lav03 laboratorijska vaja 16

(draft, feb.2025)

vsebina:

objekti

kreiranje objektov,

osnovno delo z objekti

Naloga 1

Dan je program s katerim kreiramo nekaj primerov objektov.Na označeni točki dodajte

izpise (vsebine) vseh objektov, ki so nastali v programu pred to točko in odgovorite

na spodnja vprašanja.

import java.util.Scanner;

/\*\*

\* dekalracija in kreiranje objekta

\*/

public class Zero\_00\_a{

public static void main(String[] args){

Integer oi; // deklaracija reference

oi = new Integer(121); // ustvarjanje objekta (z rezervacijo prostora v pomnilniku)

int[] oa; // deklaracija oz. najava

oa = new int[10]; // kreiranje objekta

// ---- vstavi izpis objektov

System.out.println("zaključite z ENTER");

Scanner scan; //deklaracija

scan = new Scanner("System.in"); //kreiranje objekta

scan.next();

}

}

Dopolnite:

V gornjem primeru kreiramo 3 objekte. Ti objekti so poimenovani \_\_\_\_\_\_, \_\_\_\_\_\_\_\_\_, \_\_\_\_\_\_\_\_\_\_.

Objekt dejansko nastane s tem, ko se pojavi v pomnilniku po tem, ko izvedemo

kreiranje z uporabo operatorja \_\_\_\_\_\_\_.

Naloga 2

Dan je program, ki izkorišča definicijo razreda Integer kot generator objektov.

Definicja razreda služi kot predloga za objekte. Oglejte si ga in odgovorite

na zahtevano spodaj.

import java.util.Scanner;

import java.util.Arrays;

/\*\*

\* več objektov istega tipa

\* na osnovi razreda je moč ustvariti poljubno števil objektov tega tipa

\*/

public class Zero\_00\_b{

public static void main(String[] args){

Integer oi1; // deklaracija(najava) reference

oi1 = new Integer(121); // ustvarjanje objekta (z rezervacijo prostora v pomnilniku)

Integer oi2; // deklaracija reference

oi2 = new Integer(321);

Integer[] toi; // deklaracija oz. najava

toi = new Integer[10]; // kreiranje objekta /tabele)

for (int i=0;i<toi.length;i++)

toi[i] = new Integer(i);

System.out.println(Arrays.toString(toi));

System.out.println("zaključite z ENTER");

Scanner scan; //deklaracija

scan = new Scanner("System.in"); //kreiranje objekta

scan.next();

}

}

V izvedbi zgornjega programa (driver oz. zagonske metode) smo ustvari natanko \_\_\_\_\_\_ objektov.

Ustvarjeni objekti so vrst:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.

Pri tem je bilo največ ustvarjenih objektov vrst \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ in sicer smo jih nardili \_\_\_\_\_\_ .

Naloga 3

import java.util.Arrays;

/\*\*

\* Glavna metoda (driver method) razreda v štirih ločenih zaporedih fazah (1)usvari 150 objektov

\* vrste (tipa) ZeroTocka. Vsakega naj inicializira s pseudo-naključno vrednostjo ravninske

\* koordinate iz obsega [0-200.0,0-200.0]. Nato naj izmed ustvarjenih poišče(2) tisto točko, ki

\* je najbolj oddaljena od točke(0,0), izpiše(3) koordinate te točke ter njeno oddaljenost od

\* točke (0,0). Nato najde točko vrste ZeroTocka, ki je najbližje povprečni oddaljenosti vseh takih

\* točk od točke (0,0), izpiše oddaljenost povpečja, oddaljenost najdene točke in njeno koordinato.

\*

\* No ja, ustvarjalcu opisane zamisli se je nekje vmes ustavilo. Pomagajte mu zamisel iz/s/peljati

\* do konca.

\*

\* pri realizaciji bodite 'prijazno' in se držite pravila : ena datoteka, en razred.

\*/

public class Zero\_00\_c{

public static void main(String[] args){

// (1)

ZeroTocka zt = new ZeroTocka(); zt.x=Math.random()\*100.00001; zt.y=Math.random()\*100.00001;

// (2)

// (3)

// (4)

}

}

class ZeroTocka{

double x;

double y;

}

Naloga 4

Za osežitev spomina; dan je program, ki uporabi razred z razrednim(statičnim) kontekstom.

