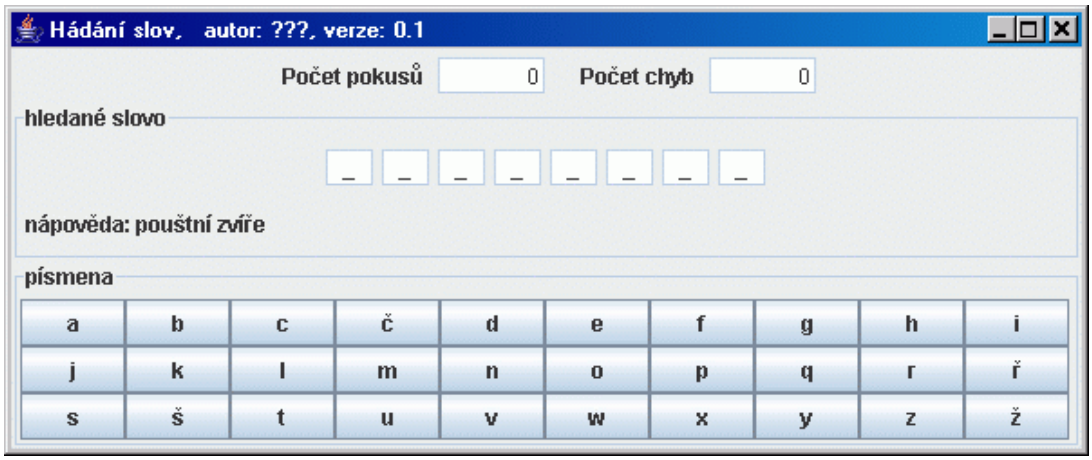
16.Projekt Hádání slov

16.1. Základní popis, zadání úkolu

Pracujeme na projektu Hádání slov, který je ke stažení na `java.vse.cz`. Po otevření v BlueJ vytvoříme instanci třídy `HadaniSlov` a zavoláme metodu `show()`. Spustí se grafická aplikace umožňující hádání slov viz obrázek 16.1.



Obrázek 16.1 Vzhled aplikace Hádání slov po spuštění

Naším úkolem je:

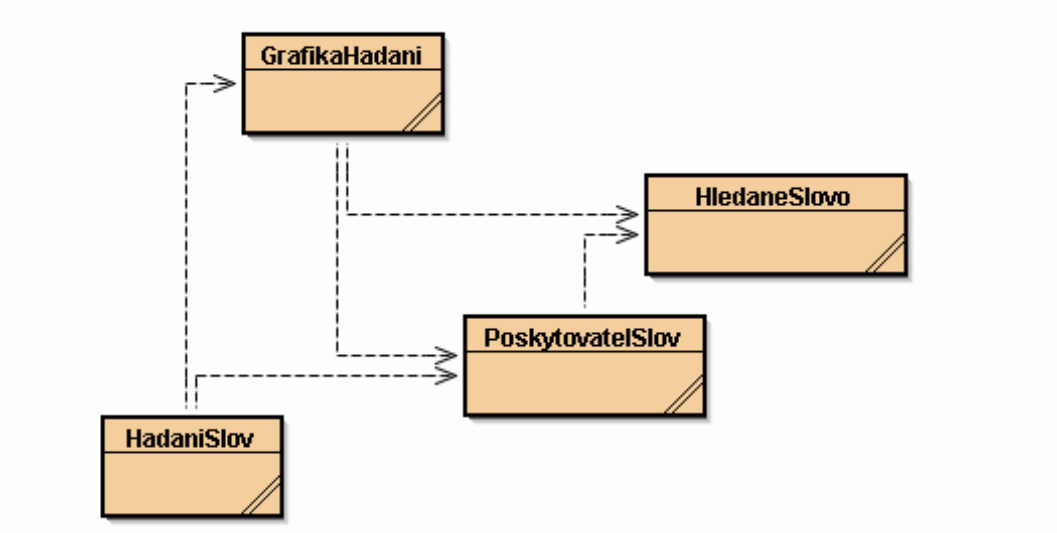
- rozšířit tuto aplikaci tak, aby poskytovala více slov k hádání,
- poskytovat slova k hádání v náhodném pořadí.

Cílem tohoto projektu je:

- naučit se používat seznamy v Javě (konkrétně třídu `java.util.ArrayList` a některé její metody),
- naučit se generovat náhodná čísla v Javě.

