ĮVADAS Į OBJEKTIŠKAI ORIENTUOTĄ PROGRAMAVIMO KALBĄ JAVA

Marius Gžegoževskis

1 paskaita

JAVA KALBOS SAVYBĖS - PAPRASTUMAS

- Nėra rodyklių (pointer) aritmetikos, goto sakinio, tipų apibrėžimų ir pan.
- Nepalaiko metodų su kintamu parametrų sąrašu.
- Neturi globalinių kintamųjų.
- Neturi daugialypio paveldėjimo.
- Java pati atlaisvina atmintį (nėra destruktorių).

KITOS JAVA KALBOS SAVYBĖS

- Nepriklausoma nuo architektūros.
- Daugiaprocesė.
- Saugumas.
- Išskirtinių situacijų valdymas.

SĄVOKOS

Pagrindinės objektiškai orientuotų (OO) kalbų sąvokos:

- Klasė – objekto aprašas;
- Objektas – konkrečios klasės esybė;
- Metodas – objekto savybė.

PIRMOJI PROGRAMA

- Pavyzdys atspausdinantis tekstą Hello World.
- ----------
- public class MyFirstJavaProgram {
- /* This is my first java program.
  - This will print 'Hello World' as the output */
  - public static void main(String []args) {
    - System.out.println("Hello World"); // prints Hello World
- }
- }

KAIP PALEISTI PIRMĄJĄ PROGRAMĄ? (1 OF 2)

Prieš tai pateikto pavyzdžio skirto atspausdinti Hello World įvykdymui reikia atlikti šiuos žingsnius:

1. Atidaryti tekstinį redaktorių pavyzdžiui „Notepad“ ir įkelti prieš tai pateiktą kodą.
2. Failas yra išsaugomas su plėtiniu ***.java** kadangi kompiliuojant kodo interpretatorius turi matyti jog tai yra java programa. MyFirstJava – tai kompiliuojamos klasės pavadinimas: **MyFirstJavaProgram.java.**
3. Tada reikia atsidaryti komandinę eilutę ir joje pereiti į tą direktoriją, kurioje išsaugojote failą pvz: **cd** C:/Users/test/Documents. Komanda **cd** skirta pereiti į nurodytą direktoriją.

KAIP PALEISTI PIRMĄJĄ PROGRAMĄ? (2 OF 2)

4. Komandinėje eilutėje įvesti komanda **' javac MyFirstJavaProgram.java** (kompiliuojamos klasės vardas). Jeigu nėra išvedamas klaidos pranešimas, tada viskas gerai jūsų vykdomasis kodas be klaidų. Dažnai pasitaiko jog nebūna nurodytas kelias iki java JDK (Java development kit) ir negali sukompiliuoti jūsų kodo yra išvedamas pranešimas jog ši komanda neegzistuoja. Kelio nurodymui žiūrėti čia. Dažniausiai Java įrankis yra instaliuotas panšioje vietoje pvz: „C:\Program Files\Java\jdk1.8.0_20\bin“. Jeigu šio įrankio nėra įdiegto jūsų kompiuteryje tada reikėtų parsiųsti iš interneto: nuoroda čia. Yra ir patogūs įrankiai turintys daug įvairiausių naudingų opcijų palengvinančių programuotojui darbą tai : NetBeans, Eclipse, JCreator. Ir t.t.

5. Galiausiai įveskite**' java MyFirstJavaProgram‘** prieš tai sukompiliuotas kodas su komanda **javac MyFirstJavaProgram.java** bus įvykdomas bei pateikiamas rezultatas bus ‚Hello World‘

Kompiliavimo bei paleidimo pvz:

C : > javac MyFirstJavaProgram.java – skirtas kompiliuoti

C : > java MyFirstJavaProgram – skirtas įvykdyti sukompiliuotą kodą.

