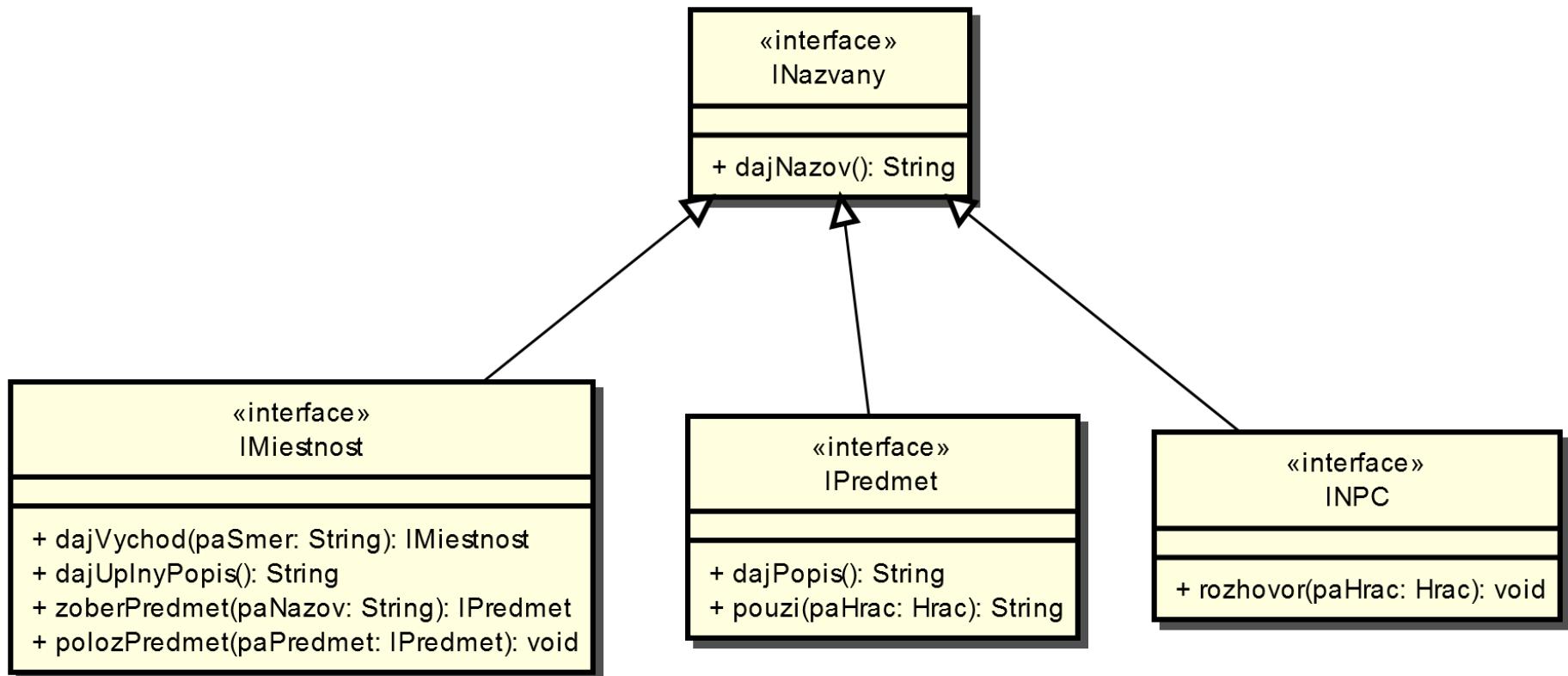
Rozširovanie interface v UML



Rozširovanie interface vo WoF

- IMiestnost
- IPredmet
- INPC
- všetky majú názov
 - spoločný interface INazvany

Interface INazvany v hre WoF



Balíčky

Balíčky

- knižnice – rozširujúca funkčnosť
 - delí sa na balíčky
 - balíčky obsahujú triedy
- podobne:
- program
 - delí sa na balíčky
 - balíčky obsahujú triedy

Balíčky – Java

- môžu byť ľubovoľne vnorené
- názvy vnorených balíčkov sa oddelujú bodkou
- príklad: java.util
- balíčky vytvárajú menné priestory

Menný priestor

- identifikátory – mená
 - premenná, trieda/enum/interface, metóda, ...
- menný priestor – situácia vyžadujúca jednoznačné identifikátory
 - blok
 - telo triedy
 - balíček
- menný priestor – kontejner identifikátorov s jednoznačným určením významu

Riešenie kolízie identifikátorov

- plne kvalifikovaný názov
- FQN – fully qualified name
- FQN identifikátor
 - postupnosť názvov balíčkov
 - bodková notácia

Využitie iného menného priestoru

- využitie FQN – ak hrozí konflikt

```
java.util.ArrayList<Integer> premenna =  
    new java.util.ArrayList<Integer>();
```

- príkaz import – ak nehrozí konflikt

```
import java.util.ArrayList;
```

```
ArrayList<Integer> premenna =
```

```
    new ArrayList<Integer>();
```

Príkaz package

package nazov;

- úplne prvý príkaz v súbore
 - súbor: trieda, enum, interface
- nad ním iba komentár
- obsahuje názov balíčka
- každý súbor v balíčku !!!
- prostredie BlueJ/Netbeans automaticky

Nepomenovaný balíček

- každý projekt (aplikácia)
- prvý balíček – nemá meno
- doteraz len také projekty

Reprezentácia balíčkov

- balíček == adresár (zložka, priečinok)
- podbalíčky == podadresáre
- súbory v balíčku == súbory .java v adresári

Názvy balíčkov

- konvencia: prvé písmeno malé, rovnako ako metódy
- názov balíčka musí byť unikátny !
- odstrašujúci príklad:
 - nový balíček „java“ s jednou triedou
 - prestane fungovať program
 - v starej verzii: prestane fungovať dokonca BlueJ

Unikátnosť názvov balíčkov

- v nepomenovanom balíčku len najvyšší balíček
- najvyšší balíček – vlastné meno, prezývka, názov firmy, webová doména (viacstupňové vnáranie)
- vnorený balíček – názov programu
- do neho vnorené balíčky – vlastné balíčky programu

Príklad názvov balíčkov

- nepomenovaný balíček obsahuje
 - balíček fri (správne sk.uniza.fri)
- balíček fri obsahuje
 - balíček hry worldOfFri
- balíček hry obsahuje
 - balíčky hra, mapaHry, príkazy, predmety, ...

