PHRUB.CZ
Hlavné klid

JAVA
úvod do programování

Mgr. Pavel Hrubý, 2017

PHRUB.CZ
Hlavně klid

Úvod, seznámení s programovacím jazykem Java

Vlastnosti jazyka JAVA

jednoduchý – jeho syntaxe je zjednodušenou (a drobně upravenou) verzí syntaxe jazyka C a C++. Odpadla většina nízkoúrovňových konstrukcí. Součástí jazyka nejsou například ukazatelé, bezznaménkové číselné datové typy, příkaz goto nebo preprocesor.

objektově orientovaný – s výjimkou osmi primitivních datových typů jsou všechny ostatní datové typy objektové.

distribuovaný – je navržen pro podporu aplikací v síti (podporuje různé úrovně síťového spojení, práce se vzdálenými soubory, umožňuje vytvářet distribuované klientské aplikace a servery.

interpretovaný – místo skutečného strojového kódu se vytváří pouze tzv. mezikód (bajtkód). Tento formát je nezávislý na architektuře počítače nebo zařízení. Program pak může pracovat na libovolném počítači nebo zařízení, který má k dispozici interpret Javy, tzv. virtuální stroj Javy – Java Virtual Machine (JVM).

- V pozdějších verzích Javy nebyl mezikód přímo interpretován, ale před prvním svým provedením dynamicky zkompilován do strojového kódu daného počítače (tzv. just in time compilation – JIT). Tato vlastnost zásadním způsobem zrychlila provádění programů v Javě, ale výrazně zpomalila start programů.
- V současnosti se převážně používají technologie zvané HotSpot compiler, které mezikód zpočátku interpretují a na základě statistik získaných z této interpretace později provedou překlad často používaných částí do strojového kódu včetně dalších dynamických optimalizací (jako je např. inlining krátkých metod atp.).
- Některé platformy nabízejí přímou hardwarovou podporu pro Javu. Existují též mikroprocesory, které dokáží spustit Javu hardwarově namísto softwarové emulace Java Virtual Machine. ARM procesory mohou mít přímou hardwarovou podporu pro spuštění binárního kódu Javy.
- robustní – je určen pro psaní vysoce spolehlivého softwaru – z tohoto důvodu neumožňuje některé programátorské konstrukce, které bývají častou příčinou chyb (např. správa paměti, příkaz goto, používání ukazatelů). Používá tzv. silnou typovou kontrolu – veškeré používané proměnné musí mít definovaný svůj datový typ.
- generační správa paměti – správa paměti je realizována pomocí automatického garbage collectoru, který automaticky vyhledává již nepoužívané části paměti a uvolňuje je pro další použití. To bylo v prvních verzích opět příčinou pomalejšího běhu programů. V posledních verzích běhových prostředí je díky novým algoritmům pro garbage collection a tzv. generační správě paměti (paměť je rozdělena na více částí, v každé se používá jiný algoritmus pro garbage collection a objekty jsou mezi těmito částmi přesunovány podle délky svého života) tento problém ze značné části eliminován.

Úvod, seznámení s programovacím jazykem Java

bezpečný – má vlastnosti, které chrání počítač v síťovém prostředí, na kterém je program zpracováván, před nebezpečnými operacemi nebo napadením vlastního operačního systému nepřátelským kódem.

nezávislý na architektuře – vytvořená aplikace běží na libovolném operačním systému nebo libovolné architektuře. Ke spuštění programu je potřeba pouze to, aby byl na dané platformě instalován správný virtuální stroj. Podle konkrétní platformy se může přizpůsobit vzhled a chování aplikace.

přenositelný – vedle zmíněné nezávislosti na architektuře je jazyk nezávislý i co se týká vlastností základních datových typů (je například explicitně určena vlastnost a velikost každého z primitivních datových typů). Přenositelností se však myslí pouze přenášení v rámci jedné platformy Javy (např. J2SE). Při přenášení mezi platformami Javy je třeba dát pozor na to, že platforma určená pro jednodušší zařízení nemusí podporovat všechny funkce dostupné na platformě pro složitější zařízení a kromě toho může definovat některé vlastní třídy doplňující nějakou speciální funkčnost nebo nahrazující třídy vyšší platformy, které jsou pro nižší platformu příliš komplikované.

výkonný – přestože se jedná o jazyk interpretovaný, není ztráta výkonu významná, neboť překladače mohou pracovat v režimu „just-in-time“ a do strojového kódu se překládá jen ten kód, který je opravdu zapotřebí.

víceúlohový – podporuje zpracování vícevláknových aplikací

dynamický – Java byla navržena pro nasazení ve vyvíjejícím se prostředí. Knihovna může být dynamicky za chodu rozšiřována o nové třídy a funkce, a to jak z externích zdrojů, tak vlastním programem.

elegantní – velice pěkně se v něm pracuje, je snadno čitelný (např. i pro publikaci algoritmů), přímo vyžaduje ošetření výjimek a typovou kontrolu.