Hello World – **rezultatas.**

PAGRINDINIAI DUOMENŲ TIPAI (1)

- Sveiki skaičiai:
  - long (64 bitai)
  - int (32 bitai)
  - short (16 bitų)
  - byte (8 bitai)

PAGRINDINIAI DUOMENŲ TIPAI (2)

- Slankaus kablelio skaičiai
  - double (64 bitai)
  - float (32 bitai)

PAGRINDINIAI DUOMENŲ TIPAI (3)

- Kiti duomenų tipai:
  - char (vienas simbolis, 16 bitų)
  - boolean (true / false)

OPERATORIAI

- Aritmetiniai: +, -, *, /, %, ++, --, +=, -=, *=, /=, %=.
- Loginiai / bitiniai: ~ (NOT), & (AND), | (OR), ^ (XOR), >> (shift right), << (shift left), &&, ||, !.
- Palyginimo: ==, !=, >, <, >=, <=.

KINTAMŲJŲ IR OPERATORIŲ PVZ.

```
class BasicMath  {
  public static void main(String args[]) {
          int a = 1;
          a++;
          int b = a *  3;
          int c = b / 4;
          int d = b - a;
          int e = -d;
          System.out.println("a =  " +    a);  // a = 2
          System.out.println("b =  " +    b);  // b = 6
          System.out.println("c =  " +    c);  // c = 1 !!!
          System.out.println("d =  " +    d);  // d = 4
          System.out.println("e =  " +    e);  // e = -4
  }
}
```

MODULIO PVZ.

```
class Modulus {
  public static void main (String args []) {
        int x = 42;
        double y = 42.3;
        System.out.println("x mod 10 = " + x % 10);
        System.out.println("y mod 10 = " + y % 10);
  }
}
```

```
c:> java Modulus
x mod 10 = 2
y mod 10 = 2.3
```

DIDINIMO / MAŽINIMO VIENETU PVZ.

```
class IncDec {
  public static void main(String args[]) {
        int a = 1;
        int b = 2;
        int c = ++b;
        int d = a++;
        c++;
        System.out.println("a = " + a); // a = 2
        System.out.println("b = " + b); // b = 3
        System.out.println("c = " + c); // c = 4
        System.out.println("d = " + d); // d = 1
  }
}
```

IF-ELSE IR SWITCH SAKINIAI

```
if (sąlyga) {
  vykdyti_jei_tenkinama_sąlyga;
} else {
  vykdyti_jei_netenkinama_sąlyga;
}

switch (išraiška) {
  case reikšmė1:
          break;
  case reikšmė2:
          break;
  case reikšmėN:
          break;
  default:
}
```

IF-ELSE PAVYZDYS

```
class IfElseDemo {
  public static void main(String args[]) {
          int a = 5;
          if (a > 0) {
                    System.out.println(“teigiamas”);
          } else if (a == 0) {
                    System.out.println(“nulis”);
          } else {
                    System.out.println(“neigiamas”);
          }
  }
}
```

C:> java IfElseDemo

teigiamas

SWITCH PAVYZDYS

```
class SwitchSeason {
  public static void main(String args[]) {
        int month = 4;
        String season;
        switch (month) {
                case 12:
                case 1:
                case 2: season = "Winter"; break;
                case 3:
                case 4:
                case 5: season = "Spring"; break;
                //...
                default: season = "Bogus Month";
        }
        System.out.println("April is in the " +    season +     ".");
  }
}
```

CIKLAI

For ciklas:

```
for (pradinė_reikšmė; pabaigos_sąlyga; pokytis) {
  // sakiniai ciklo viduje
}
```

While ciklas:

```
while (sąlyga) {
  // kartoti, kol sąlyga == true
}
```

FOR PAVYZDYS

```
class ForDemo {
 public static void main(String args[]) {
       for (int i = 0; i < 10; i++)
       System.out.println("i = " + i);
 }
}
```

WHILE PAVYZDYS

```
class WhileDemo {
  public static void main(String args[]) {
        int n = 2;
        int a = 1;
        while (n <= 1024) {
              System.out.println("2^" + a + "=" + n);
              n *= 2;
              a++;
        }
  }
}
```

KONSTRUKTORIAI

- Skirti sukurti konkrečiam klasės objektui.
- Gali būti keli konstruktoriai su skirtingais parametrais.
- Konstruktoriaus gali ir nebūti. Tada naudojamas "default" konstruktorius.
- Konstruktorius sugrąžina sukurtą objektą.
- Konstruktoriaus vardas turi sutapti su klasės vardu.