16.2. Struktura tříd

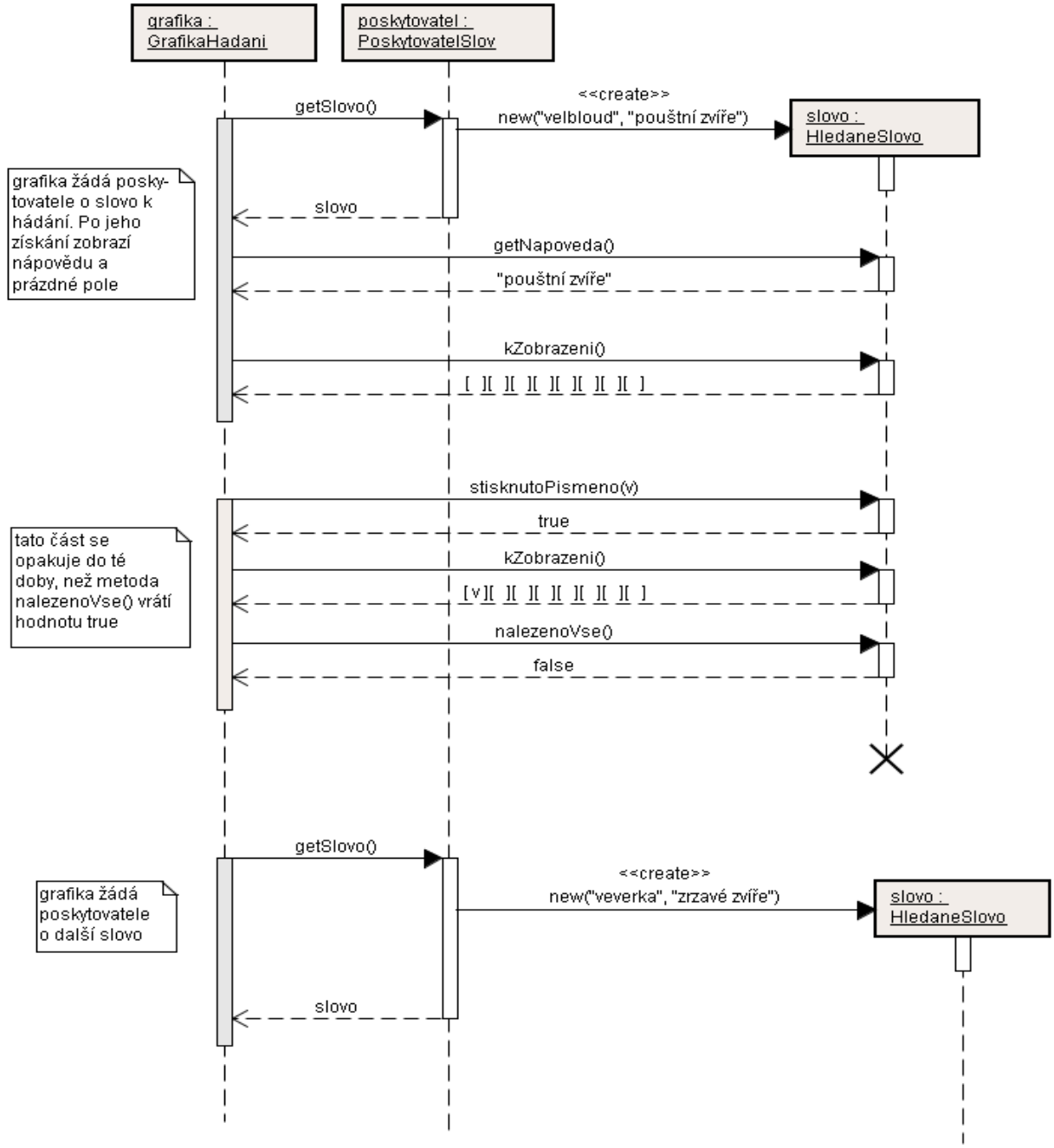
Projekt Hádání slov je tvořen těmito třídami:



Obrázek 16.2 Struktura tříd projektu Hádání slov, převzato z BlueJ

16.3. Popis komunikace mezi objekty

V konstruktoru třídy `HadaniSlov` se vytvoří instance třídy `PoskytovatelSlov` a předá se jako parametr konstruktoru při vytváření instance třídy `GrafikaHadani`. V konstruktoru třídy `PoskytovatelSlov` je vytvořena instance třídy `HledaneSlovo`.
Při zobrazení grafiky na obrazovce instance třídy `GrafikaHadani` zavolá metodu `getSlovo()` instance `PoskytovatelSlov` a získá odkaz na instanci třídy `HledaneSlovo`. Dokud hráč neuhádne slovo, posílá `GrafikaHadani` zprávy instanci třídy `HledaneSlovo`. Po uhodnutí slova `GrafikaHadani` požádá o další slovo instanci `PoskytovalSlov`. Pokud již nemá další slovo k hádání, vrátí hodnotu `null` a na obrazovce se zobrazí zpráva, že již není další slovo k hádání.
Instance třídy `GrafikaHadani` se při komunikaci s instancí třídy `HledaneSlovo` nejdříve „zeptá“ na nápovědu (voláním metody `getNapoveda()`) a na políčka k zobrazení (voláním metody `kZobrazeni()`).
Po stisknutí tlačítka uživatelem pošle grafika stisknuté písmeno instanci třídy `HledaneSlovo` (volání metody `stisknutoPismeno`) a následně požádá o políčka k zobrazení (opět volání metody `kZobrazení()`). Dále se zeptá, zda je již celé slovo uhodnuto (metoda `nalezenoVse()`) – pokud ano, tak zobrazí příslušnou zprávu na obrazovce a přejde k dalšímu slovu.
Postup komunikace mezi jednotlivými třídami je zachycen v diagramu na obrázku 16.3.



Obrázek 16.3 Diagram komunikace mezi instancemi při hádání slov

16.4. Popis kódu třídy PoskytovatelSlov

Třída `PoskytovatelSlov` obsahuje datový atribut `slovo` typu `HledaneSlovo` (řádek 13 v následujícím výpise). V konstruktoru je `slovo` inicializováno (řádek 19). Metoda `getSlovo()` pro každé zavolání vrací odkaz na tuto instanci. Mimo tuto metodu jsou zde i dvě jednoduché metody poskytující jméno autora a číslo verze (řádky 34 – 53). Díky této implementaci je možné hádat pouze jedno slovo. Jakmile hráč uhodne, je mu znovu nabídnuto stejné slovo, navíc již s doplněnými písmeny z minulého hádání.

```
 1 /**
 2  * Tato třída zpřístupňuje grafickému rozhraní slovo,
 3  * které se má hádat. Slovo je
 4  * instancí třídy HledaneSlovo a obsahuje vedle vlastního
 5  * slova i nápovědu, která se zobrazí.
 6  * Třída dále poskytuje informace o autorovi a o verzi.
 7  *
 8  * @author  Luboš Pavlíček
 9  * @version 1.0
10  */
11 public class PoskytovatelSlov {
12
13   private HledaneSlovo slovo;
14   /**
15    * konstruktor pro vytvoření instance třídy pro
16    * poskytování slov
17    */
18   public PoskytovatelSlov() {
19      slovo = new HledaneSlovo("velbloud","pouštní zvíře");
20   }
21
22   /**
23    * Metoda vrací slovo k hádání jako instanci třídy
24    * HledaneSlovo (tj. ke slovu je připojena nápověda).
25    * Pokud není již další slovo k dispozici, vrátí
26    * hodnotu null
27    *
28    * @return     slovo k hádání
29    */
30   public HledaneSlovo getSlovo() {
31      return slovo;
32      }
33
34   /**
35    * Metoda vrací jméno autora programu –
36    * jméno se zobrazuje v grafickém rozhraní
37    *
38    * @return     jméno autora, např. "L. Pavlicek"
39    */
40   public String getAutor() {
41      return "???";
42   }
43
44   /**
45    * Metoda vrací verzi programu - ta se zobrazuje
46    * v grafickém rozhraní
47    *
48    * @return   verze programu, např. "1.0"
49    */
50   public String getVerze() {
51   return "0.1";
52   }
53 }
```

16.5. Postup řešení

16.5.1. Přidání možnosti hádat více slov

Naším prvním úkolem je úprava třídy `PoskytovatelSlov`, aby nabízela více slov k hádání. Jakmile hráč vyčerpá zásobu slov, vrátí metoda `getSlovo()` hodnotu `null`. Nebudeme potřebovat datový atribut `slovo`, ale jiný, který umožní uchovat větší počet slov k hádání. Pro jejich uložení použijeme instanci třídy `ArrayList`, kterou nazveme `seznamSlov`. Třída `ArrayList` je z balíčku `java.util` a je tedy nutné psát plné jméno `java.util.ArrayList` nebo na úvodní řádky kódu doplnit `import` následujícím způsobem:

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

Do tohoto seznamu budeme ukládat instance třídy `HledaneSlovo`. Při deklaraci seznamu uvedeme toto:

```
    private ArrayList <HledaneSlovo> seznamSlov;
```

Nyní musíme v konstruktoru tento datový atribut inicializovat (řádek 22) a naplnit zásobou slov k hádání (řádky 23 – 28). Třída `HledaneSlovo` má několik nedostatků, na které je nyní nutno poukázat: neumí samohlásky s interpunkcí a nezná velká písmena. Slova k hádání tedy mohou obsahovat pouze ty znaky, které jsou uvedeny na „klávesnici“ grafického uživatelského rozhraní aplikace, např. Praha s malým p na začátku (řádek 27).

```
 1 import java.util.ArrayList;
 2
 3 /**
 4  * Tato třída zpřístupňuje grafickému rozhraní slova,
 5  * která se mají hádat. Slovo je instancí třídy HledaneSlovo,
 6  * slova jsou uložena v seznamu (ArrayList).
 7  * Třída dále poskytuje informace o autorovy a o verzi.
 8  *
 9  * @author  Luboš Pavlíček
10  * @version 1.1
11  */
12 public class PoskytovatelSlov {
13
14   private ArrayList <HledaneSlovo> seznamSlov;
15   private int index;
16
17   /**
18    * konstruktor pro vytvoření instance třídy pro
19    * poskytování slov
20    */
21   public PoskytovatelSlov() {
22       seznamSlov = new ArrayList<HledaneSlovo>();
23       seznamSlov.add(new HledaneSlovo("velbloud",
24                           "pouštní zvíře"));
25       seznamSlov.add(new HledaneSlovo ("veverka",
26                           "zrzavé zvíře"));
27       seznamSlov.add(new HledaneSlovo ("praha",
28                           "naše hlavní město"));
29       index = 0;
30   }
31
```

```
32   /**
33   * Metoda vrací slovo k hádání jako instanci třídy HledaneSlovo
34    * (tj. ke slovu je připojena nápověda).
35    * Pokud není již další slovo k dispozici, vrátí hodnotu null
36    *
37    * @return     slovo k hádání
38    */
39   public HledaneSlovo getSlovo() {
40   HledaneSlovo slovo = null;
41   if (index < seznamSlov.size() ) {
42      slovo = seznamSlov.get(index);
43      index++;
44   }
45   return slovo;
46   }
```

Metoda `getSlovo()` při každém zavolání postupně vrací jednotlivá slova, pokud už není žádné slovo k dispozici, vrátí hodnotu `null`. Je nutné přidat další datový atribut, abychom si zapamatovali, které slovo je na řadě (musí to být datový atribut, aby se v něm uchovávala pozice mezi jednotlivými voláními metody `getSlovo()`). Využijeme toho, že `ArrayList` používá indexy – budeme si uchovávat, které slovo je další na řadě k hádání. Datový atribut pojmenujeme `index`, bude typu `int` (řádek 15) a v konstruktoru mu přiřadíme hodnotu 0 (řádek 29). Hodnota nula odkazuje na první prvek v `seznamSlov`, protože Java používá indexy v rozsahu `0` až `n-1`, kde `n` je počet prvků.
V metodě `getSlovo()` si na začátku deklarujeme pomocnou proměnnou `slovo` typu `HledaneSlovo` a inicializujeme ji hodnotou `null` (řádek 40). Potom zkontrolujeme, zda hodnota atributu `index` je menší než počet prvků v seznamu slov, který zjistíme metodou `size()` – řádek 41. Pokud je `index` menší, do pomocné proměnné uložíme metodou `get(index)` odkaz na instanci třídy `HledaneSlovo`, která je na pozici udané indexem (řádek 42). Následně zvýšíme hodnotu indexu o jedna, abychom při příštím volání metody `getSlovo()` poskytli následující slovo (řádek 43).
Pokud je index větší nebo roven počtu prvků seznamu, znamená to, že již nemáme další slovo k hádání – v proměnné `slovo` tedy ponecháme hodnotu `null`, kterou jsme přiřadili při inicializaci. Větev `else` příkazu `if` proto nepotřebujeme.
Na konci metody příkazem `return` předáme obsah pomocné proměnné `slovo` (řádek 45 výpisu).