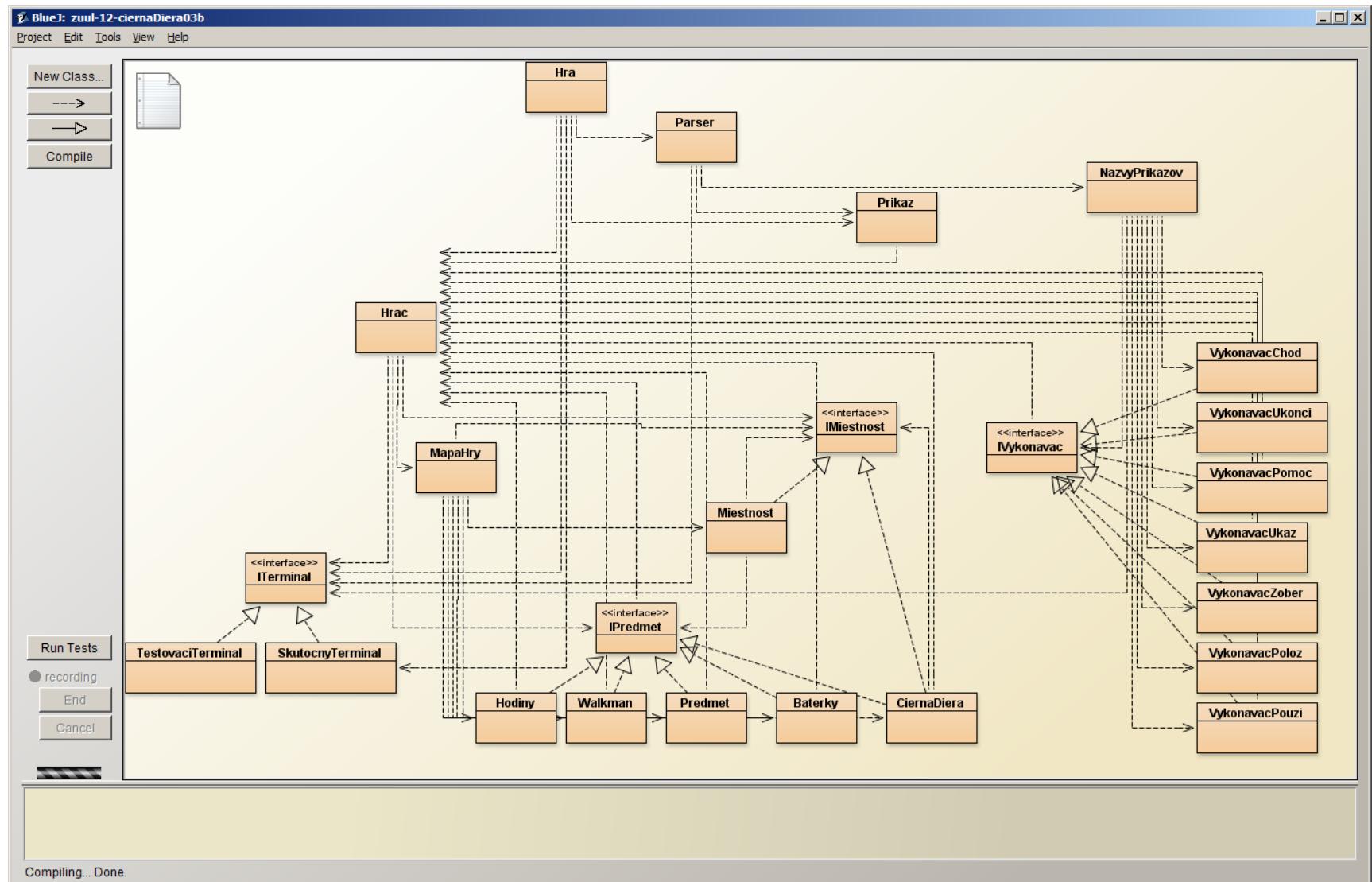
Výsledné FQN tried WoF

- fri.worldOfFri.hra.Hra
- fri.worldOfFri.predmety.Walkman
- fri.worldOfFri.prikazy.IVykonavac
- ...