Oglejti si načine naslavljanja lastnosti in njihove značilnosti, nato odgovorite na vprašanja

spodaj

/\*\*

\* razred;

\* celoten kontekst je razredne vrste

\* ( vsi člani so razredni člani /static/ )

\*/

class Zero\_01\_ra{

// class variables - replace the example below with your own

static int x;

static int y;

}

public class Zero\_01 {

public static void main(String[] args){

//--------- a

Zero\_01\_ra.x = 3;

Zero\_01\_ra.y = 4;

System.out.println("----------------------------- vsebina: class members --");

System.out.println(Zero\_01\_ra.x+" "+Zero\_01\_ra.y);

Zero\_01\_ra ra1 = new Zero\_01\_ra(); // naredimo 2 objekta vrste 'ra'

Zero\_01\_ra ra2 = new Zero\_01\_ra();

//malo preglejmo

System.out.println("----------------------------- vsebina iz obeh objektov --");

System.out.println(ra1.x+" "+ra1.y);

System.out.println(ra2.x+" "+ra2.y);

// še malo manipulacije

Zero\_01\_ra.x = 131;

ra2.y = 898;

// in preglejmo:

System.out.println("----------------------------- vsi 'trije' --");

System.out.println(Zero\_01\_ra.x+" "+Zero\_01\_ra.y);

System.out.println(ra1.x+" "+ra1.y);

System.out.println(ra2.x+" "+ra2.y);

}

}

Potrdite ali zavrzite izjave (nanašajo se na definicijo razred Zero\_01\_ra)

a) ne glede na to koliko objektov naredimo, imamo na razpolago zgolj en nabor podatkov

b) vsi objekti iste vrste (v primeru tipa Zero\_01\_ra) si očitno 'delijo' iste podatke

c) z vsebino razreda lahko manipuliramo (speminjamo, nastavljamo,..) ne da bi bilo potrebno

narediti kakršen koli objekt

d) tudi razredno metodo lahko kličemo, ne da bi bilo predhodno potrebno ustvariti objekt

Naloga 5

Malo začinimo nalogo 3 te vaje in jo vizualizirajmo. Uporabili bomo tehnologijo(ogrodje) FX,

točke pa vizualizirali kot krožnico z radijem 3. Da ne bo pretežko, je na voljo del programa

z izrisanim koordinatnim sistemom in eno točko (krožnico). Predlagam, da uporabite točke, kot

v naloga 3, FX pa uporabite zgolj kot vizualizacijo.

Naprej preglejte naslednje

import javafx.application.Application;

import javafx.scene.Group;

import javafx.scene.shape.Circle;

import javafx.scene.shape.Line;

import javafx.scene.Scene;

import javafx.stage.Stage;

import javafx.scene.paint.Color;

public class Two\_00 extends Application {

class TTocka {

double x=50,y=50;

}

public void start(Stage stage){

double centerX, centerY; centerX=centerY=250;

Group scg = new Group();

Line h = new Line(250,0,250,500); h.getStrokeDashArray().addAll(25d, 20d, 5d, 20d);

Line v = new Line(0,250,500,250); v.getStrokeDashArray().addAll(25d, 20d, 5d, 20d);

Group g = new Group(); // osnoven kontejner bo grupa - ne izkoriščamo nobenega

// prednastavljenega razporejnja komponent

// primer bloka, ki doda točko za izris v grupo g ! to morajo biti tudi ostale točke

TTocka t = new TTocka(); t.x=75; t.y=35;

{

Circle c = new Circle(centerX + t.x, centerY - t.y,3);

c.setFill(null); c.setStroke(Color.RED);

g.getChildren().add(c);

}

scg.getChildren().addAll(h,v,g); // v scg damo osi sistema h in v, ter grupo točk g

// JavaFX must have a Scene (window content) inside a Stage (window)

Scene scene = new Scene(scg, 500,500);

stage.setTitle("JavaFX Example");

stage.setScene(scene);

// Show the Stage (window)

stage.show();

}

}

- dodajte 150 točk (tabela 150 točk) - po dodajanju skušajte vizualizirati

- po kreiranju točk določi najbolj oddaljeno, določite črto med to točko in 0,0, jo dodajte

v grupo g, - vizualizirajte

- po določitvi črte, poiščite točko najbližjo poprečju razdalje, obarvajte (fill) jo zeleno

\* seveda je celoten zahtevan izpis iz naloge 3 v konzoli (terminalu) še vedno zaželen, ni

pa obvezen, ker imamo sedaj drug način vizualizacije

Naloga 6

V vsakem izmed kvadrantov bi želeli zgenerirati po 100 točk, recimo, da je celotna ravnina

v obsegu [-200,200]. S črto(ami) bi želeli medseboj povezati tiste točke, ki so v vsakem

posameznem kvadrantu najbolj oddaljene od koordinatnega izhodišča.

p.s.

če ste tole izvedli, ste naredili več kot 600 objektov pri enkratni izvedbi nalog.