16.5.2. Poskytování slov k hádání v náhodném pořadí

Nyní budeme řešit druhou část zadání, tj. poskytování slov v náhodném pořadí. Budeme postupovat následovně:

- Potřebujeme náhodné číslo z intervalu `0` až `n-1` (`n` je počet slov k hádání v kopii seznamu), neboť indexy jsou od `0` do `n-1`. Pro generování náhodných čísel poskytuje Java třídu `Random` z balíčku `java.util`. Stejně jako u třídy `ArrayList` můžeme psát buď celý název (`java.util.Random`) nebo na začátku zdrojového textu importovat tuto třídu (`import java.util.Random;`) a poté psát jenom název `Random`. V konstruktoru vytvoříme instanci třídy `Random` (řádek 16 deklarace a řádek 30 inicializace). Třída `Random` má více metod pro generování náhodných čísel, my použijeme metodu `nextInt(int n)`, která vrací celé kladné číslo z intervalu `<0, n)`, tj. od nuly (včetně) po hodnotu parametru (není v intervalu zahrnuta)[36]. Jako parametr použijeme aktuální velikost seznamu slov (řádek 43).

[36] Čtenáře může napadnout, co vrací metoda `nextInt(int n)`, pokud bude parametrem záporná hodnota. Máte tří možnosti, jak to zjistit – podívat se do dokumentace třídy `Random`, napsat si vlastní zkušební program a podívat se do zdrojového textu třídy `Random`.

- Slovo k hádání, které je uloženo v seznamu na místě s indexem odpovídajícím náhodnému číslu, uložíme do pomocné proměnné *slovo* (deklarace a inicializace na řádku 41).
- V seznamu toto slovo zrušíme, metoda vrátí vybrané slovo z pomocné proměnné.
- Pokud již v seznamu slov k hádání není žádné slovo, vrátí metoda hodnotu *null*, která se ukládá do pomocné proměnné slovo na začátku metody (zda je či není slovo k hádání k dispozici, se zjišťuje na řádku 42).

V uvedeném řešení se slovo určené k hádání maže ze seznamu slov, tj. po předání již není k dispozici v instanci třídy *PoskytovatelSlov*. Pokud bychom chtěli slova dále uchovávat, vytvořili bychom na začátku kopii seznamu a slova bychom vydávali z kopie.

```
 1  import java.util.Random;
 2  import java.util.List;
 3  import java.util.ArrayList;
 4
 5  /**
 6   * Tato třída zpřístupňuje v náhodném pořadí grafickému rozhraní
 7   * slova, která se mají hádat. Slovo je instancí třídy
 8   * HledaneSlovo, slova jsou uložena v seznamu (ArrayList).
 9   * Třída dále poskytuje informace o autorovi a o verzi.
10   *
11   * @author  Luboš Pavlíček
12   * @version 1.1
13   */
14  public class PoskytovatelSlov {
15
16    private List <HledaneSlovo> seznamSlov;
17    private Random nahoda;
18
19    /**
20     * konstruktor pro vytvoření instance třídy pro
21     * poskytování slov
22     */
23    public PoskytovatelSlov() {
24       seznamSlov = new ArrayList<HledaneSlovo>();
25       seznamSlov.add(new HledaneSlovo
26                           ("velbloud", "pouštní zvíře"));
27    seznamSlov.add(new HledaneSlovo
28                           ("panenka", "hračka pro holčičky"));
29       seznamSlov.add(new HledaneSlovo
30                           ("praha", "naše hlavní město"));
31       nahoda = new Random();
32    }
33
34    /**
35     * Metoda vrací slovo k hádání jako instanci třídy HledaneSlovo
36     * (tj. ke slovu je připojena nápověda).
37     * Pokud není již další slovo k dispozici, vrátí hodnotu null
38     *
39     * @return     slovo k hádání
40     */
```

```
41   public HledaneSlovo getSlovo() {
42   HledaneSlovo slovo = null;
43   if (seznamSlov.size() > 0 ){
44       int index = nahoda.nextInt(seznamSlov.size());
45       slovo = seznamSlov.get(index);
46       seznamuSlov.remove(index);
47   }
48   return slovo;
49   }
```

16.6. Domácí úkol

1. Upravte aplikaci tak, aby umožňovala používat malá a velká písmena (tj. aby slovo Praha bylo s velkým písmenem P) a aby se nerozlišovaly samohlásky s čárkami, háčky (např. pokud by se mělo hádat slovo „nápad“, tak by se po volbě tlačítka „a“ odkryly znaky „a“ a „á“). Neměli by jste upravovat grafické rozhraní aplikace.