Balíčky v prostredí BlueJ

- pridávanie balíčkov
- presúvanie tried medzi balíčkami

Súčasný stav World of FRI



Súčasný stav - NetBeans

The screenshot shows the NetBeans IDE interface. On the left, the Project Explorer window displays a Java project named "WorldOffRI". It contains a "Source Packages" node with a single package named "<default package>". This package contains 24 Java source files: Baterky.java, CiernaDiera.java, Hodiny.java, Hra.java, Hrac.java, IMiestnost.java, IPredmet.java, ITermal.java, IVykonavac.java, MapaHry.java, Miestnost.java, NazvyPrikazov.java, Parser.java, Predmet.java, Prikaz.java, SkutocnyTerminal.java, TestovaciTerminal.java, VykonavacChod.java, VykonavacPoloz.java, VykonavacPomoc.java, VykonavacPouzi.java, VykonavacUkaz.java, VykonavacUkoni.java, VykonavacZober.java, and Walkman.java. Below the source packages is a "Libraries" node. At the bottom of the Project Explorer is a status bar with the text "JavaProject3 - Navigator". On the right, the Code Editor window is open to the file "Hra.java". The code is as follows:

```
public class Hra
{
    private Pars
    private Hrac
    private ITer

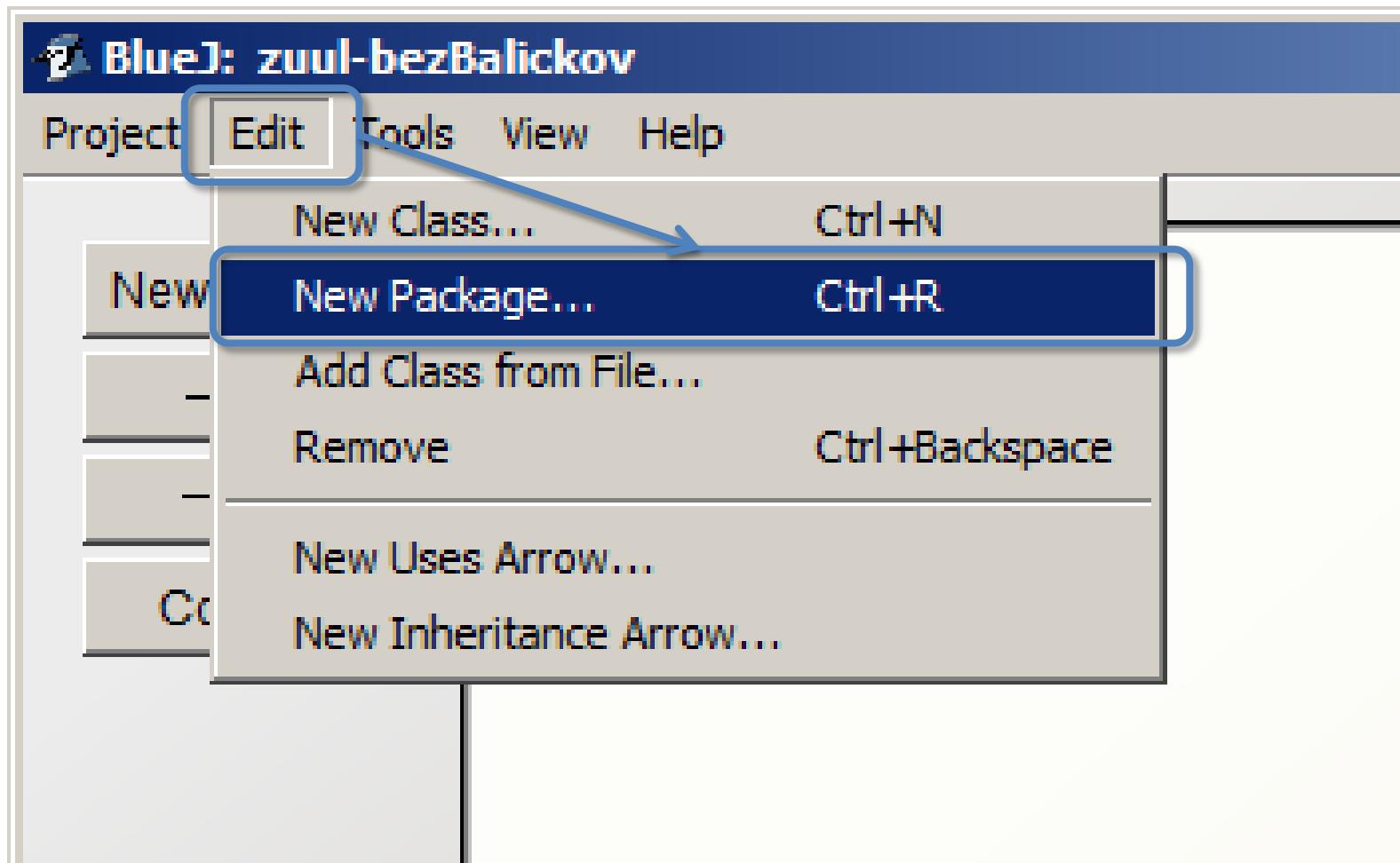
    public Hra(I
    {
        aTermina
        aParser
        aHrac =
    }

    public void
    {
        this.zac
        this.vyp

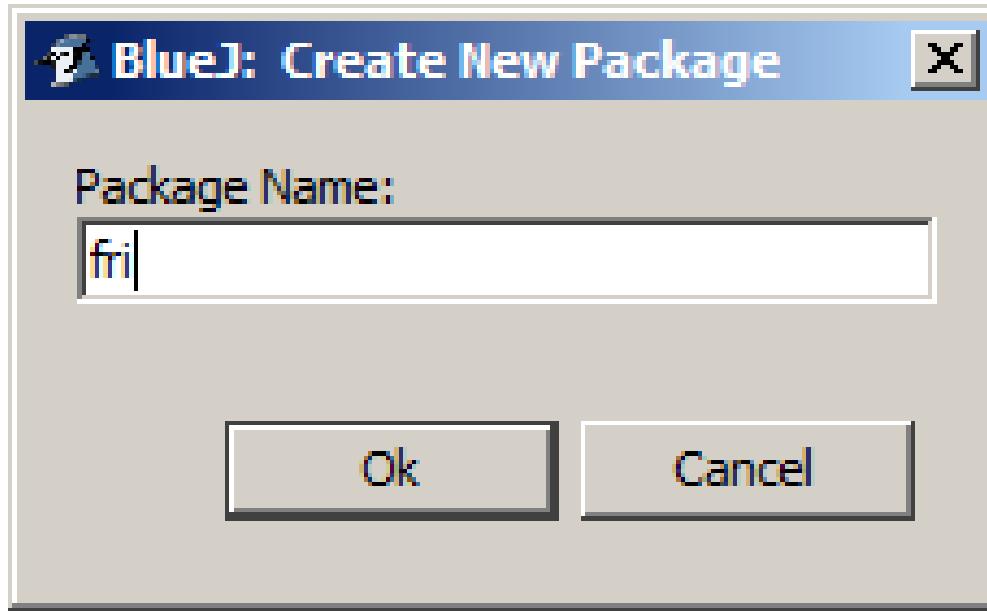
        boolean
        while (!
            Prik
            jeKc
        }
        aTermina
    }

    private void
    {
        String n
        aTermina
        menoHrac
    }
}
```

