

Informatika 1

Kontajnery s premenlivým počtom prvkov



Pojmy zavedené v 6. prednáške (1)

- logické výrazy
 - operátory &&, ||, !
 - skrátené vyhodnocovanie
- asociácia
 - vzťah dvoch rovnocenných objektov

Pojmy zavedené v 6. prednáške (2)

- referencie
- porovnanie objektov
 - operátory ==, !=
 - správa equals
- hodnota null
- zánik inštancie
 - zánik pri kompozícii
 - zánik pri asociácii

Pojmy zavedené v 6. prednáške (3)

- Posielanie správy cez this
- Súkromné metódy

Cieľ prednášky

- skupiny objektov – kontajnery
- nový prvok algoritmu – cyklus foreach
- príklad: poznámkový blok

Jednoduché a zložené objekty

- jednoduché objekty
 - atribúty, parametre – primitívne typy
- zložené objekty
 - kompozícia – celok a časť
 - digitálne hodiny
 - kontajnery – vytváranie skupín prvkov jedného druhu
 - môže obsahovať ľubovoľný počet prvkov

Príklady kontajnerov

- zoznam študentov v študijnej skupine
- katalóg kníh v knižnici
- cestujúci v trolejbuse
- index – zoznam zapísaných predmetov
- zoznam klientov banky
- diár – zoznam poznámok
- String – postupnosť znakov

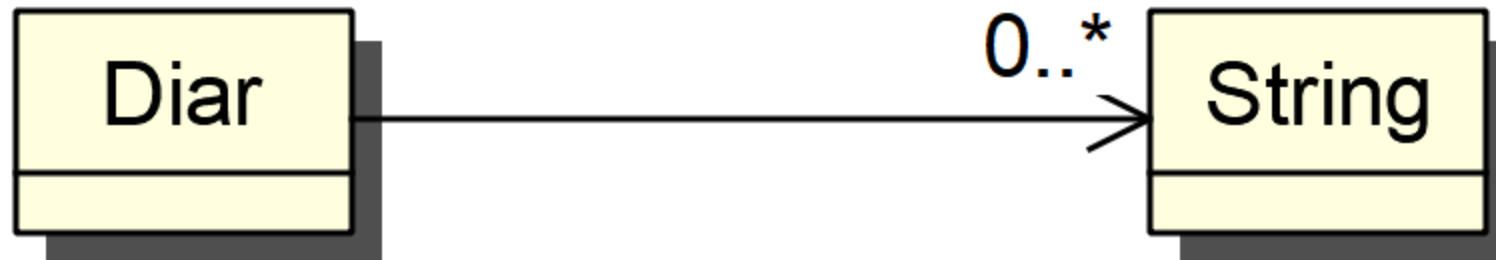
Poznámkový blok – Diár

- kontajner na poznámky
 - poznámka – text (reťazec)
- služby:
 - nové poznámky – doplnenie
 - nepotrebné poznámky – škrtanie
 - vypísanie zvolenej poznámky
 - počet všetkých poznámok

Diár – rozhranie

Diár
<ul style="list-style-type: none">+ <u>new()</u>: Diár+ vlozPoznamku(poznamka: String): void+ vypisPoznamku(poradoveCislo: int): void+ zmazPoznamku(poradoveCislo: int): void+ getPocetPoznamok(): int

Diár – vzťah s poznámkou



Diár – s doterajšími prostriedkami

- každá poznámka – atribút
- pevne daný počet poznámok
- voľná poznámka == null

Vnútrotný pohľad

Diar
<ul style="list-style-type: none">- poznamka1: String- poznamka2: String- poznamka3: String- poznamka4: String
<ul style="list-style-type: none">+ Diar()+ vložPoznamku(poznamka: String): void+ vypisPoznamku(poradoveCislo: int): void+ zmazPoznamku(poradoveCislo: int): void+ getPocetPoznamok(): int

Prázdny diár

diar : Diar

- poznamka1 = null
- poznamka2 = null
- poznamka3 = null
- poznamka4 = null

Diár s 1. poznámkou

diar : Diar

- poznamka1 = "SI - prezentácie"
- poznamka2 = null
- poznamka3 = null
- poznamka4 = null

Diár so 4 poznámkami – plný

diar : Diar

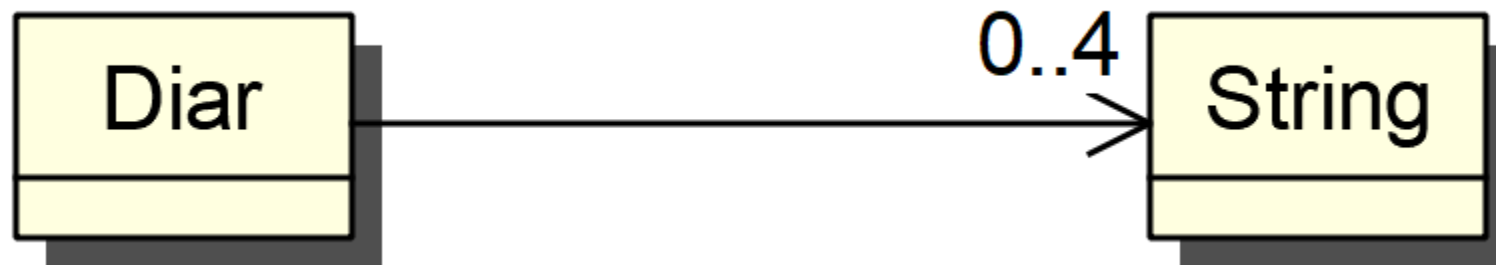
- poznamka1 = "SI - prezentácie"
- poznamka2 = "INF - prednáška"
- poznamka3 = "nákup - chlieb"
- poznamka4 = "INF1 - cvičenie"

Diár po vymazaní tretej poznámky

diar : Diar

- poznamka1 = "SI - prezentácie"
- poznamka2 = "INF - prednáška"
- poznamka3 = null
- poznamka4 = "INF1 - cvičenie"

Diár – diagram tried



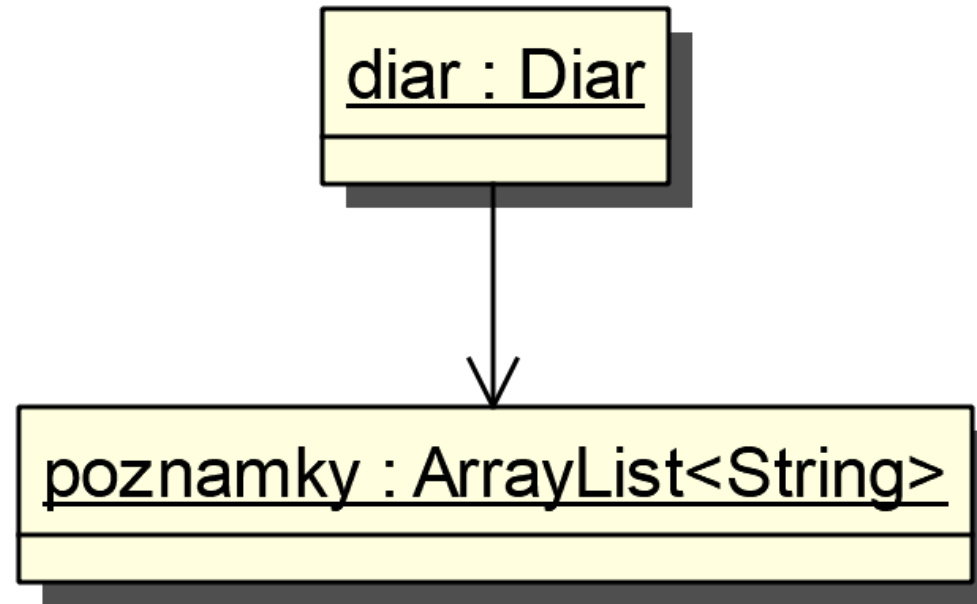
Diár – ďalšie podmienky

- neurčená horná hranica
- počet poznámok sa mení
- počet je ľubovoľný – aj žiadna

