МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

“ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”

КАФЕДРА «Програмна інженерія та інтелектуальні технології управління»

ЗВІТ

з лабораторної роботи № 4

з навчальної дисципліни

“ОСНОВИ ПРОГРАМУВАННЯ JAVA (ДВВ)”

ВИКОНАЛА

Студентка групи КН-222а

Інеса РЕПЕШКО

ПЕРЕВІРИВ

доц. каф. ПІІТУ к.т.н

доцент Валерій ВОЛОВЩИКОВ

Харків 2023

# **Тема роботи**

“Робота з узагальненнями та колекціями”.

# **1.1 Завдання №1 “Індивідуальне завдання”**

До ієрархії класів, створеної під час реалізації індивідуального завдання лабораторної роботи № 3 додати похідні класи, які представляють першу з сутностей індивідуального завдання. Перший з них зберігає дані у за допомогою списку (ArrayList), другий за допомогою множини (SortedSet).

Для сортування списку скористатися засобами класу Collections та інтерфейсу List. Працюючи з множиною, слід послідовно створювати варіанти множини, впорядковані за різними критеріями.

Здійснити тестування обох реалізацій. Результати виконання програм повинні збігатися.

Умову завдання з лабораторної роботи № 3 для варіанту 24 (номер 24 за порядком у списку групи) наведено нижче.

Изображение выглядит как текст, снимок экрана, Шрифт, чек

Автоматически созданное описание

Рисунок 1.1 – Умова 1 завдання № 1 варіант 24

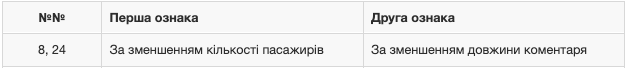


Рисунок 1.2 – Умова 2 завдання № 1 варіант 24

**1.2 Набір тестових даних завдання № 1**

**1.2.1 Тестовий випадок № 1** **“Тестування класу MetroStationWithCollection, MetroStationWithList та MetroStationWithSortedSet”**

Опис: Тестування функціональності класу MetroStationWithCollection, MetroStationWithList та MetroStationWithSortedSet.

Тестові дані:

Таблиця 1.1 – Тестові дані для класів MetroStationWithList та MetroStationWithSortedSet

|  |  |  |  |
| --- | --- | --- | --- |
| Індекс | Пасажиропотік | Коментар | |
| Масив 1 з даними про години роботи станції метро для подальшого тестування класів MetroStationWithList та MetroStationWithSortedSet: | | | |
| 0 | 578 | "Medium ridership" | |
| 1 | 256 | "Medium ridership" | |
| 2 | 817 | "High ridership" | |
| 3 | 1115 | "Very high ridership" | |
| 4 | 817 | "High ridership" | |
| Тестування конструктора класів з параметрами – валідні дані станції, – тестування отримання даних про назву та даних про рік відкриття, отримання списку годин роботи станції метро:   * назва станції: "Sportyvna" * рік відкриття: 1975 * часи роботи: масив 1 годин роботи станції метро | | | |
| Масив 2 з даними про години роботи станції метро для подальшого тестування з класу MetroStationWithCollection: | | | |
| 0 | 23 | | "Very low ridership" |
| 1 | 345 | | "Medium ridership" |
| 2 | 87 | | "Low ridership" |
| 3 | 1007 | | "Very high ridership" |
| Тестування отримання даних про назву та даних про рік відкриття станції метро:   * назва станції: "Sportyvna" * рік відкриття: 1975 | | | |
| Тестування встановлення нових даних (перезапису) про години роботи для станції метро замість даних, що містяться у масиві годин роботи для станції метро:   * нові часи роботи: масив 2 годин роботи станції метро. | | | |
| Тестування встановлення нових даних однієї години роботи у масиві годин роботи за індексом та виведення всього масиву годин роботи з оновленими даними: | | | |
| 0 | 250 | | "Medium ridership" |
| Тестування отримання по індексу даних про одну годину роботи із масиву годин роботи станції метро:   * індекс однієї години роботи: 1 | | | |
| Тестування отримання кількості всіх годин роботи в масиві годин роботи станції метро:   * масив годин роботи: масив 2 годин роботи станції метро. | | | |
| Тестування встановлення значень назви та року відкриття станції метро 3, видалення даних масиву годин роботи, сортування за зменшенням пасажиропотоку, сортування за зменшенням довжини коментаря при відсутності даних масиву годин роботи:   * назва станції: " Derzhprom" * рік відкриття: 1995 * Масив годин роботи: відсутній | | | |
| Тестування створення масиву годин роботи станції метро 3 з послідовним додаванням даних окремих годин роботи в кінець масиву годин роботи – валідні дані: | | | |
|  | 320 | | "Medium ridership" |
|  | 88 | | "Very low ridership" |
|  | 470 | | "Medium ridership" |
|  | 105 | | "Low ridership" |
|  | 688 | | "High ridership" |
|  | 1234 | | "Very high ridership" |
| Тестування створення масиву годин роботи станції метро 3 з послідовним додаванням даних окремих годин роботи в кінець масиву годин роботи – невалідні дані: | | | |
|  | -1 | | null |
| Тестування створення масиву годин роботи станції метро 3 з послідовним додаванням даних окремих годин роботи в кінець масиву годин роботи – невалідні дублюючі дані: | | | |
|  | 1234 | | "Very high ridership" |
| Тестування на даних масиву годин роботи станції метро 3 сортування за зменшенням пасажиропотоку та за зменшенням довжини коментаря: | | | |
| 0 | 320 | | "Medium ridership" |
| 1 | 88 | | "Very low ridership" |
| 2 | 470 | | "Medium ridership" |
| 3 | 105 | | "Low ridership" |
| 4 | 688 | | "High ridership" |
| 5 | 1234 | | "Very high ridership" |

Очікуваний результат: Успішне завершення виконання програми. Результати сортування повинні збігатися. Виведення у консоль повідомлення з результатами:

|  |
| --- |
| TESTING METRO STATION WITH LIST: Get data about Metro Station from the MetroStationWithList class:  Name: 'Sportyvna' Opened: 1975  Get data from the Operating Hours List:  Hour { ridership = 578, comment = 'Medium ridership' }  Hour { ridership = 256, comment = 'Medium ridership' }  Hour { ridership = 817, comment = 'High ridership' }  Hour { ridership = 1115, comment = 'Very high ridership' }  Initial Metro Station data:  Station: Name: 'Sportyvna'. Opened: 1975. Hours:  Hour { ridership = 578, comment = 'Medium ridership' }  Hour { ridership = 256, comment = 'Medium ridership' }  Hour { ridership = 817, comment = 'High ridership' }  Hour { ridership = 1115, comment = 'Very high ridership' }  Get Metro Station Name and Opened Year:  Name: Sportyvna Opened: 1975  Reset the Operating Hours for the Metro Station:  Station: Name: 'Sportyvna'. Opened: 1975. Hours:  Hour { ridership = 23, comment = 'Very low ridership' }  Hour { ridership = 345, comment = 'Medium ridership' }  Hour { ridership = 87, comment = 'Low ridership' }  Hour { ridership = 1007, comment = 'Very high ridership' }  Set the Operating Hour by index and get all Operating Hours:  Hour { ridership = 250, comment = 'Medium ridership' }  Hour { ridership = 345, comment = 'Medium ridership' }  Hour { ridership = 87, comment = 'Low ridership' }  Hour { ridership = 1007, comment = 'Very high ridership' }  Get Operating Hour by index:  Hour { ridership = 345, comment = 'Medium ridership' }  Get count of all Operating Hours: 4  SORTING RESULTS:  Sort data for Metro Station without Operating Hours:  Station: Name: 'Derzhprom'. Opened: 1995. Hours:  There are no hours for this station.  Create the Metro Station:  Add 6 valid Operating Hours at Metro Station:  true true true true true  Add one Operating Hour with invalid data at Metro Station: true  Add one Operating Hour with duplicate data at Metro Station: false  Station: Name: 'Derzhprom'. Opened: 1995. Hours:  Hour { ridership = 320, comment = 'Medium ridership' }  Hour { ridership = 88, comment = 'Very low ridership' }  Hour { ridership = 107, comment = 'Low ridership' }  Hour { ridership = 688, comment = 'High ridership' }  Hour { ridership = 1234, comment = 'Very high ridership' }  Hour { ridership = 0, comment = '' }  Sort Operating Hours by decreasing ridership:  Station: Name: 'Derzhprom'. Opened: 1995. Hours:  Hour { ridership = 1234, comment = 'Very high ridership' }  Hour { ridership = 688, comment = 'High ridership' }  Hour { ridership = 320, comment = 'Medium ridership' }  Hour { ridership = 107, comment = 'Low ridership' }  Hour { ridership = 88, comment = 'Very low ridership' }  Hour { ridership = 0, comment = '' }  Sort Operating Hours by descending comment length:  Station: Name: 'Derzhprom'. Opened: 1995. Hours:  Hour { ridership = 88, comment = 'Very low ridership' }  Hour { ridership = 1234, comment = 'Very high ridership' }  Hour { ridership = 320, comment = 'Medium ridership' }  Hour { ridership = 107, comment = 'Low ridership' }  Hour { ridership = 688, comment = 'High ridership' }  Hour { ridership = 0, comment = '' }  TESTING METRO STATION WITH SORTED SET: Get data about Metro Station from the MetroStationWithSortedSet class:  Name: 'Sportyvna' Opened: 1975  Get data from the Operating Hours SortedSet:  Hour { ridership = 1115, comment = 'Very high ridership' }  Hour { ridership = 817, comment = 'High ridership' }  Hour { ridership = 578, comment = 'Medium ridership' }  Hour { ridership = 256, comment = 'Medium ridership' }  Initial Metro Station data:  Station: Name: 'Sportyvna'. Opened: 1975. Hours:  Hour { ridership = 1115, comment = 'Very high ridership' }  Hour { ridership = 817, comment = 'High ridership' }  Hour { ridership = 578, comment = 'Medium ridership' }  Hour { ridership = 256, comment = 'Medium ridership' }  Get Metro Station Name and Opened Year:  Name: Sportyvna Opened: 1975  Reset the Operating Hours for the Metro Station:  Station: Name: 'Sportyvna'. Opened: 1975. Hours:  Hour { ridership = 345, comment = 'Medium ridership' }  Hour { ridership = 87, comment = 'Low ridership' }  Hour { ridership = 1007, comment = 'Very high ridership' }  Hour { ridership = 23, comment = 'Very low ridership' }  Set the Operating Hour by index and get all Operating Hours:  Hour { ridership = 87, comment = 'Low ridership' }  Hour { ridership = 1007, comment = 'Very high ridership' }  Hour { ridership = 250, comment = 'Medium ridership' }  Hour { ridership = 23, comment = 'Very low ridership' }  Get Operating Hour by index:  Hour { ridership = 1007, comment = 'Very high ridership' }  Get count of all Operating Hours: 4  SORTING RESULTS:  Sort data for Metro Station without Operating Hours:  Station: Name: 'Derzhprom'. Opened: 1995. Hours:  There are no hours for this station.  Create the Metro Station:  Add 6 valid Operating Hours at Metro Station:  true true true true true  Add one Operating Hour with invalid data at Metro Station: true  Add one Operating Hour with duplicate data at Metro Station: false  Station: Name: 'Derzhprom'. Opened: 1995. Hours:  Hour { ridership = 88, comment = 'Very low ridership' }  Hour { ridership = 1234, comment = 'Very high ridership' }  Hour { ridership = 320, comment = 'Medium ridership' }  Hour { ridership = 107, comment = 'Low ridership' }  Hour { ridership = 688, comment = 'High ridership' }  Hour { ridership = 0, comment = '' }  Sort Operating Hours by decreasing ridership:  Station: Name: 'Derzhprom'. Opened: 1995. Hours:  Hour { ridership = 1234, comment = 'Very high ridership' }  Hour { ridership = 688, comment = 'High ridership' }  Hour { ridership = 320, comment = 'Medium ridership' }  Hour { ridership = 107, comment = 'Low ridership' }  Hour { ridership = 88, comment = 'Very low ridership' }  Hour { ridership = 0, comment = '' }  Sort Operating Hours by descending comment length:  Station: Name: 'Derzhprom'. Opened: 1995. Hours:  Hour { ridership = 88, comment = 'Very low ridership' }  Hour { ridership = 1234, comment = 'Very high ridership' }  Hour { ridership = 320, comment = 'Medium ridership' }  Hour { ridership = 107, comment = 'Low ridership' }  Hour { ridership = 688, comment = 'High ridership' }  Hour { ridership = 0, comment = '' } |

**1.3 Програмний код завдання № 1**

Лістинг 1.1 - Програмний код реалізований у файлі “Hour.java”

