**MINISTERUL EDUСAŢIEI ŞI СERСETĂRII AL REPUBLIСII MOLDOVA**

**UNIVERSITATEA DE STAT „ALEСU RUSSO” DIN BĂLŢI**

**FAСULTATEA DE ŞTIINŢE REALE, EСONOMIСE ȘI ALE MEDIULUI**

**СATEDRA DE MATEMATIСĂ ȘI INFORMATIСĂ**

**LUCRARE DE LABORATOR NR. 8**

**„Colecții”**

**Curs „Platforma JAVA Enterprise II”**

**Coordonator:**

**Olga CERBU**

dr.,conf.univ

\_\_\_\_\_\_\_\_\_\_\_\_

(semnătura)

**Autor:**

Studentul grupei AW21M

**Dumitru BAZAOСHI**

\_\_\_\_\_\_\_\_\_\_\_\_

(semnătura)

**BĂLȚI, 2023**

**CUPRINS**

[Scop. Condiția problemei 2](#_Toc149313454)

[Cod Java – Clasa EnumerationTest 2](#_Toc149313455)

[Cod Java – Clasa BitSetTest 3](#_Toc149313456)

[Cod Java – Clasa VectorTest 4](#_Toc149313457)

[Cod Java – Clasa StackTest 6](#_Toc149313458)

[Cod Java – Clasa HasTableTest 8](#_Toc149313459)

[Cod Java – Clasa PropertiesTest 9](#_Toc149313460)

[Cod Java – Clasa GenericMethodTest 10](#_Toc149313461)

[Rezultate Consolă 11](#_Toc149313462)

[Concluzii 13](#_Toc149313463)

[Referințe bibliografice 14](#_Toc149313464)

Scop. Condiția problemei

Scopul acestei lucrări de laborator este să ofere o înțelegere și o experiență practică în utilizarea unor colecții și structuri de date specifice limbajului Java, cum ar fi Enumerația, BitSet, Vector, Stack, HashTable, Properties și metodele generice. Aceasta implică familiarizarea cu caracteristicile, funcționalitățile și utilizările acestor clase pentru manipularea datelor și gestionarea resurselor.

Cod Java – Clasa EnumerationTest

import java.util.Vector;

import java.util.Enumeration;

public class EnumerationTest {

public static void main(String args[]) {

// Aici se declară o variabilă de tip Enumeration numită days. Interfața Enumeration permite extragerea secvențială a elementelor dintr-o structură de date.

Enumeration days;

// Se creează un obiect Vector numit dayNames pentru a stoca numele zilelor săptămânii

Vector dayNames = new Vector();

dayNames.add("Luni"); //adaugarea

dayNames.add("Marti");

dayNames.add("Miercuri");

dayNames.add("Joi");

dayNames.add("Vineri");

dayNames.add("Sambata");

dayNames.add("Duminica");

days = dayNames.elements();

// O buclă while este folosită pentru a parcurge elementele. Aceasta se execută atâta timp cât Enumeration are mai multe elemente de extras.

while (days.hasMoreElements()) {

System.out.println(days.nextElement()); //Obiectul nextElement() - este folosit pentru a obține următorul element din secvență

}

}

}

Cod Java – Clasa BitSetTest

import java.util.BitSet;

public class BitSetTest {

public static void main(String args[]) {

// Se creează două obiecte BitSet cu o dimensiune inițială de 16 de biți fiecare.

BitSet bits1 = new BitSet(16);

BitSet bits2 = new BitSet(16);

// Se setează anumiți biți în fiecare BitSet.

for (int i = 0; i < 20; i++) {

if ((i % 2) == 0)

bits1.set(i); // Se setează biți pe pozițiile pare în bits1.

if ((i % 5) != 0)

bits2.set(i); // Se setează biți pe pozițiile care nu sunt divizibile cu 5 în bits2.

}

// Afișarea stării originale a BitSet-urilor.

