Projekto ataskaita

Atliko: Egidijus Brazaitis, IFF-5/7, 2016 m. Gruodžio 12 d.

# Bendriniai sąrašai

# Darbo Tikslai

1. Išmokti kurti naujas bendro naudojimo duomenų struktūrų klases pagal duotą (pasirinktą) ADT aprašą;

2. Įtvirtinti projektavimo, programavimo, testavimo ir tyrimo procesų žinias;

3. Išmokti kurti programinės įrangos dokumentaciją;

# Praktinio darbo atlikimas

Realizuotas interfeisas bei multiset duomenų struktūra.

|  |
| --- |
| public interface MultiSetClassic<T> |
| public interface MultiSetClassic<T> {    int put (T obj);    T pick ();    int remove (T obj);    int contains (T obj);    int size ();  } |
| **Komentarai: Bazinis sąrašo klasės interfeisas, pagal kurį buvo sukurtas mano multiset** |
| **public class Multiset<T> implements MultiSetClassic<T>** |
| private List<T> values;  private List<Integer> occurences;    /\*\*  12.\* Constructor  13.\* @param initialCapacity initial capacity of underlying lists  14  \* @param initialCapacity.\*/  public Multiset(int initialCapacity)  {  values = new ArrayList<>(initialCapacity);  occurences = new ArrayList<>(initialCapacity);    }  public Multiset ()  {  this(64);  }  /\*\*  21.\* Inserts a value into the multiset  22.\* @param val value  23.\* @return number of occurences of the value in the set after the addition  24  \* @param val.  \* @return \*/  @Override  public int put(T val) {  int index = getIndex(val);  if (index == -1) {  values.add(val);  occurences.add(1);  return 1;  }  else  {  int currCount = occurences.get(index);  occurences.set(index, currCount + 1);  return currCount + 1;  }  }  public int getIndex(T val)  {  int index = -1;  int i = 0;  for (T value : values)  {  if (value.equals(val)) index = i;  i++;  }    return index;  }    /\*\*  39.\* Returns and deletes (decrements number of uccurences) some value from the multiset  40.\* @return value, null if the multiset is empty  41  \* @return .\*/  @Override  public T pick() {  if (values.isEmpty())  {  return null;  }  if (occurences.get(0) == 1)  {  T v = values.remove(0);  occurences.remove(0);  return v;  }  else {  T v = values.get(0);  occurences.set(0, occurences.get(0) - 1);  return v;  }  }    /\*\*  58.\* Deletes a given value from the multiset (removes one occurrence)  59.\* @param val value  60.\* @return number of occurences of the value in the set after the deletion  61  \* @param e.  \* @return \*/  @Override  public int remove(T e) {  int index = getIndex(e);  int curr = occurences.get(index);  if (curr != 1)  {  occurences.set(index, curr - 1);  return curr - 1;  }  else  {  values.remove(index);  occurences.remove(index);  return 0;  }  }    /\*\*  76.\* Query, if the multiset contains a given value  77.\* @param val value  78.\* @return number of occurences of the given value in the set  79  \* @param e.  \* @return \*/  @Override  public int contains(T e) {  int index = getIndex(e);  if(index == -1) return 0;  return occurences.get(index);  }  public void SpausdintiUnikalius()  {  for (int i = 0; i < values.size();i++)  System.out.println((i+1) + ". " + values.get(i));  }  public void SpausdintiVisa()  {  if (values.isEmpty())  {  System.out.println("Sąrašas tuščias.");  }  int kintamasis = 1;  for (int i = 0; i < values.size(); i++)  for (int j = 0; j < occurences.get(i); j++)  {  System.out.println(kintamasis + ". " + values.get(i));  kintamasis++;  }  }  public void GalimybeIsigyti(T e, int kiekis)  {  int galimybe = -1;  int index = getIndex(e);  if (index >= 0)  {  galimybe = occurences.get(index);  }  if (galimybe >= kiekis) System.out.println("Yra galimybė įsigyti "+ kiekis + " vienetus " + e + " .");  else System.out.println("Nėra galimybės įsigyti "+ kiekis + " vienetus " + e + " .");  }  /\*\*  87.\* Size of the multiset (including all multiplicities)  88.\* @return number of stored entities (values\*occurences)  89  \* @return .\*/  public int size()  {  int count = 0;  for(Integer i : occurences){  count += i;  }  return count;  }  } |

# Multiset duomenų struktūros aprašymas

MultiSet – tai tokia duomenų struktūra, kurioje galima saugoti tokias pačias reikšmes tačiau jos neįrašomos į atmintį, tiesiog nurodomas jų pasikartojimų skaičius . MultiSet gali būti sukuriamas nurodant pradinę talpą arba nenurodant. Jei ji nenurodyta – sukuriamas pagal programuotojo nustatytą bazinį dydį. MultiSet turi du sąrašus – values(reikšmės), occurences(pasikartojimai).

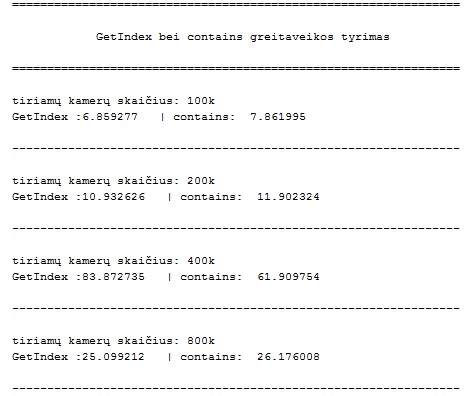
Atliekant laboratorinį darbą susikūriau MultiSetClassic interfeisą su **put**,**remove**, **getIndex**, **size, contains, pick** metodais. Pagal šį interfeisą sukūriau MultiSet klasę. Šią klasę dar papildžiau įvairiais metodais, kurie būdingi MultiSet : **getIndex**, **pick**, ,**galimybėIsigyti, printAll, printUnique**.

# Greitaveikos tyrimas

Greitaveikos tyrimo metu lyginau MultiSet pagrindinių metodų **getIndex** bei **contains** metodus.

**Greitaveikos tyrimai bei rezultatai**

Mano MultiSet metodo **GetIndex(element E)** greitaveikos palyginimas su **contains** metodu:



2. Palyginimas

**Greitaveikos išvados**

Kaip matome iš rezultatų, GetIndex metodas yra greitesnis nei contains esant mažesniam duomenų kiekiui. Kuomet duomenų kiekis didelis, contains veikia greičiau nei GetIndex.

# Projekto išvados

Atliekant šį projektą įsigilinau į MultiSet duomenų struktūrą, įvirtinau programavimo, testavimo bei dokumentacijos rašymo žinias. MultiSet – nėra pati populiariausia duomenų struktūra, tačiau buvo įdomu įsigilinti kaip ji veikia bei suprasti jos pranašumus bei trūkumus.

Bendros projekto atlikimo laiko sąnaudos: ~10 valandų.