|  |
| --- |
| ***Teme:***  *Seznami, javanske kolekcije, LinkedList, Record*  *Realizacija Izbranih postopkov* |

|  |
| --- |
| Za razliko od ArrayList, ki je v osnovi razširljiva tabela z metodami iz List vmesnika, predstavlja LinkedList pravi seznam (iz dokumentacije: implementira List, Queue, Dequeue, …). Strukturno ustreza objestransko povezanemu seznama z močno več metodami, kot smo jih v naši realizaciji (krožni, obejestransko povezan) za primer mi implementirali. V naslednjih primerih bomo naslavljali elemente seznama zaporedno na obeh koncih seznama in za ta primer bo LinkedList idealen. |

**Naloga 1**

Izvedimo par testov/poizvedovanj:

1. Ugotovite, ali LinkedList implementira Serializable. Če ga implementira, je mogoče celotno strukturo predstaviti kot niz, torej tudi izpisati s preprostim print
2. Izvedite test:

LinkedList l1= **new LinkedList<Integer>()**;

**var l2** = new LinkedList<Integer>();

System.out.println(**l1.getClass().getName()**);

System.out.println(l2.getClass().getName());

l1.**add(3)**; System.out.println(**l1**);

l1.**add(new Integer(6))**; System.out.println(l1);

System.out.println( l1.**remove()** ); System.out.println(l1);

In ugotovite:

* katerega izmed paketov je potrebno vključiti, da lahko uporabite zgornjo kodo;
* ali sta l1 in l2 strukturi enakega tipa;
* kako lahko dodajate na začetek/konec oziroma jemljete element z začetka/konca seznama (spišite demonstracijo!)

**Naloga 2**

Recimo, da je element seznama (LinkedList) obarvana ravninska točka. Ker vrednosti v enkrat definirani točki ne bomo spreminjali, jo definiramo kot record(Java 17+). Test rabe recorda:

public record RTocka(int x, int y, java.awt.Color barva) {}

RTocka r1= new RTocka(3,4,java.awt.Color.red); // konstruktor se generira sam

System.out.println(r1.x); // lastnosti so finalne in javne

System.out.println(r1); // je Serializable !

Seznam naredimo kot : var ll = new LinkedList<RTocka>();

V seznam vstavimo 7 točk, vse naj imajo enako y koordinato, x koordinate naj bodo zaporedne od 5 naprej, barva vseh naj bo rumena. Spišite sekvenci a) in b), ki bosta:

1. izvede krožen premik elementov za en cel 'krog/cikel' tako da: vzame 1. element iz seznama, ga vstavi na konec istega seznama in izpiše seznam. Nato postopek ponovi, dokler ne prestavimo vseh elementov.
2. Izvede krožni premik elementov za cel cikel; vzame 1. iz seznama, na konec doda element, ki ima enak y, kot pred dodajanjem zadnji element, x je za 1 večji od x-a prej zadnjega elementa, barva novega pa je rdeča; no barva njegovega predhodnika je rumena. Izpišite seznam, ponavljajte čez celoten cikel.

Če ste b) izvedli prav, ste dejansko zaporedje premaknili po pozicijah desno !

**Naloga 3**

Uporabite JavaFX in seznam iz 2b) izrišite na sceno odra. Točke lahko vizualizirate z majhno krožnico (javafx.scene.shape.Circle) , barvo v record nadomestite z javafx.scene.paint.Color, pri čemer konstrukcija barve lahko ostane enaka.

Realizirajte gibanje kače (event driven) : na sceno dodamo 3 gumbe; CW-premakne zaporedje za 1 v novi smeri urinega kazalca, CCW premakne zaporedje za 1 v novi smeri nasprotni urinega kazalca, NAP – premakne zaporedje v obstoječi smeri za 1.

No, pa spišite zahtevano FX aplikacijo.

**Naloga 4**

Ponovite 3, element povezanega seznama (LinkedList) naj bo namesto zapisa RTocka element seznama objekt vrste Circle.

Rok za oddajo je 7 dni od oddaje/prejema naloge.