System.out.println("BitSet inițial bits1:");

System.out.println(bits1);

System.out.println("\nBitSet inițial bits2:");

System.out.println(bits2);

// Operația de AND între cele două BitSet-uri (bits2 și bits1).

bits2.and(bits1);

System.out.println("\nBitSet-ul rezultat după operația AND între bits2 și bits1:");

System.out.println(bits2);

// Operația de OR între cele două BitSet-uri (bits2 și bits1).

bits2.or(bits1);

System.out.println("\nBitSet-ul rezultat după operația OR între bits2 și bits1:");

System.out.println(bits2);

// Operația de XOR între cele două BitSet-uri (bits2 și bits1).

bits2.xor(bits1);

System.out.println("\nBitSet-ul rezultat după operația XOR între bits2 și bits1:");

System.out.println(bits2);

}

}

Cod Java – Clasa VectorTest

// Importăm clasa pentru a folosi Vector și alte clase necesare

import java.util.\*;

public class VectorTest {

public static void main(String args[]) {

// Inițializăm un obiect Vector cu un anumit capacitate inițială (3) și un pas (2)

Vector v = new Vector(3, 2);

// Afișăm dimensiunea inițială a vectorului

System.out.println("Marimea initiala: " + v.size());

// Afișăm capacitatea inițială a vectorului

System.out.println("Capacitatea initiala: " + v.capacity()); // Metoda capacity() returnează capacitatea actuală a vectorului.

// Adăugăm elemente în vector folosind metoda addElement(Object obj)

v.addElement(Integer.valueOf(1));

v.addElement(Integer.valueOf(2));

v.addElement(Integer.valueOf(3));

v.addElement(Integer.valueOf(4));

// Afișăm capacitatea după adăugarea celor patru elemente

System.out.println("Capacitate după patru pliuri: " + v.capacity());

// Adăugăm elemente de tip Double și Integer în vector

v.addElement(Double.valueOf(5.45));

System.out.println("Capacitatea actuală: " + v.capacity());

v.addElement(Double.valueOf(6.08));

v.addElement(Integer.valueOf(7));

System.out.println("Capacitatea actuală: " + v.capacity());

// Adăugăm elemente de tip Float și Integer

v.addElement(new Float(9.4));

v.addElement(Integer.valueOf(10));

System.out.println("Capacitatea actuală: " + v.capacity());

// Adăugăm două elemente de tip Integer

v.addElement(Integer.valueOf(11));

v.addElement(Integer.valueOf(12));

// Afișăm primul element din vector folosind metoda firstElement()

System.out.println("Al cincilea element: " + (Integer)v.firstElement());

// Metoda firstElement() returnează primul element (elementul cu indexul 0) din vector.

// Afișăm ultimul element din vector folosind metoda lastElement()

System.out.println("Ultimul element: " + (Integer)v.lastElement());

// Metoda lastElement() returnează ultimul element din vector.

// Verificăm dacă vectorul conține elementul cu valoarea 3 folosind metoda contains(Object elem)

if(v.contains(new Integer(3)))

System.out.println("Vectorul contine 3.");

// Enumerăm elementele din vector folosind un obiect de tip Enumeration și afișăm elementele

Enumeration vEnum = v.elements();

System.out.println("\nElementele din vector:");

// Folosim un ciclu while pentru a parcurge și afișa elementele

while(vEnum.hasMoreElements())

System.out.print(vEnum.nextElement() + " ");

System.out.println();

}

}

Cod Java – Clasa StackTest

// Importăm clasa necesară pentru a folosi Stack

import java.util.\*;

public class StackTest {

// Metoda pentru a introduce (push) un element în stivă și a-l afișa

static void showpush(Stack st, int a) {

st.push(new Integer(a));

System.out.println("Impinge(" + a + ")");

System.out.println("Stiva: " + st);

}

// Metoda pentru a elimina (pop) un element din stivă și a-l afișa

static void showpop(Stack st) {

System.out.print("Impusca -> ");