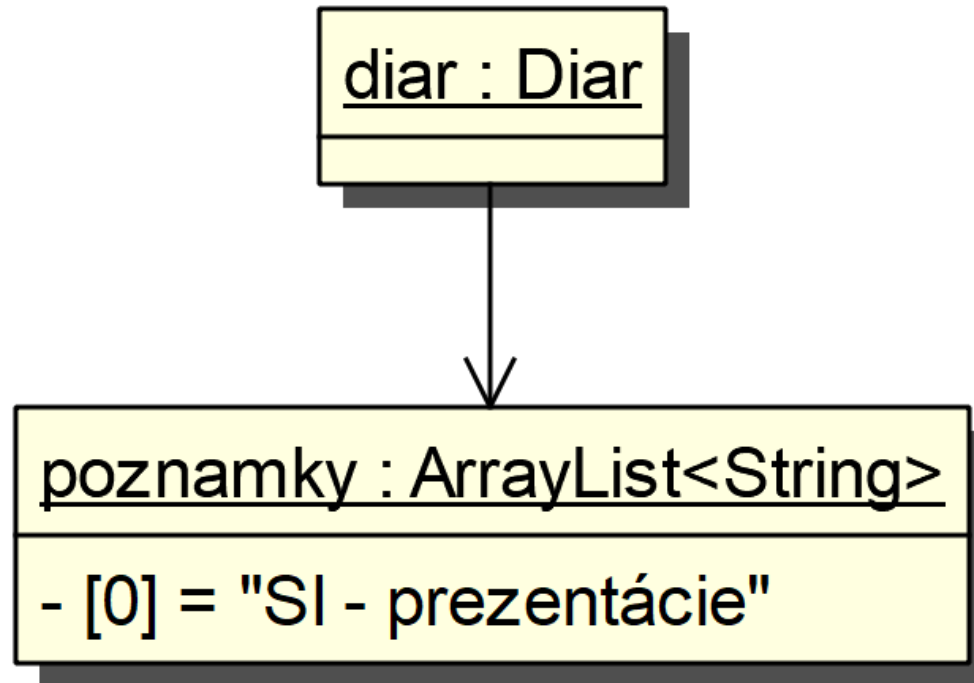
ArrayList – Java

- ArrayList<TypPrvkov>
 - TypPrvkov – ľubovoľný objektový typ
- kontajner pre zvolený typ prvkov
- môže obsahovať ľubovoľný počet prvkov
- ktorýkoľvek prvok sa dá vymazať
- ku prvkom sa dá pristupovať pomocou poradového čísla – indexu
 - číslovanie začína od 0

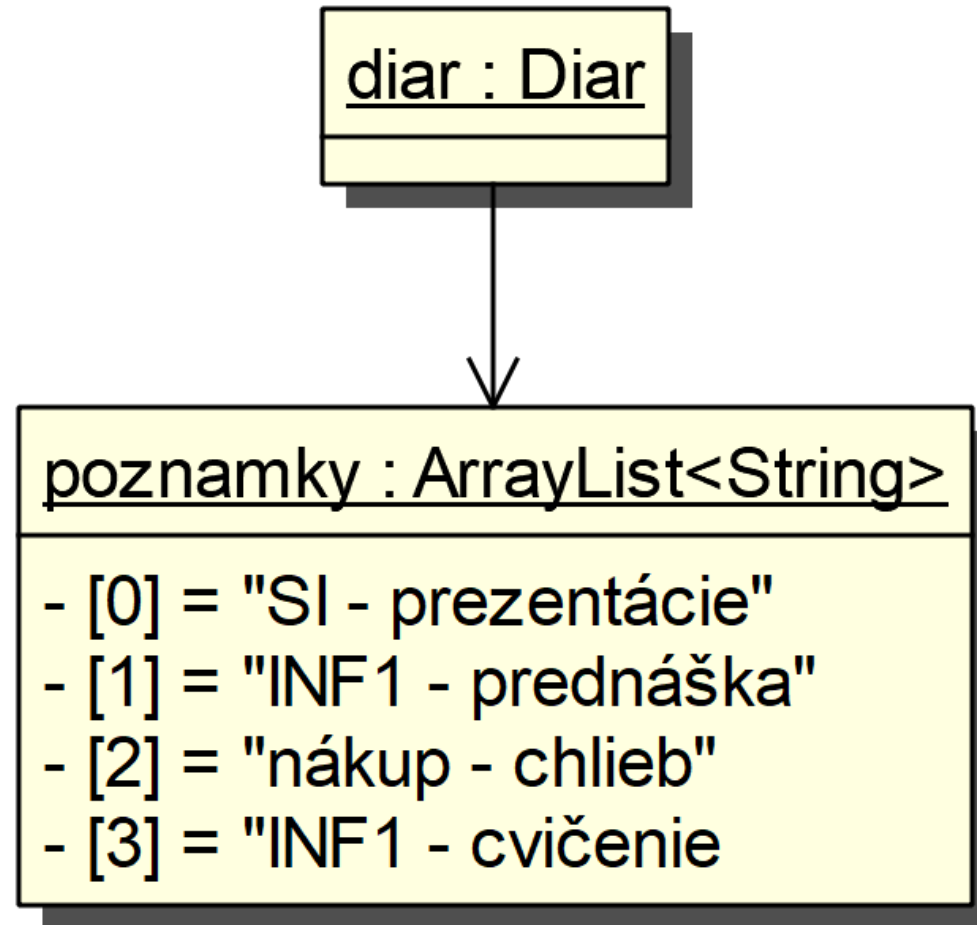
Prázdny diár



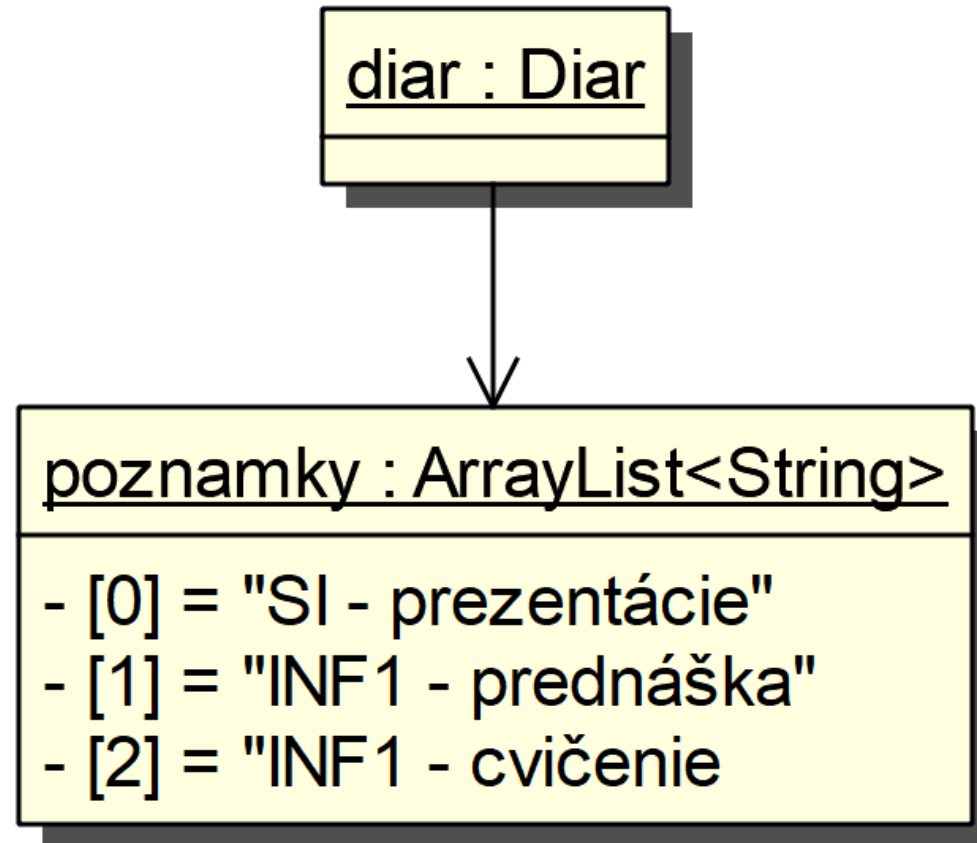
Diár s jednou poznámkou



Diár so 4 poznámkami



Diár po vymazaní tretej poznámky



ArrayList<String> – rozhranie

ArrayList<String>
+ <u>new(): ArrayList<String></u>
+ add(prvok: String): void
+ get(index: int): String
+ remove(index: int): String
+ size(): int