PHRUB.CZ
Hlavně klid

Úvod do objektově orientovaného programování

Seznámení s pojmy objekt, dědičnost, třída, polymorfismus

Základní principy OOP

Java je objektově orientovaný programovací jazyk přenositelný na různé platformy. Pracuje tedy s objekty. Co to vlastně jsou objekty?

Jedná se o abstrakci z reality, každý objekt představuje spojení dat (údajů, proměnných, datových atributů) a činností s těmito daty (metod). Tato abstrakce je vždy účelová, o každém reálném objektu se sledují ty údaje, které jsou relevantní pro aplikaci.

Pokud chceme pracovat s objekty, je nutné vědět, jaké jsou obecné vlastnosti objektů.

Objekty a abstrakce

Abstrakce je základní objektovou vlastností. Skutečnost, kterou chceme do programu promítnout musíme vždy zjednodušit, pracovat jen s těmi daty, která jsou důležitá.

Příklad 1

Pokud budeme psát aplikaci pro počítání obvodů a obsahů dvourozměrných tvarů, budou zde podobné objekty, ale s jinými daty a metodami.
U jednotlivých tvarů stačí sledovat rozměry stran a popřípadě velikosti úhlů.

Metody, které budou k dispozici, budou metody pro výpočet obsahu a obvodu.

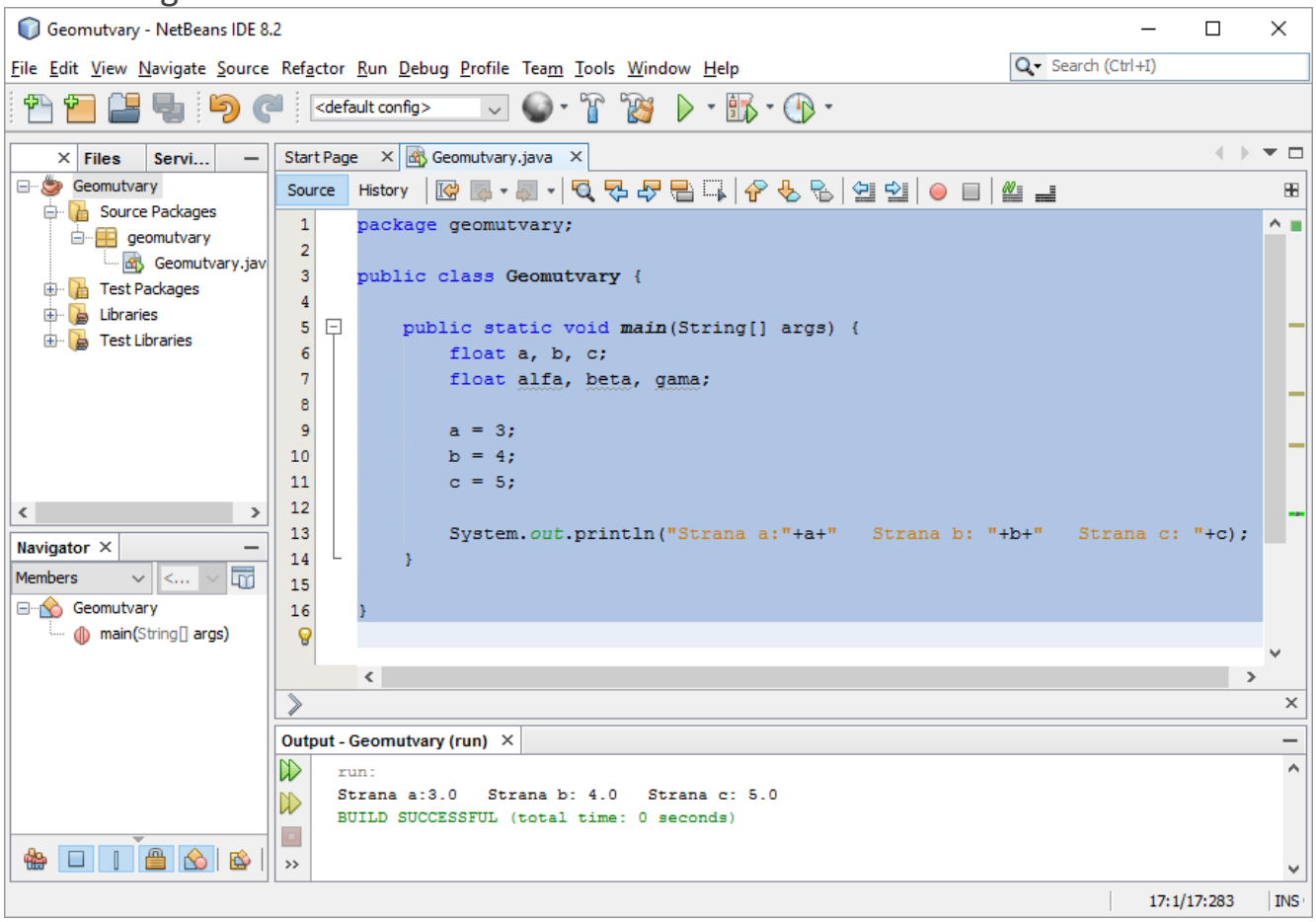
Úvodní program a základní struktura

```
package geomutvary;
 public class Geomutvary {
   public static void main(String[] args) {
      float a, b, c;
      float alfa, beta, gama;
      a = 3;
      b = 4;
      c = 5;
      System.out.println("Strana a:"+a+"   Strana b: "+b+"   Strana c: "+c);
   }
}
```