Integer a = (Integer) st.pop();

System.out.println(a);

System.out.println("Stiva: " + st);

}

public static void main(String args[]) {

Stack st = new Stack();

System.out.println("Stiva: " + st);

// Adăugăm câteva elemente în stivă și le afișăm

showpush(st, 42);

showpush(st, 66);

showpush(st, 99);

// Scoatem elemente din stivă și le afișăm

showpop(st);

showpop(st);

showpop(st);

try {

showpop(st);

} catch (EmptyStackException e) {

System.out.println("Stiva pustie"); // Tratăm excepția EmptyStackException în cazul unui stack gol

}

}

}

Cod Java – Clasa HashTableTest

// Importăm clasa necesară pentru a utiliza Hashtable și Enumeration

import java.util.\*;

public class HashTableTest {

public static void main(String args[]) {

// Creăm o hartă hash (Hashtable)

Hashtable balance = new Hashtable();

Enumeration names;

String str;

double bal;

// Adăugăm câteva perechi cheie-valoare în Hashtable

balance.put("Maria", new Double(3434.34));

balance.put("Mihai", new Double(123.22));

balance.put("Oleg", new Double(1378.00));

balance.put("Denis", new Double(99.22));

balance.put("Anton", new Double(-19.08));

// Afișăm toate balanțele din Hashtable

names = balance.keys();

while(names.hasMoreElements()) {

str = (String) names.nextElement();

System.out.println(str + ": " + balance.get(str));

}

System.out.println();

// Adăugăm 1,000 la soldul lui Maria

bal = ((Double)balance.get("Maria")).doubleValue();

// Returnează un obiect get(Object key), care conține valoarea asociată cu cheia.

// În cazul în care cheia nu se află în tabelul hash, se returnează un obiect nul.

balance.put("Maria", new Double(bal + 1000));

//Object put(Object key, Object value)

// Inserați cheia și valoarea în tabelul hash. Returnează null dacă cheia nu este încă

// în tabelul hash, returnează valoarea anterioară asociată cu cheia dacă aceasta se află deja în tabelul hash.

System.out.println("New Balance Maria: " + balance.get("Maria"));

}

}

Cod Java – Clasa PropertiesTest

// Importăm clasa necesară pentru a utiliza Properties, Set și Iterator

import java.util.\*;

public class PropertiesTest {

public static void main(String args[]) {

// Creăm un obiect Properties pentru a stoca perechile cheie-valoare

Properties capitals = new Properties();

Set states;

String str;

// Adăugăm câteva perechi cheie-valoare în obiectul Properties

capitals.put("Illinois", "Springfield");

capitals.put("Missouri", "Jefferson City");

capitals.put("Washington", "Olympia");

capitals.put("California", "Sacramento");

capitals.put("Indiana", "Indianapolis");

// Afișăm toate statele și capitalele din obiectul Properties

states = capitals.keySet(); // Obținem un set cu cheile

Iterator itr = states.iterator();

while(itr.hasNext()) {

str = (String) itr.next();

System.out.println("Capitala " + str + " este " +

capitals.getProperty(str) + ".");

}

// String getProperty(String key)

// Returnează valoarea asociată cu cheia. Se returnează un obiect nul

// dacă cheia nu se află nici în listă, nici în lista de proprietăți implicite.

