|  |
| --- |
| **Utrjevanje:**   * ponavljanja. * mehak uvod v JavaFX   Opombi:  *izvedba je zasnovana s poudarkom na grafični vizualizaciji rezultata ponavljanj. Uvodni del uporabe tehnologije JavaFx je obrazložen v* [*https://openjfx.io/openjfx-docs/#introduction*](https://openjfx.io/openjfx-docs/#introduction)*, dost. nov.2022.*  *za razliko od predhodne rabe FX ne bomo uporabili preprostega risanja na platno, temveč bodo uporabljeni elementi pravi objekti.* |

|  |
| --- |
| Izhodišče in uvodni del vaje  Vse realizacije so izhodiščno osnovane in izhajajo iz naslednje predloge razreda, kot ga je moč najti na Example, <https://openjfx.io/javadoc/17/javafx.graphics/javafx/application/Application.html>, dost. nov 2022:  import javafx.application.Application;  import javafx.scene.Group;  import javafx.scene.Scene;  import javafx.scene.shape.Circle;  import javafx.stage.Stage;  public class MyApp extends **Application** {  public void start(**Stage** stage) {  **Circle** circ = new Circle(40, 40, 30);  **Group** root = new Group(circ);  **Scene** scene = new Scene(root, 400, 300);  stage.setTitle("My JavaFX Application");  stage.setScene(scene);  stage.show();  }  }  JavaFX je tehnologija, ki omogoča vizualizacijo aplikacije na grafičen način (izgradnjo funkcionalnega uporabniškega vmesnika). Vse kar vidimo v prikazu, si lahko zamislimo kot dogajanje na enem velikem **odru** (Stage), na katerem imamo v vsakem trenutka prikazano eno izmed **scen** (Scene). Sceno sestavljajo posamezni elementi ali skupine elementov (za skupine je značilna sočasna aktivnost vseh elementov v skupini). V levem primeru je na sceni zgolj ena skupina (Group) z enim samim elementom (Circle). |

**Naloga 1**

Zaženi izhodiščni primer in si oglejte rezultat izvajanja. Ugotovite, kako lahko spremenite velikost scene (naredite 'kvadratno' velikost), velikost in pozicijo 'krogca'(naj bo usrediščen na sceni in naj bo največji možen, pa da je prikazan še v celoti). Posvetite pozornost barvi edinega elementa scene.

***! – iz izvorne kode vidimo tudi to, da je potrebno definicije vseh elementov, ki sestavljajo našo FX aplikacijo uvoziti v našo kodo (import).***

**Naloga 2**

Program iz predhodne naloge dopolnite z naslednjo kodo tako, da dodano kodo vrinete v vrstice za programskim stavkom stage.show(); .

Circle circ2 = new Circle();

circ2.setCenterX(62);

circ2.setCenterY(162);

circ2.setRadius(25);

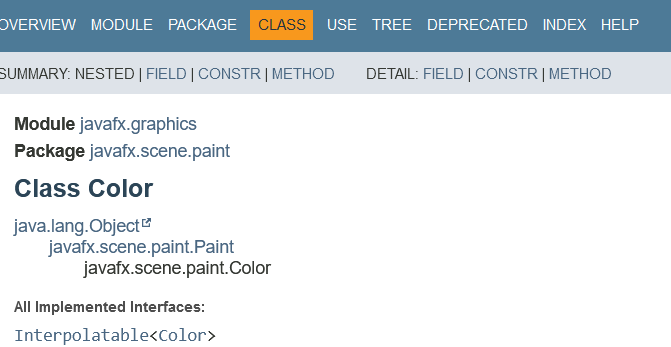
circ2.setStroke(Color.RED);

circ2.setFill(Color.LIGHTGRAY);

root.getChildren().add(circ2);

//root.setTranslateX(166);

Dan del kode kaže alternativni način kreiranja krogca, kjer vse lastnosti (izhodišče, radi, barve) nastavljate posamično. Predzadnja vrstica kaže običajen način za dodajanje elementa v skupino 'root' . Izvedite:

1. Dopolnjen program prevedite (in zaženite?). Prevajanje verjetno ne bo možno, ker platforma ne pozna definicije razreda Color. K obstoječim vključitvam boste morali dodati še vključitev (include) za ta razred. Če vam ga razvojno okolje samo ne najde, uporabite v iskalniku povpraševanje/query z vsebnino 'JavaFX + 17 + Color'. Uradna FXjavanska dokumentacija vam poda vrstico za vključitev, ki se nahaja v zadnjem delu drevesa poti pod Class Color. Iskani razred Color se v strukturi nahaja na poti javafx.scene.paint.Color (s črno barvo) in to predstavlja tudi pot za vključitev:

import javafx.scene.paint.Color;

RED in LIGHTGRAY sta konstanti opredeljeni v tem istem razredu, ogledate si lahko tudi ostale.

1. Preverite, da se je tudi drugi 'krogec' dodaj v isto skupino elementov kjer je prvi. Če je novi dodan k skupini, se bo premik skupine izvršil tako, da so bodo premaknili vsi elementi, ki so člani skupine. V dani spremenjeni kodi odstranite komentar pri zadnji vrstici, zaženite in izvedbo primerjajte s predhodno izvedbo (brez premika).

*! – stage.show(); je metoda, ki vse kar imate na odru tudi prikaže. Opazili ste, da smo nov krogec naredili po tem prikazu, premik pa tudi. Kljub temu so se vse spremembe uveljavile na prikazanem UI ! Načeloma to pomeni, da JavaFX 'čuti' vse spremembe, ki so narejene na UI in jih tudi uveljavi. To velja za celotno življensko dobo aplikacije.*

*!- java ne vsebuje (več) izvajanega okolja za FX, zato ga je potrebno pri zagonu vključiti. Spodnji zgled je dan za primer, kjer je FX naložen v okviru distribucije BlueJ (naložen v c:\program files\BlueJ524). Sicer morate poti ustrezno prilagoditi:*

1. *V razred dodate zagonsko metodo z vsebino kot:*

*public static void main(String[] args) {*

*launch(); // Run this Application.*

*}*

1. *Zaženete MyApp iz okolja (lupine) OS kot:*

*java --module-path C:\Progra~1\BlueJ524\lib\javafx\lib --add-modules javafx.controls MyApp*

|  |
| --- |
| Izvedbeni del  Naloge skušajte rešiti s sredstvi, ki so bila obdelana v okviru predmeta Računalništvo(3), odstopanja dopustite zgolj v delu, ki je vezan na vizualizacijo in se nanaša na tehnologijo FX.  Pri pisanju kode pazite, da boste vse, kar boste izvajali, izvedli znotraj metode start/1, kot je bilo to storjeno v nalogi 2. |

**Naloga 3**

Program Nal3307 vizualizira število 1487503 v besedilnem/tekstovnem načinu tako, da je posamezna števka izpisana v lastni vrstici, z odmikom velikost števke od levega roba zaslona

> java Nal3307 1487503

1\*1

4\*\*\*\*4

8\*\*\*\*\*\*\*\*8

7\*\*\*\*\*\*\*7

5\*\*\*\*\*5

00

3\*\*\*3

Program bi oblikovno želeli predelati tako, da bi posamezne števke vizualizirali kot krožnico z radijem 25pik na odmiku (velikost\_števke\*50) pik od levega roba zaslona. Programsko realizirajte »vrstice« ustrezne višine, da se krožnice v nobenem primeru ne bodo prekrivale. Število, ki ga predstavljate, lahko (z)generirate kot 2-8 mestno celo število.

**Naloga 4**

Variacija vizualizacije števila iz naloge 3 naj vse števke števila vizualizira v eni sami 'vrstici', drugo ob drugi. Števke naj bodo predstavljene s krožnicami, katerih velikost naj se določi v pikah (pikslih) kot 15+(vrednost\_števke\*15). Krožnice se lahko pri izrisu kvečjemu dotikajo. Če vam uspe, naj bo vsaka različna vrednost predstavljena z drugo barvo.

**Naloga 5**

Popravite realizacijo naloge 3. Namesto objekta vrste Circle za vizualizacij števke uporabite objekt vrste Text, ki naj vsebuje realno vrednost števke. Rabo 'Text'a si oglejte v dokumentaciji.