KONSTRUKTORIAI IR JŲ PANAUDOJIMAS

```
class Point {
 int x;
 int y;
 // konstruktorius be parametrų
 public Point() {
      this.x = 0;
      this.y = 0;
 }
 // konstruktorius su parametrais
 public Point(int x, int y) {
      this.x = x;
      this.y = y;
 }
}
```

KONSTRUKTORIAI IR JŲ PANAUDOJIMAS

```
class PointDemo {
      public static void main(String args[]) {
            Point p1 = new Point();
            Point p2 = new Point(1, 3);
      }
}
```

METODAI

Sintaksė:

```
[metodo_modifikatoriai] [grąžinama_reikšmė] vardas(
        parametro1_tipas parametras1,
        parametro2_tipas parametras2, ...) {


        metodo_kūnas;
}
```

METODŲ MODIFIKATORIAI

- **public** – pasiekiamas visur, kur pasiekiama metodo klasė.
- **“nieko”** – pasiekiamas tik šio paketo klasėse.
- **private** – pasiekiamas tik šioje klasėje.
- **abstract** – metodas neturi kūno ir privalo būti perdengtas.
- **final** – metodas negali būti perdengtas.
- **static** – galima taikyti klasei (be objekto). Nepersidengia.

METODŲ PVZ.

```
public class MethodTest {
  public static void main(String args[]) {
        int i = getX(); // kompiliatoriaus klaida
  }

  public int getX() {
        return 10;
  }
}
/*
Klaida, nes neaišku, kuriam objektui kviesti metodą getX()
*/
```

KLAIDOS IŠTAISYMAS NR. 1

```
/* reikia sukurti objektą */

public class MethodTest {
  public static void main(String args[]) {
        MethodTest test = new MethodTest();
        int i = test.getX();
  }

  public int getX() {
        return 10;
  }
}
```

KLAIDOS IŠTAISYMAS NR. 2

```
/* reikia paskelbti metodą statiniu */

public class MethodTest {
  public static void main(String args[]) {
          // klasės vardas nebūtinas, jei tai yra tos
          // pačios klasės viduje (kaip šiuo atveju)
          int i = MethodTest.getX();
          // arba: int i = getX();
  }

  public static int getX() {
          return 10;
  }
}
```

METODŲ PERKROVIMAS (OVERLOADING)

Tai polimorfizmo atvejis, kai klasė turi kelis metodus tais pačiais vardais:

- Metodai privalo skirtis savo parametrų tipais ir / arba kiekiu.
- Metodai, besiskiriantys tik grąžinamos reikšmės tipu, negalimi.
- Metodo tipą nusako jo iškvietimo formatas.

METODŲ PERKROVIMAS

```
public class Point {
  int x;
  int y;
  public void setCoords(int x, int y) {
         this.x = x;
         this.y = y;
  }
  public void setCoords(float x, float y) {
         this.x = Math.round(x);
         this.y = Math.round(y);
  }
  public static void main(String args[]) {
         Point p1, p2;
         p1.setCoords(0, 5);
         p2.setCoords(3.0, -2.4);
  }
}
```

STATINIAI IR OBJEKTO KINTAMIEJI

- Jei prieš kintamąjį yra modifikatorius “static”, tai kintamasis yra bendras visiems tos klasės objektams. Kitu atveju kintamasis būna atskiras kiekvienam objektui.