Generické triedy – Java

- ArrayList je jedna z generických tried

```
NazovTriedy<ZoznamParametrov>
```

- parametrami musia byť objektové typy
 - typové parametre

Generické triedy – Java

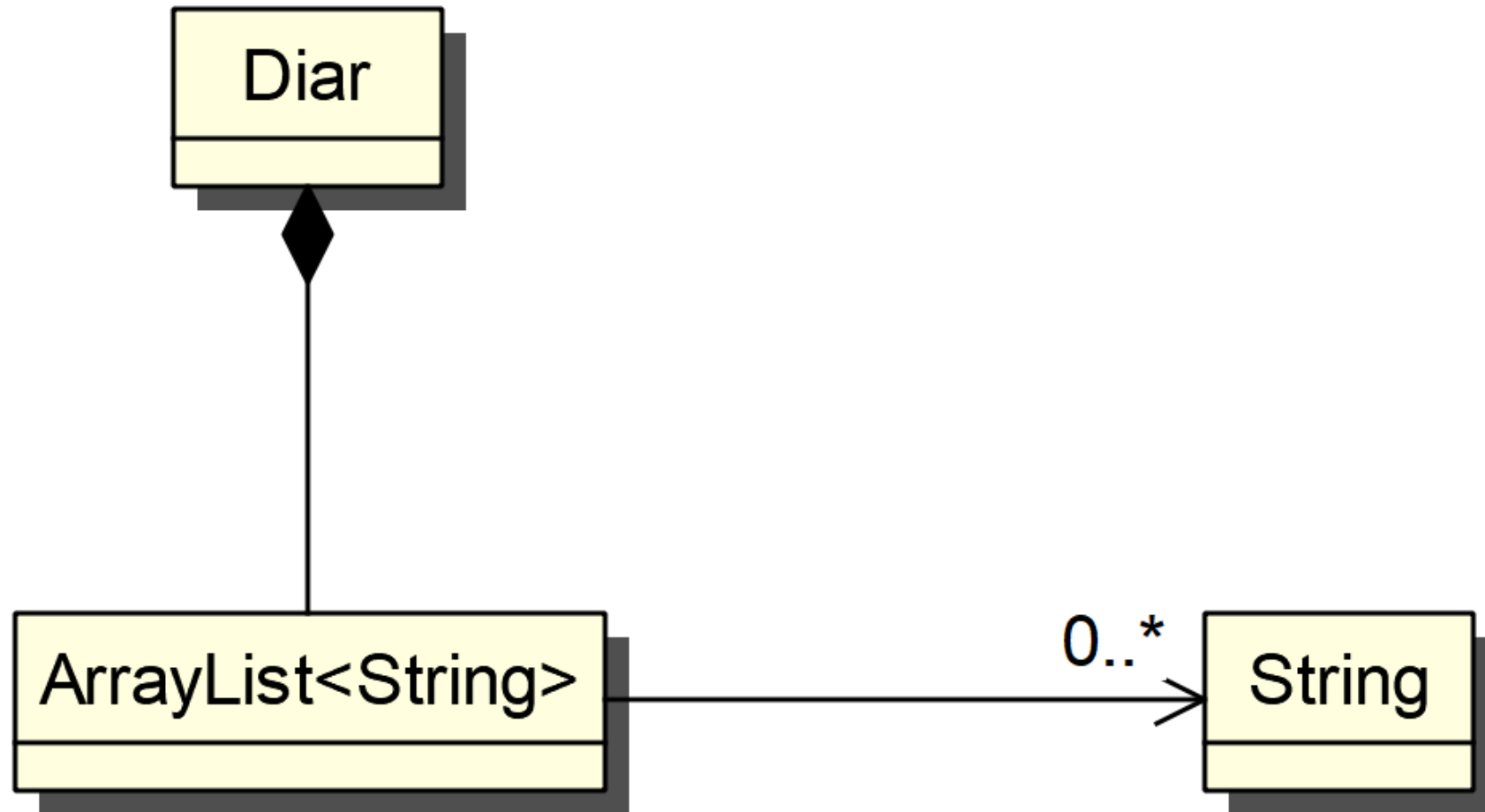
- deklarácia premennej:

```
NazovTriedy<ZoznamParametrov> premenna;
```

- vytvorenie inštancie:

```
new NazovTriedy<zoznamParametrov>(parametre)
```

Diar – diagram tried



Diár – vnútorný pohľad

Diár
- poznamky: <code>ArrayList<String></code>
+ Diar() + vložPoznamku(poznamka: String): void + vypisPoznamku(poradoveCislo: int): void + zmazPoznamku(poradoveCislo: int): void + getPocetPoznamok(): int

Diár – definícia triedy, atribút

```
public class Diar {  
    private ArrayList<String> poznamky;  
  
    ...  
}
```

Diár – konštruktor

```
public Diar() {  
    this.poznamky = new ArrayList<String>();  
}
```

Diár – vloženie poznámky

```
public void vložPoznamku(String poznamka) {  
    this.poznamky.add(poznamka);  
}
```

Diár – zobrazenie poznámky

```
public void vypisPoznamku(int poradoveCislo) {  
    int pocet = this.poznamky.size();  
    if (poradoveCislo >= 0 && poradoveCislo < pocet) {  
        System.out.println(this.poznamky.get(poradoveCislo));  
    }  
}
```


Diár – počet poznámok

```
public int getPocetPoznamok() {  
    return this.poznamky.size();  
}
```

Diár – zmazanie poznámky

```
public void zmazPoznamku(int poradoveCislo) {  
    int pocet = this.poznamky.size();  
    if (poradoveCislo >= 0 && poradoveCislo < pocet) {  
        this.poznamky.remove(poradoveCislo);  
    }  
}
```

Chyba pri preklade

```
public class Diar {  
    private ArrayList<String> poznamky;  
  
    public Diar() {  
        this.poznamky = new ArrayList<String>();  
    }  
}
```

Cannot find symbol - class ArrayList

...

Diár – príkaz import

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

```
public class Diar {  
    private ArrayList<String> poznamky;  
  
    ...  
}
```

Používanie knižníc (1)

- knižnice – rozširujúca funkčnosť
 - delí sa na balíčky
 - balíčky obsahujú triedy
 - poznáme len rozhrania
- štandardná knižnica – súčasť jazyka Java
 - String – balíček java.lang
 - ArrayList – balíček java.util

Používanie knižníc (2)

- príkaz import:

```
import nazovBalicka.NazovTriedy;
```

- príklad:

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

- pre triedy z balíčka java.lang netreba import

Zobrazenie všetkých poznámok

- dokážeme zobrazit' ktorúkoľvek poznámku
- ďalšia požiadavka:
 - výpis všetkých poznámok do okna terminálu
- výpis musíme teda postupne zopakovať pre každú poznámku v kontajneri

Cyklus foreach

- jeden z cyklov
- pravidlo:
 - vykonaj pre každý prvok kontajnera

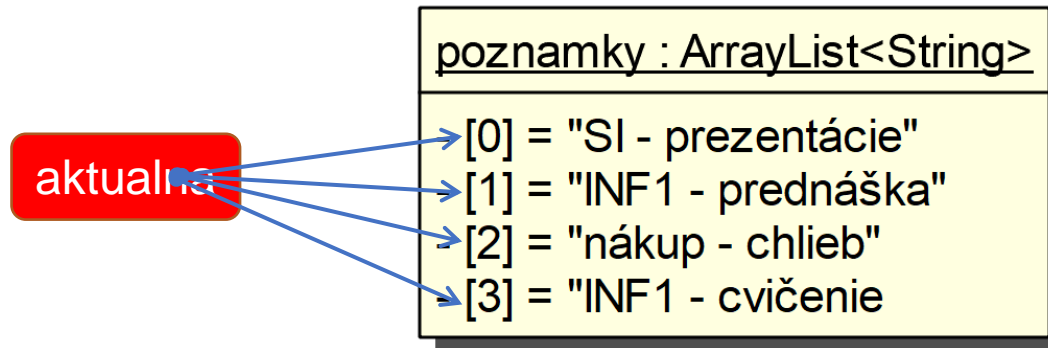
```
for (TypPrvku prvok : kontajner) {  
    // telo cyklu  
}
```

- !!! kontajner sa nesmie v tele cyklu meniť

Diár – výpis všetkých poznámok


```
public void zobrazVsetko() {  
    for (String aktualna : this.poznamky) {  
        System.out.println(aktualna);  
    }  
}
```

Cyklus foreach – vykonanie



```
BlueJ: Terminal Window - ...  
Options  
SI - prezentácia  
INF1 - prednáška  
nákup - chlieb  
INF1 - cvičenie  
Can only enter input while your
```

```
for (String aktualna : this.poznamky) {  
    System.out.println(aktualna);  
}
```



Rozšírenie možností diára

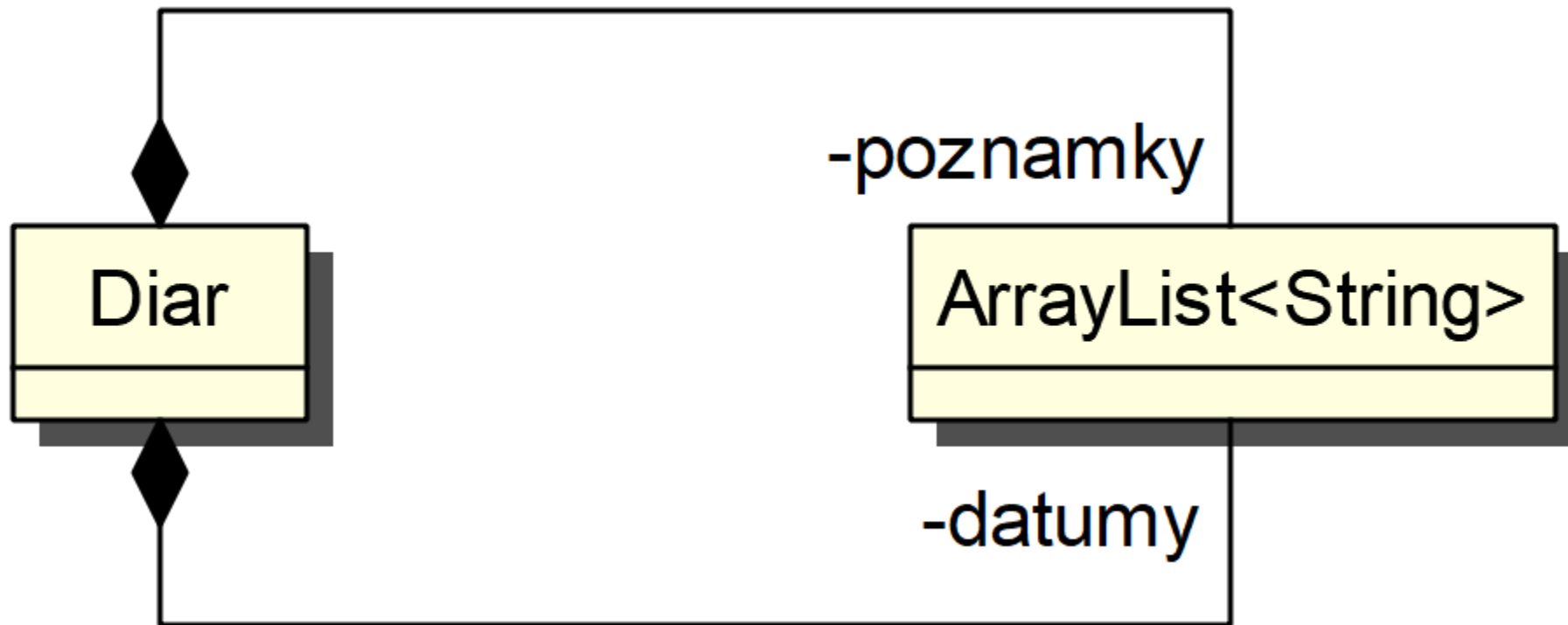
- aktuálna verzia – poznámka = len text
- požiadavka – ukladanie dátumu v poznámke