**Naloga 6**

Program V02\_06 predstavlja simulacijo streljanja serije desetih strelov v tarčo. Odstopanje strela od centra tarče znaša največ 60% oziroma +-6 v **x** in v **y** smeri (6 pomeni razdaljo od centra tarče oz. 10, kar pomeni, da je strelec pač zadel 4). Programsko izvedite simulacijo 10-tih strelov in določite rezultat glede na doseženo vrednost strela. Točnost strela se določa na ena decimalko natančno.

Primer izpisa simulacije streljanja:

serija 1: Vrednost strela: 10.0 akumulacija: 10.0

serija 2: Vrednost strela: 7.3 akumulacija: 17.3

serija 3: Vrednost strela: 4.6 akumulacija: 21.9

serija 4: Vrednost strela: 8.2 akumulacija: 30.1

serija 5: Vrednost strela: 5.9 akumulacija: 36.0

serija 6: Vrednost strela: 5.2 akumulacija: 41.2

serija 7: Vrednost strela: 5.4 akumulacija: 46.6

serija 8: Vrednost strela: 8.9 akumulacija: 55.5

serija 9: Vrednost strela: 6.9 akumulacija: 62.4

serija 10: Vrednost strela: 6.2 akumulacija: 68.6

Rezultat: 68.6

Program naj dodatno izvede še :

* izpiše vrednost najboljšega strela in kateri po vrsti v seriji je bil dosežen,
* izpiše vrednost najslabšega strela ter kateri po vrsti v seriji je bil sprožen, v kateri je bil dosežen;
* vizualizira vse strele serije, najboljši in najslabši strel naj bosta pri tem razlikovalno obarvana.

**Naloga 7**

Obstoječi program V07\_N07 zapolni zaslonsko sliko s poljubnim znakom. Sredi zaslonske slike pusti prazen prostor, ki predstavlja romb višine n. Program se kliče kot : V07\_N07 ***znak* *visina***, oziroma za izrisan primer:

* java N02\_06 c 5 V obstoječi programski dokumentaciji pa najdemo še:

ccccccccccccccccccccc *pri generiranju leve slike je bilo uporabljeno:*

cccccccccc cccccccccc *visina = 5 + 2 = 7*

ccccccccc ccccccccc *širina = (5+2) \* 3 = 21*

cccccccc cccccccc

ccccccccc ccccccccc *romb je usredinjen v polju velikosti 21x7*

cccccccccc cccccccccc

ccccccccccccccccccccc

Realizacijo bi želeli posodobiti z uporabo grafičnega načina, pri čemer namesto znaka v realizaciji uporabimo krožnico. Namesto izbire črke(argument) izvajamo izbiro barve. Recimo, da argumente znak(barva)/visina ali zajamete ali generirate.

**Naloga 8**

Zaslonski izris pravokotnika in v njem ekvidistančno porazdeljenih manjših pravokotnikov prikazuje:

+------------------------------------------------------------------------------+

| +----------------------------------------------------------------------+ |

| | +--------------------------------------------------------------+ | |

| | | +------------------------------------------------------+ | | |

| | | | +----------------------------------------------+ | | | |

| | | | | +--------------------------------------+ | | | | |

| | | | | | +------------------------------+ | | | | | |

| | | | | | | +----------------------+ | | | | | | |

| | | | | | | +----------------------+ | | | | | | |

| | | | | | +------------------------------+ | | | | | |

| | | | | +--------------------------------------+ | | | | |

| | | | +----------------------------------------------+ | | | |

| | | +------------------------------------------------------+ | | |

| | +--------------------------------------------------------------+ | |

| +----------------------------------------------------------------------+ |

+------------------------------------------------------------------------------+

Spisati bi želeli program, ki bi podobno izvedel v grafičnem načinu za vsak podan pravokotnik z razsežnostmi med 20x5 in dimenzijskimi omejitvami razpoložljive scene v FX. Upoštevajte, da je vsak gnezdeni pravokotnik dolžinsko in širinsko za 4 pike manjši od tistega, v katerem je vgnezden.