KINTAMŲJŲ PVZ.

```
class Number {
  public int a = 0;
  public static int b = 0;

  public static void main(String args[]) {
        Number n1, n2;
        n1.a = 1;
        n1.b = 2; // arba (dažniau naudojama) Number.b = 2;
        n2.a = 3;
        n2.b = 4;
        System.out.println(n1.a + " " + n1.b + " " + n2.a +
                " " + n2.b);
  }
}
```

INTERFEISAI

- Interfeisas – tai griežtai apibrėžtas metodų rinkinys, kurie turi būti įgivendinti, tą interfeisą realizuojančioje klasėje.
- Vieną interfeisą gali realizuoti daug klasių.
- Viena klasė gali realizuoti daug interfeisų.

INTERFEISO PVZ. (1)

```
interface Sort {
  public Object sort(Object array);
}

public class QuickSort implements Sort {
  public Object sort(Object array) {
        // masyvo rūšiavimo algoritmas (pvz., QuickSort)
  }
}

public class BubbleSort implements Sort {
  public Object sort(Object array) {
        // masyvo rūšiavimo algoritmas (pvz., BubbleSort)
  }
}              //Tęsinys kitoje skaidrėje
```

INTERFEISO PVZ. (2)

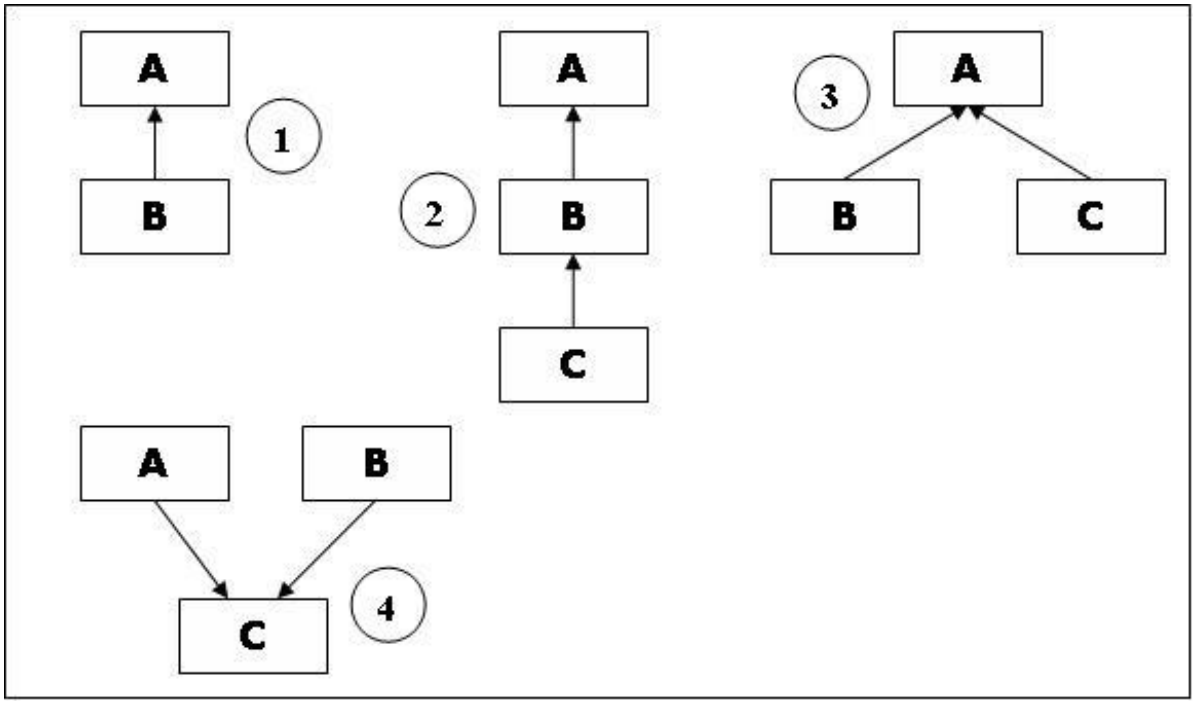
```
public class SortDemo {
  public static void main(String args[]) {
        SortDemo demo = new SortDemo();
        Sort s1 = new QuickSort();
        Sort s2 = new BubbleSort();
        Object sorted = demo.sortArray(s1);
  }

  public Object sortArray(Sort s) {
        Object array = new Object[10];
        // ... atsitiktinai sugeneruojam masyvo reikšmes
        return s.sort(array);
  }
}
```