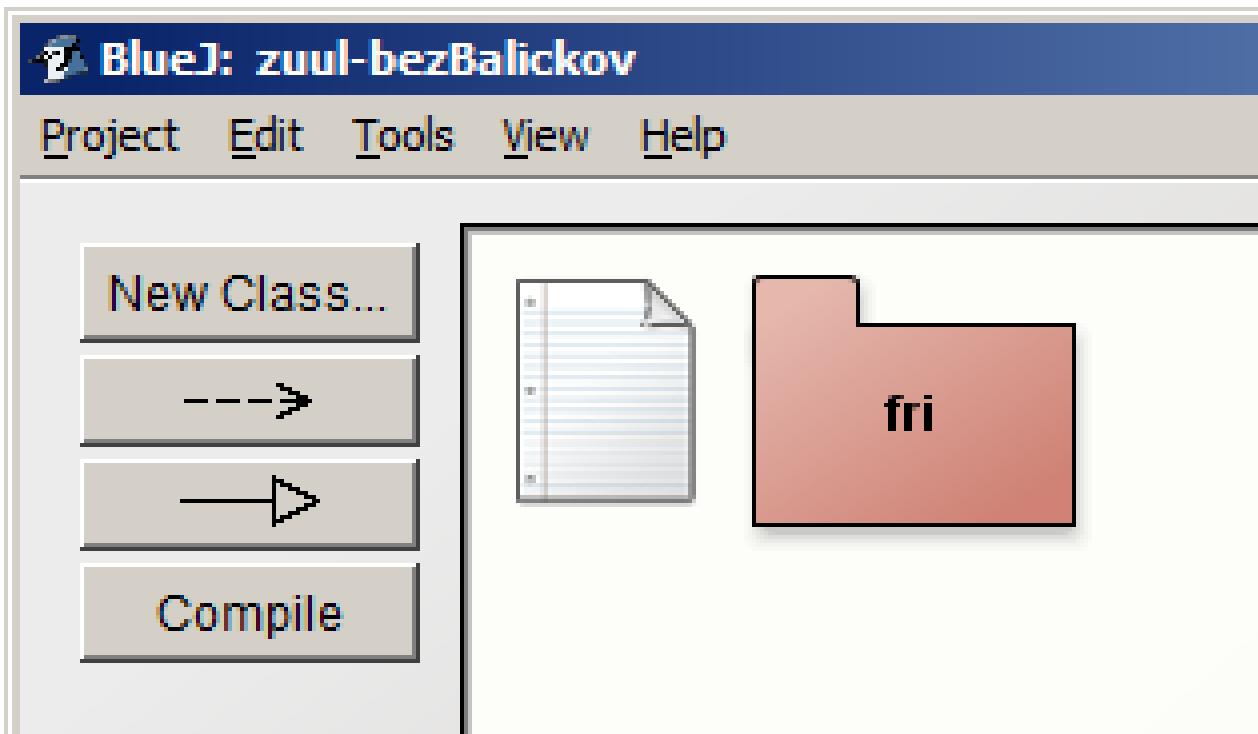
Pridanie balíčka



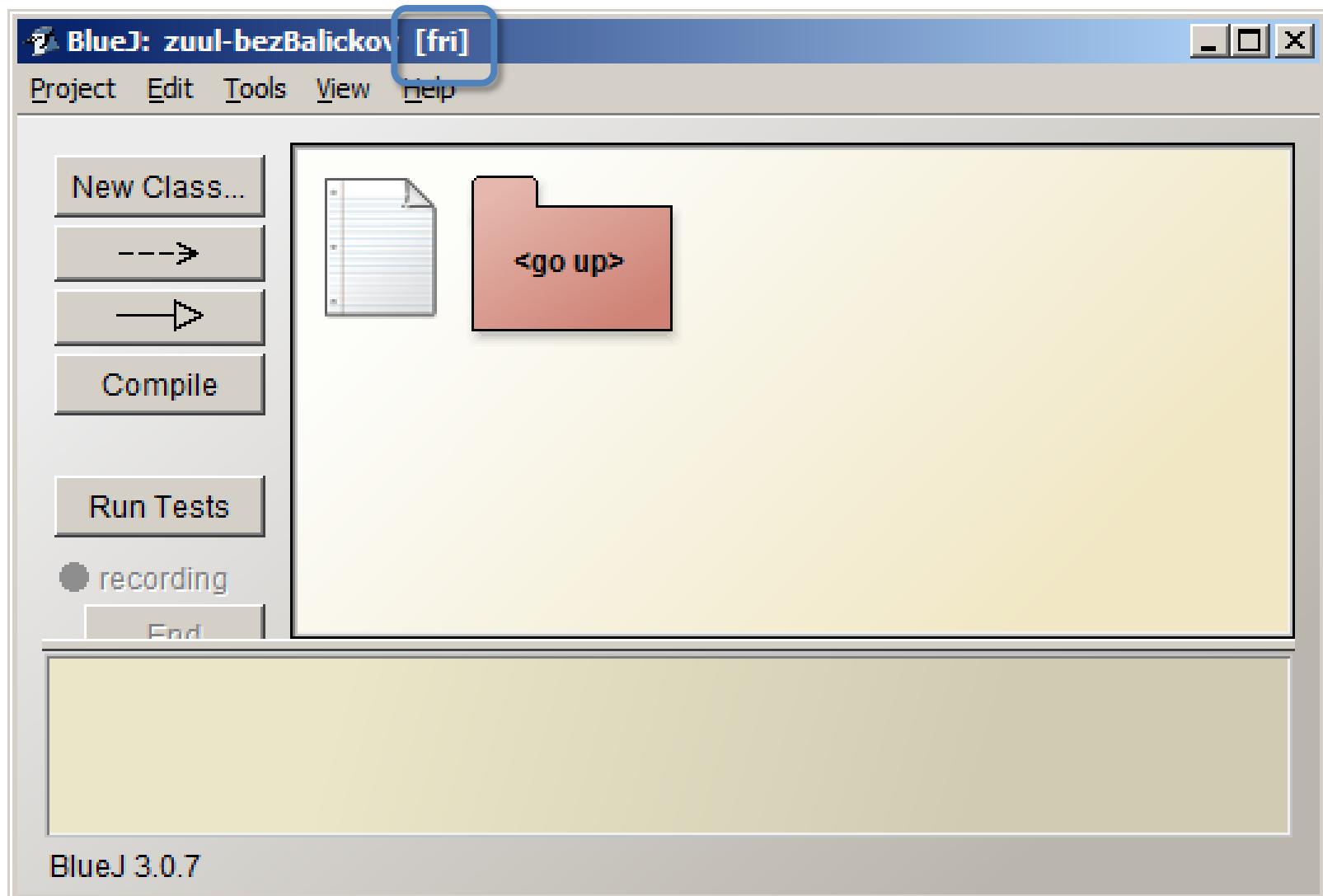
Pomenovanie balíčka



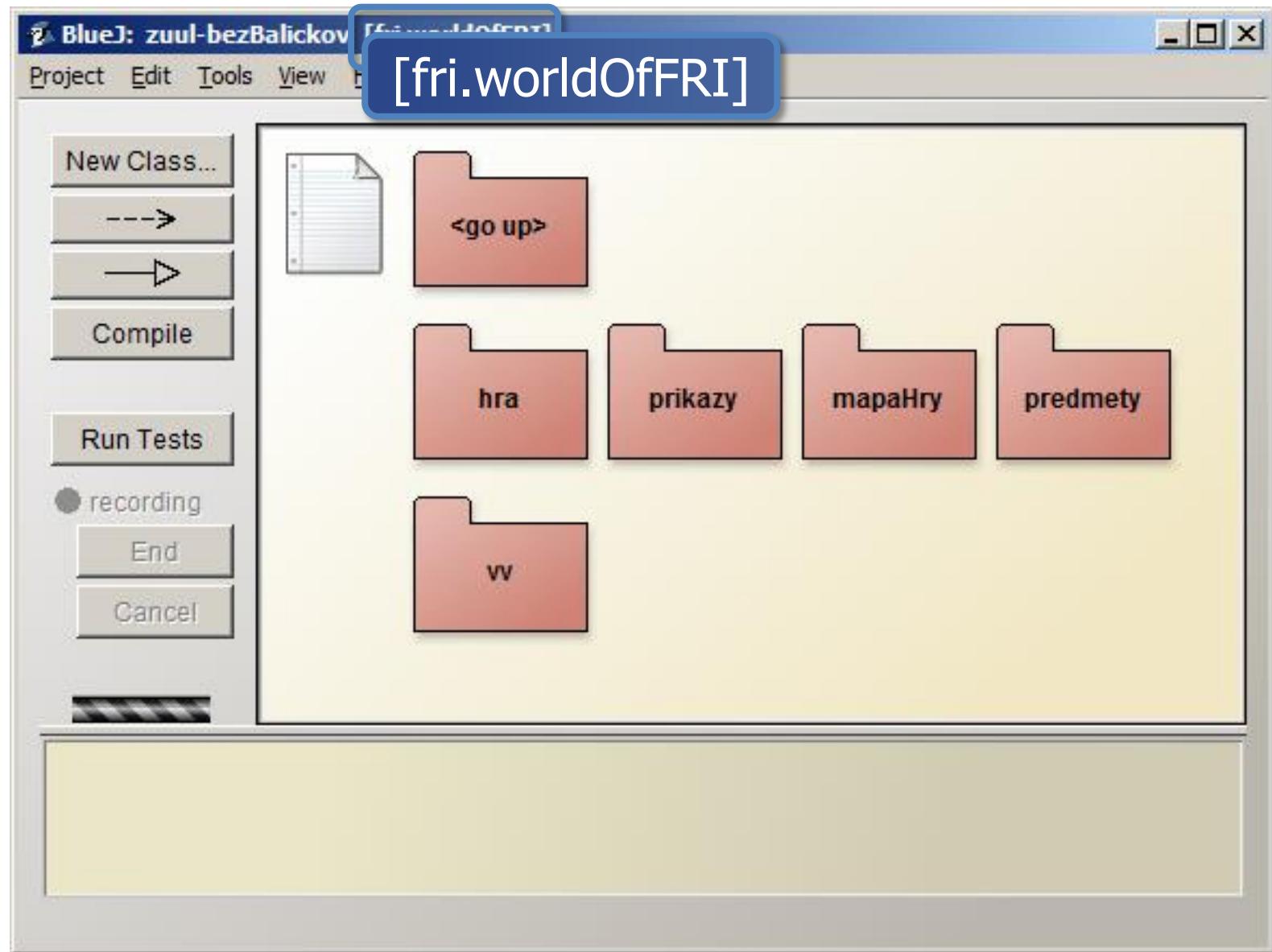
Zobrazenie balíčka



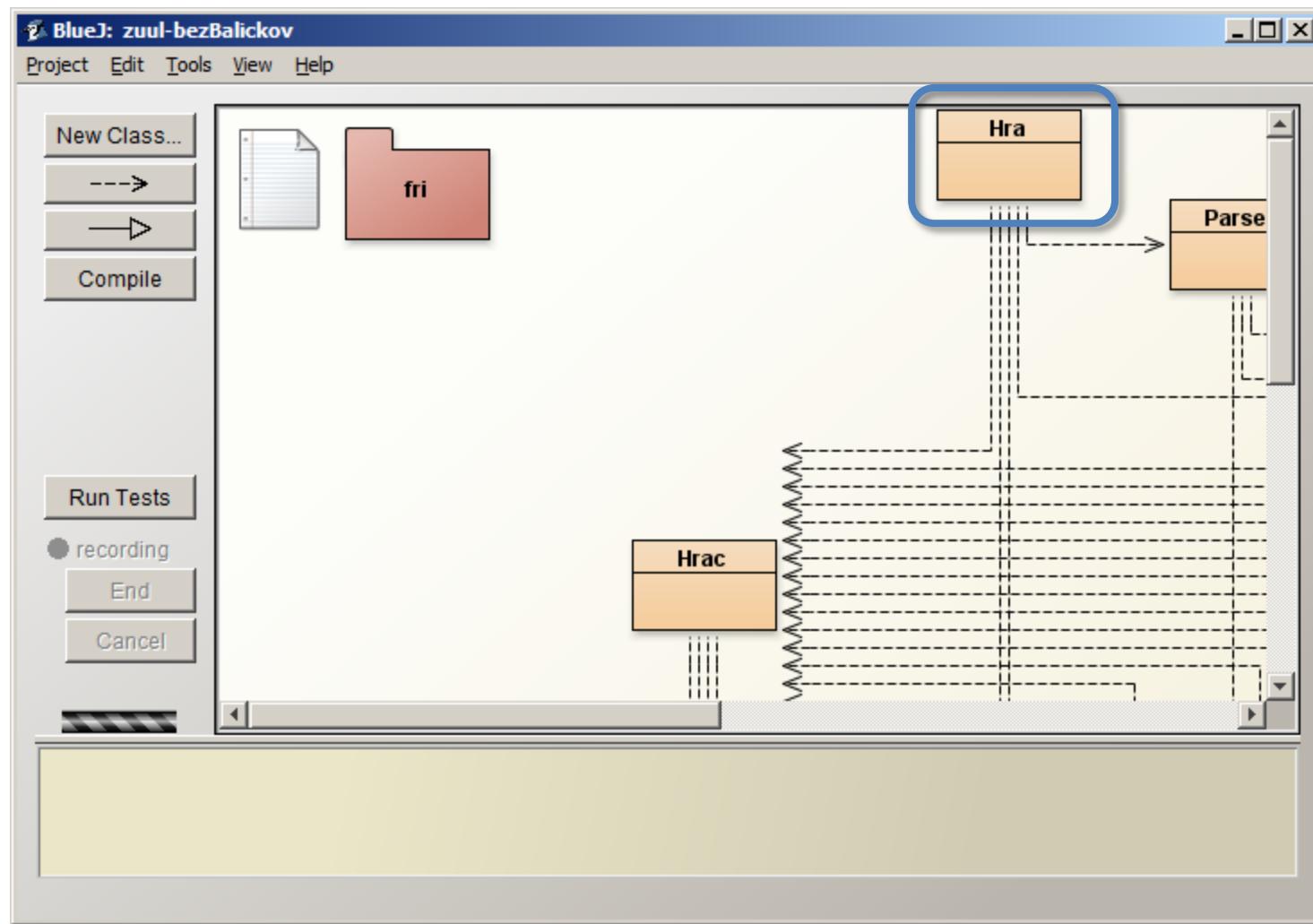
Balíček otvorený v BlueJ



Štruktúra balíčkov vo World of FRI



Presun triedy do balíčka



Kód pred presunom

```
public class Hra  
{  
    private Parser aParser;  
    private Hrac aHrac;  
    private ITerminal aTerminal;  
}
```

...