|  |
| --- |
| package task1;  import java.util.Arrays;  */\*\*  \* The {****@code*** *Hour} class performs hour with {****@code*** *ridership} and {****@code*** *comment}.  \*/* public class Hour implements Comparable<Hour> {  */\*\* Ridership is the number of passengers visiting a metro station per hour. \*/* private int ridership;   */\*\* Comment on the {****@code*** *ridership} metric. \*/* private String comment;   */\*\*  \* The constructor initialises the hour object with the default values.  \*/* public Hour() {  }   */\*\*  \* The constructor initialises the hour object with the specified values.  \** ***@param*** *ridership the ridership;  \** ***@param*** *comment the comment.  \*/* public Hour(int ridership, String comment) {  if (ridership < 0) {  this.ridership = 0;  }   if (comment == null) {  this.comment = "";  }   this.ridership = ridership;  this.comment = comment;  }   */\*\*  \* Gets the {****@code*** *ridership} of the hour.  \** ***@return*** *the {****@code*** *ridership}.  \*/* public int getRidership() {  if (ridership < 0) {  return 0;  }  return ridership;  }   */\*\*  \* Sets the {****@code*** *ridership} of the hour.  \** ***@param*** *ridership the {****@code*** *ridership} to be set.  \*/* public void setRidership(int ridership) {  if (ridership < 0) {  this.ridership = 0;  }   this.ridership = ridership;  }   */\*\*  \* Gets the {****@code*** *comment} for the hour.  \** ***@return*** *the {****@code*** *comment}.  \*/* public String getComment() {  if (comment == null) {  return "";  }   return comment;  }   */\*\*  \* Sets the {****@code*** *comment} for the hour.  \** ***@param*** *comment the {****@code*** *comment} to be set.  \*/* public void setComment(String comment) {  if (comment == null) {  this.comment = "";  }   this.comment = comment;  }   */\*\*  \* Gets the length of a comment in the hour.  \** ***@return*** *the length of a comment.  \*/* public int getCommentLength() {  if (comment == null) {  return 0;  }   return getComment().length();  }   */\*\*  \* Calculates the count of words of a comment in the hour.  \** ***@return*** *the length of a comment.  \*/* public int calculateWordCountOfComment() {  if (comment == null  || comment.isEmpty()) {  return 0;  }   String[] wordArray = comment.split(" ");   return wordArray.length;  }   */\*\*  \* Provides the string representing the Hour object.  \** ***@return*** *the string representing the Hour object.  \*/* @Override  public String toString() {  return "Hour\t{ "  + "ridership = " + getRidership()  + ",\tcomment = \'" + getComment() + "\' }";  }   */\*\*  \* Checks metro station this hour is equivalent to another.  \** ***@param*** *obj the hour with which check the equivalence;  \** ***@return*** *{****@code*** *true}, if two hours are the same and {****@code*** *false} otherwise.  \*/* @Override  public boolean equals(Object obj) {  if (this == obj) {  return true;  }   if (!(obj instanceof Hour hour)) {  return false;  }   return Integer.*compare*(hour.getRidership(), getRidership()) == 0  && hour.getComment().equals(getComment());  }   */\*\*  \* Calculates the hash code of the hour.  \* If two objects are equal, they must have the same hash code.  \* If this method is called multiple times on the same object, it must return the same number each time.  \** ***@return*** *the hash code of the hour.  \*/* @Override  public int hashCode() {  return Integer.*hashCode*(getRidership()) \* getComment().hashCode();  }   */\*\*  \* Compares this Hour object with another Hour object based on ridership.  \** ***@param*** *h the object to be compared;  \** ***@return*** *negative number, if this object is smaller, zero, if they are equal,  \* positive number, if this object is larger.  \*/* @Override  public int compareTo(Hour h) {  return Integer.*compare*(h.getRidership(), getRidership());  }   */\*\*  \* Prints the array of hours.  \** ***@param*** *hours array of hours to print.  \*/* public void printHourArray(Hour[] hours) {  System.*out*.println("Array of hours:");   for (Hour hour : hours) {  System.*out*.println(hour);  }  }   */\*\*  \* Tests of the functionality of the {****@code*** *Hour} class.  \*/* public void testHour() {  System.*out*.println("Create Hour with default constructor:");  Hour hour = new Hour();  System.*out*.println(hour);  System.*out*.println("Length of comment:\t" + hour.getCommentLength());  System.*out*.println("Count of words in comment:\t" + hour.calculateWordCountOfComment());   System.*out*.println("\nCreate Hour with parameterized constructor:");  System.*out*.println("Valid data for hour:");  hour = new Hour(100, "Low ridership");  System.*out*.println(hour);  System.*out*.println("Invalid data for hour:");  Hour invalidHour = new Hour(-200, null);  System.*out*.println(invalidHour);   System.*out*.println("\nSet values for the Hour:");  hour.setRidership(200);  hour.setComment("Medium ridership");  System.*out*.println(hour);   System.*out*.println("\nGet values for the Hour:");  System.*out*.println("Hour\t{ "  + "ridership = " + hour.getRidership()  + ",\tcomment = \'" + hour.getComment() + "\' }");  System.*out*.println("Get length of comment:\t" + hour.getCommentLength());  System.*out*.println("Get count of words in comment:\t" + hour.calculateWordCountOfComment() + "\n");   Hour[] hours = { hour,  new Hour(50, "Very low ridership"),  new Hour(200, "Medium ridership"),  new Hour(100, "Low ridership"),  new Hour(700, "High ridership"),  new Hour(1200, "Very high ridership"),  invalidHour  };  printHourArray(hours);   System.*out*.println("\nCheck for equal values of Hours at index 0 and 1:\t" + hours[0].equals(hours[1]));  System.*out*.println("Hour at index 0:\t" + hours[0]);  System.*out*.println("Hour at index 1:\t" + hours[1]);  System.*out*.println("Check for equal values of Hours at index 0 and 2:\t" + hours[0].equals(hours[2]));  System.*out*.println("Hour at index 0:\t" + hours[0]);  System.*out*.println("Hour at index 2:\t" + hours[2]);   System.*out*.println("\nComparison of Hours at index 0 and 1:\t" + hours[0].compareTo(hours[1]));  System.*out*.println("Hashcode of Hour at index 0:\t" + hours[0].hashCode());  System.*out*.println("Hashcode of Hour at index 1:\t" + hours[1].hashCode());  System.*out*.println("Comparison of Hours at index 0 and 2:\t" + hours[0].compareTo(hours[2]));  System.*out*.println("Hashcode of Hour at index 0:\t" + hours[0].hashCode());  System.*out*.println("Hashcode of Hour at index 2:\t" + hours[2].hashCode());  System.*out*.println("Comparison of Hours at index 1 and 2:\t" + hours[1].compareTo(hours[2]));  System.*out*.println("Hashcode of Hour at index 1:\t" + hours[1].hashCode());  System.*out*.println("Hashcode of Hour at index 2:\t" + hours[2].hashCode());   System.*out*.println("\nSort array of Hours by descending ridership:");  Arrays.*sort*(hours);  printHourArray(hours);  } } |

Лістинг 1.2 - Програмний код реалізований у файлі “AbstractMetroStation.java”

|  |
| --- |
| package task1;  import java.util.Arrays;  */\*\*  \* Abstract class representing metro station with {****@code*** *name}, {****@code*** *opened} year and operating hour data.  \* Access to the sequence of hours, {****@code*** *name} and {****@code*** *opened} year is represented by abstract methods.  \*/* public abstract class AbstractMetroStation {  */\*\*  \* Gets the {****@code*** *name} for the metro station.  \* The derived class must provide an implementation of this method.  \** ***@return*** *the {****@code*** *name}.  \*/* public abstract String getName();   */\*\*  \* Sets the {****@code*** *name} for the metro station.  \* The derived class must provide an implementation of this method.  \** ***@param*** *name the {****@code*** *name} to be set.  \*/* public abstract void setName(String name);   */\*\*  \* Gets the {****@code*** *opened} year for the metro station.  \* The derived class must provide an implementation of this method.  \** ***@return*** *the {****@code*** *opened}.  \*/* public abstract int getOpened();   */\*\*  \* Sets the {****@code*** *opened} year for the metro station.  \* The derived class must provide an implementation of this method.  \** ***@param*** *opened the {****@code*** *opened} year to be set.  \*/* public abstract void setOpened(int opened);   */\*\*  \* Gets the {****@code*** *hour} with index {****@code*** *i}.  \* The derived class must provide an implementation of this method.  \** ***@param*** *i the index of hour array element;  \** ***@return*** *the object of class {****@code*** *Hour} with index {****@code*** *i}.  \*/* public abstract Hour getHour(int i);   */\*\*  \* Sets the {****@code*** *hour} with index {****@code*** *i}.  \* The derived class must provide an implementation of this method.  \** ***@param*** *i index of {****@code*** *hour} in array of hours;  \** ***@param*** *hour the object of class {****@code*** *Hour} with index {****@code*** *i} to be set.  \*/* public abstract void setHour(int i, Hour hour);   */\*\*  \* Gets the array of operating hours for the metro station.  \* The derived class must provide an implementation of this method.  \** ***@return*** *the array of operating hours.  \*/* public abstract Hour[] getHours();   */\*\*  \* Sets the array of operating hours for the metro station.  \* The derived class must provide an implementation of this method.  \** ***@param*** *hours the array of operating hours to be set.  \*/* public abstract void setHours(Hour[] hours);   */\*\*  \* Adds a link to the new operating {****@code*** *hour} at the end of the hour array.  \* The derived class must provide an implementation of this method.  \** ***@param*** *hour the object of class {****@code*** *Hour} to be added;  \** ***@return*** *{****@code*** *true}, if the link was added successfully, {****@code*** *false} otherwise.  \*/* public abstract boolean addHour(Hour hour);   */\*\*  \* Creates a new operating {****@code*** *hour} and adds a link to it at the end of the hour array.  \* The derived class must provide an implementation of this method.  \** ***@param*** *ridership the ridership;  \** ***@param*** *comment the comment;  \** ***@return*** *{****@code*** *true}, if the link was added successfully, {****@code*** *false} otherwise.  \*/* public abstract boolean addHour(int ridership, String comment);   */\*\*  \* Counts the number of hours in the hours array.  \* The derived class must provide an implementation of this method.  \** ***@return*** *the number of hours.  \*/* public abstract int countHours();   */\*\*  \* Removes the sequence of hours from hours array.  \* The derived class must provide an implementation of this method.  \*/* public abstract void removeHours();   */\*\*  \* Provides the string representing the object that is inherited from this abstract class.  \** ***@return*** *the string representing the object that is inherited from this abstract class.  \*/* @Override  public String toString() {  StringBuilder string = new StringBuilder();  string.append("Station:\t")  .append("Name: \'").append(getName()).append("\'.\t")  .append("Opened: ").append(getOpened()).append(".\t")  .append("Hours:\n");   if (countHours() <= 0) {  string.append("There are no hours for this station.\n");  } else {  for (Hour h : getHours()) {  string.append(h).append("\n");  }  }   return string.toString();  }   */\*\*  \* Checks whether this metro station is equivalent to another.  \** ***@param*** *obj the metro station with which check the equivalence.  \** ***@return*** *{****@code*** *true}, if two weathers are the same, {****@code*** *false} otherwise.  \*/* @Override  public boolean equals(Object obj) {  if (this == obj) {  return true;  }   if (!(obj instanceof AbstractMetroStation ms)) {  return false;  }   if (!ms.getName().equals(getName())  || Integer.*compare*(ms.getOpened(), getOpened()) != 0) {  return false;  }   return Arrays.*equals*(getHours(), ms.getHours());  }   */\*\*  \* Calculates the hash code of the metro station.  \* If two objects are equal, they must have the same hash code.  \* If this method is called multiple times on the same object, it must return the same number each time.  \** ***@return*** *the hash code of the metro station.  \*/* @Override  public int hashCode() {  return getName().hashCode() \* Integer.*hashCode*(getOpened()) \* Arrays.*hashCode*(getHours());  }   */\*\*  \* An additional static function for adding hour reference to the provided array of hours.  \** ***@param*** *hours the array to which the hour is added;  \** ***@param*** *hour the link that is added;  \** ***@return*** *updated array of hours.  \*/* public static Hour[] addHourToArray(Hour[] hours, Hour hour) {  if (hour.getRidership() < 0  || hour.getComment() == null) {  return hours;  }   Hour[] newHours;   if (hours == null) {  newHours = new Hour[1];  } else {  newHours = new Hour[hours.length + 1];  System.*arraycopy*(hours, 0, newHours, 0, hours.length);  }   newHours[newHours.length - 1] = hour;   return newHours;  }   */\*\*  \* Calculates the total ridership for an array of metro station operating hours.  \** ***@return*** *null, if there is no pointer to the hours array, or it is empty, the total ridership otherwise.  \*/* public Integer calculateTotalRidership() {  if (countHours() == 0) {  return null;  }   int totalRidership = 0;   for (Hour hour : getHours()) {  totalRidership += hour.getRidership();  }   return totalRidership;  }   */\*\*  \* Finds the hours with the minimal ridership in the array of metro station operating hours.  \** ***@return*** *null, if there is no pointer to the hours array, or it is empty,  \* array of hours with minimal ridership otherwise.  \*/* public Hour[] findHoursWithMinRidership() {  if (countHours() == 0) {  return null;  }   Hour minHour = getHours()[0];   for (Hour hour : getHours()) {  if (hour.getRidership() < minHour.getRidership()) {  minHour = hour;  }  }   Hour[] hours = null;   for (Hour hour : getHours()) {  if (hour.getRidership() == minHour.getRidership()) {  hours = *addHourToArray*(hours, hour);  }  }   return hours;  }   */\*\*  \* Finds the hours with the maximum count of words in the comment in the array of metro station operating hours.  \** ***@return*** *null, if there is no pointer to the hours array, or it is empty,  \* array of hours with the maximum word count in comment otherwise  \*/* public Hour[] findHoursWithMaxWordCountOfComment() {  if (countHours() == 0) {  return null;  }   Hour maxHour = getHours()[0];   for (Hour hour : getHours()) {  if (hour.calculateWordCountOfComment() > maxHour.calculateWordCountOfComment()) {  maxHour = hour;  }  }   Hour[] hours = null;   for (Hour hour : getHours()) {  if (hour.calculateWordCountOfComment() == maxHour.calculateWordCountOfComment()) {  hours = *addHourToArray*(hours, hour);  }  }   return hours;  }   */\*\*  \* Finds the total ridership for an array of metro station operating hours and prints the result to the console.  \*/* private void printTotalRidership() {  Integer totalRidership = calculateTotalRidership();  System.*out*.print("Total ridership for station:\t");   if (totalRidership == null) {  System.*out*.println("There is no ridership hours.");  } else {  System.*out*.println(totalRidership);  }  }   */\*\*  \* Prints the array of hours.  \** ***@param*** *hours the array of hours to be printed.  \*/* private void printHours(Hour[] hours) {  for (Hour hour : hours) {  System.*out*.println(hour);  }  }   */\*\*  \* Finds the hours with the minimal ridership in the array of metro station operating hours  \* and prints the result to the console.  \*/* private void printHoursWithMinRidership() {  Hour[] hours = findHoursWithMinRidership();  System.*out*.print("Hours with minimal ridership:\t");   if (hours == null) {  System.*out*.println("There is no ridership hours.");  } else {  System.*out*.println();  printHours(hours);  }  }   */\*\*  \* Finds the hours with the maximum count of words in the comment in the array of metro station operating hours  \* and prints the result to the console.  \*/* private void printHoursWithMaxWordCountOfComment() {  Hour[] hours = findHoursWithMaxWordCountOfComment();  System.*out*.print("Hours with the maximum word count in a comment:\t");   if (hours == null) {  System.*out*.println("There is no ridership hours.");  } else {  System.*out*.println();  printHours(hours);  }  }   */\*\*  \* Sorts a sequence of hours by decreasing ridership using bubble sorting.  \*/* public void sortByDecreasingRidership() {  if (countHours() == 0) {  return;  }   boolean unsorted = true;   while (unsorted) {  unsorted = false;   for (int i = 0; i < getHours().length - 1; i++) {  if (getHours()[i].getRidership() < getHours()[i + 1].getRidership()) {  Hour temp = getHours()[i];  getHours()[i] = getHours()[i + 1];  getHours()[i + 1] = temp;  unsorted = true;  }  }  }  }   */\*\*  \* Sorts a sequence of hours by descending comment length using insertion sorting.  \*/* public void sortByDescendingCommentLength() {  if (countHours() == 0) {  return;  }   for (int i = 0; i < getHours().length; i++) {  Hour key = getHours()[i];  int j;   for (j = i - 1; j >= 0  && Integer.*compare*(getHours()[j].getCommentLength(), key.getCommentLength()) < 0; j--) {  getHours()[j + 1] = getHours()[j];  }   getHours()[j + 1] = key;  }  }   */\*\*  \* An additional function for adding hours to a sequence of hours in hours array.  \** ***@return*** *The object is inherited from this abstract class.  \*/* public AbstractMetroStation createMetroStationHours() {  System.*out*.println("Add 6 valid Operating Hours at Metro Station:");  System.*out*.print(addHour(320, "Medium ridership") + "\t");  Hour hour = new Hour(88, "Very low ridership");  System.*out*.println(addHour(hour) + "\t"  + addHour(107, "Low ridership") + "\t"  + addHour(688, "High ridership") + "\t"  + addHour(1234, "Very high ridership"));   System.*out*.println("Add one Operating Hour with invalid data at Metro Station:\t"  + addHour(-1, null));   System.*out*.println("Add one Operating Hour with duplicate data at Metro Station:\t"  + addHour(1234, "Very high ridership"));   return this;  }   */\*\*  \* Calls up search methods and print results of searching.  \*/* private void showSearchResults() {  printTotalRidership();  printHoursWithMinRidership();  printHoursWithMaxWordCountOfComment();  }   */\*\*  \* Performs testing of search methods.  \*/* public void testSearchData() {  System.*out*.println("SEARCHING RESULTS:");  setName("Universytet");  setOpened(1984);  System.*out*.println("Search data for Metro Station without Operating Hours:");  removeHours();  showSearchResults();  System.*out*.println();   System.*out*.println("Create the Metro Station:");  createMetroStationHours();  System.*out*.println(this);  showSearchResults();  System.*out*.println();   System.*out*.println("Add new two Operating Hours with min ridership and max word count in comment for searching:");  System.*out*.println(addHour(75, "Very low ridership"));  System.*out*.println(addHour(2000, "Maximum possible ridership for station"));  System.*out*.println(this);  showSearchResults();  }   */\*\*  \* Performs testing of sorting methods.  \*/* public void testSortingData() {  System.*out*.println();  System.*out*.println("SORTING RESULTS:");  setName("Derzhprom");  setOpened(1995);  System.*out*.println("Sort data for Metro Station without Operating Hours:");  removeHours();  sortByDecreasingRidership();  sortByDescendingCommentLength();  System.*out*.println(this);   System.*out*.println("Create the Metro Station:");  createMetroStationHours();  System.*out*.println(this);   System.*out*.println("Sort Operating Hours by decreasing ridership:");  sortByDecreasingRidership();  System.*out*.println(this);   System.*out*.println("Sort Operating Hours by descending comment length:");  sortByDescendingCommentLength();  System.*out*.println(this);  } } |