PHRUB.CZ
Hlavně klid

Seznámení s vývojovým prostředím

Program v Netbeans



Program 2 - trojúhelník

Podmíněný příkaz, matematické a logické operace

```
package geomutvary;

public class Geomutvary {

    public static void main(String[] args) {

        double a, b, c;

        double alfa, beta, gama;

        a = 3;

        b = 4;

        c = 5;

        System.out.println("Je zadána strana a:" + a + "    strana b: " + b + "    strana c: " + c);

        //trojúhelníková nerovnost

        if (((c + a) < b) | ((a + b) < c) | ((b + c) < a)) {

            System.out.println("Neni trojuhelnik");

        } else {

            if ((a == b) & (a == c) & (a == b)) {

}

}
```

```
 System.out.println("Rovnostranný trojuhelnik");


            } else {

                if (((a == b) & (a != c)) | ((a == c) & (a != b)) | ((b == c) & (a != b))) {


System.out.println("Rovnoramenný trojuhelnik");

                }

            }

        }

        double obvod = a + b + c;

        System.out.println("Obvod trojuhelnika je " + obvod);

        //Heronuv vzorec

        double obsah = Math.sqrt(obvod / 2 * (obvod / 2 - a) * (obvod / 2 - b) * (obvod / 2 - c));

        System.out.println("Obsah trojuhelnika je " + obsah);
```

