|  |
| --- |
| **Teme:** **Enorazsežne tabele - utrjevanje**   * Realizacija enostavnih struktur * Realizacija enostavnih postopkov nad sekvenco istovrstnih podatkov |

|  |  |
| --- | --- |
| PO-djetje  Izdeluje in distribuira lesene palice, s katerih je moč sestaviti trikotnike z obsegom N v paketih po 3 palice. Proces poteka tako, da palico dolžine N dvakrat sočasno razrežemo. Pri tem dobimo 3 kose, ki lahko sestavijo trikotnik (dober izdelek) ali pa ne (škart oz. izmet). V dosedanjih procesih je bilo ugotovljeno, da za določeno dolžino N izhodiščne palice ni rentabilno izvesti razreza za manj kot 1000 palic. Rezultat procesa je A paketov dobrih izdelkov in B paketov slabih.V osnovi nas zanima : kolikšen je procent izmeta v enem takem procesu ? kolikšen je povprečen % izmeta v K ponovitvah takega procesa (recimo, da je K=100)? seznam in specifikacija vseh dobrih izdelkov. seznam in specifikacija vseh slabih izdelkov    Del procesa je že bil avtomatiziran in je specificiran kot.  a) Razredni diagram obstoječe implementacije:  [RezalnikTrikotnikov|-del1 : int［1..\*］; -del2 : int［1..\*］; -del3 : int［1..\*］;-dolzinaPalice : int; -velikostSerije : int;|+inicializacijaSerije(); +razreziPalice()]  izris na: YUML, Draw Class Diagram, https://yuml.me/diagram/scruffy/class/draw , 2017, dost.12.2019  b) Popis/Obrazložitev  b1.) stroj pred postopkom prilagodimo/nastavimo : nastavimo velikost serije (minimalno 1000) in želen obseg sestavljenega trikotnika.  b2.) omejitev stroja: minimalen pomik posamezne žage znaša 1 cm; najmanjši odrezan kos je velikosti 1 cm; širina reza je zanemarljiva  b3.) po razrezu vsak izmed odrezanih delov aka stranica trikotnika pade v svoj razdelek škatle ; morebitna stranica 'a' v ustrezen razdelek škatle 'del1', 'b' v razdelek škatle 'del2' in 'c' v razdelek škatle 'del3'. Enako oštevilčeni razdelki posameznih škatel vsebujejo stranice istega trikotnika.  c) delna implementacija, dana z razrednim diagramom in prototipom implementacije:    /\*\*\*\*  \*  \* razred RezalnikTrikotnikov  \*  \* model proizvodnje linije za razrez palic in pakiranje stranic trikotnikov  \*/  class RezalnikTrikotnikov{  /\*\*  \* nastavitve za serijo:  \* velikost serije in dolžina palice  \* priprava ustreznega števila paketov  \*/  public static void inicializacijaSerije(){  velikostSerije = 1000;  dolzinaPalice = 100; // v cm  del1 = new int[velikostSerije];  del2 = new int[velikostSerije];  del3 = new int[velikostSerije];  }  /\*\*  \* razred palice na na tri stranice trikotnika  \*/  public static void razreziPalice(){    for(int i=0; i< velikostSerije; i++) {  int rez1 = (int)(Math.random()\*98+1);  int rez2 = (int)(Math.random()\*(dolzinaPalice-rez1-1 + 1);    del1[i] = rez1; del2[i] = rez2; del3[i] = rez1 + rez2;  }  }    /\*\*  \* izpise vsebino vseh paketov, dobrih in izmeta, v obliki (a,b,c), pri čemer so a,b,c dolžine palic v paketih;  \* za prvi paket torej : (del1[0],del2[0],del3[0])  \*/  public static void izpisiVsebinoVsehPaketov(){  }  }  d) testni program : | |
| d1) okvir testnega razreda  public class TestRezalnikaTrikotnikov{  public static void main(String[] args){  // testna aktivnost ena  }  } | d2) prva testna aktivnost je podana z diagramom:  (start)->(inicializacija serije)->(razrezi palice)->(izpisi vsebino vseh paketov)->(end)    Diagram si lahko izrišete na: YUML, Draw Class Diagram,https://yuml.me/diagram/scruffy/activity/draw, 2017, dost.12.2019 |
|  | |

**Naloga 1**

a) Izvedite realizacijo in testiranje izvedenega programa do implementacije, ki je opisana v izhodiščnih opisih aplikacije zgoraj; do vključene izvedbe prve testne aktivnosti v d2

b) Ker je v seriji 1000 razrezov, organizirajte mehanizem 'izpisiVsebinoPaketov' tako, da se bo v vsaki izpisanih po 10 trojčkov, urejenih

po stolpcih enakih širin, kot npr.:

( 6, 9,8), (12, 3,89), (10,31,1), ....

(33,42,9), ( 3,22, 6), ....

opomba: v prime izrisanih vrstic številke niso realne

**Naloga 2**

v proces dodajte metodo izpisiVsebinoVsehPaketovDobrih/0 , ki izpiše zgolj pakete, ki so opredeljeni kot dobri. oz. za vsakega izmed paketov izvedemo aktivnost, podano z diagramom:

(start)-><c>[je dober paket]->(izpisi paket)->(end)

**Naloga 3**

V proces dodajte metodo izpisiVsebinoVsehPaketovUrejeno/0, ki izvede aktivnost, da z diagramom:

(start)->(izpisi vsebino vseh dobrih paketov)->(izpisi vse skart pakete)->(end)

**Naloga 4**

Iz procesa lahko dobimo tudi naslednje informacije:

- koliko paketov je dobrih; npr.: public static int stDobrih(); ;

- koliko paketov je slabih/škart; npr. kot stSkart/0 ;

- procent dobrih paketov,

- procent slabih paketov.

Za vsako od naštetih realizirajte tudi testno metodo z ustrezno vizualizacijo

**Naloga 5**

v procesu realizirajte metodo, ki ugotovi in vrne številko paketa v katerem je najdaljša v seriji odrezana stranica.

test naj rezultat te metode vizualizira kot:

paket št. : <<ugotovljena številka paketa>>

vsebina paketa : (<<a>>,<<b>>,<<c>>)

meta podatek : <<dober paket>>

**Naloga 6**

v procesu realizirajte metodo, ki ugotovi in vrne številko paketa, ki je dober in v katerem je najdaljša v seriji odrezana stranica. test naj rezultat te metode vizualizira kot:

paket št. : <<ugotovljena številka paketa>>

vsebina paketa : (<<a>>,<<b>>,<<c>>)

meta podatek : dober paket

**Naloga 7**

v procesu bi želeli ločiti dobre pakete od škarta. Izvedite metodo prepakiraj/0, ki bo izločila vse dobre pakete v strukture 'dobri', škart pa v 'slabi'. Iz strukture mora biti razvidno, koliko elementov je v njej.

/op.: v razred dodajte 6 tabel : 3 za 'dobri' in 3 za 'slabi' ; gled del1-del3 !. V omenjeni metodi skreirajte tabele takih dolžin, da bo npr. dobri1.length ==dobri2.length==dobri3.length== število dobrih paketov in enako za slabe. Iz tabel del1-3 prepišite pakete v eno od ustreznih tabel.