Лістинг 1.3 - Програмний код реалізований у файлі “MetroStationWithCollection.java”

|  |
| --- |
| package task1;  */\*\*  \* An abstract class {****@link*** *MetroStationWithCollection} representing a Metro Station with a collection of operating  \* hours. Extends the {****@link*** *AbstractMetroStation} class.  \*/* public abstract class MetroStationWithCollection extends AbstractMetroStation {  */\*\* The name of the metro station. \*/* private String name;   */\*\* The opened year of the metro station. \*/* private int opened;   */\*\*  \* The constructor initialises the object with the default values.  \*/* public MetroStationWithCollection() {}   */\*\*  \* The constructor initialises the object with the specified values with metro station {****@code*** *name}  \* and {****@code*** *opened} year.  \** ***@param*** *name the name of metro station;  \** ***@param*** *opened the opened year of metro station.  \*/* public MetroStationWithCollection(String name, int opened) {  this.name = name;  this.opened = opened;  }   public abstract void setHour(int i, Hour hour);   */\*\*  \* Gets the {****@code*** *name} for the metro station.  \** ***@return*** *the {****@code*** *name} of metro station.  \*/* @Override  public String getName() {  return name;  }   */\*\*  \* Sets the {****@code*** *name} for the metro station.  \** ***@param*** *name the {****@code*** *name} of metro station to be set.  \*/* @Override  public void setName(String name) {  this.name = name;  }   */\*\*  \* Gets the {****@code*** *opened} year for the metro station.  \** ***@return*** *the {****@code*** *opened} year of metro station.  \*/* @Override  public int getOpened() {  return opened;  }   */\*\*  \* Sets the {****@code*** *opened} year for the metro station.  \** ***@param*** *opened the {****@code*** *opened} year of metro station to be set.  \*/* @Override  public void setOpened(int opened) {  this.opened = opened;  }   */\*\*  \* Performs testing of the functionality of the {****@code*** *MetroStationWithCollection} class.  \*/* public void testMetroStationWithCollection() {  System.*out*.println("Initial Metro Station data:");  System.*out*.println(this);   Hour[] hoursArray = {  new Hour(23, "Very low ridership"),  new Hour(345, "Medium ridership"),  new Hour(87, "Low ridership"),  new Hour(1007, "Very high ridership")  };   System.*out*.println("Get Metro Station Name and Opened Year:");  System.*out*.println("Name:\t" + getName() + "\tOpened:\t" + getOpened());  System.*out*.println();   System.*out*.println("Reset the Operating Hours for the Metro Station:");  setHours(hoursArray);  System.*out*.println(this);   System.*out*.println("Set the Operating Hour by index and get all Operating Hours:");  setHour(0, new Hour(250, "Medium ridership"));  hoursArray = getHours();  for (Hour hour : hoursArray) {  System.*out*.println(hour);  }  System.*out*.println();   System.*out*.println("Get Operating Hour by index:");  System.*out*.println(getHour(1));  System.*out*.println("Get count of all Operating Hours:\t" + countHours());  System.*out*.println();  } } |

Лістинг 1.4 - Програмний код реалізований у файлі “MetroStationWithList.java”

|  |
| --- |
| package task1;  import java.util.ArrayList; import java.util.Arrays; import java.util.Comparator; import java.util.List; import java.util.Collections; import java.util.Set; import java.util.LinkedHashSet;  */\*\*  \* Represents metro station data with an ArrayList of operating hours.  \* This class is inherited from the abstract {****@link*** *MetroStationWithCollection}.  \*/* public class MetroStationWithList extends MetroStationWithCollection {  */\*\* List of operating hours of the metro station. \*/* private List<Hour> hours = new ArrayList<>();   */\*\*  \* The constructor initialises the metro station object with the default values.  \*/* public MetroStationWithList() {}   */\*\*  \* The constructor initialises the metro station object with the specified values with {****@code*** *name},  \* {****@code*** *opened} year and operating {****@code*** *hours}.  \** ***@param*** *name the name of metro station;  \** ***@param*** *opened the opened year of metro station;  \** ***@param*** *hours the operating hours of metro station.  \*/* public MetroStationWithList(String name, int opened, ArrayList<Hour> hours) {  super(name, opened);  Set<Hour> uniqueSet = new LinkedHashSet<>(hours);  this.hours = new ArrayList<>(uniqueSet);  }   */\*\*  \* The constructor initialises the metro station object with the specified values with {****@code*** *name} and {****@code*** *opened} year.  \** ***@param*** *name the name of metro station;  \** ***@param*** *opened the opened year of metro station.  \*/* public MetroStationWithList(String name, int opened) {  super(name, opened);  }   */\*\*  \* Gets the array of operating hours for the metro station.  \** ***@return*** *the array of hours.  \*/* @Override  public Hour[] getHours() {  return hours.toArray(new Hour[0]);  }   */\*\*  \* Gets the list of operating hours for the metro station.  \** ***@return*** *the list of operating hours for the metro station.  \*/* public List<Hour> getHoursList() {  return hours;  }   */\*\*  \* Sets the list of operating hours for the metro station.  \** ***@param*** *hours the array of hours to be set.  \*/* @Override  public void setHours(Hour[] hours) {  Set<Hour> uniqueSet = new LinkedHashSet<>(Arrays.*asList*(hours));  this.hours = new ArrayList<>(uniqueSet);  }   */\*\*  \* Gets the {****@code*** *hour} with index {****@code*** *i} from the hours list.  \** ***@return*** *the object of class {****@code*** *Hour} with index {****@code*** *i}.  \*/* @Override  public Hour getHour(int i) {  return hours.get(i);  }   */\*\*  \* Sets the {****@code*** *hour} with index {****@code*** *i} to hours list.  \** ***@param*** *i index of {****@code*** *hour} in hours list;  \** ***@param*** *hour the object of class {****@code*** *Hour} with index {****@code*** *i} to be set.  \*/* @Override  public void setHour(int i, Hour hour) {  if (hours.contains(hour)) {  return;  }   hours.set(i, hour);  }   */\*\*  \* Adds a link to the new {****@code*** *hour} at the end of the hours list.  \** ***@param*** *hour the object of class {****@code*** *Hour} to be added to the hours list;  \** ***@return*** *{****@code*** *true}, if the link was added successfully, {****@code*** *false} otherwise.  \*/* @Override  public boolean addHour(Hour hour) {  if (hours.contains(hour)) {  return false;  }   return hours.add(hour);  }   */\*\*  \* Creates a new {****@code*** *hour} and adds a link to it at the end of the sequence at the hours list.  \** ***@param*** *ridership the ridership;  \** ***@param*** *comment the comment;  \** ***@return*** *{****@code*** *true}, if the link was added successfully, {****@code*** *false} otherwise.  \*/* @Override  public boolean addHour(int ridership, String comment) {  return addHour(new Hour(ridership, comment));  }   */\*\*  \* Counts the number of hours in the sequence at hours list.  \** ***@return*** *the number of hours.  \*/* @Override  public int countHours() {  return hours.size();  }   */\*\*  \* Removes the sequence of hours from hours list.  \*/* @Override  public void removeHours() {  hours.clear();  }   */\*\*  \* Overridden decreasing ridership sorting method using the standard sort function of class {****@code*** *Collections}.  \* Is provided by the implementation of the Comparable interface for the {****@code*** *Hour} class.  \*/* @Override  public void sortByDecreasingRidership() {  Collections.*sort*(hours);  }   */\*\*  \* Overridden descending comment length sorting method using the default sort function of interface {****@code*** *List}.  \* Is provided by {****@code*** *Comparator}.  \*/* @Override  public void sortByDescendingCommentLength() {  hours.sort(Comparator.*comparing*(Hour::getCommentLength).reversed());  } } |

Лістинг 1.5 - Програмний код реалізований у файлі “MetroStationWithSortedSet.java”

|  |
| --- |
| package task1;  import java.util.Arrays; import java.util.Comparator; import java.util.SortedSet; import java.util.TreeSet;  */\*\*  \* Represents metro station data with an SortedSet of operating hours.  \* This class is inherited from the abstract {****@link*** *MetroStationWithCollection}.  \*/* public class MetroStationWithSortedSet extends MetroStationWithCollection {  */\*\*  \* Private field representing a SortedSet of Hour objects.  \* The set is initialized with a TreeSet using a custom comparator based on the hashCode of the Hour objects.  \*/* private SortedSet<Hour> hours = new TreeSet<>(Comparator.*comparing*(Hour::hashCode));   */\*\*  \* The constructor initialises the object with the default values.  \*/* public MetroStationWithSortedSet() {  }   */\*\*  \* The constructor initialises the metro station object with the specified values with {****@code*** *name},  \* {****@code*** *opened} year and operating {****@code*** *hours}.  \** ***@param*** *name the name of metro station;  \** ***@param*** *opened the opened year of metro station;  \** ***@param*** *hours the operating hours of metro station.  \*/* public MetroStationWithSortedSet(String name, int opened, SortedSet<Hour> hours) {  super(name, opened);  this.hours = hours;  }   */\*\*  \* The constructor initialises the metro station object with the specified values with {****@code*** *name}  \* and {****@code*** *opened} year.  \** ***@param*** *name the name of metro station;  \** ***@param*** *opened the opened year of metro station.  \*/* public MetroStationWithSortedSet(String name, int opened) {  super(name, opened);  }   */\*\*  \* Gets the array of operating hours for the metro station.  \** ***@return*** *the array of hours.  \*/* @Override  public Hour[] getHours() {  return hours.toArray(new Hour[0]);  }   */\*\*  \* Gets the SortedSet of operating hours for the metro station.  \** ***@return*** *the SortedSet of operating hours.  \*/* public SortedSet<Hour> getHoursSortedSet() {  return hours;  }   */\*\*  \* Sets the SortedSet of operating hours for the metro station.  \** ***@param*** *hours the array of operating hours.  \*/* @Override  public void setHours(Hour[] hours) {  this.hours = new TreeSet<>(Comparator.*comparing*(Hour::hashCode));  this.hours.addAll(Arrays.*asList*(hours));  }   */\*\*  \* Gets the operating {****@code*** *hour} with index {****@code*** *i}.  \** ***@return*** *the object of class {****@code*** *Hour} with index {****@code*** *i}.  \*/* @Override  public Hour getHour(int i) {  return hours.toArray(new Hour[0])[i];  }   */\*\*  \* Sets the operating {****@code*** *hour} with index {****@code*** *i}.  \** ***@param*** *i index of {****@code*** *hour};  \** ***@param*** *hour the object of class {****@code*** *Hour} with index {****@code*** *i} to be set.  \*/* @Override  public void setHour(int i, Hour hour) {  hours.remove(getHour(i));  hours.add(hour);  }   */\*\*   \* Adds a link to the new {****@code*** *hour}.  \** ***@param*** *hour the object of class {****@code*** *Hour} to be added;  \** ***@return*** *{****@code*** *true}, if the link was added successfully, {****@code*** *false} otherwise.  \*/* @Override  public boolean addHour(Hour hour) {  return hours.add(hour);  }   */\*\*  \* Creates a new {****@code*** *hour} and adds a link to it.  \** ***@param*** *ridership the ridership;  \** ***@param*** *comment the comment;  \** ***@return*** *{****@code*** *true}, if the link was added successfully, {****@code*** *false} otherwise.  \*/* @Override  public boolean addHour(int ridership, String comment) {  return addHour(new Hour(ridership, comment));  }   */\*\*  \* Counts the number of hours in the sequence.  \** ***@return*** *the number of hours.  \*/* @Override  public int countHours() {  return hours.size();  }   */\*\*  \* Removes the sequence of hours.  \*/* @Override  public void removeHours() {  hours.clear();  }   */\*\*  \* Overridden decreasing ridership sorting method using {****@code*** *TreeSet}.  \* Is provided by the implementation of the Comparable interface for the {****@code*** *Hour} class.  \*/* @Override  public void sortByDecreasingRidership() {  SortedSet<Hour> newSet = new TreeSet<>();  newSet.addAll(hours);  hours = newSet;  }   */\*\*  \* Overridden descending comment length sorting method using the {****@code*** *TreeSet}.  \* Is provided by {****@code*** *Comparator}.  \*/* @Override  public void sortByDescendingCommentLength() {  SortedSet<Hour> newSet = new TreeSet<>(Comparator.*comparing*(Hour::getCommentLength).reversed());  newSet.addAll(hours);  hours = newSet;  } } |

Лістинг 1.6 - Програмний код реалізований у файлі “Main.java”

|  |
| --- |
| package task1;  import java.util.ArrayList; import java.util.Arrays; import java.util.Comparator; import java.util.TreeSet; import java.util.SortedSet; import java.util.List;  */\*\*  \* The {****@code*** *Main} class performs testing of the {****@link*** *MetroStationWithCollection}, {****@link*** *MetroStationWithList}  \* and {****@link*** *MetroStationWithSortedSet} classes functionality.  \*/* public class Main {  */\*\*  \* Calls the test methods for testing the functionality of the {****@link*** *MetroStationWithCollection} class.  \** ***@param*** *metroStation the MetroStationWithCollection object with test data.  \*/* private static void testMetroStation(MetroStationWithCollection metroStation) {  metroStation.testMetroStationWithCollection();  metroStation.testSortingData();  System.*out*.println();  }   */\*\*  \* Performs testing of the {****@link*** *MetroStationWithList} class functionality.  \** ***@param*** *hourArray the {****@code*** *Hour} array with test data.  \*/* private static void testMetroStationWithList(Hour[] hourArray) {  System.*out*.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  System.*out*.println("TESTING METRO STATION WITH LIST:");  System.*out*.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  MetroStationWithList metroStation = new MetroStationWithList(  "Sportyvna",  1975,  new ArrayList<>(Arrays.*asList*(hourArray))  );   System.*out*.println("Get data about Metro Station from the MetroStationWithList class:\n"  + "Name:\t\'" + metroStation.getName() + "\'\tOpened:\t" + metroStation.getOpened() + "\n");  System.*out*.println("Get data from the Operating Hours List:");  List<Hour> hours = metroStation.getHoursList();  hours.forEach(System.*out*::println);  System.*out*.println();   *testMetroStation*(metroStation);  System.*out*.println();  }   */\*\*  \* Performs testing of the {****@link*** *MetroStationWithSortedSet} class functionality.  \** ***@param*** *hourArray the {****@code*** *Hour} array with test data.  \*/* private static void testMetroStationWithSortedSet(Hour[] hourArray) {  System.*out*.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  System.*out*.println("TESTING METRO STATION WITH SORTED SET:");  System.*out*.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  SortedSet<Hour> hours = new TreeSet<>(Comparator.*comparing*(Hour::hashCode));  hours.addAll(Arrays.*asList*(hourArray));  MetroStationWithSortedSet metroStation = new MetroStationWithSortedSet(  "Sportyvna",  1975,  hours  );   System.*out*.println("Get data about Metro Station from the MetroStationWithSortedSet class:\n"  + "Name:\t\'" + metroStation.getName() + "\'\tOpened:\t" + metroStation.getOpened() + "\n");  System.*out*.println("Get data from the Operating Hours SortedSet:");  SortedSet<Hour> hourSortedSet = metroStation.getHoursSortedSet();  hourSortedSet.forEach(System.*out*::println);  System.*out*.println();   *testMetroStation*(metroStation);  System.*out*.println();  }   */\*\*  \* Performs testing of the {****@link*** *MetroStationWithCollection}, {****@link*** *MetroStationWithList} and  \* {****@link*** *MetroStationWithSortedSet} classes functionality. The {****@code*** *args} are not used.  \** ***@param*** *args the command-line arguments.  \*/* public static void main(String[] args) {  Hour[] hours = {  new Hour(578, "Medium ridership"),  new Hour(256, "Medium ridership"),  new Hour(817, "High ridership"),  new Hour(1115, "Very high ridership"),  new Hour(817, "High ridership"),  };  *testMetroStationWithList*(hours);  *testMetroStationWithSortedSet*(hours);  } } |