Nový diár – rozhranie

Diar

- + new(): Diar
- + vlozPoznamku(datum: String, poznamka: String): void
- + vypisPoznamku(poradoveCislo: int): void
- + zmazPoznamku(poradoveCislo: int): void
- + getPocetPoznamok(): int

Nový diár – možné riešenie



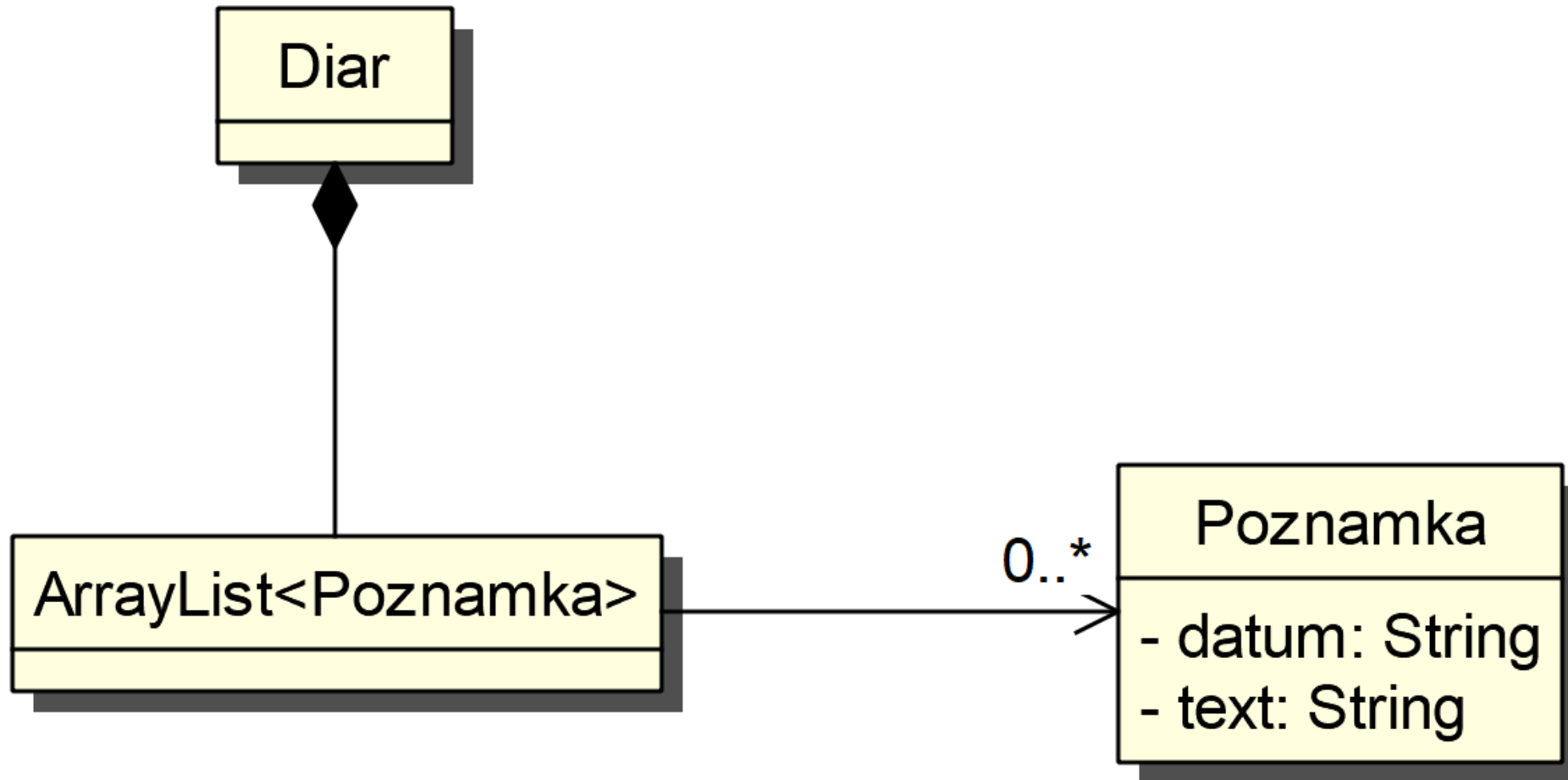
Nový diár – problémy

- samostatné dátumy + samostatné poznámky
- nie je možné jednoducho spárovať dátum a poznámku
- komplikované vypisovanie dátumov a poznámok
- je ľahké spraviť chybu
- ...

Riešenie – nová trieda Poznamka

- iba jeden zoznam – zoznam poznámok
- každá poznámka obsahuje
 - dátum
 - text

Nový diár – lepšie riešenie



Nový diár – vnútorný pohľad

Diar

- poznamky: `ArrayList<Poznamka>`

- + `Diar()`
- + `vlozPoznamku(datum: String, poznamka: String): void`
- + `vypisPoznamku(poradoveCislo: int): void`
- + `zmazPoznamku(poradoveCislo: int): void`
- + `getPocetPoznamok(): int`

ArrayList<Poznamka> – rozhranie

ArrayList<Poznamka>

- + new(): ArrayList<Poznamka>
- + add(prvok: Poznamka): void
- + get(index: int): Poznamka
- + remove(index: int): Poznamka
- + size(): int

Nový diár – definícia triedy, atribút

```
public class Diar {  
    private ArrayList<Poznamka> poznamky;  
  
    ...  
}
```

Nový diár – vloženie poznámky

```
public void vložPoznamku(String datum, String text) {  
    Poznamka nova = new Poznamka(datum, text);  
    this.poznamky.add(nova);  
}
```

Nový diár – vloženie poznámky

```
public void vložPoznamku(String datum, String text) {  
    this.poznamky.add(new Poznamka(datum, text));  
}
```

anonymný objekt



Anonymný objekt

- anonymná inštancia
 - nemá názov \Rightarrow nie je priradená do premennej
- použitie:
 - ako parameter správy
 - ako adresát správy

Anonymný objekt ako parameter správy

```
public void vložPoznamku(String datum, String text) {  
    this.poznamky.add(new Poznamka(datum, text));  
}
```

Anonymný objekt ako adresát správy

```
public void vypisPoznamku(int poradoveCislo) {  
    int pocet = this.poznamky.size();  
    if (poradoveCislo >= 0 && poradoveCislo < pocet) {  
        this.poznamky.get(poradoveCislo).zobrazPoznamku();  
    }  
}
```


ArrayList čísel

- toto v jazyku Java nefunguje:

```
ArrayList<int> cisla = new ArrayList<int>();
```

```
unexpected type  
required: reference  
found:    int
```

- prečo?
 - typovým parametrom generickej triedy musí byť objektový typ
- riešenie: obal'ovacia trieda

Oblažovacia trieda CeleCislo

```
public class CeleCislo {  
    private int cislo;  
  
    public CeleCislo(int cislo) {  
        this.cislo = cislo;  
    }  
  
    public int getCislo() {  
        return this.cislo;  
    }  
}
```

Použitie obalovacej triedy

```
ArrayList<CeleCislo> cc = new ArrayList<CeleCislo>();  
cc.add(new CeleCislo(5));  
  
// Vypíše 5  
System.out.println(cc.get(0).getCislo());
```

Obalovacie triedy v jazyku Java

primitívny typ	obaľovacia trieda
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double
char	Character
boolean	Boolean

Použitie štandardnej obalovacej triedy

```
int celeInt = 15;  
  
Integer obaleneCislo = new Integer(celeInt);  
  
int druheCeleInt = obaleneCislo.intValue();
```