PHRUB.CZ
Hlavně klid

Datové typy, operátory, řídící příkazy

Pole a řetězce - jejich zpracování

Vlastnosti řetězců, porovnávání řetězců, imutabilita

Metody třídy String

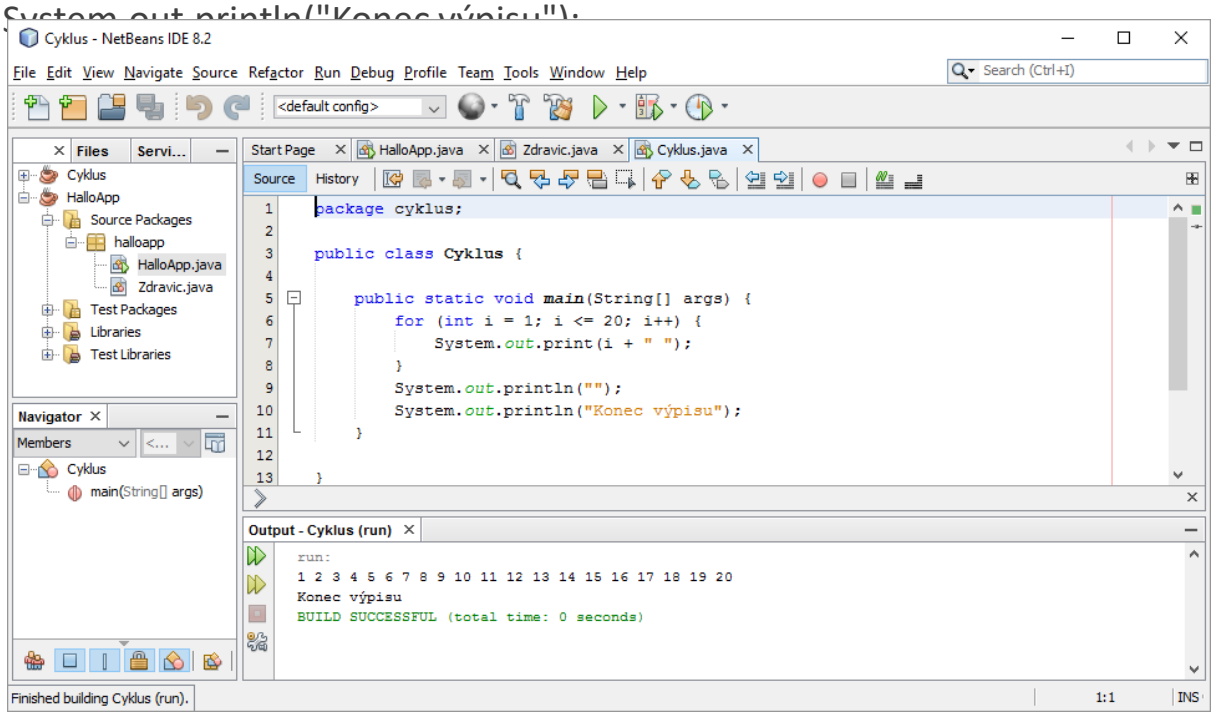
Standardní jazykové pole, omezení a jejich výhody, vytváření, cyklus for a iteratovatelné objekty

Příklad - řetězce

```
package papousek;
import java.util.Scanner;
public class Papousek {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in, "Windows-1250");
       System.out.println("Ahoj, jsem virtuální papoušek Lóra, rád opakuji!");
       System.out.println("Napiš něco: ");
       String vstup;
       vstup = sc.nextLine();
       String vystup;
       vystup = vstup + ", " + vstup + "!";
       System.out.println(vystup);
   }
}
```

Příklad – cyklus s pevným počtem opakování

```
package cyklus;

public class Cyklus {

    public static void main(String[] args) {

        for (int i = 1; i <= 20; i++) {

            System.out.print(i + " ");

        }

        System.out.println("");

        System.out.println("Konec výpisu");

    }

}
```



PHRUB.CZ
Hlavně klid

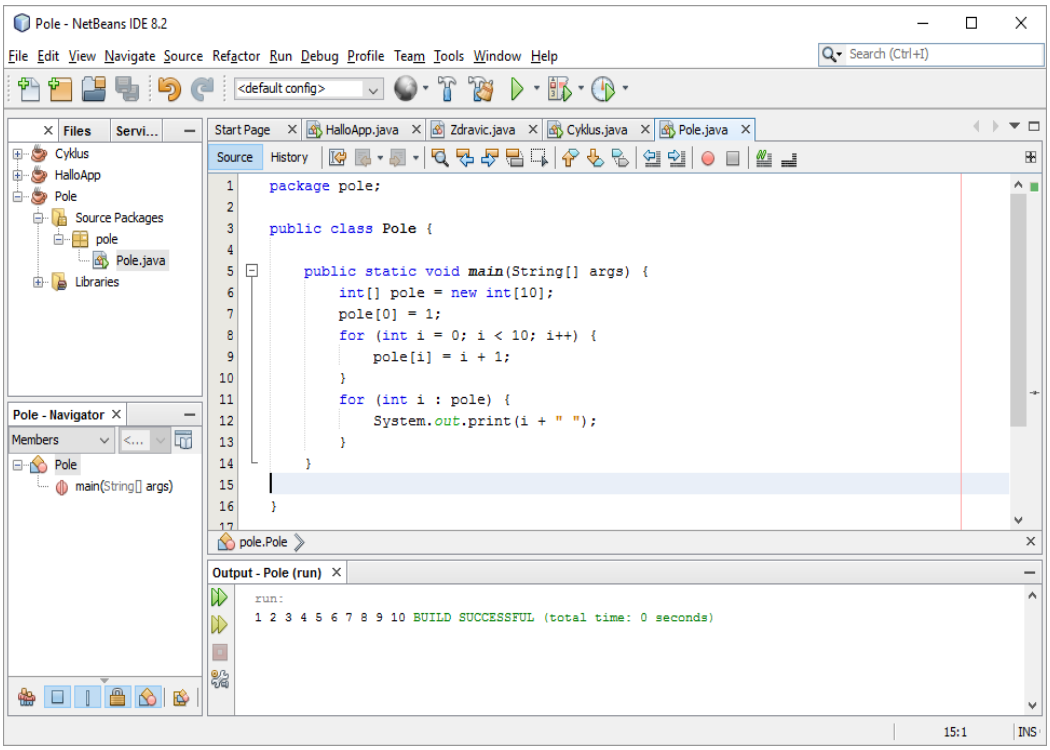
Příklad – cyklus a ukládání hodnoty proměnné, parsování

```
package mocninator;

import java.util.Scanner;

public class Mocninator {

    public static void main(String[] args)
{
        Scanner sc = new
Scanner(System.in, "Windows-1250");

System.out.println("Mocninátor");

System.out.println("==========");
        System.out.println("Zadejte
základ mocniny: ");
        int a =
Integer.parseInt(sc.nextLine());


}
```

```
        System.out.println("Zadejte
exponent: ");
        int n =
Integer.parseInt(sc.nextLine());


        int vysledek = a;
        for (int i = 0; i < (n - 1); i++) {
            vysledek = vysledek * a;
        }


        System.out.println("Výsledek: " +
vysledek);
        System.out.println("Děkuji za
použití mocninátoru");
    }
```