Сформований Javadoc для всієї лабораторної роботи міститься у прикріплених файлах до роботи. Структура файлів в проекті наступна:

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 1.3 - Структура файлів в проекті “lab4”

**1.4** **Екранні форми за результатами роботи програмного коду завдання № 1**

Изображение выглядит как текст, снимок экрана, Шрифт, меню

Автоматически созданное описание

Рисунок 1.4 - Результати роботи програмного коду відповідно до тестового випадку № 1 “Тестування класу MetroStationWithCollection, MetroStationWithList та MetroStationWithSortedSet” частина 1

Изображение выглядит как текст, снимок экрана, меню, черно-белый

Автоматически созданное описание

Рисунок 1.5 - Результати роботи програмного коду відповідно до тестового випадку № 1 “Тестування класу MetroStationWithCollection, MetroStationWithList та MetroStationWithSortedSet” частина 2

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

Рисунок 1.6 - Результати роботи програмного коду відповідно до тестового випадку № 1 “Тестування класу MetroStationWithCollection, MetroStationWithList та MetroStationWithSortedSet” частина 3

**Изображение выглядит как текст, снимок экрана, черно-белый, шаблон

Автоматически созданное описание**

Рисунок 1.7 - Результати роботи програмного коду відповідно до тестового випадку № 1 “Тестування класу MetroStationWithCollection, MetroStationWithList та MetroStationWithSortedSet” частина 4

**2.1 Завдання № 2 “Перелік для опису місяців року”**

Створити перелік "Місяць". Необхідно визначати у конструкторі і зберігати кількість днів (для невисокосного року). Додати методи отримання попереднього та наступного місяця, а також функцію, яка повертає сезон для кожного місяця. Передбачити виведення місяців українською (російською) мовою. Створити статичну функцію виведення даних про усі місяці. Протестувати переліку в функції main() тестового класу.

**2.2 Набір тестових даних завдання № 2**

**2.2.1 Тестовий випадок № 1** **“Тестування пори року”**

Опис: Тестування виведення даних про один пори року із вказанням поточного, попереднього і наступного сезонів від поточного. І тестування виведення переліку всіх сезонів року.

Тестові дані:

Таблиця 2.1 – Тестові дані для пори року

|  |  |
| --- | --- |
| № з/п | Назва сезону з переліку |
| 1 | Тестування виведення даних про один сезон року із вказанням поточного, попереднього і наступного сезонів від поточного: |
| «WINTER» |
| 2 | Тестування виведення даних про один сезон року із вказанням поточного, попереднього і наступного сезонів від поточного: |
| «SUMMER» |
| 3 | Тестування виведення переліку всіх сезонів року: |
| «WINTER» |
| «SPRING» |
| «SUMMER» |
| «AUTUMN» |

Очікуваний результат: Успішне завершення виконання програми. Виведення у консоль повідомлення з результатами:

|  |
| --- |
| Current season:  The season 'Зима' has ordinal number 0.  Previous season:  The season 'Осінь' has ordinal number 3.  Next season:  The season 'Весна' has ordinal number 1.  Current season:  The season 'Літо' has ordinal number 2.  Previous season:  The season 'Весна' has ordinal number 1.  Next season:  The season 'Осінь' has ordinal number 3.  All seasons:  The season 'Зима' has ordinal number 0.  The season 'Весна' has ordinal number 1.  The season 'Літо' has ordinal number 2.  The season 'Осінь' has ordinal number 3. |

**2.2.2 Тестовий випадок № 2** **“Тестування місяців року”**

Опис: Тестування виведення даних про один місяць року із вказанням поточного, попереднього і наступного місяців від поточного із тестуванням переходу на місяць наступного або попереднього року. І тестування виведення переліку всіх місяців року.

Тестові дані:

Таблиця 2.2 – Тестові дані для місяців року

|  |  |
| --- | --- |
| № з/п | Назва місяця з переліку |
| 1 | Тестування виведення даних про один місяць року із вказанням поточного, попереднього і наступного місяців від поточного: |
| «FEBRUARY» |
| 2 | Тестування виведення даних про один місяць року із вказанням поточного, попереднього і наступного місяців від поточного із тестуванням переходу на місяць наступного року: |
| «DECEMBER» |
|  | Тестування виведення даних про один місяць року із вказанням поточного, попереднього і наступного місяців від поточного із тестуванням переходу на місяць попереднього року: |
| «JANUARY» |
| 3 | Тестування виведення переліку всіх місяців року: |
| «JANUARY» |
| «FEBRUARY» |
| «MARCH» |
| «APRIL» |
| «MAY» |
| «JUNE» |
| «JULY» |
| «AUGUST» |
| «SEPTEMBER» |
| «OCTOBER» |
| «NOVEMBER» |
| «DECEMBER» |

Очікуваний результат: Успішне завершення виконання програми. Виведення у консоль повідомлення з результатами:

|  |
| --- |
| Current month:  The month of 'Лютий' has 28 days and ordinal number 1.  Season of the month: The season 'Зима' has ordinal number 0.  Previous month:  The month of 'Січень' has 31 days and ordinal number 0.  Season of the month: The season 'Зима' has ordinal number 0.  Next month:  The month of 'Березень' has 31 days and ordinal number 2.  Season of the month: The season 'Весна' has ordinal number 1.  Current month:  The month of 'Грудень' has 31 days and ordinal number 11.  Season of the month: The season 'Зима' has ordinal number 0.  Previous month:  The month of 'Листопад' has 30 days and ordinal number 10.  Season of the month: The season 'Осінь' has ordinal number 3.  Next month:  The month of 'Січень' has 31 days and ordinal number 0.  Season of the month: The season 'Зима' has ordinal number 0.  Current month:  The month of 'Січень' has 31 days and ordinal number 0.  Season of the month: The season 'Зима' has ordinal number 0.  Previous month:  The month of 'Грудень' has 31 days and ordinal number 11.  Season of the month: The season 'Зима' has ordinal number 0.  Next month:  The month of 'Лютий' has 28 days and ordinal number 1.  Season of the month: The season 'Зима' has ordinal number 0.  All months:  The month of 'Січень' has 31 days and ordinal number 0.  Season of the month: The season 'Зима' has ordinal number 0.  The month of 'Лютий' has 28 days and ordinal number 1.  Season of the month: The season 'Зима' has ordinal number 0.  The month of 'Березень' has 31 days and ordinal number 2.  Season of the month: The season 'Весна' has ordinal number 1.  The month of 'Квітень' has 30 days and ordinal number 3.  Season of the month: The season 'Весна' has ordinal number 1.  The month of 'Травень' has 31 days and ordinal number 4.  Season of the month: The season 'Весна' has ordinal number 1.  The month of 'Червень' has 30 days and ordinal number 5.  Season of the month: The season 'Літо' has ordinal number 2.  The month of 'Липень' has 31 days and ordinal number 6.  Season of the month: The season 'Літо' has ordinal number 2.  The month of 'Серпень' has 31 days and ordinal number 7.  Season of the month: The season 'Літо' has ordinal number 2.  The month of 'Вересень' has 30 days and ordinal number 8.  Season of the month: The season 'Осінь' has ordinal number 3.  The month of 'Жовтень' has 31 days and ordinal number 9.  Season of the month: The season 'Осінь' has ordinal number 3.  The month of 'Листопад' has 30 days and ordinal number 10.  Season of the month: The season 'Осінь' has ordinal number 3.  The month of 'Грудень' has 31 days and ordinal number 11.  Season of the month: The season 'Зима' has ordinal number 0. |

**2.3 Програмний код завдання № 2**

Лістинг 2.1 - Програмний код реалізований у файлі “Season.java”

|  |
| --- |
| package task2;  public enum Season {  *WINTER*,  *SPRING*,  *SUMMER*,  *AUTUMN*;   public String getSeasonNameUA() {  return switch (this) {  case *WINTER* -> "Зима";  case *SPRING* -> "Весна";  case *SUMMER* -> "Літо";  case *AUTUMN* -> "Осінь";  default -> "Невідомо";  };  }   Season next() {  return (*values*())[(ordinal() + 1) % *values*().length];  }   Season previous() {  return (*values*())[(ordinal() + 3) % *values*().length];  }   @Override  public String toString() {  return "\tThe season\t\'" + getSeasonNameUA() + "\'\thas ordinal number " + ordinal() + ".";  }   public static void printAllSeasons() {  System.*out*.println("All seasons:");   for (Season s : *values*()) {  System.*out*.println(s);  }   System.*out*.println();  } } |

Лістинг 2.2 - Програмний код реалізований у файлі “Month.java”

|  |
| --- |
| package task2;  public enum Month {  *JANUARY*(31),  *FEBRUARY*(28),  *MARCH*(31),  *APRIL*(30),  *MAY*(31),  *JUNE*(30),  *JULY*(31),  *AUGUST*(31),  *SEPTEMBER*(30),  *OCTOBER*(31),  *NOVEMBER*(30),  *DECEMBER*(31);   private final int daysPerMonth;   Month(int daysPerMonth) {  this.daysPerMonth = daysPerMonth;  }   public int getDaysPerMonth() {  return daysPerMonth;  }   public String getMonthNameUA() {  return switch (this) {  case *JANUARY* -> "Січень";  case *FEBRUARY* -> "Лютий";  case *MARCH* -> "Березень";  case *APRIL* -> "Квітень";  case *MAY* -> "Травень";  case *JUNE* -> "Червень";  case *JULY* -> "Липень";  case *AUGUST* -> "Серпень";  case *SEPTEMBER* -> "Вересень";  case *OCTOBER* -> "Жовтень";  case *NOVEMBER* -> "Листопад";  case *DECEMBER* -> "Грудень";  default -> "Невідомо";  };  }   Month next() {  return (*values*())[(ordinal() + 1) % *values*().length];  }   Month previous() {  return (*values*())[(ordinal() + 11) % *values*().length];  }   public Season getSeason() {  return switch (this) {  case *DECEMBER*, *JANUARY*, *FEBRUARY* -> Season.*WINTER*;  case *MARCH*, *APRIL*, *MAY* -> Season.*SPRING*;  case *JUNE*, *JULY*, *AUGUST* -> Season.*SUMMER*;  case *SEPTEMBER*, *OCTOBER*, *NOVEMBER* -> Season.*AUTUMN*;  default -> null;  };  }   @Override  public String toString() {  return "\tThe month of\t\'" + getMonthNameUA() + "\' \thas " + getDaysPerMonth()  + " days and ordinal number " + ordinal()  + ".\n\tSeason of the month:" + getSeason();  }   public static void printAllMonths() {  System.*out*.println("All months:");   for (Month m : *values*()) {  System.*out*.println(m);  }   System.*out*.println();  } } |

Лістинг 2.3 - Програмний код реалізований у файлі “EnumTest.java”

|  |
| --- |
| package task2;  public class EnumTest {  private static void testSeason(Season season) {  System.*out*.println("Current season:\n" + season);  System.*out*.println("Previous season:\n" + season.previous());  System.*out*.println("Next season:\n" + season.next() + "\n");  }  private static void testSeasons() {  System.*out*.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  System.*out*.println("TEST SEASONS:");  System.*out*.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  *testSeason*(Season.*WINTER*);  *testSeason*(Season.*SUMMER*);  Season.*printAllSeasons*();  }   private static void testMonth(Month month) {  System.*out*.println("Current month:\n" + month);  System.*out*.println("Previous month:\n" + month.previous());  System.*out*.println("Next month:\n" + month.next() + "\n");  }   private static void testMonths() {  System.*out*.println("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  System.*out*.println("TEST MONTHS:");  System.*out*.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  *testMonth*(Month.*FEBRUARY*);  *testMonth*(Month.*DECEMBER*);  *testMonth*(Month.*JANUARY*);  Month.*printAllMonths*();  }   public static void main(String[] args) {  *testSeasons*();  *testMonths*();  } } |

**2.4 Екранні форми за результатами роботи програмного коду завдання № 2**

Изображение выглядит как текст, снимок экрана, Шрифт, меню

Автоматически созданное описание

Рисунок 2.1 - Результати роботи програмного коду відповідно до тестового випадку № 1 “Тестування сезонів року”

Изображение выглядит как текст, снимок экрана, меню, Шрифт

Автоматически созданное описание

Рисунок 2.2 - Результати роботи програмного коду відповідно до тестового випадку № 2 “Тестування місяців року” частина 1

Изображение выглядит как текст, снимок экрана, черно-белый, Шрифт

Автоматически созданное описание

Рисунок 2.3 - Результати роботи програмного коду відповідно до тестового випадку № 2 “Тестування місяців року” частина 2

**3.1 Завдання № 3 “Створення бібліотеки узагальнених функцій для роботи з масивами”**

Створити клас зі статичними узагальненими методами, які реалізують таку функціональність:

* обмін місцями двох груп елементів
* обмін місцями усіх пар сусідніх елементів (з парним і непарним індексом)
* заміна групи елементів іншим масивом елементів

Здійснити демонстрацію роботи усіх методів з використанням даних різних типів (Integer, Double, String) .

**3.2 Набір тестових даних завдання № 3**

**3.2.1 Тестовий випадок №** **1** **“Тестування з типом** **Integer”**

Опис: Тестування статичних узагальнених функцій для типу Integer.

Тестові дані типу Integer:

Таблиця 3.1 – Тестові дані з типом Integer

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер та розмір масиву | Значення елементів тестових масивів даних відповідно до індексу елемента в масиві | | | | |
| 0 | 1 | 2 | 3 | 4 |
| Масив 1, 5 елементів | 5 | 3 | 1 | 0 | -4 |
| Масив 2, 1 елемент | 7 |  |  |  |  |
| Масив 3, 3 елементи | 4 | -6 | -7 |  |  |

* Обмін місцями двох груп елементів:
  + Невалідний індекс масиву, куди копіюються дані, та валідні індекси масиву, звідки копіюються дані:
    - srcBegin = 0,
    - srcEnd = 1,
    - dstBegin = -3.
  + Валідні індекси масивів:
    - srcBegin = 1;
    - srcEnd = 3;
    - dstBegin = 2.
* Обмін місцями усіх пар сусідніх елементів (з парним і непарним індексом).
* Заміна групи елементів іншим масивом елементів:
  + Невалідний індекс масиву, звідки копіюються дані, та валідні індекси масиву, куди копіюються дані:
    - srcBegin = 1,
    - srcEnd = 0,
    - dstBegin = 2.
  + Валідні індекси масивів:
    - srcBegin = 0;
    - srcEnd = 0;
    - dstBegin = 3.