PAVELDIMUMAS

- Siekiant išvengti kodo kartojimo, objektinėje paradigmoje įprasta naudoti paveldėjimą (angl. inheritance).
- Paveldėjimas leidžia sukurti naują klasę panaudojant ankstesnės klasės savybes.
- Tėvinės klasės savybės būdingos ir vaikinei klasei.
- Galima pridėti naujų savybių.
- Paveldėjimui nusakyti naudojamas žodis **extends.**
- Java nepalaiko daugybinio paveldėjimo, bet galima pasinaudoti java **Interfeisais** šiai problemai išspręsti. Kaip matote Klasė **A** paveldi klasę **B** ir Implementuoja Interfeisą klasės **C.**
- Nariai tėvinėje klasėje su modifikatoriais **public** ir **protected** gali paveldėti iš vaiko (angl. child) klasės . **Private** modifikatorius neleidžia prieiti nariams iš už klasės srities ribų.

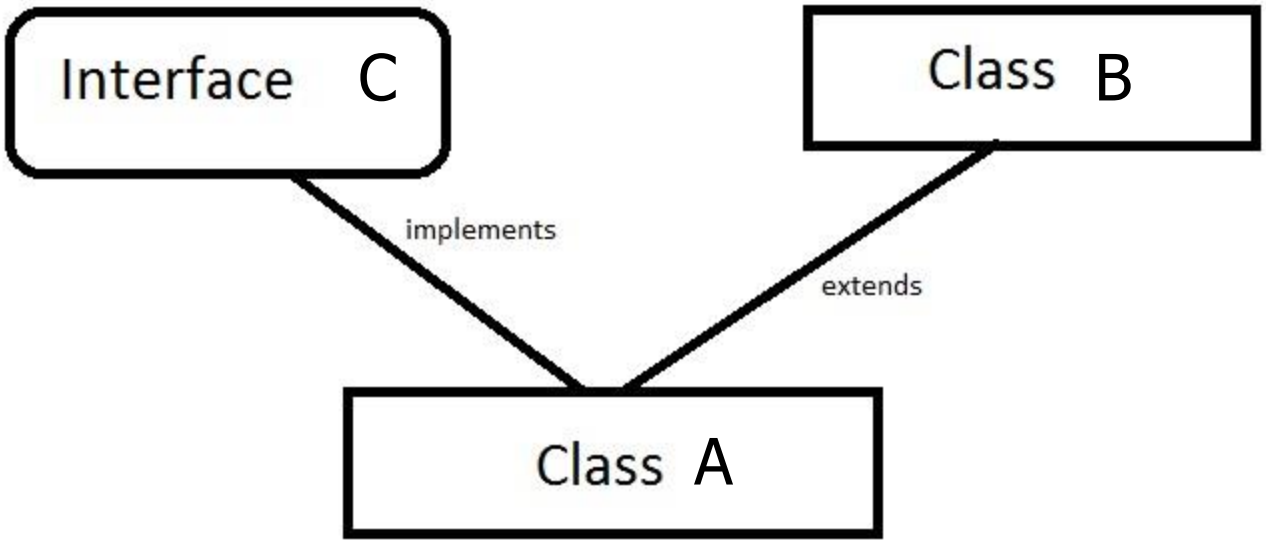
PAVELDĖJIMAS JAVA



1. Paveldėjimas vieno lygmens (angl. Single Inheritance)
2. Daugialygmenis paveldėjimas (angl. Multilevel Inheritance)
3. Hierarchinis paveldėjimas (angl. Hierarchical Inheritance)
4. Daugybinis paveldėjimas (angl. Multiple Inheritance)