Pole - Arrays

```
package pole;

public class Pole {

   public static void main(String[] args) {

      int[] pole = new int[10];

      pole[0] = 1;

      for (int i = 0; i < 10; i++) {

         pole[i] = i + 1;

      }

      for (int i : pole) {

         System.out.print(i + " ");

      }

   }

}
```



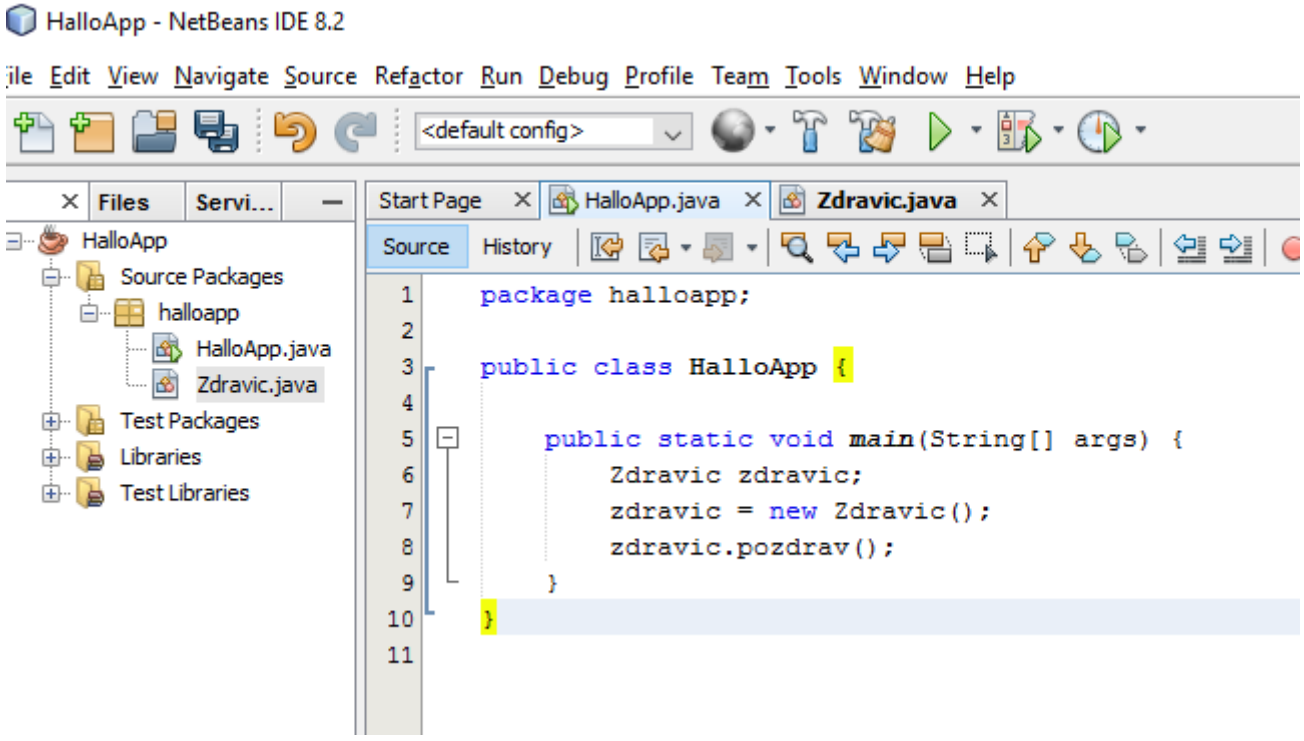
Návrh a tvorba tříd, metod, objektů

Vytvoření vlastní třídy

Proměnné objektu, metody objektu

Vytváření objektů pomocí new

Úrovně viditelnosti