Automatické konverzie

- typová kompatibilita
 - primitívny typ \leftrightarrow obal'ovacia trieda
- boxing
 - prevod primitívneho typu na objektový
- unboxing
 - prevod objektového typu na primitívny

Boxing/unboxing

```
int celeInt = 15;
```

```
// Integer obaleneCislo = new Integer(celeInt);
```

```
Integer obaleneCislo = celeInt;
```

```
// int druheCeleInt = obaleneCislo.intValue();
```

```
int druheCeleInt = obaleneCislo;
```

```
// ???
```

```
obaleneCislo = obaleneCislo + 1;
```

Obalovacia trieda a ArrayList

```
ArrayList<Integer> ciska = new ArrayList<Integer>();  
  
ciska.add(15);  
ciska.add(cislo);  
  
for (int cislo : ciska) {  
    System.out.println(cislo);  
}
```


Informatika 1

Základy vstupu-výstupu



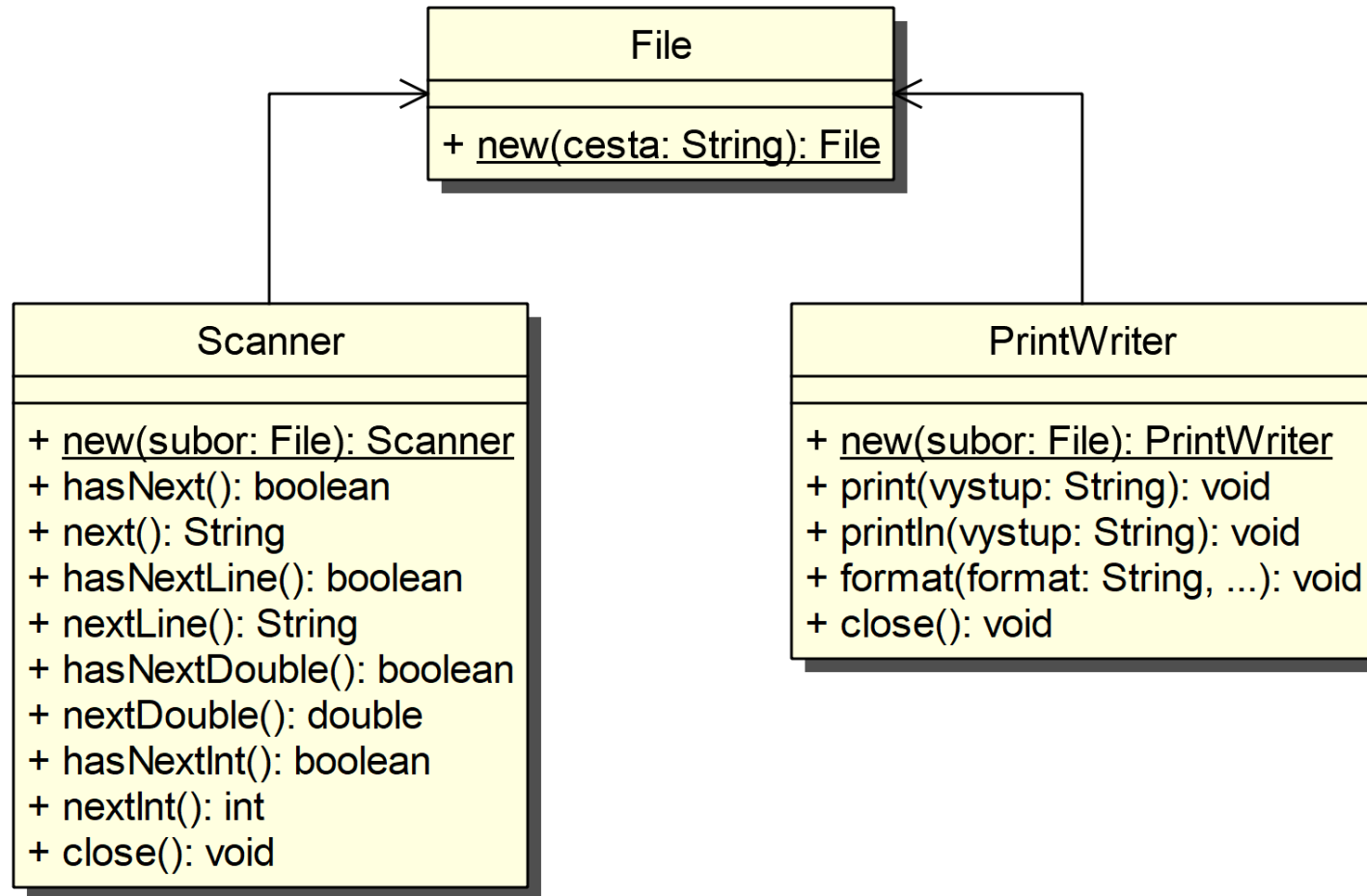
Vstup-výstup

- hromadnosť – vlastnosť algoritmu
- program je tiež algoritmus
 - pre univerzálnosť potrebuje vstupy
 - potrebuje prezentovať výstupy
- mnoho možností
- zatiaľ dve:
 - textový súbor
 - komunikácia s používateľom

Práca so súborom

- čítanie
 - otvorenie súboru na čítanie
 - postupné čítanie obsahu
 - zatvorenie súboru
- zápis
 - otvorenie súboru na zápis
 - postupný zápis nového obsahu
 - zatvorenie súboru

Práca so súbormi v jazyku Java



Práca so súborom – príkazy import

```
import java.util.Scanner;  
import java.io.File;  
import java.io.IOException;  
import java.io.PrintWriter;
```

Diár – zápis poznámok

- súbor diar.txt
- otvorenie
- pre každý záznam
 - dátum – print
 - medzera – print
 - text – println
 - alebo zápis všetkého cez format(...)
- zatvorenie

Diar – metóda zapis

```
public void zapis() throws IOException {
```

```
File subor = new File("diar.txt");
```

```
PrintWriter zapisovac = new PrintWriter(subor);
```

otvor

```
for (Poznamka poznamka : this.poznamky) {
```

```
    subor.format("%s %s%n", poznamka.getDatum(),
```

```
        poznamka.getText());
```

```
}
```

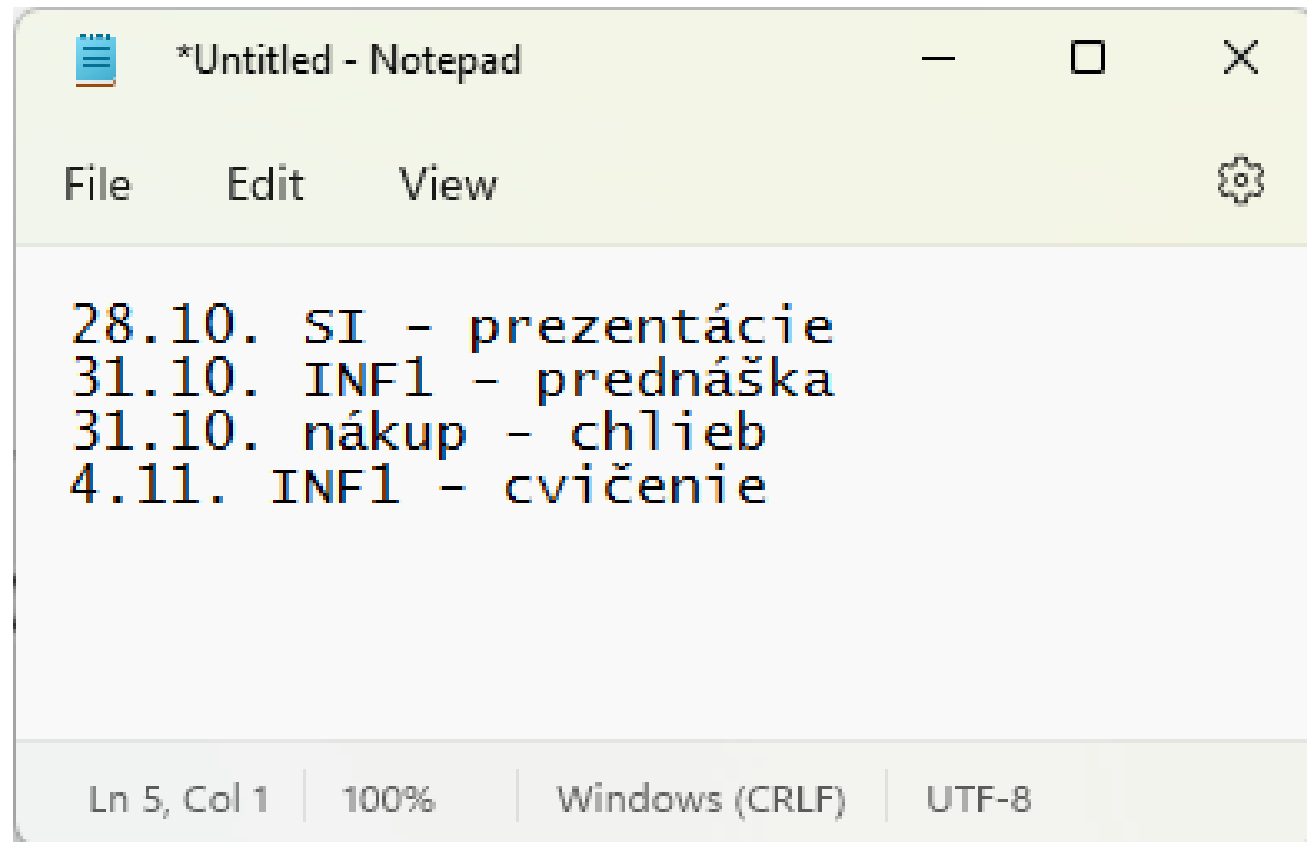
zapiš

```
zapisovac.close();
```

zatvor

```
}
```

Poznámky v súbore



Diár – čítanie poznámok

- súbor diar.txt
- otvorenie
- čítanie kým existuje ďalší riadok
 - dátum – next
 - text poznámky – nextLine
- zatvorenie