DAUGYBINIS PAVELDĖJIMAS JAVA

- Java nepalaiko daugybinio paveldėjimo, bet galima pasinaudoti java **Interfeisais** šiai problemai išspręsti. Kaip matote Klasė **A** paveldi klasę **B** ir Implementuoja Interfeisą klasės **C.**



Multiple inheritance in java

VIENO LYGMENS PAVELDĖJIMO PAVYZDYS

- class A {int i;}
- class B extends A {
- int j;
- int didinti() {return i++;}
- }

DAUGYBINIO PAVELDĖJIMO PAVYZDYS

```
class A {
                    int a = 5 ;
                    public void showA()
                    {
                       System.out.println("Value of a is : "+a) ;
                    }
          }
Interface I1 {
                    int b = 6 ;
                    public void showI1()
               }
Interface I2 extends I1
 {
                    int c = 7 ; public void showI2() ;
}
class test extends A implements I1
{
                     public void showI1()
                    {
                    System.out.println("Value of b is : "+b) ;
                     }
public void showI2()
                    {
                    System.out.println("Value of c is : "+c) ;
                    }
 }
class enter
{
                    public static void main(String args[])
                    {
                    test t = new test() ;
                    t.showA() ; t.showI1() ; t.showI2() ; }
 }
```

PAKETAI

Dažnai vieno projekto klasės yra apjungiamos į paketą (arba kelis paketus).

Tokiu būdu galima išvengti besidubliojančių skirtingų programuotojų klasių, galima greičiau orientuotis, ką konkreti klasė atlika.

PAKETAI IR JŲ PANAUDOJIMAS

```
package lt.ktu.dsplab.figures;

public class Rectangle {
  //klasės kūnas
}

-------------------------------------
package lt.ktu.dsplab.rts;

import lt.ktu.dsplab.figures.*;
//arba import lt.ktu.dsplab.figures.Rectangle;

public class RTSystem {
  Rectangle r = new Rectangle();
  // …
}
```

PAGRINDINIAI TEIGINIAI IŠ JAVA CODING CONVENTIONS

- Eilutės atitraukimas (indentation) – 4 simboliai.
- Vengti ilgesnių eilučių nei 80 simbolių.
- if-else sakiniuose ir cikluose visada naudoti { ir }, net jei viduj tik vienas sakinys.
- Naudoti javaDoc kometarus;
- Aprašyti kintamuosius po vieną vienoje eilutėje.
- Vienoje eilutėje – vienas sakinys.
- Po metodo pavadinimo, tarpas prieš skliaustelį nededamas.
- Po tipo keitimo (cast) skliaustų palikti tarpą.
- Metodai ir kintamieji prasideda mažąja raide, jei žodis sudurtinis, kitų žodžių pradžios – iš didžiosios raidės.
- Konstantos rašomos vien didžiosiomis raidėmis.

JAVADOC KOMENTARAI

- Programose naudojant JavaDoc komentarus, vėliau galima automatiškai sugeneruoti programos dokumentaciją, naudojant komandą “javadoc”.
- JavaDoc komentarai turi būti prieš: klasę, klasės kintamuosius, metodus.

PAGRINDINIAI JAVADOC REZERVUOTI ŽODŽIAI

- @param – aibūdinamas metodo parametras.
- @return – apibūdinama grąžinama metodo reikšmė.

PROGRAMOS SU JAVADOC PVZ. (1)

```
package lt.ktu.fcm;
/**
 *  Point with two int type coordinates
 */
public class Point {

    /**
     *  The x coordinate
     */
    private int x;

    /**
     *  The y coordinate
     */
    private int y;

  // tęsinys kitoje skaidrėje
```

PROGRAMOS SU JAVADOC PVZ. (2)

```
    /**
     *  Moves the point to the given location
     *
     * @param  x  The new x coordinate value
     * @param  y  The new y coordinate value
     */
    public int move(int x, int y) {
        this.x = x;
        this.y = y;
  }

   /**
     *  Gets the Y coordinate
     *
     * @return    The y coordinate value
     */
    public int getY() {
        return this.y;
    }
}
```