Міністерство освіти і науки, молоді та спорту України

Національний університет «Львівська політехніка»

Інститут комп’ютерних наук та інформаційних технологій

Кафедра автоматизованих систем управління



**Звіт**

**до лабораторної роботи №4**

**на тему**

**“Collections Framework”**

**Виконав:**

Ст. гр. КН-203 Павлишин  Д. А.

**Перевірив:**

Скрибайло-Леськів Д.Ю.

**Завдання**

**Exercise l:** (2) Create a new class called Gerbil with an int gerbilNumber that's initialized

in the constructor. Give it a method called hop( ) that displays which gerbil number this is, and that it's hopping. Create an ArrayList and add Gerbil objects to the List. Now use the get( ) method to move through the List and call hop( ) for each Gerbil.

**Exercise 2:** (3) Create a generator class that produces character names (as String objects) from your favorite movie (you can use Snow White or Star Wars as a fallback) each time you call next( ), and loops around to the beginning of the character list when it runs out of names. Use this generator to fill an array, an ArrayList, a LinkedList, a HashSet, a LinkedHashSet, and a TreeSet, then print each container.

**Exercise 3:** (3) Create a class, then make an initialized array of objects of your class. Fill a List

from your array. Create a subset of your List by using subList( ), then remove this subset from your List.

**Exercise 4:** (1) Modify Exercise l so it uses an Iterator to move through the List while calling hop( ).

**Exercise 5:** (2) Write a method that uses an Iterator to step through a Collection and print

the toString( ) of each object in the container. Fill all the different types of Collections with objects and apply your method to each container.

**Exercise 6:** (3) Create and populate a List<Integer>. Create a second List < Integer > of

the same size as the first, and use Listlterators to read elements from the first List and insert them into the second in reverse order. (You may want to explore a number of different ways to solve this problem.)

**Exercise 7:** (3) Create an empty LinkedList<Integer>. Using a Listlterator, add Integers to the List by always inserting them in the middle of the List.

**Exercise 8:** (2) Take the Gerbil class in Exercise 1 and put it into a Map instead, associating

each Gerbil's name (e.g. "Fuzzy" or "Spot") as a String (the key) for each Gerbil (the value) you put in the table. Get an Iterator for the keySet( ) and use it to move through the Map, looking up the Gerbil for each key and printing out the key and telling the Gerbil to hop( ).

**Exercise 9:** (3) Fill a HashMap with key-value pairs. Print the results to show ordering by

hash code. Extract the pairs, sort by key, and place the result into a LinkedHashMap. Show that the insertion order is maintained.

**Exercise 10:** (2) Repeat the previous exercise with a HashSet and LinkedHashSet.

**Exercise 11:** (2) Fill a LinkedHashMap with String keys and objects of your choice. Now extract the pairs, sort them based on the keys, and reinsert them into the Map.

**Exercise 12:** (2) Write a class called Command that contains a String and has a method operation( ) that displays the String. Write a second class with a method that fills a Queue

with Command objects and returns it. Pass the filled Queue to a method in a third class that

consumes the objects in the Queue and calls their operation( ) methods.

**Exercise 13:** (2) Fill a PriorityQueue (using offer( )) with Double values created using java.util.Random, then remove the elements using poll( ) and display them.

**Код програми:**

**LabThree.java:**

package com.pavlyshyn;  
  
import java.util.\*;  
  
public class LabThree {  
  
 public static void main(String[] args) {  
 System.*out*.println("Task 1");  
 List<Gerbil> gerbils = new ArrayList<>(3);  
 for (int i = 0; i < 3; i++) {  
 gerbils.add(new Gerbil(i));  
 }  
 for (int i = 0; i < gerbils.size(); i++) {  
 gerbils.get(i).hop();  
 }

System.*out*.println("Task 4");  
 ShowMovie.*main*();  
 System.*out*.println("Task 7");  
 UniqueElement.*main*();  
 System.*out*.println("Task 8");  
 for (Iterator<Gerbil> iterator = gerbils.iterator(); iterator.hasNext(); iterator.next().hop()) ;  
  