Очікуваний результат: Успішне завершення виконання програми. Виведення у консоль повідомлення з результатами:

|  |
| --- |
| Initial first array:  [5, 3, 1, 0, -4]  Test functionality of swapping two groups of elements,  with srcBegin=0 srcEnd=1 dstBegin=-3: false  First array after swapping:  [5, 3, 1, 0, -4]  Test functionality of swapping two groups of elements,  with srcBegin=1 srcEnd=3 dstBegin=2: true  First array after swapping:  [5, 1, 0, -4, 3]  Test swapping of all pairs of adjacent elements  (with even and odd index) for first array: true  First array after swapping:  [1, 5, -4, 0, 3]  Initial second array:  [7]  Test swapping of all pairs of adjacent elements  (with even and odd index) for second array: false  Second array after swapping:  [7]  Initial third array:  [4, -6, -7]  Test swapping of all pairs of adjacent elements  (with even and odd index) for third array: true  Third array after swapping:  [-6, 4, -7]  Test replacing a group of elements of first array with elements of second array  with srcBegin=1 dstBegin=0 dstEnd=2: false  First array after replacing:  [1, 5, -4, 0, 3]  Second array after replacing:  [7]  Test replacing a group of elements of first array with elements of third array  with srcBegin=0 dstBegin=0 dstEnd=3: true  First array after replacing:  [-6, 4, -7, 0, 3]  Third array after replacing:  [-6, 4, -7] |

**3.2.2 Тестовий випадок № 2****“Тестування з типом Double”**

Опис: Тестування статичних узагальнених функцій для типу Double.

Тестові дані типу Double:

Таблиця 3.2 – Тестові дані з типом Double

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер та розмір масиву | Значення елементів тестових масивів даних відповідно до індексу елемента в масиві | | | | |
| 0 | 1 | 2 | 3 | 4 |
| Масив 1, 5 елементів | 5.9 | -3.5 | 1.8 | 1.6 | -2.7 |
| Масив 2, 1 елемент | 2.6 |  |  |  |  |
| Масив 3, 3 елементи | 4.8 | -6.9 | -7.9 |  |  |

* Обмін місцями двох груп елементів:
  + Невалідний індекс масиву, куди копіюються дані, та валідні індекси масиву, звідки копіюються дані:
    - srcBegin = 0,
    - srcEnd = 1,
    - dstBegin = -3.
  + Валідні індекси масивів:
    - srcBegin = 1;
    - srcEnd = 3;
    - dstBegin = 2.
* Обмін місцями усіх пар сусідніх елементів (з парним і непарним індексом).
* Заміна групи елементів іншим масивом елементів:
  + Невалідний індекс масиву, звідки копіюються дані, та валідні індекси масиву, куди копіюються дані:
    - srcBegin = 1,
    - srcEnd = 0,
    - dstBegin = 2.
  + Валідні індекси масивів:
    - srcBegin = 0;
    - srcEnd = 0;
    - dstBegin = 3.

Очікуваний результат: Успішне завершення виконання програми. Виведення у консоль повідомлення з результатами:

|  |
| --- |
| Initial first array:  [5.9, -3.5, 1.8, 1.6, -2.7]  Test functionality of swapping two groups of elements,  with srcBegin=0 srcEnd=1 dstBegin=-3: false  First array after swapping:  [5.9, -3.5, 1.8, 1.6, -2.7]  Test functionality of swapping two groups of elements,  with srcBegin=1 srcEnd=3 dstBegin=2: true  First array after swapping:  [5.9, 1.8, 1.6, -2.7, -3.5]  Test swapping of all pairs of adjacent elements  (with even and odd index) for first array: true  First array after swapping:  [1.8, 5.9, -2.7, 1.6, -3.5]  Initial second array:  [2.6]  Test swapping of all pairs of adjacent elements  (with even and odd index) for second array: false  Second array after swapping:  [2.6]  Initial third array:  [4.8, -6.9, -7.9]  Test swapping of all pairs of adjacent elements  (with even and odd index) for third array: true  Third array after swapping:  [-6.9, 4.8, -7.9]  Test replacing a group of elements of first array with elements of second array  with srcBegin=1 dstBegin=0 dstEnd=2: false  First array after replacing:  [1.8, 5.9, -2.7, 1.6, -3.5]  Second array after replacing:  [2.6]  Test replacing a group of elements of first array with elements of third array  with srcBegin=0 dstBegin=0 dstEnd=3: true  First array after replacing:  [-6.9, 4.8, -7.9, 1.6, -3.5]  Third array after replacing:  [-6.9, 4.8, -7.9] |

**3.2.3 Тестовий випадок № 3** **“Тестування з типом String”**

Опис: Тестування статичних узагальнених функцій для типу String.

Тестові дані типу String:

Таблиця 3.3 – Тестові дані з типом String

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер та розмір масиву | Значення елементів тестових масивів даних відповідно до індексу елемента в масиві | | | | |
| 0 | 1 | 2 | 3 | 4 |
| Масив 1, 5 елементів | "Apple" | "Banana" | "Orange" | "Grape" | "Mango" |
| Масив 2, 1 елемент | "Peach" |  |  |  |  |
| Масив 3, 3 елементи | "Kiwi" | "Lemon" | "Plum" |  |  |

* Обмін місцями двох груп елементів:
  + Невалідний індекс масиву, куди копіюються дані, та валідні індекси масиву, звідки копіюються дані:
    - srcBegin = 0,
    - srcEnd = 1,
    - dstBegin = -3.
  + Валідні індекси масивів:
    - srcBegin = 1;
    - srcEnd = 3;
    - dstBegin = 2.
* Обмін місцями усіх пар сусідніх елементів (з парним і непарним індексом).
* Заміна групи елементів іншим масивом елементів:
  + Невалідний індекс масиву, звідки копіюються дані, та валідні індекси масиву, куди копіюються дані:
    - srcBegin = 1,
    - srcEnd = 0,
    - dstBegin = 2.
  + Валідні індекси масивів:
    - srcBegin = 0;
    - srcEnd = 0;
    - dstBegin = 3.

Очікуваний результат: Успішне завершення виконання програми. Виведення у консоль повідомлення з результатами:

|  |
| --- |
| Initial first array:  [Apple, Banana, Orange, Grape, Mango]  Test functionality of swapping two groups of elements,  with srcBegin=0 srcEnd=1 dstBegin=-3: false  First array after swapping:  [Apple, Banana, Orange, Grape, Mango]  Test functionality of swapping two groups of elements,  with srcBegin=1 srcEnd=3 dstBegin=2: true  First array after swapping:  [Apple, Orange, Grape, Mango, Banana]  Test swapping of all pairs of adjacent elements  (with even and odd index) for first array: true  First array after swapping:  [Orange, Apple, Mango, Grape, Banana]  Initial second array:  [Peach]  Test swapping of all pairs of adjacent elements  (with even and odd index) for second array: false  Second array after swapping:  [Peach]  Initial third array:  [Kiwi, Lemon, Plum]  Test swapping of all pairs of adjacent elements  (with even and odd index) for third array: true  Third array after swapping:  [Lemon, Kiwi, Plum]  Test replacing a group of elements of first array with elements of second array  with srcBegin=1 dstBegin=0 dstEnd=2: false  First array after replacing:  [Orange, Apple, Mango, Grape, Banana]  Second array after replacing:  [Peach]  Test replacing a group of elements of first array with elements of third array  with srcBegin=0 dstBegin=0 dstEnd=3: true  First array after replacing:  [Lemon, Kiwi, Plum, Grape, Banana]  Third array after replacing:  [Lemon, Kiwi, Plum] |

**3.3 Програмний код завдання № 3**

Лістинг 3.1 - Програмний код реалізований у файлі “ArrayOperations.java”

|  |
| --- |
| package task3;  public class ArrayOperations {  private ArrayOperations() {}   private static <T> void swapElementsByIndex(T[] array, int srcIndex, int dstIndex) {  T temp = array[srcIndex];  array[srcIndex] = array[dstIndex];  array[dstIndex] = temp;  }   public static <T> boolean swapGroupsOfElements(int srcBegin, int srcEnd, T[] array, int dstBegin) {  if (srcBegin < 0  || dstBegin < 0  || srcEnd <= 0  || array.length < srcBegin + srcEnd  || array.length < dstBegin + srcEnd  || srcBegin == dstBegin) {  return false;  }   for (int i = 0; i < srcEnd; i++) {  *swapElementsByIndex*(array, srcBegin + i, dstBegin + i);  }   return true;  }   public static <T> boolean swapAdjacentElements(T[] array) {  if (array.length <= 1) {  return false;  }   for (int i = 0; i < array.length - 1; i += 2) {  *swapElementsByIndex*(array, i, i + 1);  }   return true;  }   public static <T> boolean replaceGroupOfElements(T[] src, int srcBegin, T[] dst, int dstBegin, int dstEnd) {  if (dstBegin < 0  || srcBegin < 0  || dstEnd <= 0  || dst.length < dstBegin + dstEnd  || src.length < srcBegin + dstEnd) {  return false;  }   for (int i = 0; i < dstEnd; i++) {  dst[dstBegin + i] = src[srcBegin + i];  }   return true;  } } |

Лістинг 3.2 - Програмний код реалізований у файлі “ArrayOperationsTest.java”

|  |
| --- |
| package task3;  import java.util.Arrays;  public class ArrayOperationsTest {  private static <T> void printArray(T[] array) {  System.*out*.println(Arrays.*toString*(array));  }   private static <T> void testSwapGroupsOfElements(T[] firstArray) {  System.*out*.println("Initial first array:");  *printArray*(firstArray);  System.*out*.println();   int srcBegin = 0, srcEnd = 1, dstBegin = -3;  System.*out*.println("Test functionality of swapping two groups of elements,\n"  + "with srcBegin=" + srcBegin + "\t\tsrcEnd=" + srcEnd + "\tdstBegin=" + dstBegin + ":\t"  + ArrayOperations.*swapGroupsOfElements*(srcBegin, srcEnd, firstArray, dstBegin));  System.*out*.println("First array after swapping:");  *printArray*(firstArray);  System.*out*.println();   srcBegin = 1; srcEnd = 3; dstBegin = 2;  System.*out*.println("Test functionality of swapping two groups of elements,\n"  + "with srcBegin=" + srcBegin + "\t\tsrcEnd=" + srcEnd + "\tdstBegin=" + dstBegin + ": \t"  + ArrayOperations.*swapGroupsOfElements*(srcBegin, srcEnd, firstArray, dstBegin));  System.*out*.println("First array after swapping:");  *printArray*(firstArray);  System.*out*.println();  }   private static <T> void testSwapAdjacentElements(T[] firstArray, T[] secondArray, T[] thirdArray) {  System.*out*.println("Test swapping of all pairs of adjacent elements\n"  + "(with even and odd index) for first array:\t" + ArrayOperations.*swapAdjacentElements*(firstArray));  System.*out*.println("First array after swapping:");  *printArray*(firstArray);  System.*out*.println();   System.*out*.println("Initial second array:");  *printArray*(secondArray);  System.*out*.println();   System.*out*.println("Test swapping of all pairs of adjacent elements\n"  + "(with even and odd index) for second array:\t" + ArrayOperations.*swapAdjacentElements*(secondArray));  System.*out*.println("Second array after swapping:");  *printArray*(secondArray);  System.*out*.println();   System.*out*.println("Initial third array:");  *printArray*(thirdArray);  System.*out*.println();   System.*out*.println("Test swapping of all pairs of adjacent elements\n"  + "(with even and odd index) for third array:\t" + ArrayOperations.*swapAdjacentElements*(thirdArray));  System.*out*.println("Third array after swapping:");  *printArray*(thirdArray);  System.*out*.println();  }   private static <T> void testReplaceGroupOfElements(T[] firstArray, T[] secondArray, T[] thirdArray) {  int srcBegin = 1, dstBegin = 0, dstEnd = 2;  System.*out*.println("Test replacing a group of elements of first array with elements of second array\n"  + "with srcBegin=" + srcBegin + "\tdstBegin=" + dstBegin + "\tdstEnd=" + dstEnd + ":\t"  + ArrayOperations.*replaceGroupOfElements*(secondArray, 1, firstArray, 0, 2));  System.*out*.println("First array after replacing:");  *printArray*(firstArray);  System.*out*.println("Second array after replacing:");  *printArray*(secondArray);  System.*out*.println();   srcBegin = 0; dstBegin = 0; dstEnd = 3;  System.*out*.println("Test replacing a group of elements of first array with elements of third array\n"  + "with srcBegin=" + srcBegin + "\tdstBegin=" + dstBegin + "\tdstEnd=" + dstEnd + ":\t"  + ArrayOperations.*replaceGroupOfElements*(thirdArray, srcBegin, firstArray, dstBegin, dstEnd));  System.*out*.println("First array after replacing:");  *printArray*(firstArray);  System.*out*.println("Third array after replacing:");  *printArray*(thirdArray);  System.*out*.println();  }   private static <T> void testOperations(T[] firstArray, T[] secondArray, T[] thirdArray) {  *testSwapGroupsOfElements*(firstArray);  *testSwapAdjacentElements*(firstArray, secondArray, thirdArray);  *testReplaceGroupOfElements*(firstArray, secondArray, thirdArray);  }  private static <T> void testOperationsWithInteger() {  System.*out*.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  System.*out*.println("Test ArrayOperations class generic functions for Integer type:");  System.*out*.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  Integer[] firstArrayOfIntegers = { 5, 3, 1, 0, -4};  Integer[] secondArrayOfIntegers = {7};  Integer[] thirdArrayOfIntegers = {4, -6, -7};  *testOperations*(firstArrayOfIntegers, secondArrayOfIntegers, thirdArrayOfIntegers);  System.*out*.println();  }   private static <T> void testOperationsWithDouble() {  System.*out*.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  System.*out*.println("Test ArrayOperations class generic functions for Double type:");  System.*out*.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  Double[] firstArrayOfDoubles = {5.9, -3.5, 1.8, 1.6, -2.7};  Double[] secondArrayOfDoubles = {2.6};  Double[] thirdArrayOfDoubles = {4.8, -6.9, -7.9};  *testOperations*(firstArrayOfDoubles, secondArrayOfDoubles, thirdArrayOfDoubles);  System.*out*.println();  }   private static <T> void testOperationsWithString( ) {  System.*out*.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  System.*out*.println("Test ArrayOperations class generic functions for String type:");  System.*out*.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  String[] firstArrayOfStrings = {"Apple", "Banana", "Orange", "Grape", "Mango"};  String[] secondArrayOfStrings = {"Peach"};  String[] thirdArrayOfStrings = {"Kiwi", "Lemon", "Plum"};  *testOperations*(firstArrayOfStrings, secondArrayOfStrings, thirdArrayOfStrings);  System.*out*.println();  }   public static void main(String[] args) {  *testOperationsWithInteger*();  *testOperationsWithDouble*();  *testOperationsWithString*();  } } |

**3.4 Екранні форми за результатами роботи програмного коду завдання № 3**

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 3.1 - Результати роботи програмного коду відповідно до тестового випадку № 1 “Тестування з типом Integer” частина 1

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 3.2 - Результати роботи програмного коду відповідно до тестового випадку № 2 “Тестування з типом Integer” частина 2

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 3.3 - Результати роботи програмного коду відповідно до тестового випадку № 2 “Тестування з типом Double” частина 1

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 3.4 - Результати роботи програмного коду відповідно до тестового випадку № 2 “Тестування з типом Double” частина 2

Изображение выглядит как текст, снимок экрана, Шрифт, меню

Автоматически созданное описание

Рисунок 3.5 - Результати роботи програмного коду відповідно до тестового випадку № 3 “Тестування з типом String” частина 1

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 3.6 - Результати роботи програмного коду відповідно до тестового випадку № 3 “Тестування з типом String” частина 2

## **4.1 Завдання № 4 “Створення бібліотеки узагальнених функцій для роботи зі списком чисел”**

Створити клас, який містить статичні узагальнені функції для реалізації таких дій зі списком чисел (об'єктів, похідних від Number):

* знаходження індексу першого нульового елемента
* визначення кількості від'ємних чисел
* повернення останнього від'ємного елементу

Здійснити тестування всіх функцій на списках чисел різних типів.