System.out.println();

// Obținem capitala statului "Florida" și furnizăm un mesaj dacă cheia nu există

str = capitals.getProperty("Florida", "Nu a fost gasita");

System.out.println("Capitala statului Florida " + str + ".");

}

}

Cod Java – Clasa GenericMethodTest

public class GenericMethodTest {

public static <E> void printArray(E[] inputArray) {

// Afișăm elementele din array

for (E element : inputArray) {

System.out.printf("%s ", element);

}

System.out.println();

}

public static void main(String args[]) {

// Creăm array-uri de tip Integer, Double și Character

Integer[] intArray = {1, 2, 3, 4, 5};

Double[] doubleArray = {1.1, 2.2, 3.3, 4.4};

Character[] charArray = {'S', 'A', 'L', 'U', 'T'};

System.out.println("Matricea intArray conține:");

printArray(intArray); // Apelăm metoda cu un array de tip Integer

System.out.println("\nMatricea doubleArray conține:");

printArray(doubleArray); // Apelăm metoda cu un array de tip Double

System.out.println("\nMatricea charArray conține:");

printArray(charArray); // Apelăm metoda cu un array de tip Character

}

}

Rezultate Consolă

*Clasa EnumerationTest:*

Luni

Marti

Miercuri

Joi

Vineri

Sambata

Duminica

*Clasa BitSetTest:*

BitSet inițial bits1:

{0, 2, 4, 6, 8, 10, 12, 14, 16, 18}

BitSet inițial bits2:

{1, 2, 3, 4, 6, 7, 8, 9, 11, 12, 13, 14, 16, 17, 18, 19}

BitSet-ul rezultat după operația AND între bits2 și bits1:

{2, 4, 6, 8, 12, 14, 16, 18}

BitSet-ul rezultat după operația OR între bits2 și bits1:

{0, 2, 4, 6, 8, 10, 12, 14, 16, 18}

BitSet-ul rezultat după operația XOR între bits2 și bits1:

{}

*Clasa VectorTest:*

Marimea initiala: 0

Capacitatea initiala: 3

Capacitate după patru pliuri: 5

Capacitatea actuală: 5

Capacitatea actuală: 7

Capacitatea actuală: 9

Al cincilea element: 1

Ultimul element: 12

Vectorul contine 3.

Elementele din vector:

1 2 3 4 5.45 6.08 7 9.4 10 11 12

*Clasa StackTest:*

Stiva: []

Impinge(42)

Stiva: [42]

Impinge(66)

Stiva: [42, 66]

Impinge(99)

Stiva: [42, 66, 99]

Impusca -> 99

Stiva: [42, 66]

Impusca -> 66

Stiva: [42]

Impusca -> 42

Stiva: []

Impusca -> Stiva pustie

*Clasa HashTabelTest:*

Denis: 99.22

Oleg: 1378.0

Maria: 3434.34

Anton: -19.08

Mihai: 123.22

New Balance Maria: 4434.34

*Clasa PropertiesTest:*

Capitala Indiana este Indianapolis.

Capitala Illinois este Springfield.

Capitala Missouri este Jefferson City.

Capitala California este Sacramento.

Capitala Washington este Olympia.

Capitala statului Florida Nu a fost gasita.

*Clasa GenericMethodTest:*

Matricea intArray conține:

1 2 3 4 5

Matricea doubleArray conține:

1.1 2.2 3.3 4.4

Matricea charArray conține:

S A L U T

Concluzii

În concluzie, această lucrare de laborator a oferit oportunitatea de a explora și de a lucra cu diverse colecții și structuri de date din limbajul Java, precum Enumerația, BitSet, Vector, Stack, HashTable, Properties și metodele generice. Am învățat cum să le utilizăm eficient pentru stocarea, manipularea și afișarea datelor, cât și pentru parcurgerea elementelor acestor colecții.

Referințe bibliografice

1. *Collections in Java* [online] [vizitat 28.10.2023]. Disponibil: <https://www.javatpoint.com/collections-in-java>
2. *Collections in Java* *8* [online] [vizitat 28.10.2023]. Disponibil: <https://www.geeksforgeeks.org/collections-in-java-2/>
3. *Collections Framework Overview* [online] [vizitat 29.10.2023]. Disponibil: <https://docs.oracle.com/javase/8/docs/technotes/guides/collections/overview.html>
4. *Generics in Java* [online] [vizitat 29.10.2023]. Disponibil: <https://www.geeksforgeeks.org/generics-in-java/>