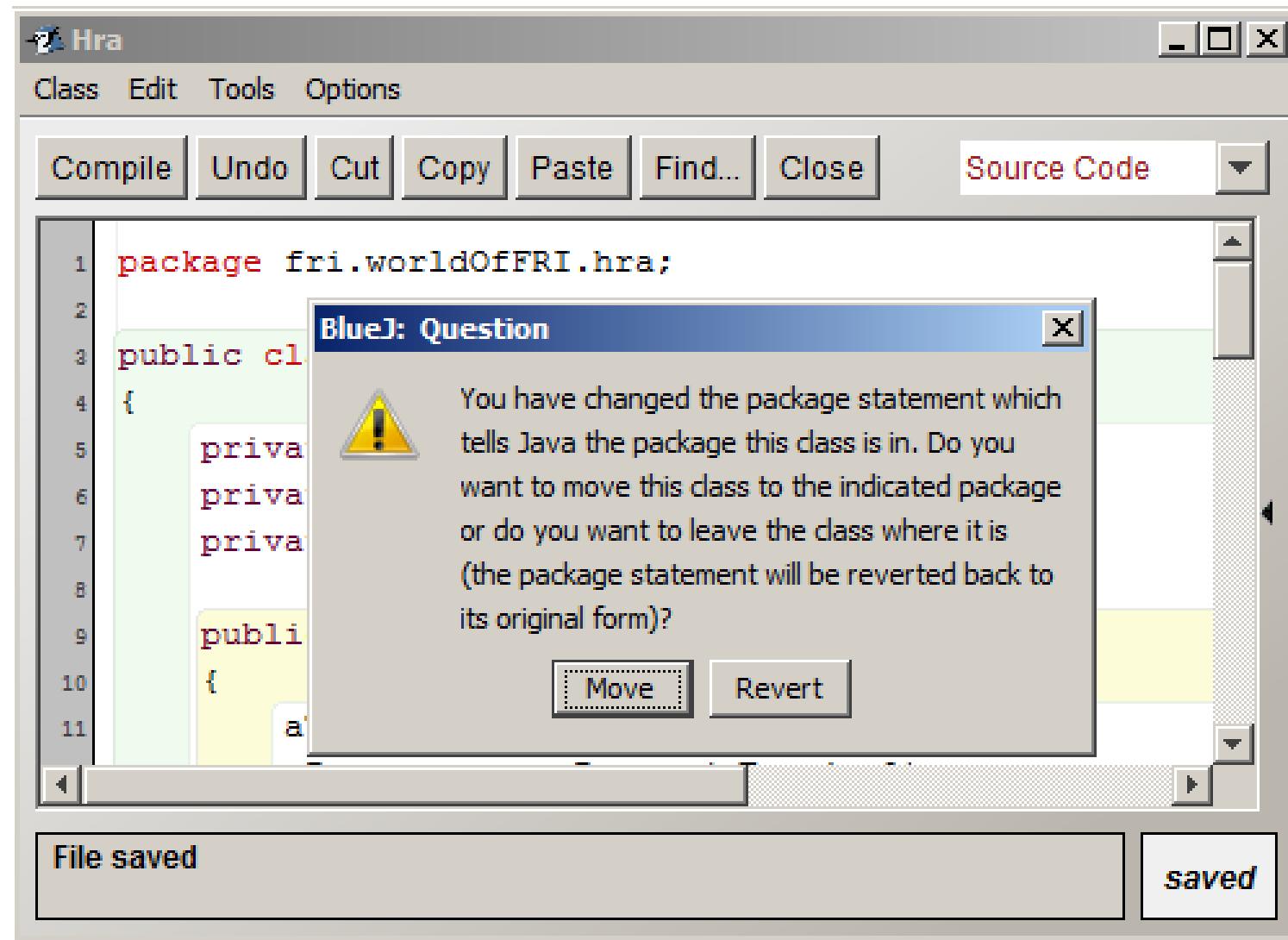
Pridanie názvu balíčka

```
package fri.worldOfFRI.hra;
```