**4.2 Набір тестових даних завдання № 4**

**4.2.1 Тестовий випадок № 1 “Тестування з типом Integer”**

Опис: Тестування статичних узагальнених функцій для типу Integer:

* знаходження індексу першого нульового елемента
* визначення кількості від'ємних чисел
* повернення останнього від'ємного елементу

Тестові дані типу Integer:

Таблиця 4.1 – Тестові дані з типом Integer

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер та розмір списку | Значення елементів тестових списків даних відповідно до індексу елемента у списку | | | | |
| 0 | 1 | 2 | 3 | 4 |
| Список 1, 5 елементів | -2 | 0 | 1 | -3 | -5 |
| Список 2, 4 елементи | 4 | 8 | 3 | 5 |  |

Очікуваний результат: Успішне завершення виконання програми. Виведення у консоль повідомлення з результатами:

|  |
| --- |
| 1. Test for first List of Integers:  Initial List:  [-2, 0, 1, -3, -5]  Test finding the index of the first zero element: 1  Test finding the count of negative numbers: 3  Test getting the last negative element: -5  2. Test for second List of Integers:  Initial List:  [4, 8, 2, 3, 5]  Test finding the index of the first zero element: -1  Test finding the count of negative numbers: 0  Test getting the last negative element: No negative elements found. |

**4.2.2 Тестовий випадок № 2 “Тестування з типом Double”**

Опис: Тестування статичних узагальнених функцій для типу Double:

* знаходження індексу першого нульового елемента
* визначення кількості від'ємних чисел
* повернення останнього від'ємного елементу

Тестові дані типу Double:

Таблиця 4.2 – Тестові дані з типом Double

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер та розмір списку | Значення елементів тестових списків даних відповідно до індексу елемента у списку | | | | |
| 0 | 1 | 2 | 3 | 4 |
| Список 1, 5 елементів | -2.8 | 0.0 | 2.2 | -3.0 | -6.9 |
| Список 2, 4 елементи | 3.2 | 4.5 | 1.0 | 3.5 | 2.8 |

Очікуваний результат: Успішне завершення виконання програми. Виведення у консоль повідомлення з результатами:

|  |
| --- |
| 1. Test for first List of Doubles:  Initial List:  [-2.8, 0.0, 2.2, -3.0, -6.9]  Test finding the index of the first zero element: 1  Test finding the count of negative numbers: 3  Test getting the last negative element: -6.9  2. Test for second List of Doubles:  Initial List:  [3.2, 4.5, 1.0, 3.5, 2.8]  Test finding the index of the first zero element: -1  Test finding the count of negative numbers: 0  Test getting the last negative element: No negative elements found. |

**4.3 Програмний код завдання № 4**

Лістинг 4.1 - Програмний код реалізований у файлі “NumberListOperations.java”

|  |
| --- |
| package task4;  import java.util.List; import java.util.Optional;  public class NumberListOperations {  private NumberListOperations() {}   public static <T extends Number> int findIndexOfFirstZeroNumber(List<T> numberList) {  for (int i = 0; i < numberList.size(); i++) {  if (numberList.get(i).doubleValue() == 0.0) {  return i;  }  }   return -1;  }   public static <T extends Number> int findCountOfNegativeNumbers(List<T> numberList) {  int countOfNegativeNumbers = 0;   for (T number : numberList) {  if (number.doubleValue() < 0.0) {  countOfNegativeNumbers++;  }  }   return countOfNegativeNumbers;  }   public static <T extends Number> Optional<T> getLastNegativeNumber(List<T> numberList) {  T lastNegativeNumber = null;   for (T number : numberList) {  if (number.doubleValue() < 0.0) {  lastNegativeNumber = number;  }  }   return Optional.*ofNullable*(lastNegativeNumber);  } } |

Лістинг 4.2 - Програмний код реалізований у файлі “NumberListOperationsTest.java”

|  |
| --- |
| package task4;  import java.util.List; import java.util.Optional;  public class NumberListOperationsTest {  private static <T extends Number> void printNumberList(List<T> list) {  System.*out*.println(list);  }   private static <T extends Number> void printLastNegativeNumber(Optional<T> lastNegativeNumber) {  if (lastNegativeNumber.isPresent()) {  System.*out*.print(lastNegativeNumber.get());  } else {  System.*out*.println("No negative elements found.");  }  }   private static <T extends Number> void testNumberListOperations(List<T> numberList) {  System.*out*.println("Initial List:");  *printNumberList*(numberList);   System.*out*.println("Test finding the index of the first zero element:\t"  + NumberListOperations.*findIndexOfFirstZeroNumber*(numberList));   System.*out*.println("Test finding the count of negative numbers:\t\t\t"  + NumberListOperations.*findCountOfNegativeNumbers*(numberList));   Optional<T> lastNegativeNumber = NumberListOperations.*getLastNegativeNumber*(numberList);  System.*out*.print("Test getting the last negative element:\t\t\t\t");  *printLastNegativeNumber*(lastNegativeNumber);  System.*out*.println("\n");  }   private static <T extends Number> void testOperationsWithInteger() {  System.*out*.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  System.*out*.println("Test NumberListOperations class generic functions for Integer type:");  System.*out*.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  List<Integer> firstListOfIntegers = List.*of*(-2, 0, 1, -3, -5);  List<Integer> secondListOfIntegers = List.*of*(4, 8, 2, 3, 5);  System.*out*.println("1. Test for first List of Integers:");  *testNumberListOperations*(firstListOfIntegers);  System.*out*.println("2. Test for second List of Integers:");  *testNumberListOperations*(secondListOfIntegers);  }   private static <T extends Number> void testOperationsWithDouble() {  System.*out*.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  System.*out*.println("Test NumberListOperations class generic functions for Double type:");  System.*out*.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  List<Double> firstListOfDoubles = List.*of*(-2.8, 0.0, 2.2, -3.0, -6.9);  List<Double> secondListOfDoubles = List.*of*(3.2, 4.5, 1.0, 3.5, 2.8);  System.*out*.println("1. Test for first List of Doubles:");  *testNumberListOperations*(firstListOfDoubles);  System.*out*.println("2. Test for second List of Doubles:");  *testNumberListOperations*(secondListOfDoubles);  }   public static void main(String[] args) {  *testOperationsWithInteger*();  *testOperationsWithDouble*();  } } |

**4.4 Екранні форми за результатами роботи програмного коду завдання № 4**

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 4.1 - Результати роботи програмного коду відповідно до тестового випадку № 1 “Тестування з типом Integer”

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 4.2 - Результати роботи програмного коду відповідно до тестового випадку № 2 “Тестування з типом Double”

## **5.1 Завдання № 5 “Пошук різних слів у реченні”**

Увести речення, створити колекцію (SortedSet) різних слів речення та вивести ці слова в алфавітному порядку.

**5.2 Набір тестових даних завдання № 5**

**5.2.1 Тестовий випадок № 1 “Речення зі словами”**

Опис: Тестування виводу слів речення у алфавітному порядку для речення зі звичайними словами.

Тестові дані:

* Речення: «"Quick brown fox jumps over the lazy dog."»

Очікуваний результат: Успішне завершення виконання програми. Виведення у консоль повідомлення з результатами:

|  |
| --- |
| [brown, dog, fox, jumps, lazy, over, quick, the] |

**5.2.2 Тестовий випадок № 2 “Речення зі словами і символами”**

Опис: Тестування виводу слів речення у алфавітному порядку для речення зі словами і різними символами, які можуть бути роздільниками.

Тестові дані:

* Речення: «Your password "A .,:;?!-()@’”b c d z y x" is not good enough to protect your account!»

Очікуваний результат: Успішне завершення виконання програми. Виведення у консоль повідомлення з результатами:

|  |
| --- |
| [@’”b, a, account, c, d, enough, good, is, not, password, protect, to, x, y, your, z, »] |

**5.2.3 Тестовий випадок № 3 “Пусте речення”**

Опис: Тестування виводу слів речення у алфавітному порядку для речення без слів і знаків.

Тестові дані:

* Речення: «»

Очікуваний результат: Успішне завершення виконання програми. Виведення у консоль повідомлення з результатами:

|  |
| --- |
| [] |

**5.3 Програмний код завдання № 5**

Лістинг 5.1 - Програмний код реалізований у файлі “Circle.java”

|  |
| --- |
| package task5;  import java.util.Set; import java.util.HashSet; import java.util.SortedSet; import java.util.TreeSet; import java.util.Scanner; import java.util.Arrays;  public class SentenceProcessor {  public static SortedSet<String> processSentence(String sentence) {  Set<Character> delimiters = new HashSet<>(  Arrays.*asList*(' ', '.', ',', ':', ';', '?', '!', '-', '(', ')', '\"', '\''));   SortedSet<String> words = new TreeSet<>();  StringBuilder currentWord = new StringBuilder();   for (int i = 0; i < sentence.length(); i++) {  char currentChar = sentence.charAt(i);   if (!delimiters.contains(currentChar)) {  currentWord.append(currentChar);  } else {  if (!currentWord.isEmpty()) {  words.add(currentWord.toString().toLowerCase());  currentWord.setLength(0);  }  }  }   if (!currentWord.isEmpty()) {  words.add(currentWord.toString().toLowerCase());  }   return words;  }   public static void main(String[] args) {  Scanner scanner = new Scanner(System.*in*);  System.*out*.print("Enter a sentence:\t");  String inputSentence = scanner.nextLine();   SortedSet<String> result = *processSentence*(inputSentence);  System.*out*.println(result);  } } |

**5.4 Екранні форми за результатами роботи програмного коду завдання № 5**

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

Рисунок 5.1 - Результати роботи програмного коду відповідно до тестового випадку № 1 “Речення зі словами”

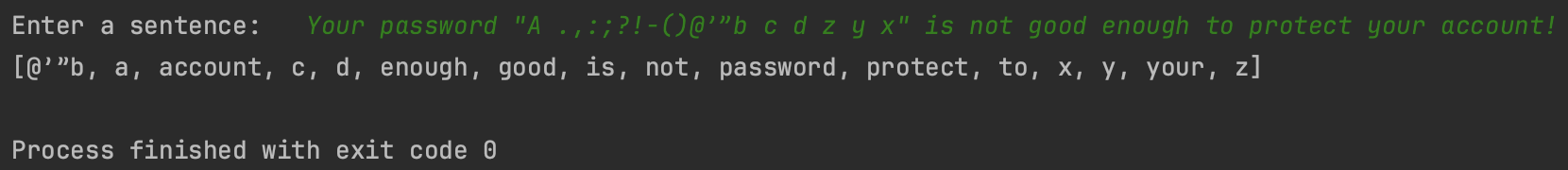


Рисунок 5.2 - Результати роботи програмного коду відповідно до тестового випадку № 2 “Речення зі словами і символами”

Изображение выглядит как текст, Шрифт, снимок экрана, черный

Автоматически созданное описание

Рисунок 5.3 - Результати роботи програмного коду відповідно до тестового випадку № 3 “Пусте речення”

## **6.1 Завдання № 6 “Дані про користувачів”**

Представити дані про користувачів у вигляді асоціативного масиву (ім'я / пароль) з припущенням, що всі імена користувачів різні. Вивести дані про користувачів з довжиною пароля більше 6 символів.

**6.2 Набір тестових даних завдання № 6**

**6.2.1 Тестовий випадок № 1 “****Дані користувачів”**

Опис: Тестування пошуку даних в асоціативному масиві користувачів з правилом довжини пароля більше 6 символів.

Тестові дані:

Таблиця 6.1 – Дані користувачів

|  |  |  |
| --- | --- | --- |
| Індекс | Імʼя користувача | Пароль |
| 0 | “David” | “Lmn2” |
| 1 | “Henry” | “S3cretPwd” |
| 2 | “Frank” | “1a2b3c4d” |
| 3 | “Alice” | “Abcd5” |
| 4 | “Charlie” | “Pqr7890” |
| 5 | “Grace” | “Jkl0pqr” |
| 6 | “Isabel” | “9AbCdEfG” |
| 7 | “Emily” | “Ghi456” |
| 8 | “Jack” | “XYZ45678” |
| 9 | “Bob” | “Xyz123” |

Очікуваний результат: Успішне завершення виконання програми. Виведення у консоль повідомлення з результатами:

|  |
| --- |
| Users with passwords longer than 6 letters:  Username: 'Charlie' Password: 'Pqr7890'  Username: 'Frank' Password: '1a2b3c4d'  Username: 'Grace' Password: 'Jkl0pqr'  Username: 'Henry' Password: 'S3cretPwd'  Username: 'Isabel' Password: '9AbCdEfG'  Username: 'Jack' Password: 'XYZ45678' |

**6.3 Програмний код завдання № 6**

Лістинг 6.1 - Програмний код реалізований у файлі “UsersProcessor.java”

|  |
| --- |
| package task6;  import java.util.Map; import java.util.SortedMap; import java.util.TreeMap;  public class UsersProcessor {  public static SortedMap<String, String> createUsers() {  SortedMap<String, String> users = new TreeMap<>();  users.put("David", "Lmn2");  users.put("Henry", "S3cretPwd");  users.put("Frank", "1a2b3c4d");  users.put("Alice", "Abcd5");  users.put("Charlie", "Pqr7890");  users.put("Grace", "Jkl0pqr");  users.put("Isabel", "9AbCdEfG");  users.put("Emily", "Ghi456");  users.put("Jack", "XYZ45678");  users.put("Bob", "Xyz123");   return users;  }   public static void findUsersWithPasswordLengthMoreThanSix(SortedMap<String, String> users) {  if (users.isEmpty()) {  System.*out*.println("There are no users with passwords longer than 6 letters.");  }   System.*out*.println("Users with passwords longer than 6 letters:");   for (Map.Entry<?, String> entry : users.entrySet()) {  if (entry.getValue().length() > 6) {  System.*out*.println("\tUsername:\t\'" + entry.getKey() + "\' \tPassword:\t\'" + entry.getValue() + "\'");  }  }  }   public static void main(String[] args) {  *findUsersWithPasswordLengthMoreThanSix*(*createUsers*());  } } |

**6.4 Екранні форми за результатами роботи програмного коду завдання № 6**

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 6.2 - Результати роботи програмного коду відповідно до тестового випадку № 1 “Дані користувачів”

## **7.1 Завдання № 7 “ Створення власного контейнера на базі масиву (додаткове завдання)”**

Створити узагальнений клас для представлення одновимірного масиву, індекс елементів якого змінюється від певного значення from до значення to включно. Ці значення можуть бути як додатними, так і від'ємними. Клас повинен реалізовувати інтерфейс Collection. Доцільно використати клас AbstractList як базовий.

**7.2 Набір тестових даних завдання № 7**

**7.2.1 Тестовий випадок № 1** **“****Тестування приватних методів класу OneDimBoundedArray”**

Опис: Тестування приватних методів класу OneDimBoundedArray з типом Integer.

Тестові дані:

* Початковий індекс: 5
* Кінцевий індекс: 7
* Масив типу Integer: { 2, 7, 9 }
* Пошук індексу у масиві: 0
* Пошук індексу у масиві: 7

Очікуваний результат: Успішне завершення виконання програми. Виведення у консоль повідомлення з результатами:

|  |
| --- |
| Initial array of Integers with indexes from = 5 to = 7:  Elements: [2, 7, 9]  Verify that the element index 0 exists in the array: false  Verify that the element index 7 exists in the array: true |

**7.2.2 Тестовий випадок № 2** **“Тестування з типом Integer”**

Опис: Тестування класу OneDimBoundedArray з типом Integer з позитивними і негативними індексами елементів масиву.