 System.*out*.println("Task 11");  
 List<Collection<String>> ListOfCollections =  
 Arrays.<Collection<String>>*asList*(  
 new ArrayList<String>(),  
 new LinkedList<String>(),  
 new HashSet<String>(),  
 new TreeSet<String>());  
 for (Collection<String> item : ListOfCollections) {  
 ShowMovie.*fill*(item);  
 }  
 for (Collection<String> item : ListOfCollections) {  
 *iteratorPrint*(item.iterator());  
 }  
 System.*out*.println("Task 11");  
 /\*Linked\*/  
 List<Integer> integers = new LinkedList<>(Arrays.*asList*(1, 2, 3, 4, 5)), integers2 = new LinkedList<>(integers);  
 System.*out*.println(integers);  
 *iteratorReverse*(integers2);  
 System.*out*.println(integers2);  
 System.*out*.println("Task 14");  
 LinkedList<Integer> integers3 = new LinkedList<>();  
 ListIterator<Integer> integerListIterator = integers3.listIterator();  
 for (int i = 0; i < 6; i++) {  
 integerListIterator.add(i);  
 if (i % 2 == 0) {  
 integerListIterator.previous();  
 }  
 }  
 System.*out*.println(integers3);  
  
 System.*out*.println("Task 17");  
 Map<String, Gerbil> stringGerbilMap = new HashMap<>();  
 stringGerbilMap.put("Fuzzy", new Gerbil(1));  
 stringGerbilMap.put("Joe", new Gerbil(2));  
 stringGerbilMap.put("Mike", new Gerbil(3));  
 stringGerbilMap.put("Jerry", new Gerbil(4));  
 stringGerbilMap.put("Kitty", new Gerbil(5));  
 stringGerbilMap.put("Ben", new Gerbil(6));  
 stringGerbilMap.put("Josh", new Gerbil(7));  
 stringGerbilMap.put("Perry", new Gerbil(8));  
 Iterator<Map.Entry<String, Gerbil>> stringIterator = stringGerbilMap.entrySet().iterator();  
 while (stringIterator.hasNext()) {  
 Map.Entry<String, Gerbil> stringGerbilEntry = stringIterator.next();  
 System.*out*.print(stringGerbilEntry.getKey() + " = ");  
 stringGerbilEntry.getValue().hop();  
 }  
 System.*out*.println("Task 18");  
 Map<String, Gerbil> linkedStringGerbilMap = new LinkedHashMap<>();  
 String[] listOfKeys = stringGerbilMap.keySet().toArray(new String[0]);  
 Arrays.*sort*(listOfKeys);  
 for (String key : listOfKeys) {  
 linkedStringGerbilMap.put(key, stringGerbilMap.get(key));  
 }  
 for (String key : linkedStringGerbilMap.keySet()) {  
 System.*out*.print(key + " = ");  
 linkedStringGerbilMap.get(key).hop();  
 }  
 System.*out*.println("Task 19");  
 Set<Integer> integerHashSet = new HashSet<>(), integerLinkedHashSet = new LinkedHashSet<>();  
 for(int i=0; i< 8; i++) {  
 integerHashSet.add(new Random().nextInt(20));  
 }  
 System.*out*.println(integerHashSet);  
 Integer[] ints = integerHashSet.toArray(new Integer[0]);  
 Arrays.*sort*(ints);  
 for(int item:ints){  
 integerLinkedHashSet.add(item);  
 }  
 System.*out*.println(integerLinkedHashSet);  
 System.*out*.println("Task 24");  
 Map<String,Gerbil> stringGerbilMap2 = new LinkedHashMap<>(stringGerbilMap);  
 Map<String,Gerbil> stringGerbilMap3 = new LinkedHashMap<>();  
  
 String[] strings = stringGerbilMap2.keySet().toArray(new String[0]);  
 Arrays.*sort*(strings);  
 for(String key: strings){  
 stringGerbilMap3.put(key,stringGerbilMap2.get(key));  
 System.*out*.println(key+ " = " + stringGerbilMap3.get(key));  
 }  
  
 System.*out*.println("Task 27");  
 Queue<Command> commands = new LinkedList<>();  
 Producer.*produce*(commands);  
 Consumer.*consume*(commands);  
 System.*out*.println("Task 28");  
 PriorityQueue<Double>doublePriorityQueue = new PriorityQueue<Double>();  
 for(int i =0 ; i<10; i++){  
 doublePriorityQueue.offer(new Random().nextDouble());  
 }  
 *printPrirityQueue*(doublePriorityQueue);  
 }  
 static void printPrirityQueue(PriorityQueue<Double> priorityQueue){  
 for(Object data = priorityQueue.poll(); data != null;  
 data = priorityQueue.poll())  
 System.*out*.print(data + " ");  
 System.*out*.println();  
 }  
 static void iteratorReverse(List<Integer> integers) {  
 ListIterator<Integer> fwd = integers.listIterator();  
 ListIterator<Integer> rev =  
 integers.listIterator(integers.size());  
 int mid = integers.size() >> 1;  
 for (int i = 0; i < mid; i++) {  
 Integer tmp = fwd.next();  
 fwd.set(rev.previous());  
 rev.set(tmp);  
 }  
 }  
  
 static void iteratorPrint(Iterator iterator) {  
 while (iterator.hasNext()) {  
 System.*out*.print(iterator.next() + ", ");  
 }  
 System.*out*.println();  
 }  
  