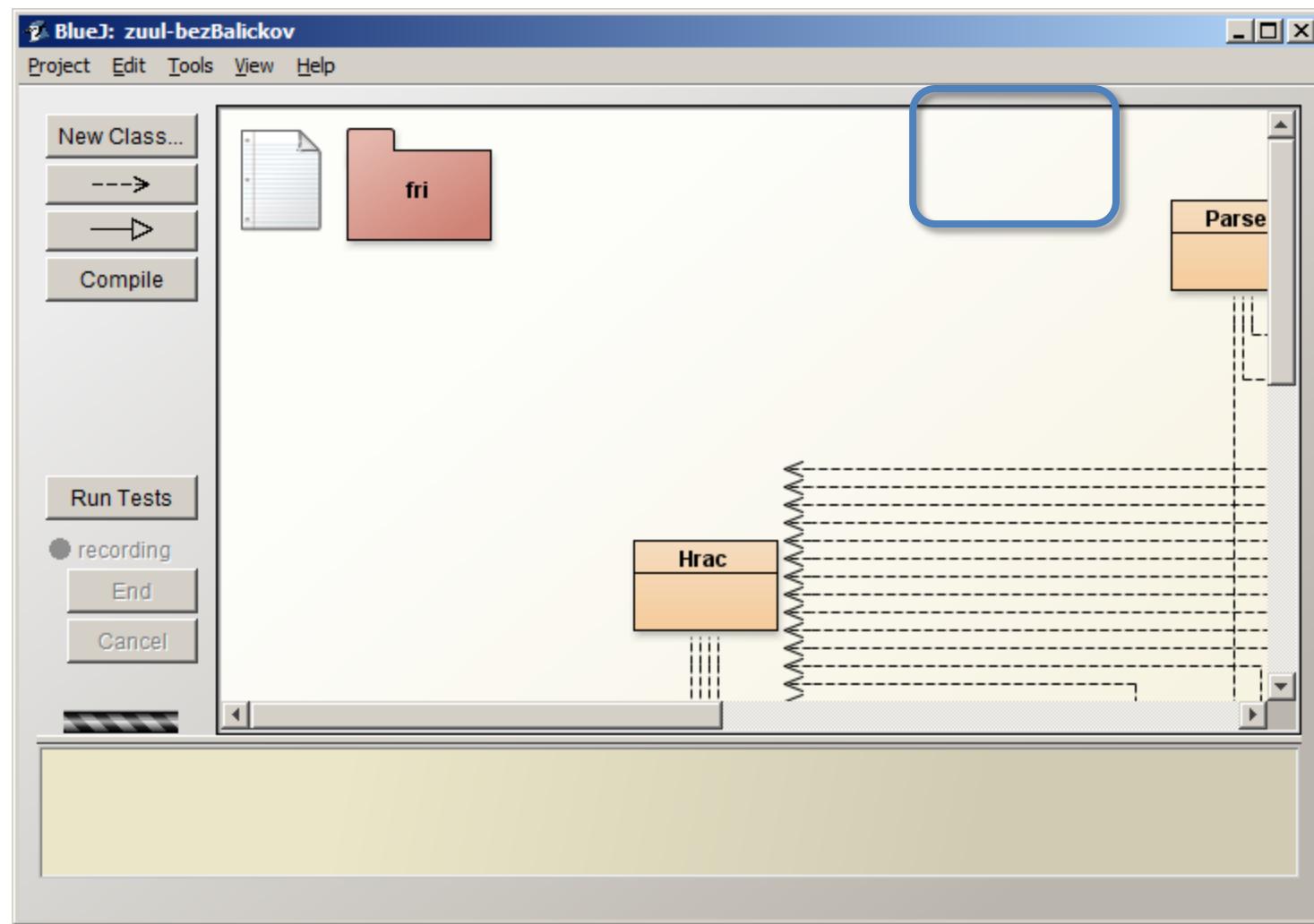
```
public class Hra
{
    private Parser aParser;
    private Hrac aHrac;
    private ITerminal aTerminal;

    ...
}
```

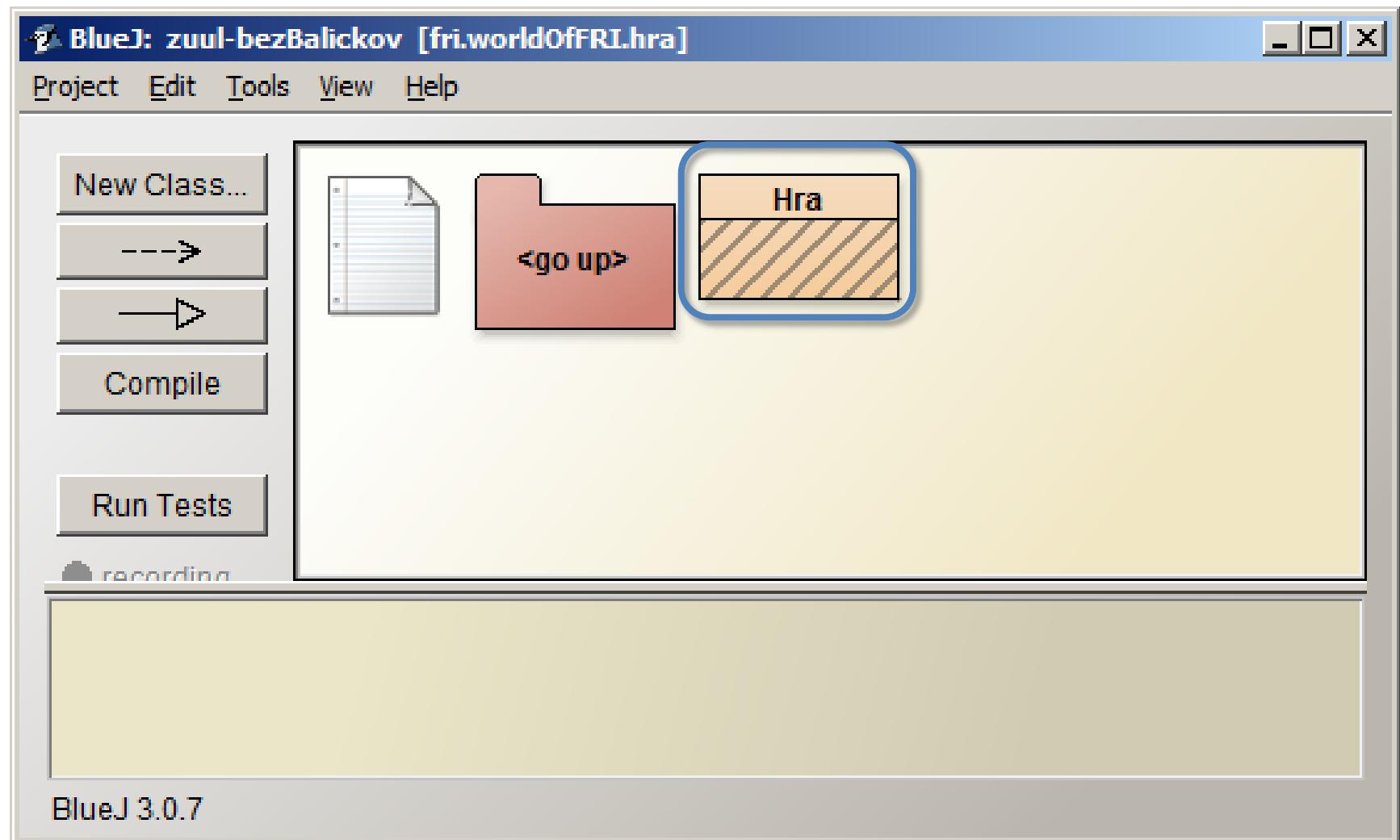
Automatický presun súboru(1)



Automatický presun súboru(2)



Automatický presun súboru(3)