Тестові дані:

Таблиця 7.1 – Тестові дані з типом Integer

|  |
| --- |
| Створення тестового масиву типу Integer з позитивними індексами елементів масиву:   * Початковий індекс: 20 * Кінцевий індекс: 26 * Масив № 1 типу Integer: {-1, 8, 12, 16, 20, 47, 29} |
| Отримання елементу масиву за валідним індексом:   * Індекс: 10 * Масив із даними: Масив № 1 {-1, 8, 12, 16, 20, 47, 29} |
| Отримання елементу масиву за невалідним індексом:   * Індекс: -10 * Масив із даними: Масив № 1 {-1, 8, 12, 16, 20, 47, 29} |
| Встановлення значення елементу масиву за валідним індексом:   * Індекс: 11 * Значення, яке потрібно встановити: 1 * Масив із даними: Масив № 1 {-1, 8, 12, 16, 20, 47, 29} |
| Встановлення значення елементу масиву за невалідним індексом:   * Індекс: -11 * Значення, яке потрібно встановити: 1 * Масив із даними: Масив № 1 {-1, 8, 12, 16, 20, 47, 29} |
| Вивід всіх елементів масиву із їх індексами, отримання довжини поточного масиву даних:   * Масив із даними: Масив № 1 {-1, 8, 12, 16, 20, 47, 29} |
| Отримання індексу елементу масиву, що містить значення, та отримання останнього індексу елементу масиву, якому відповідає задане значення:   * Значення, яке необхідно знайти: 1 |
| Створення тестового масиву типу Integer з негативними індексами елементів масиву:   * Початковий індекс: -14 * Кінцевий індекс: -8 * Масив № 1 типу Integer: {-1, 8, 12, 16, 20, 47, 29} |
| Отримання елементу масиву за невалідним індексом:   * Індекс: 10 * Масив із даними: Масив № 1 {-1, 8, 12, 16, 20, 47, 29} |
| Отримання елементу масиву за валідним індексом:   * Індекс: -10 * Масив із даними: Масив № 1 {-1, 8, 12, 16, 20, 47, 29} |
| Встановлення значення елементу масиву за невалідним індексом:   * Індекс: 11 * Значення, яке потрібно встановити: 16 * Масив із даними: Масив № 1 {-1, 8, 12, 16, 20, 47, 29} |
| Встановлення значення елементу масиву за невалідним індексом:   * Індекс: -11 * Значення, яке потрібно встановити: 16 * Масив із даними: Масив № 1 {-1, 8, 12, 16, 20, 47, 29} |
| Вивід всіх елементів масиву із їх індексами, отримання довжини поточного масиву даних:   * Масив із даними: Масив № 1 {-1, 8, 12, 16, 20, 47, 29} |
| Отримання індексу елементу масиву, що містить значення, та отримання останнього індексу елементу масиву, якому відповідає задане значення:   * Значення, яке необхідно знайти: 16 |

Очікуваний результат: Успішне завершення виконання програми. Виведення у консоль повідомлення з результатами:

|  |
| --- |
| Test Array with the Integer type values and positive indexes:  Initial array of Integers with indexes from = 20 to = 26:  Elements: [-1, 8, 12, 16, 20, 47, 29]  Get an array element at index 10: Error: invalid index. null  Get an array element at index -10: Error: invalid index. null  Set an array element at index 11 with value 1: Error: invalid index. null  Set an array element at index -11 with value 1: Error: invalid index. null  Current values of the array elements with indexes from = 20 to = 26:  Elements: [-1, 8, 12, 16, 20, 47, 29]  Get the length of the current array: 7  Get the index of the array element with a value 1: -1  Get the last index of the array element with a value 1: -1  Test Array with the Integer type values and negative indexes:  Initial array of Integers with indexes from = -14 to = -8:  Elements: [-1, 8, 12, 16, 20, 47, 29]  Get an array element at index 10: Error: invalid index. null  Get an array element at index -10: 20  Set an array element at index 11 with value 16: Error: invalid index. null  Set an array element at index -11 with value 16: 16  Current values of the array elements with indexes from = -14 to = -8:  Elements: [-1, 8, 12, 16, 20, 47, 29]  Get the length of the current array: 7  Get the index of the array element with a value 16: -11  Get the last index of the array element with a value 16: -11 |

**7.2.3 Тестовий випадок № 3** **“Тестування з типом String”**

Опис: Тестування класу OneDimBoundedArray з типом String з позитивними елементів масиву та валідною і невалідною довжиною масиву.

Тестові дані:

Таблиця 7.2 – Тестові дані з типом String

|  |
| --- |
| Створення тестового масиву типу String з позитивними індексами елементів масиву:   * Початковий індекс: 4 * Кінцевий індекс: 11 * Масив № 2 типу String: {"Apple", "Banana ", "Orange", "Grape", "Mango", "Kiwi", "Lemon", "Plum"} |
| Отримання елементу масиву за валідним індексом:   * Індекс: 10 * Масив із даними: Масив № 2 {"Apple", "Banana ", "Orange", "Grape", "Mango", "Kiwi", "Lemon", "Plum"} |
| Отримання елементу масиву за невалідним індексом:   * Індекс: -10 * Масив із даними: Масив № 2 {"Apple", "Banana ", "Orange", "Grape", "Mango", "Kiwi", "Lemon", "Plum"} |
| Встановлення значення елементу масиву за валідним індексом:   * Індекс: 11 * Значення, яке потрібно встановити: "Grape" * Масив із даними: Масив № 2 {"Apple", "Banana ", "Orange", "Grape", "Mango", "Kiwi", "Lemon", "Plum"} |
| Встановлення значення елементу масиву за невалідним індексом:   * Індекс: -11 * Значення, яке потрібно встановити: "Grape" * Масив із даними: Масив № 2 {"Apple", "Banana ", "Orange", "Grape", "Mango", "Kiwi", "Lemon", "Plum"} |
| Вивід всіх елементів масиву із їх індексами, отримання довжини поточного масиву даних:   * Масив із даними: Масив № 2 {"Apple", "Banana ", "Orange", "Grape", "Mango", "Kiwi", "Lemon", "Plum"} |
| Отримання індексу елементу масиву, що містить значення, та отримання останнього індексу елементу масиву, якому відповідає задане значення:   * Значення, яке необхідно знайти: "Grape" |
| Створення тестового масиву типу String з невалідними позитивними індексами елементів масиву, де початковий індекс більший за кінцевий індекс, а кількість заданих індексів менша за кількість елементів масиву::   * Початковий індекс: 3 * Кінцевий індекс: 0 * Масив № 2 типу String: {"Apple", "Banana ", "Orange", "Grape", "Mango", "Kiwi", "Lemon", "Plum"} |
| Створення тестового масиву типу String з невалідними позитивними індексами елементів масиву, де початковий індекс менший за кінцевий індекс, а кількість заданих індексів більша за кількість елементів масиву:   * Початковий індекс: 4 * Кінцевий індекс: 15 * Масив № 2 типу String: {"Apple", "Banana ", "Orange", "Grape", "Mango", "Kiwi", "Lemon", "Plum"} |

Очікуваний результат: Завершення виконання програми із помилками. Виведення у консоль повідомлення з результатами:

|  |
| --- |
| Test Array with the String type values and positive indexes:  Initial array of Integers with indexes from = 4 to = 11:  Elements: [Apple, Banana , Orange, Grape, Mango, Kiwi, Lemon, Plum]  Get an array element at index 10: Lemon  Get an array element at index -10: Error: invalid index. null  Set an array element at index 11 with value Grape: Plum  Set an array element at index -11 with value Grape: Error: invalid index. null  Current values of the array elements with indexes from = 4 to = 11:  Elements: [Apple, Banana , Orange, Grape, Mango, Kiwi, Lemon, Plum]  Get the length of the current array: 8  Get the index of the array element with a value Grape: 7  Get the last index of the array element with a value Grape: 7  Create first array with indexes from = 3, to = 0:  Incorrect element indexes.  The 'from' index must be less than or equal to the 'to' index.  Create second array with indexes from = 4, to = 15:  Incorrect element indexes and/or array length.  The length of the array must be equal to the number of array element indexes plus one. |

**7.3 Програмний код завдання № 7**

Лістинг 7.1 - Програмний код реалізований у файлі “OneDimBoundedArray.java”

|  |
| --- |
| package task7;  import java.util.AbstractList; import java.util.Arrays; import java.util.Iterator;  */\*\*  \* The {****@link*** *OneDimBoundedArray} is a class that represents a one-dimensional bounded array.  \* It extends AbstractList to provide a limited range of indexes for array elements.  \** ***@param*** <*T*> *the type of elements stored in the array.  \*/* public class OneDimBoundedArray<T> extends AbstractList<T> {  */\*\* The underlying array storing elements. \*/* private final T[] array;   */\*\* The starting index of the bounded array. \*/* private final int from;   */\*\* The ending index of the bounded array. \*/* private final int to;   */\*\*  \* Constructs a OneDimBoundedArray with specified range and array.  \** ***@param*** *from the start index of the array;  \** ***@param*** *to the end index of the array;  \** ***@param*** *array the array of elements.  \*/* public OneDimBoundedArray(int from, int to, T[] array) {  if (from > to) {  System.*err*.println("Incorrect element indexes.\n"  + "The \'from\' index must be less than or equal to the \'to\' index.");  System.*exit*(1);  }   if ((to - from + 1) != array.length) {  System.*err*.println("Incorrect element indexes and/or array length.\n"  + "The length of the array must be equal to the number of array element indexes plus one.");  System.*exit*(1);  }   this.from = from;  this.to = to;  this.array = Arrays.*copyOf*(array, array.length);  }   */\*\*  \* Gets the start index of the array.  \** ***@return*** *the start index of the array.  \*/* public int getFromIndex() {  return from;  }   */\*\*  \* Gets the end index of the array.  \** ***@return*** *the end index of the array.  \*/* public int getToIndex() {  return to;  }   */\*\*  \* Verifies if the given index is valid for accessing elements in the array.  \** ***@param*** *index the index to be verified;  \** ***@return*** *{****@code*** *true} if the index is valid, {****@code*** *false} otherwise.  \*/* private boolean verifyIndex(int index) {  return index >= from && index <= to;  }   */\*\*  \* Gets the element at the specified index in the array.  \** ***@param*** *index the index of the element to be retrieved.  \** ***@return*** *the element at the specified index, or {****@code*** *null} if the index is invalid.  \*/* @Override  public T get(int index) {  if (!verifyIndex(index)) {  System.*out*.print("Error: invalid index. ");   return null;  }   return array[index - from];  }   */\*\*  \* Returns the size of the array.  \** ***@return*** *the size of the array.  \*/* @Override  public int size() {  return array.length;  }   */\*\*  \* Sets the element at the specified index in the array.  \** ***@param*** *index the index of the element to be set;  \** ***@param*** *element the new value for the element.  \** ***@return*** *the previous value of the element, or {****@code*** *null} if the index is invalid.  \*/* @Override  public T set(int index, T element) {  if (!verifyIndex(index)) {  System.*out*.print("Error: invalid index. ");   return null;  }   T oldValue = array[index - from];  array[index - from] = element;   return oldValue;  }   */\*\*  \* Returns the index of the first occurrence of the specified element in the array.  \** ***@param*** *obj the element to find in the array.  \** ***@return*** *the index of the first occurrence, or {****@code*** *-1} if not found.  \*/* @Override  public int indexOf(Object obj) {  for (int i = 0; i < array.length; i++) {  if ((obj == null  && array[i] == null)  || (obj != null  && obj.equals(array[i]))) {  return i + from;  }  }   return -1;  }   */\*\*  \* Returns the index of the last occurrence of the specified element in the array.  \** ***@param*** *obj the element to find in the array.  \** ***@return*** *the index of the last occurrence, or {****@code*** *-1} if not found.  \*/* @Override  public int lastIndexOf(Object obj) {  for (int i = array.length - 1; i >= 0; i--) {  if ((obj == null  && array[i] == null)  || (obj != null  && obj.equals(array[i]))) {  return i + from;  }  }   return -1;  }   */\*\*  \* Iterator for traversing the elements of the array.  \*/* private class OneDimBoundedArrayIterator implements Iterator<T> {  */\*\* The current position of the iterator within the bounded array. \*/* private int cursor = from;   */\*\*  \* Checks if there is another element in the array to iterate.  \** ***@return*** *{****@code*** *true} if there is another element, {****@code*** *false} otherwise.  \*/* @Override  public boolean hasNext() {  return cursor <= to;  }   */\*\*  \* Retrieves the next element in the array.  \** ***@return*** *the next element in the array.  \*/* @Override  public T next() {  return array[cursor++ - from];  }  }   */\*\*  \* Returns an iterator over the elements of the array.  \** ***@return*** *an iterator over the elements of the array.  \*/* @Override  public Iterator<T> iterator() {  return new OneDimBoundedArrayIterator();  }   */\*\*  \* Returns a string representation of the array with index range information.  \** ***@return*** *a string representation of the array.  \*/* @Override  public String toString() {  StringBuilder result = new StringBuilder("\tfrom = " + getFromIndex() + "\tto = "  + getToIndex() + ":\nElements:\t[");   for (T data : this) {  result.append(data);  result.append(", ");  }   result.replace(result.length() - 2, result.length(), "]");   return result.toString();  }   */\*\*  \* Tests the {****@code*** *OneDimBoundedArray} functionality with Integer elements.  \*/* private static void testCreateIntegerArray() {  OneDimBoundedArray<Integer> arrayOfIntegers = new OneDimBoundedArray<>(5, 7, new Integer[]{2, 7, 9});  System.*out*.println("Initial array of Integers with indexes" + arrayOfIntegers + "\n");   int index = 0;  System.*out*.println("Verify that the element index " + index + " exists in the array:\t"  + arrayOfIntegers.verifyIndex(index));  index = 7;  System.*out*.println("Verify that the element index " + index + " exists in the array:\t"  + arrayOfIntegers.verifyIndex(index));  }   */\*\*  \* Main method to execute the test for creating a OneDimBoundedArray with Integer elements.  \** ***@param*** *args the command-line arguments (not used).  \*/* public static void main(String[] args) {  *testCreateIntegerArray*();  } } |