Výklad - metoda a instance metody, třída

Třída Zdravic.java *//uložena v Zdravic.java*

```
package halloapp;
public class Zdravic { //třída
  public void pozdrav() { //metoda třídy bez návratové hodnoty
            System.out.println("Hello object world!");
      }
}
```

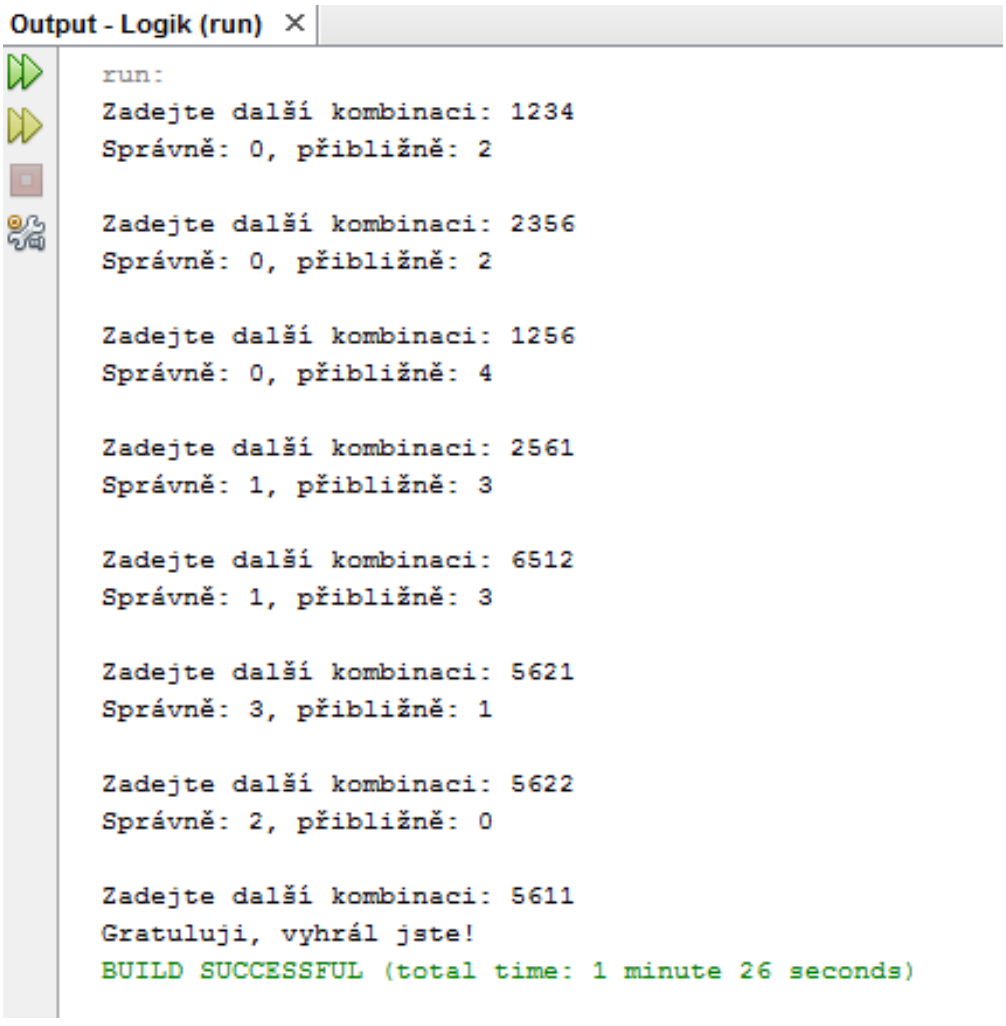
Hlavní program

```
package halloapp;
public class HalloApp {
   public static void main(String[] args) {
      Zdravic zdravic;
      zdravic = new Zdravic(); //instance třídy
      zdravic.pozdrav();
   }
}
```