Diar – metóda nacitaj (1)

```
public void nacitaj() throws IOException {
```

```
    File subor = new File("diar.txt");  
    Scanner citac = new Scanner(subor);
```

otvor

```
    this.poznamky.clear();  
    while (citac.hasNextLine()) {  
        ...  
    }
```

čítaj

```
    citac.close();
```

zatvor

```
}
```

Diar – metóda nacitaj (2)

```
String datum = citac.next();  
String text = citac.nextLine();  
  
this.poznamky.add(new Poznamka(datum, text));
```

Komunikácia s používateľom

- prostredníctvom terminálu
 - vstup z klávesnice
 - výstup do okna terminálu
- grafické prostredie JOptionPane
 - napr. pre hry naprogramované s knižnicou tvaryV3
- grafické prostredie
 - budúci semester

Používateľský vstup z terminálu

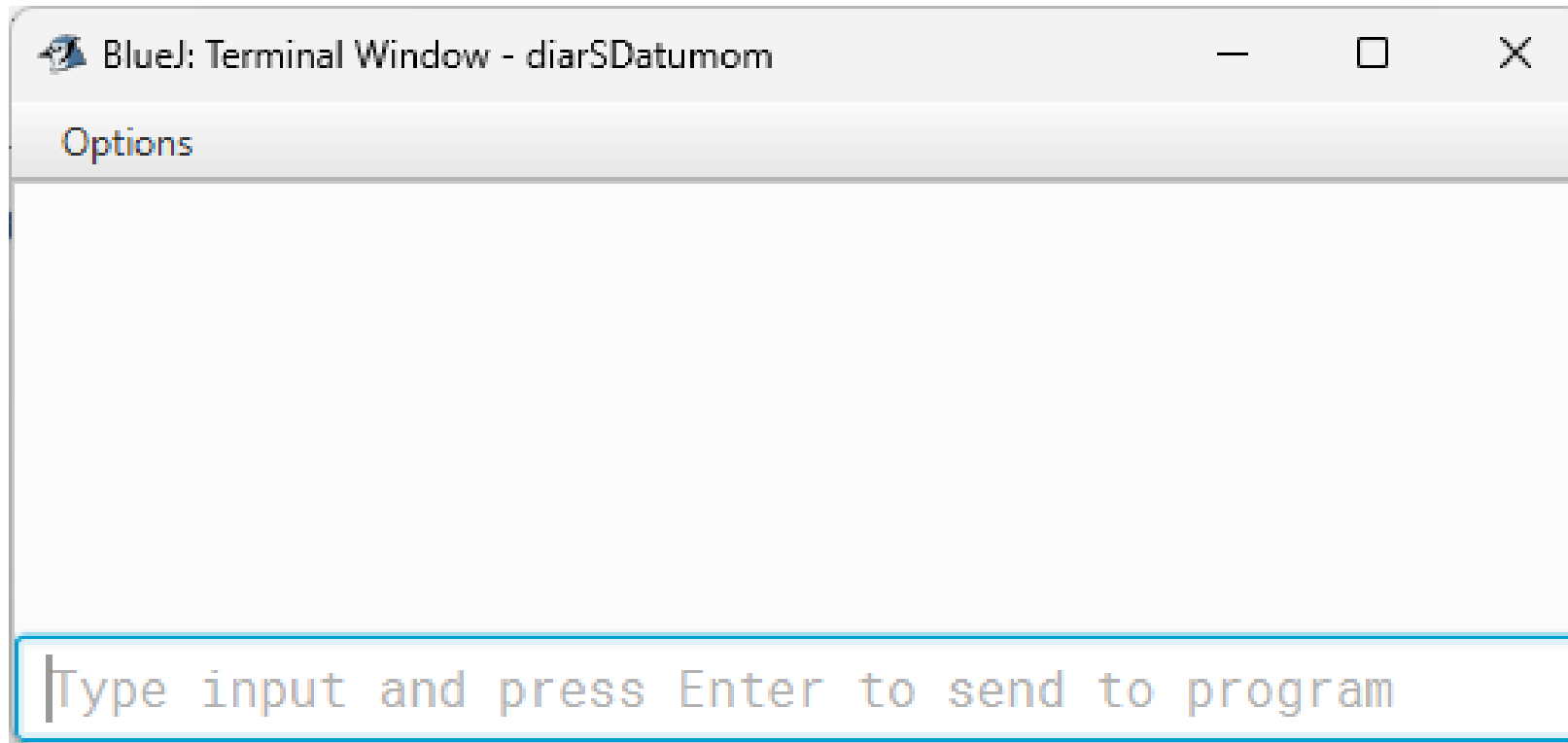
- trieda Scanner
 - rovnako ako čítanie zo súboru
- objekt System.in ako parameter konštruktora

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

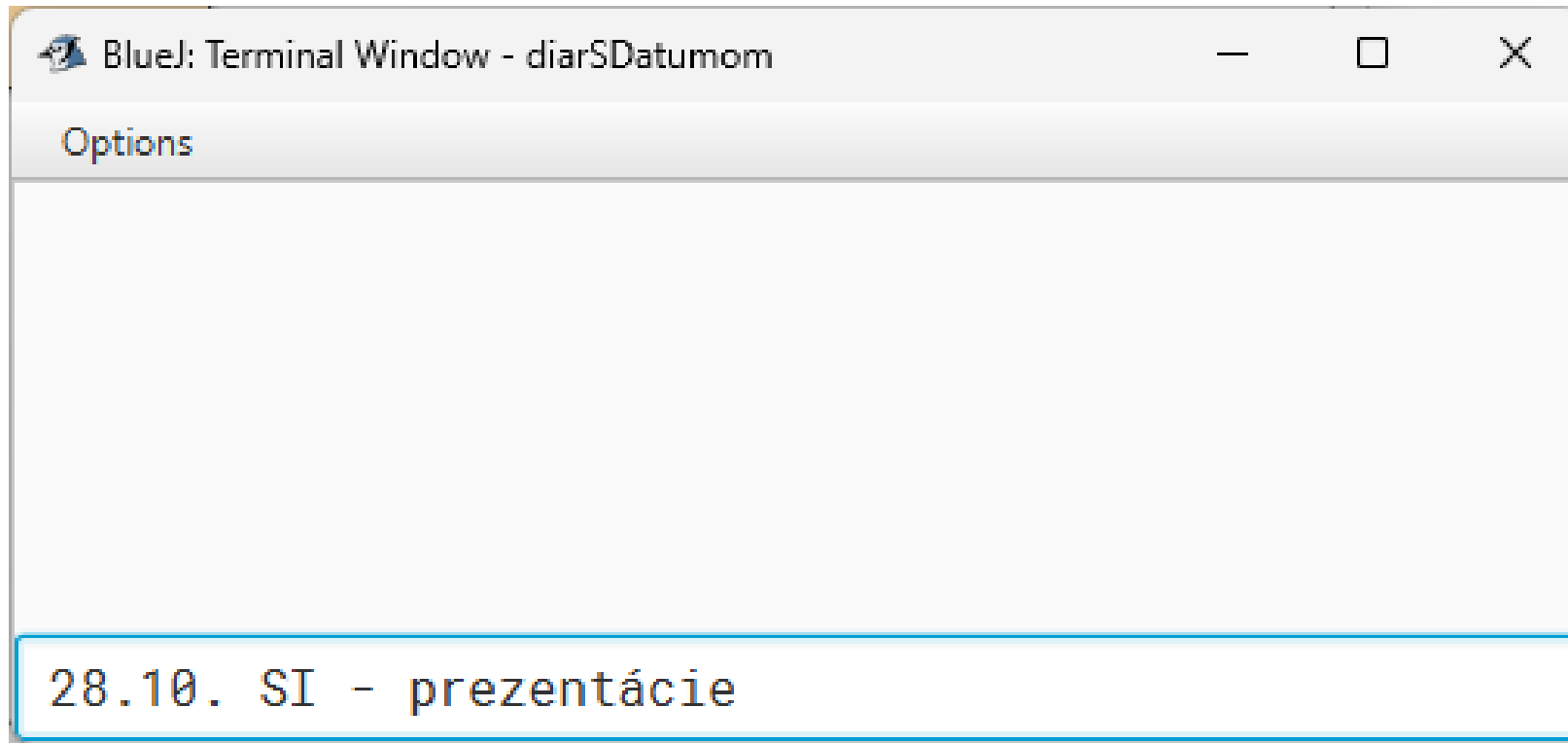
Diar – metóda nacitaj (1)

```
public void nacitajZTerminalu() {  
    Scanner vstup = new Scanner(System.in);  
  
    for (;;) {  
        String datum = vstup.next();  
        if (datum.equals("hotovo")) {  
            break;  
        }  
        String text = vstup.nextLine();  
        this.poznamky.add(new Poznamka(datum, text));  
    }  
}
```