Rozdelenie tried hry do balíčkov₍₁₎

- hra
 - Hra, Hrac
- mapaHry
 - MapaHry, IMiestnost, Miestnost
- prikazy
 - Parser, Prikaz, NazvyPrikazov, IVykonavac, VykonavacPomoc, VykonavacChod, VykonavacUkonci, VykonavacUkaz, VykonavacZober, VykonavacPoloz, VykonavacPouzi

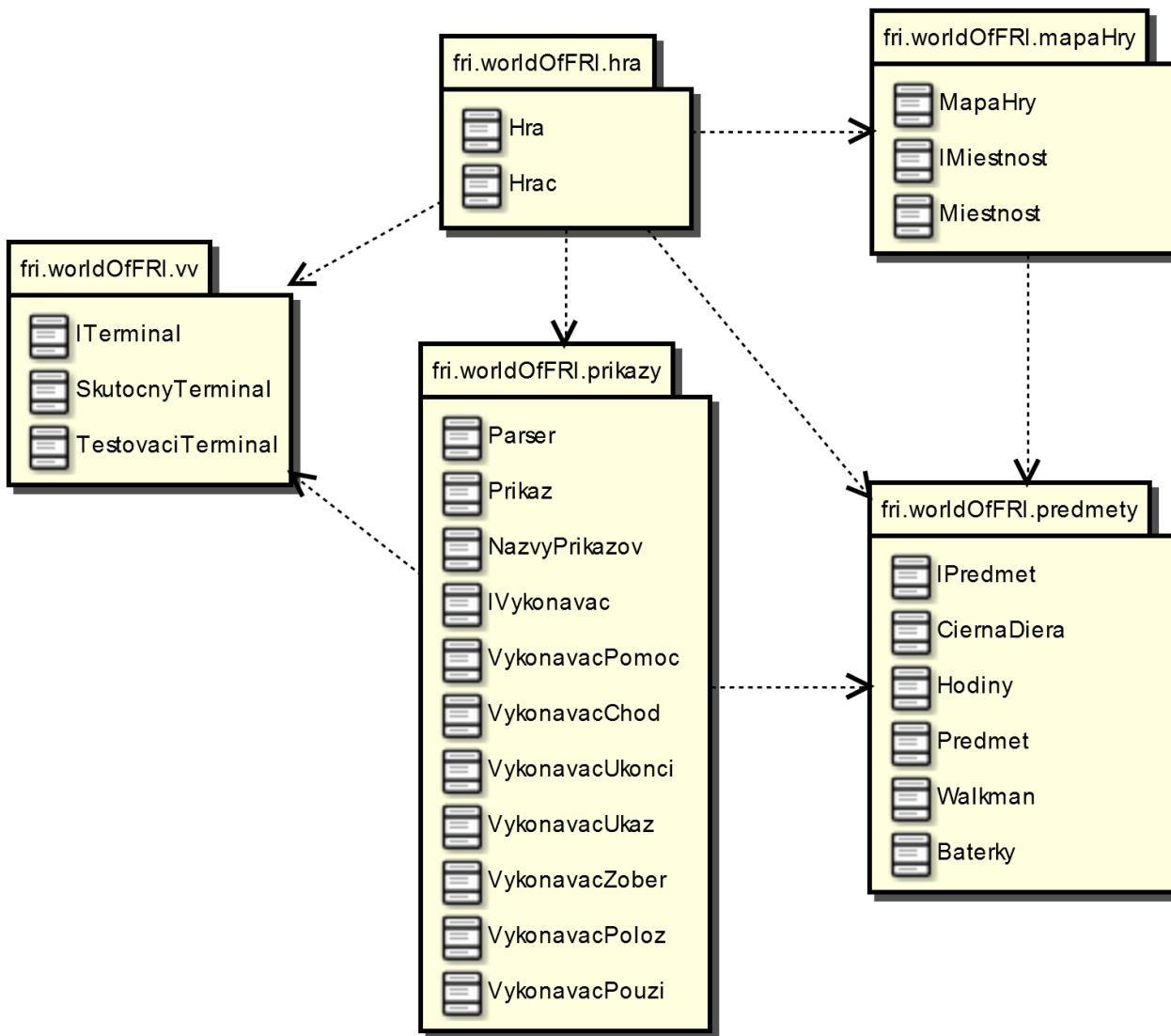
Rozdelenie tried hry do balíčkov₍₂₎

- predmety
 - IPredmet, CiernaDiera, Hodiny, Predmet, Walkman, Baterky
- vv
 - ITerminal, SkutocnyTerminal, TestovaciTerminal

Balíčky v UML – UML .FRI

- ikona balíčka rovnaká
- zoznam tried (enum, interface) v ikone
- vzťahy závislosti medzi balíčkami

Balíčky „World of Fri“ v UML .FRI



Balíček – dokumentačné komentáre

- dokumentačný komentár pre balíček – význam balíčka ako celku
- nedá sa napísat do súboru s triedou – javadoc nenájde
- špeciálny súbor package-info.java

Súbor package-info.java

- BlueJ – bez podpory
- Netbeans
 - File>New File...
 - Other>Empty File
- vytvorenie v ľubovoľnom textovom editore
 - Notepad, PSPad, Notepad++, Scite, CodePad...
 - nie Word – textový procesor

Súbor package-info.java – príklad

```
/**  
 * Obsahuje vsetky triedy ktore pracuju s hrou.  
 * Je to hlavne trieda Hra, ktoru hru riadi a trieda  
 * Hrac, ktoru reprezentuje hraca hry.  
 */  
package fri.worldOfFri.hra;
```

Javadoc pre balíčky

[Overview](#) [Package](#) [Class](#) [Tree](#) [Index](#) [Help](#)

[PREV](#) [NEXT](#)

[FRAMES](#) [NO FRAMES](#) [All Classes](#)

worldOfFri

Packages

fri.worldOfFri.hra	Obsahuje vsetky triedy ktore pracuju s hrou.
fri.worldOfFri.mapaHry	Obsahuje vsetky triedy reprezentujuce virtualny svet.
fri.worldOfFri.predmety	Obsahuje triedy určené na spracovanie jednotlivých príkazov.
fri.worldOfFri.prikazy	Obsahuje vsetky triedy ktore vykonavaju jednotlive prikazy.
fri.worldOfFri.vv	Obsahuje triedy obsluhujúce vstup a výstup.

[Overview](#) [Package](#) [Class](#) [Tree](#) [Index](#) [Help](#)

[PREV](#) [NEXT](#)

[FRAMES](#) [NO FRAMES](#) [All Classes](#)

Vďaka za pozornosť