Cvičení - LOGIK

```
package logik;

import java.util.Scanner;

public class Logik {

   public static void main(String[] args) {
      int pocetCisel = 4;
      int cislo[] = new int[pocetCisel];
      for (int i = 0; i < pocetCisel; i++) {
         cislo[i] = (int) (Math.random() * 10);
      }
      String volba;
      int presne = 0;
      int priblizne = 0;
      Scanner sc = new Scanner(System.in, "UTF-8");

      for (int a = 0; a < 10; a++) {

         System.out.printf("Zadejte další kombinaci: ");
         volba = sc.nextLine() + "      ";
         presne = priblizne = 0;
      } for (int i = 0; i < pocetCisel; i++) {
         if (cislo[i] == volba.charAt(i) - 48) {
            presne++;
         } else {
            for (int j = 0; j < pocetCisel; j++) {
               if (cislo[i] == volba.charAt(j) - 48) {
                  priblizne++;
                  j = 10;
               }        }          }
      if (presne == pocetCisel) {
         System.out.println("Gratuluji, vyhrál jste!");
         a = 10;
      } else {
         System.out.printf("Správně: %d, přibližně:
%d\n\n", presne, priblizne);
         }
      }
      if (presne != pocetCisel) {
         System.out.printf("Prohra! Hledaná kombinace byla:
");
         for (int i : cislo) {
            System.out.printf("%d", i);
         }     }    }    }
```

Cvičení - LOGIK



Zpracování výjimek v programu

Typy výjimek v Javě

Standardní výjimky, kontrolované a běhové výjimky

Ošetřování výjimek

Blok finally

```
private void vypisZpravu(String zprava)
{
System.out.println(zprava);
try { Thread.sleep(500);
}
 catch (InterruptedException ex)
{
System.err.println("Chyba, nepovedlo se uspat vlákno");
}
}
```

Závěrečná cvičení – cvičení 1

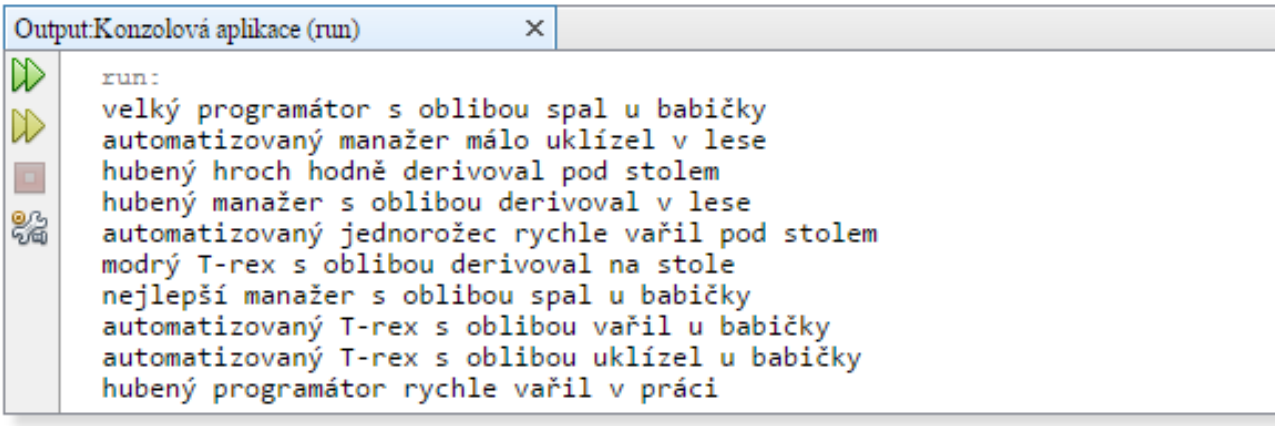
Naprogramujte aplikaci, která obsluhuje člověka. Člověk má jméno a únavu, která je zpočátku 0. Může uběhnout určitou vzdálenost a také spát určitou dobu. Běháním se jeho únava zvyšuje (1 jednotka únavy na 1 km), spaním se snižuje (10 jednotek únavy na 1 hodinu). Navrhněte třídu tak, aby se únava nikdy nemohla dostat z rozmezí 0-20 jednotek. Smozřejmě vám k tomu pomůže zapouzdření, únava určitě nebude veřejným atributem.

Program vyzkoušíte tak, že necháte člověk 3x uběhnout 10 km. Třetí uběhnutí by se nemělo povést. Když člověka necháte po druhém uběhnutí hodinu spát, zvládne i třetí běh.