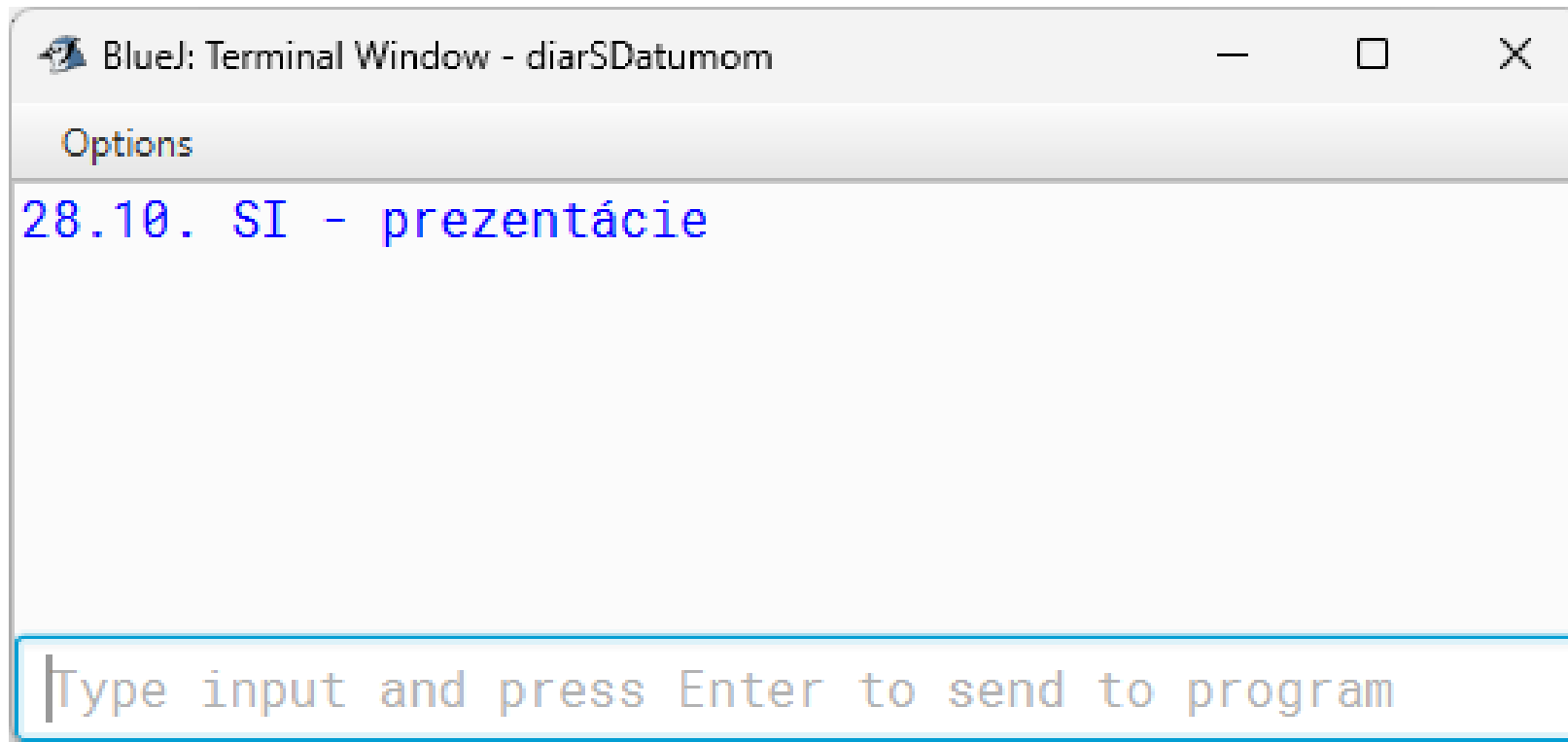
Zadávanie vstupu v nástroji BlueJ (1)



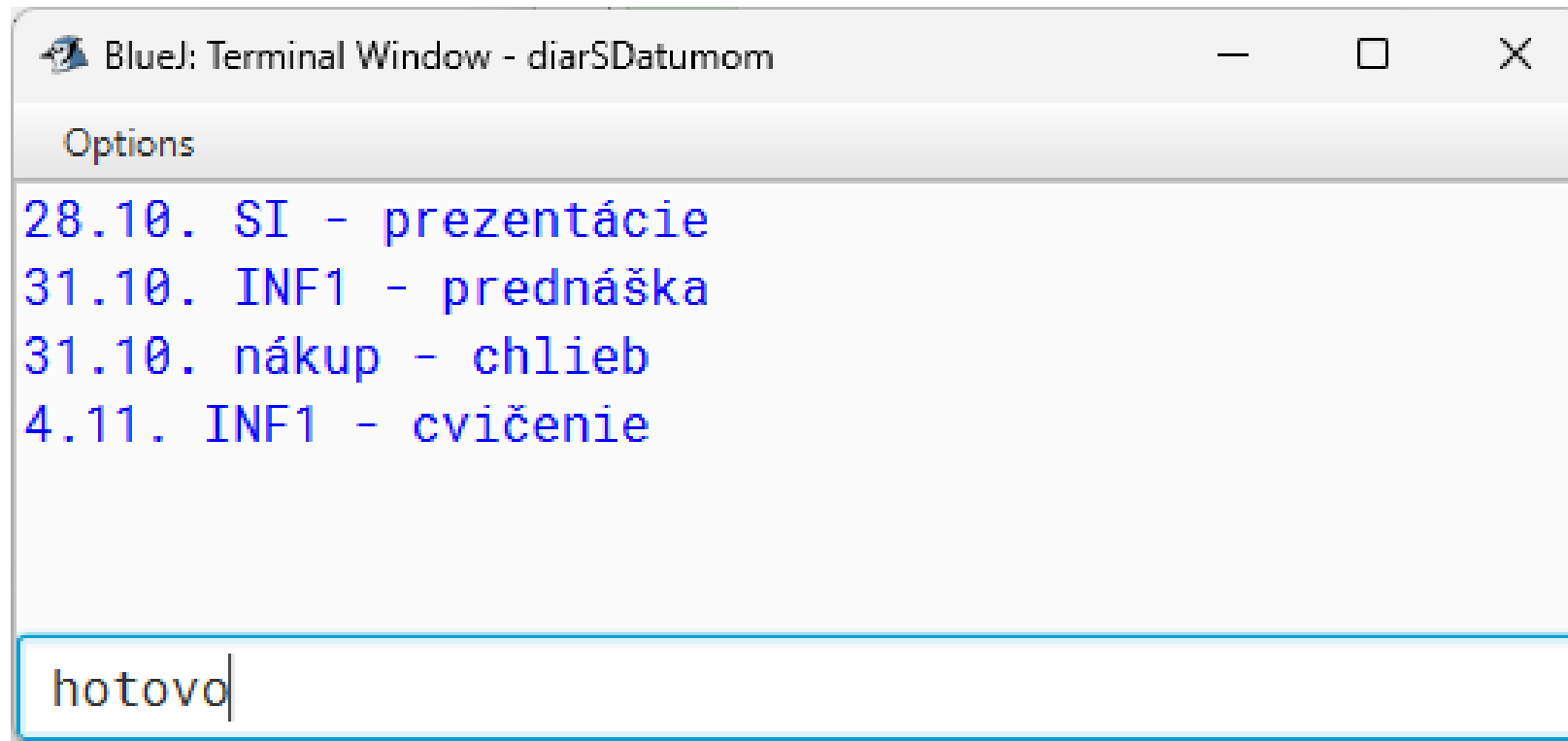
Zadávanie vstupu v nástroji BlueJ (2)