}

**Gerbil.java:**

package com.pavlyshyn;  
  
public class Gerbil {  
 int gerbilNumber;  
 public Gerbil(int gerbilNumber) {  
 this.gerbilNumber = gerbilNumber;  
 }  
  
 public void hop() {  
 System.*out*.println("Gerbil №" + gerbilNumber+" is hopping");  
 }  
  
 @Override  
 public String toString() {  
 return "Gerbil{" +  
 "gerbilNumber=" + gerbilNumber +  
 '}';  
 }  
**}**

**ShowMovie.java:**

package com.pavlyshyn;  
  
import java.util.\*;  
  
class GeneratorStarWars {  
 String[] CharacterNames = {"Luke Skywalker", "R2D2", "Han Solo", "Darth Vader" };  
 private int next= 0;  
 String next(){  
 String s = CharacterNames[next];  
 next = (next+1)%CharacterNames.length;  
 return s;  
 }  
  
}  
  
public class ShowMovie{  
 private static final GeneratorStarWars *GENERATOR\_STAR\_WARS* = new GeneratorStarWars();  
 static String[] fill(String[]strings){  
 for(int i = 0; i<strings.length; i++){  
 strings[i] = *GENERATOR\_STAR\_WARS*.next();  
 }  
 return strings;  
 }  
 static Collection<String> fill(Collection<String> stringCollection){  
 for(int i = 0; i<5; i++){  
 stringCollection.add(*GENERATOR\_STAR\_WARS*.next());  
 }  
 return stringCollection;  
 }  
 public static void main(String...args) {  
 System.*out*.println(Arrays.*toString*(*fill*(new String[5])));  
 System.*out*.println(*fill*(new ArrayList<>()));  
 System.*out*.println(*fill*(new LinkedList<>()));  
 System.*out*.println(*fill*(new HashSet<>()));  
 System.*out*.println(*fill*(new LinkedList<>()));  
 System.*out*.println(*fill*(new TreeSet<>()));  
 }  
}

**UniqueElement:**

package com.pavlyshyn;  
  
import com.sun.javafx.UnmodifiableArrayList;  
  
import java.util.ArrayList;  
import java.util.Arrays;  
import java.util.List;  
  
public class UniqueElement{  
 private int ID = *counter*++;  
 private static int *counter*;  
  
 @Override  
 public String toString() {  
 return "UniqueElement{" +  
 "ID=" + ID +  
 '}';  
 }  
 static UniqueElement[] fill(UniqueElement[]uniqueElements){  
 for(int i = 0; i<uniqueElements.length; i++){  
 uniqueElements[i] = new UniqueElement();  
 }  
 return uniqueElements;  
 }  
 public static void main(String...args) {  
 UniqueElement[] uniqueElements = new UniqueElement[5];  
 *fill*(uniqueElements);  
 System.*out*.println(Arrays.*asList*(uniqueElements));  
 List<UniqueElement> listOfUniqueElements=new ArrayList<>(Arrays.*asList*(uniqueElements));  
 System.*out*.println("List of unique elements: "+listOfUniqueElements);  
 List<UniqueElement> sublistOfUniqueElements = listOfUniqueElements.subList(2,4);  
 System.*out*.println("Sublist of unique elements [2,4]: " + sublistOfUniqueElements);  
 listOfUniqueElements.removeAll(sublistOfUniqueElements);  
 System.*out*.println("List without sublist"+listOfUniqueElements);  
 }  
}

**Producer.java**

package com.pavlyshyn;  
  
import java.util.Queue;  
  
public class Producer {  
 public static Queue<Command> produce(Queue<Command> stringQueue){  
 stringQueue.offer(new Command("moveLeft"));  
 stringQueue.offer(new Command("moveRight"));  
 stringQueue.offer(new Command("moveUp"));  
 stringQueue.offer(new Command("moveDown"));  
 stringQueue.offer(new Command("jump"));  
 return stringQueue;  
 }  
}