Závěrečná cvičení – cvičení 2

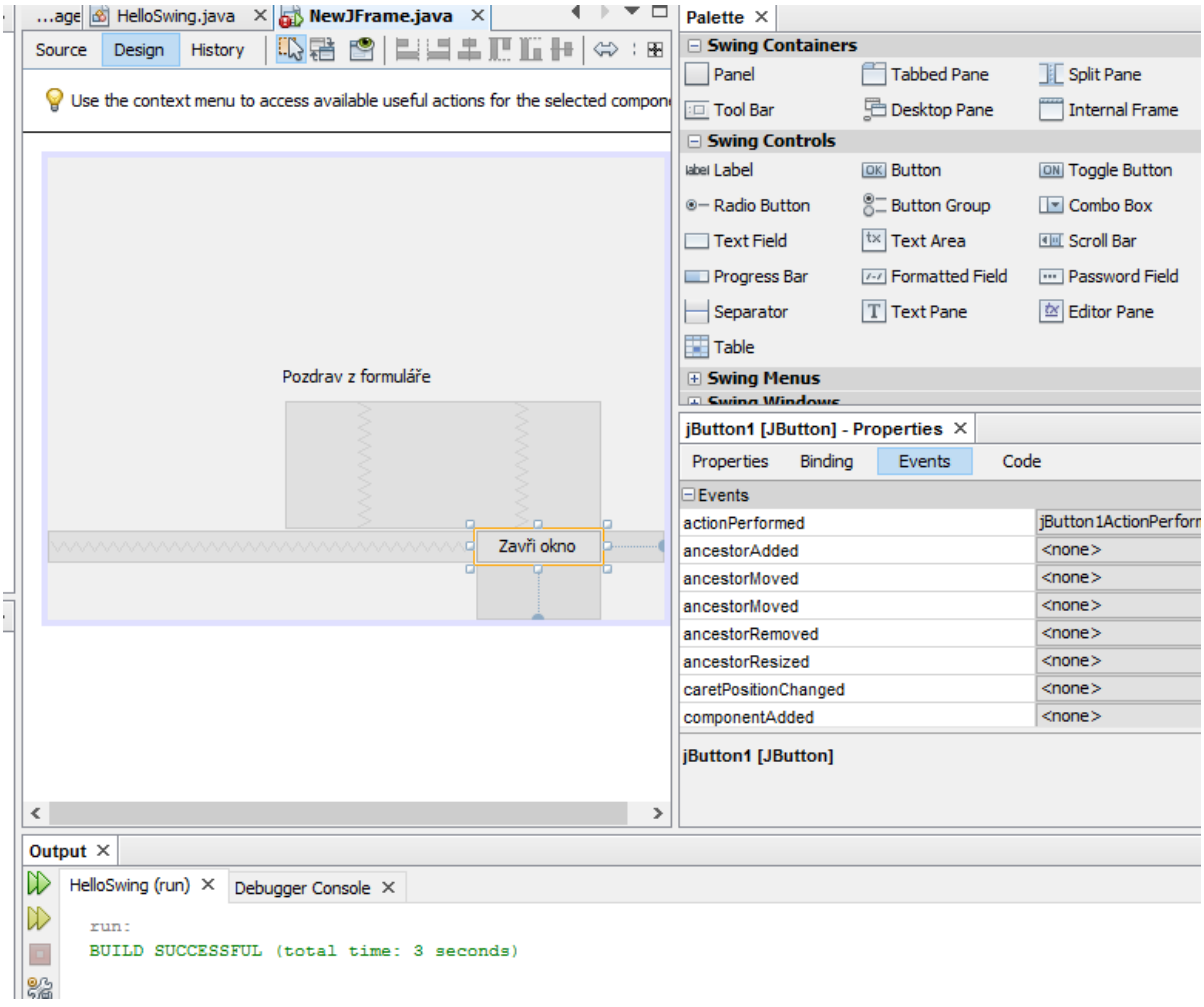
Naprogramujte za pomoci OOP generátor náhodných vět. Nechte ho vygenerovat 10 vět a tyto věty vypište do konzole. Z výstupu programu níže analyzujte jak věty vypadají a vymyslete, jak takovou větu vytvořit.

Ukázka obrazovky programu:



Praktická ukázka programu v Javě

- konzolová aplikace, základy tvorby grafických aplikací v prostředí Swing
- tvorba a oživení samostatných aplikací



Zdroje

http://www.itnetwork.cz/java

https://www.banan.cz/serialy/Java/