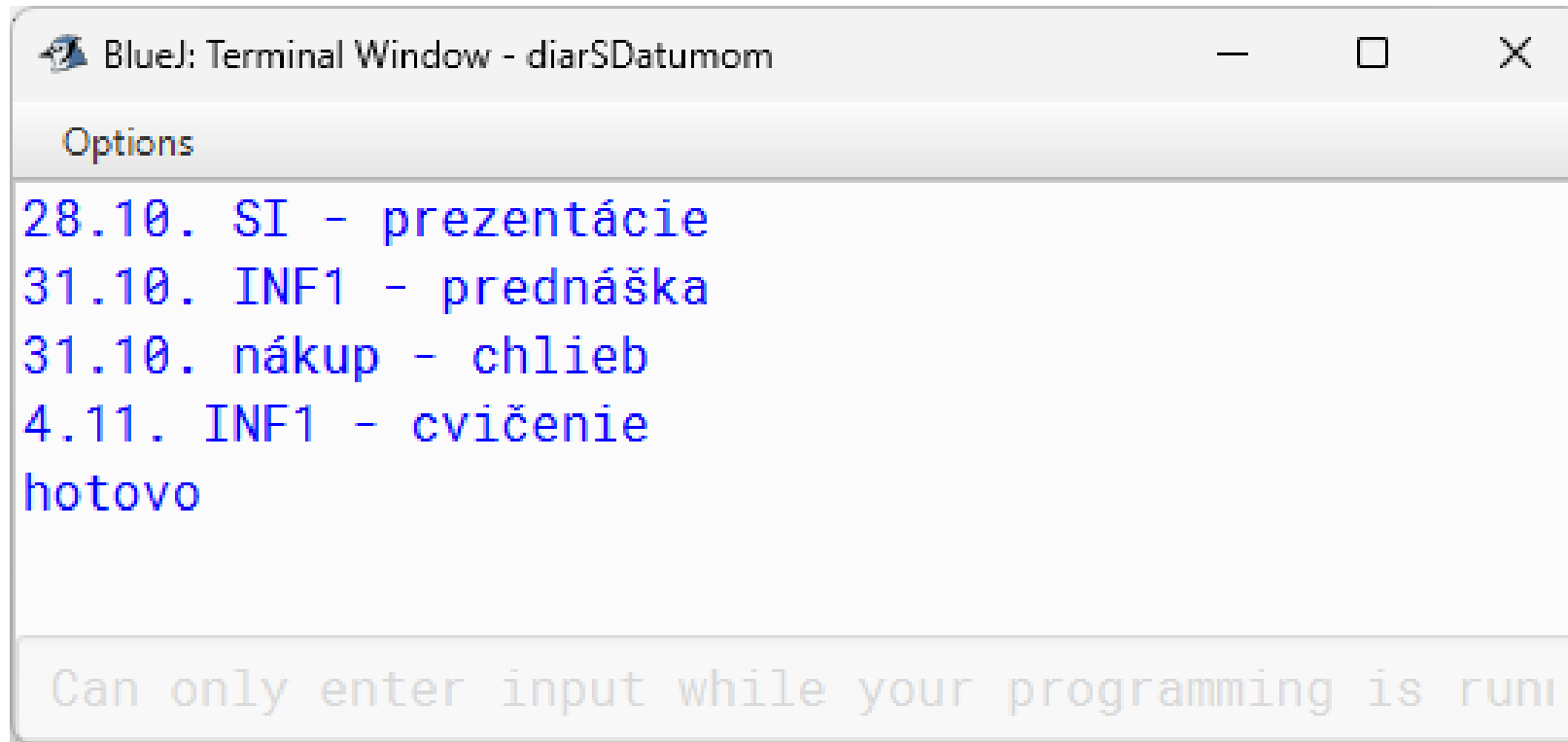
Zadávanie vstupu v nástroji BlueJ (3)



Zadávanie vstupu v nástroji BlueJ (4)



Zadávanie vstupu v nástroji BlueJ (5)

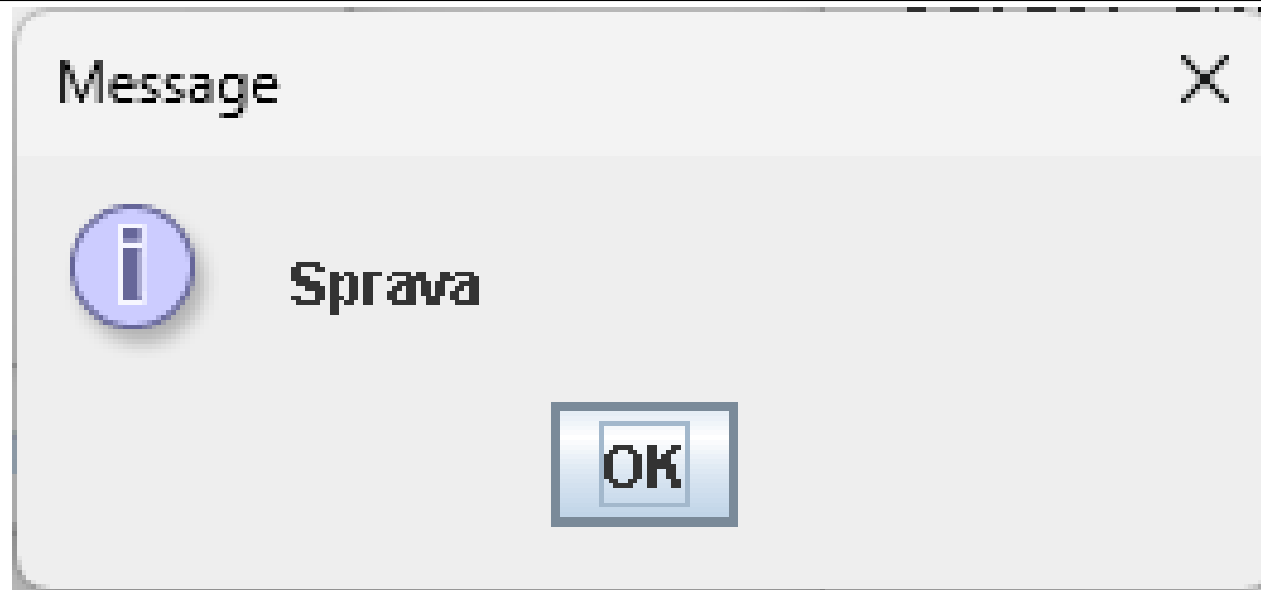


JOptionPane – výstup

```
import javax.swing.JOptionPane;
```

- zobrazenie správy:

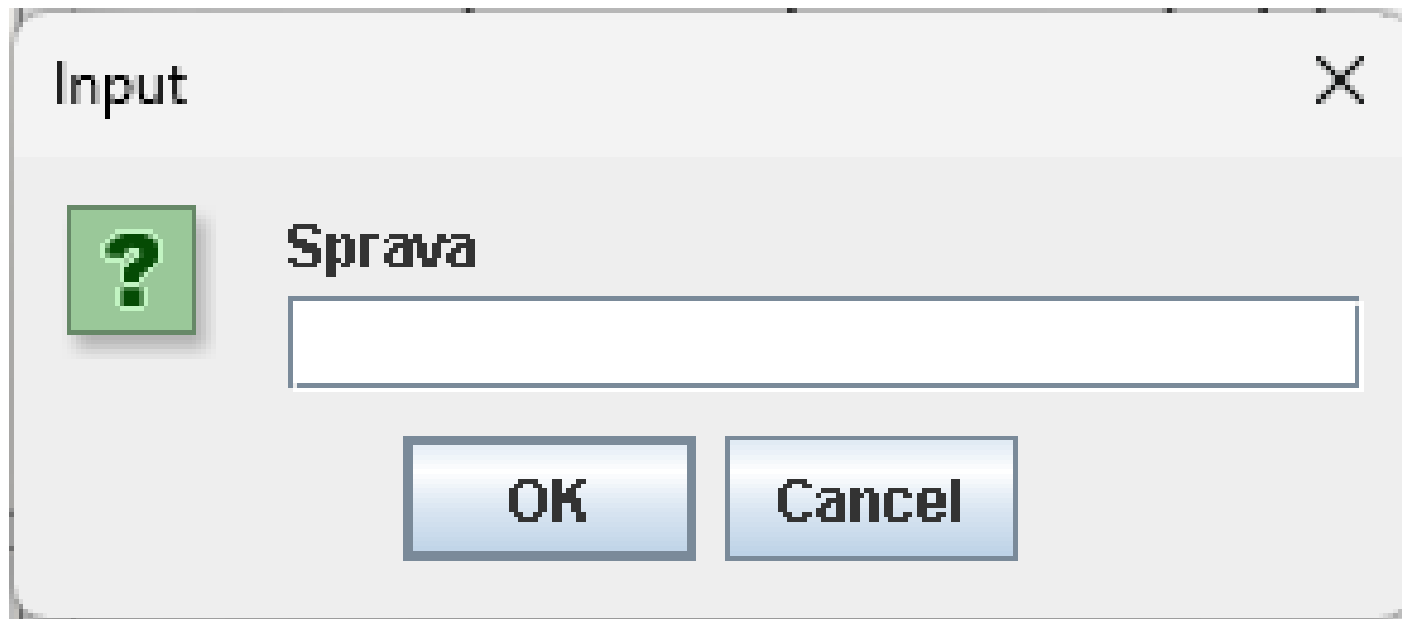
```
JOptionPane.showMessageDialog(null, "Sprava");
```



JOptionPane – vstup

- načítanie reťazca:

```
String text = JOptionPane.showInputDialog(null, "Sprava");
```



JOptionPane – vstup

- potvrdzovací dialóg

```
int moznost = JOptionPane.showConfirmDialog(null, "Sprava");
```

- 0 = Yes
- 1 = No
- 2 = Cancel