**Consumer.java**

package com.pavlyshyn;  
  
import java.util.Queue;  
  
public class Consumer {  
 public static void consume(Queue<Command> stringQueue){  
 while (stringQueue.peek()!=null){  
 System.*out*.println(stringQueue.remove().operation());  
 }  
 }  
}

**Реалізація програми:**

**"C:\Program Files\Java\jdk1.8.0\_181\bin\java.exe"**

**Connected to the target VM, address: '127.0.0.1:64466', transport: 'socket'**

**Task 1**

**Gerbil №0 is hopping**

**Gerbil №1 is hopping**

**Gerbil №2 is hopping**

**Task 2**

**0, 1, 2, 3, 4, 5, 6, 7, 8, 9, Task 4**

**[Luke Skywalker, R2D2, Han Solo, Darth Vader, Luke Skywalker]**

**[R2D2, Han Solo, Darth Vader, Luke Skywalker, R2D2]**

**[Han Solo, Darth Vader, Luke Skywalker, R2D2, Han Solo]**

**[Luke Skywalker, R2D2, Han Solo, Darth Vader]**

**[Luke Skywalker, R2D2, Han Solo, Darth Vader, Luke Skywalker]**

**[Darth Vader, Han Solo, Luke Skywalker, R2D2]**

**Task 7**

**[UniqueElement{ID=0}, UniqueElement{ID=1}, UniqueElement{ID=2}, UniqueElement{ID=3}, UniqueElement{ID=4}]**

**List of unique elements: [UniqueElement{ID=0}, UniqueElement{ID=1}, UniqueElement{ID=2}, UniqueElement{ID=3}, UniqueElement{ID=4}]**

**Sublist of unique elements [2,4]: [UniqueElement{ID=2}, UniqueElement{ID=3}]**

**List without sublist[UniqueElement{ID=0}, UniqueElement{ID=1}, UniqueElement{ID=4}]**

**Task 8**

**Gerbil №0 is hopping**

**Gerbil №1 is hopping**

**Gerbil №2 is hopping**

**Task 11**

**Han Solo, Darth Vader, Luke Skywalker, R2D2, Han Solo,**

**Darth Vader, Luke Skywalker, R2D2, Han Solo, Darth Vader,**

**Luke Skywalker, R2D2, Han Solo, Darth Vader,**

**Darth Vader, Han Solo, Luke Skywalker, R2D2,**

**Task 11**

**[1, 2, 3, 4, 5]**

**[5, 4, 3, 2, 1]**

**Task 14**

**[1, 3, 5, 4, 2, 0]**

**Task 17**

**Joe = Gerbil №2 is hopping**

**Mike = Gerbil №3 is hopping**

**Perry = Gerbil №8 is hopping**

**Kitty = Gerbil №5 is hopping**

**Josh = Gerbil №7 is hopping**

**Ben = Gerbil №6 is hopping**

**Fuzzy = Gerbil №1 is hopping**

**Jerry = Gerbil №4 is hopping**

**Task 18**

**Ben = Gerbil №6 is hopping**

**Fuzzy = Gerbil №1 is hopping**

**Jerry = Gerbil №4 is hopping**

**Joe = Gerbil №2 is hopping**

**Josh = Gerbil №7 is hopping**

**Kitty = Gerbil №5 is hopping**

**Mike = Gerbil №3 is hopping**

**Perry = Gerbil №8 is hopping**

**Task 19**

**[1, 18, 2, 7, 11, 13, 15]**

**[1, 2, 7, 11, 13, 15, 18]**

**Task 24**

**Ben = Gerbil{gerbilNumber=6}**

**Fuzzy = Gerbil{gerbilNumber=1}**

**Jerry = Gerbil{gerbilNumber=4}**

**Joe = Gerbil{gerbilNumber=2}**

**Josh = Gerbil{gerbilNumber=7}**

**Kitty = Gerbil{gerbilNumber=5}**

**Mike = Gerbil{gerbilNumber=3}**

**Perry = Gerbil{gerbilNumber=8}**

**Task 27**

**moveLeft**

**moveRight**

**moveUp**

**moveDown**

**jump**

**Task 28**

**0.09316360385815814**

**0.21271665309287213**

**0.28107068978034633**

**0.541207159632896**

**0.5874076078124224**

**0.7115874648807695**

**0.7922251513469845**

**0.8170835253940414**

**0.8593886627790768**

**0.9844137307607878**

**Disconnected from the target VM, address: '127.0.0.1:64466', transport: 'socket'**

**Process finished with exit code 0**