Лістинг 7.2 - Програмний код реалізований у файлі “OneDimBoundedArrayTest.java”

|  |
| --- |
| package task7;  */\*\*  \* The {****@link*** *OneDimBoundedArrayTest} class that contains test scenarios for the {****@link*** *OneDimBoundedArray} class.  \*/* public class OneDimBoundedArrayTest {  */\*\*  \* Tests the {****@link*** *OneDimBoundedArray} class functionality with the given array and element value.  \** ***@param*** <*T*> *the type of elements stored in the array;  \** ***@param*** *array The OneDimBoundedArray instance to test.  \** ***@param*** *elementValue The element value to use in the tests.  \*/* private static <T> void testArray(OneDimBoundedArray<T> array, T elementValue) {  System.*out*.println("Initial array of Integers with indexes" + array + "\n");   int index = 10;  System.*out*.print("Get an array element at index " + index + ": \t");  System.*out*.println(array.get(index));  index = -10;  System.*out*.print("Get an array element at index " + index + ": \t");  System.*out*.println(array.get(index) + "\n");   index = 11;  System.*out*.print("Set an array element at index " + index + " with value " + elementValue + ": \t");  System.*out*.println(array.get(index));  index = -11;  System.*out*.print("Set an array element at index " + index + " with value " + elementValue + ": \t");  System.*out*.println(array.get(index) + "\n");   System.*out*.println("Current values of the array elements with indexes\t" + array + "\n");   System.*out*.println("Get the length of the current array:\t" + array.size());  System.*out*.println("Get the index of the array element with a value " + elementValue + ":\t"  + array.indexOf(elementValue));  System.*out*.println("Get the last index of the array element with a value " + elementValue + ":\t"  + array.lastIndexOf(elementValue));  System.*out*.println();  }   */\*\*  \* Test the {****@link*** *OneDimBoundedArray} class for the Integer type.  \*/* private static void testIntegerArray() {  System.*out*.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  System.*out*.println("Test OneDimBoundedArray class for Integer type:");  System.*out*.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  System.*out*.println("Test Array with the Integer type values and positive indexes:");  Integer[] arrayOfIntegers = {-1, 8, 12, 16, 20, 47, 29};  OneDimBoundedArray<Integer> array1 = new OneDimBoundedArray<>(20, 26, arrayOfIntegers);  *testArray*(array1, 1);  System.*out*.println();   System.*out*.println("Test Array with the Integer type values and negative indexes:");  OneDimBoundedArray<Integer> array2 = new OneDimBoundedArray<>(-14, -8, arrayOfIntegers);  *testArray*(array2, 16);  System.*out*.println();  }   */\*\*  \* Test the {****@link*** *OneDimBoundedArray} class for the String type.  \*/* private static void testStringArray() {  System.*out*.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  System.*out*.println("Test OneDimBoundedArray class for String type:");  System.*out*.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  String[] arrayOfStrings = {"Apple", "Banana ", "Orange", "Grape", "Mango", "Kiwi", "Lemon", "Plum"};  int fromIndex = 4, toIndex = 11;  OneDimBoundedArray<String> array1 = new OneDimBoundedArray<>(fromIndex, toIndex, arrayOfStrings);  System.*out*.println("Test Array with the String type values and positive indexes:");  *testArray*(array1, "Grape");   fromIndex = 3; toIndex = 0;  System.*out*.println("Create first array with indexes from = " + fromIndex + ", to = " + toIndex + ":");  OneDimBoundedArray<String> array2 = new OneDimBoundedArray<>(fromIndex, toIndex, arrayOfStrings);  System.*out*.println();   /\*  fromIndex = 4; toIndex = 15;  System.out.println("Create second array with indexes from = " + fromIndex + ", to = " + toIndex + ":");  OneDimBoundedArray<String> array3 = new OneDimBoundedArray<>(fromIndex, toIndex, arrayOfStrings);  \*/  }   */\*\*  \* Main method to run the tests for the {****@link*** *OneDimBoundedArray} class.  \** ***@param*** *args Command-line arguments (not used in this context).  \*/* public static void main(String[] args) {  *testIntegerArray*();  *testStringArray*();  } } |

**7.4 Екранні форми за результатами роботи програмного коду завдання № 7**

Изображение выглядит как текст, снимок экрана, Шрифт, алгебра

Автоматически созданное описание

Рисунок 7.1 - Результати роботи програмного коду відповідно до тестового випадку № 1 “Тестування приватних методів класу OneDimBoundedArray” частина 1

Изображение выглядит как текст, снимок экрана, Шрифт, документ

Автоматически созданное описание

Рисунок 7.2 - Результати роботи програмного коду відповідно до тестового випадку № 1 “Тестування з типом Integer” частина 1

Изображение выглядит как текст, снимок экрана, Шрифт, алгебра

Автоматически созданное описание

Рисунок 7.3 - Результати роботи програмного коду відповідно до тестового випадку № 1 “Тестування з типом Integer” частина 2

Изображение выглядит как текст, снимок экрана, Шрифт, меню

Автоматически созданное описание

Рисунок 7.4 - Результати роботи програмного коду відповідно до тестового випадку № 3 “Тестування з типом String” частина 1

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 7.5 - Результати роботи програмного коду відповідно до тестового випадку № 3 “Тестування з типом String” частина 2

## **8.1 Завдання № 8 “Реалізація двобічно зв'язаного списку (додаткове завдання)”**

## Реалізувати узагальнений клас, що представляє двобічно зв'язаний список.

**8.2 Набір тестових даних завдання № 8**

**8.2.1 Тестовий випадок № 1 “Тестування приватних методів класу DoublyLinkedList”**

Опис: Тестування приватних методів класу DoublyLinkedList з типом Integer.

Тестові дані:

Таблиця 8.1 – Тестові дані

|  |  |  |
| --- | --- | --- |
| № з/п | Опис дії | Тестові дані |
| Тестування створення списку типу Integer: | | |
| 1 | Додавання елемента в кінець списку | 18 |
| 2 | Додавання елемента в кінець списку | 25 |
| 3 | Додавання елемента в кінець списку | 37 |
| 4 | Додавання елемента в кінець списку | 5 |
| 5 | Додавання елемента в початок списку | 1 |
| Виведення поточних даних списку із його розміром:   * Список даних: Список № 1 : 1 -> 18 -> 25 -> 37 -> 5 -> 1 -> null | | |
| Видалення елемента зі списку за індексом із пошуком заданого індексу у списку та виведення даних списку і його розміру:   * Індекс: 1 * Список, де видаляються дані:   Список № 1 : 1 -> 18 -> 25 -> 37 -> 5 -> 1 -> null | | |
| Перевірка існування значення по індексу, чи існує воно у списку:   * Індекс: 1 | | |
| Перевірка існування значення по індексу, чи існує воно у списку:   * Індекс: 7 | | |

Очікуваний результат: Успішне завершення виконання програми. Виведення у консоль повідомлення з результатами:

|  |
| --- |
| Initial list of Integers:  List size: 5  Elements: [1, 18, 25, 37, 5]  Remove an element from the list at index 1:  List size: 4  Elements: [1, 25, 37, 5]  Verify that the element at index 1 exists in the list: true  Verify that the element at index 7 exists in the list: false |

**8.2.2 Тестовий випадок № 2 “Тестування із типом Integer”**

Опис: Тестування класу DoublyLinkedList з типом Integer.

Тестові дані:

Таблиця 8.2 – Тестові дані з типом Integer

|  |  |  |  |
| --- | --- | --- | --- |
| № з/п | Опис дії | | Тестові дані |
| Тестування створення списку типу Integer: | | | |
| 1 | Додавання елемента в кінець списку | | -1 |
| 2 | Додавання елемента в кінець списку | | 8 |
| 3 | Додавання елемента в кінець списку | | 12 |
| 4 | Додавання елемента в кінець списку | | 15 |
| 5 | Додавання елемента в початок списку | | 21 |
| 6 | Додавання елемента в початок списку | | 29 |
| Виведення поточних даних списку із його розміром:   * Список даних:   Список № 1 : 15 -> 12 -> 8 -> -1 -> 21 -> 29 -> null | | | |
| Додавання елемента до списку за індексом та виведення даних списку після операцій:   * Список, куди додаються дані:   Список № 1 : 15 -> 12 -> 8 -> -1 -> 21 -> 29 -> null | | | |
| № з/п | | Індекс елемента, до якого додається значення | Значення, що додається |
| 1 | | 1 | 8 |
| 2 | | 3 | 8 |
| 3 | | -4 | 8 |
| Видалення першого елемента зі списку, останнього елемента зі списку та елемента за індексом та виведення даних списку після операцій:   * Індекс: 1 | | | |
| Знаходження індекса елемента списку за значенням:   * Значення 1: 8 * Значення 2: null | | | |
| Видалення першого входження значення у списку та виведення списку після операцій:   * Значення 1: 8 * Значення 2: null | | | |
| Отримання значення за індексом, очищення списку та виведення даних списку після операцій:   * Індекс 1: 1 * Індекс 2: 28 | | | |

Очікуваний результат: Успішне завершення виконання програми. Виведення у консоль повідомлення з результатами:

|  |
| --- |
| Initial list:  List size: 6  Elements: [15, 12, 8, -1, 21, 29]  Add list item data '8' by index 1: true  Add list item data '8' by index 3: true  Add list item data '8' by index -4: false  List after adding items:  List size: 8  Elements: [15, 8, 12, 8, 8, -1, 21, 29]  Remove first list item: 15  Remove last list item: 29  Remove list item by index 1: 12  List after removing items:  List size: 5  Elements: [8, 8, 8, -1, 21]  Find list item index by data '8': 0  Find list item index by data 'null': -1  Remove the first occurrence of data '8' from the list: true  Remove the first occurrence of data 'null' from the list: false  List after removing items data:  List size: 4  Elements: [8, 8, -1, 21]  Get list item data by index 1: 8  Get list item data by index 28: Error: invalid index. null  Clear all items from list:  List after cleaning items:  List size: 0  List is empty. |

**8.2.3 Тестовий випадок № 3 “Тестування із типом String”**

Опис: Тестування класу DoublyLinkedList з типом String.

Тестові дані:

Таблиця 8.3 – Тестові дані з типом String

|  |  |  |  |
| --- | --- | --- | --- |
| № з/п | Опис дії | | Тестові дані |
| Тестування створення списку типу Integer: | | | |
| 1 | Додавання елемента в кінець списку | | "Apple" |
| 2 | Додавання елемента в кінець списку | | "Orange" |
| 3 | Додавання елемента в кінець списку | | "Banana" |
| 4 | Додавання елемента в кінець списку | | "Grape" |
| 5 | Додавання елемента в початок списку | | "Plum" |
| 6 | Додавання елемента в початок списку | | "Kiwi" |
| Виведення поточних даних списку із його розміром:   * Список даних:   Список № 1 : Kiwi -> Plum -> Apple -> Orange -> Banana -> Grape -> null | | | |
| Додавання елемента до списку за індексом та виведення даних списку після операцій:   * Список, куди додаються дані:   Список № 1 : Kiwi -> Plum -> Apple -> Orange -> Banana -> Grape -> null | | | |
| № з/п | | Індекс елемента, до якого додається значення | Значення, що додається |
| 1 | | 1 | "Banana" |
| 2 | | 3 | "Banana" |
| 3 | | -4 | "Banana" |
| Видалення першого елемента зі списку, останнього елемента зі списку та елемента за індексом та виведення даних списку після операцій:   * Індекс: 1 | | | |
| Знаходження індекса елемента списку за значенням:   * Значення 1: "Banana" * Значення 2: null | | | |
| Видалення першого входження значення у списку та виведення списку після операцій:   * Значення 1: "Banana" * Значення 2: null | | | |
| Отримання значення за індексом, очищення списку та виведення даних списку після операцій:   * Індекс 1: 1 * Індекс 2: 28 | | | |

Очікуваний результат: Успішне завершення виконання програми. Виведення у консоль повідомлення з результатами:

|  |
| --- |
| Initial list:  List size: 6  Elements: [Kiwi, Plum, Apple, Orange, Banana, Grape]  Add list item data 'Banana' by index 1: true  Add list item data 'Banana' by index 3: true  Add list item data 'Banana' by index -4: false  List after adding items:  List size: 8  Elements: [Kiwi, Banana, Plum, Banana, Apple, Orange, Banana, Grape]  Remove first list item: Kiwi  Remove last list item: Grape  Remove list item by index 1: Plum  List after removing items:  List size: 5  Elements: [Banana, Banana, Apple, Orange, Banana]  Find list item index by data 'Banana': 0  Find list item index by data 'null': -1  Remove the first occurrence of data 'Banana' from the list: true  Remove the first occurrence of data 'null' from the list: false  List after removing items data:  List size: 4  Elements: [Banana, Apple, Orange, Banana]  Get list item data by index 1: Apple  Get list item data by index 28: Error: invalid index. null  Clear all items from list:  List after cleaning items:  List size: 0  List is empty. |

**8.3 Програмний код завдання № 8**

Лістинг 8.1 - Програмний код реалізований у файлі “DoublyLinkedList.java”

|  |
| --- |
| package task8;  import java.util.Iterator;  */\*\*  \* The {****@link*** *DoublyLinkedList} represents a generic doubly linked list.  \** ***@param*** <*E*> *the type of elements stored in the list.  \*/* public class DoublyLinkedList<E> implements Iterable<E> {  */\*\*  \* Node represents a node in the doubly linked list.  \** ***@param*** <*E*> *the type of elements stored in the node.  \*/* private static class Node<E> {  */\*\* The data stored in the node. \*/* E data;   */\*\* The reference to the next node in the list. \*/* Node<E> next;   */\*\* The reference to the previous node in the list. \*/* Node<E> previous;   */\*\*  \* Constructor for creating a new node with the specified previous, element, and next nodes.  \** ***@param*** *previous the previous node in the list;  \** ***@param*** *element the data of the node;  \** ***@param*** *next the next node in the list.  \*/* Node(Node<E> previous, E element, Node<E> next) {  this.data = element;  this.next = next;  this.previous = previous;  }  }   */\*\* Field representing the first node of the list. \*/* private Node<E> first = null;   */\*\* Field representing the last node of the list. \*/* private Node<E> last = null;   */\*\* Field representing the size of the list. \*/* private int size = 0;   */\*\*  \* Adds an element to the beginning of the list.  \** ***@param*** *element the element to be added.  \*/* public void addFirst(E element) {  Node<E> linkToFirst = first;  Node<E> newNode = new Node<>(null, element, first);  first = newNode;   if (linkToFirst == null) {  last = newNode;  } else {  linkToFirst.previous = newNode;  }   size++;  }   */\*\*  \* Adds an element to the end of the list.  \** ***@param*** *element the element to be added.  \*/* public void addLast(E element) {  Node<E> linkToLast = last;  Node<E> newNode = new Node<>(linkToLast, element, null);  last = newNode;   if (linkToLast == null) {  first = newNode;  } else {  linkToLast.next = newNode;  }   size++;  }   */\*\*  \* Verifies if the given index is valid for accessing elements in the list.  \** ***@param*** *index the index to be verified;  \** ***@return*** *{****@code*** *true} if the index is valid, {****@code*** *false} otherwise.  \*/* private boolean verifyIndex(int index) {  return index >= 0 && index < size;  }   */\*\*  \* Finds and returns the node at the specified index in the list.  \** ***@param*** *index the index of the node to be found;  \** ***@return*** *the node at the specified index.  \*/* private Node<E> findByIndex(int index) {  Node<E> current;   if (index < (size >> 1)) {  current = first;   for (int i = 0; i < index; i++) {  current = current.next;  }   return current;  } else {  current = last;   for (int i = size - 1; i > index; i--) {  current = current.previous;  }   return current;  }  }   */\*\*  \* Adds an element at the specified index in the list.  \** ***@param*** *index the index at which the element will be added;  \** ***@param*** *element the element to be added to the list.  \** ***@return*** *{****@code*** *true} if the element is added successfully, {****@code*** *false} otherwise.  \*/* public boolean addByIndex(int index, E element) {  if (index == size) {  addLast(element);   return true;  }   if (!verifyIndex(index)) {  return false;  }   Node<E> current = findByIndex(index);  Node<E> prevCurrent = current.previous;  Node<E> newNode = new Node<>(prevCurrent, element, current);  current.previous = newNode;   if (prevCurrent == null) {  first = newNode;  } else {  prevCurrent.next = newNode;  }   size++;   return true;  }   */\*\*  \* Removes and returns the first element in the list.  \** ***@return*** *the first element in the list, or {****@code*** *null} if the list is empty.  \*/* public E removeFirst() {  Node<E> linkToFirst = first;   if (linkToFirst == null) {  System.*out*.println("The list is empty.");   return null;  }   E element = linkToFirst.data;  Node<E> next = linkToFirst.next;  linkToFirst.data = null;  linkToFirst.next = null;  first = next;   if (next == null) {  last = null;  } else {  next.previous = null;  }   size--;   return element;  }   */\*\*  \* Removes and returns the last element in the list.  \** ***@return*** *the last element in the list, or {****@code*** *null} if the list is empty.  \*/* public E removeLast() {  Node<E> linkToLast = last;   if (linkToLast == null) {  System.*out*.println("The list is empty.");   return null;  }   E element = linkToLast.data;  Node<E> prev = linkToLast.previous;  linkToLast.data = null;  linkToLast.previous = null;  last = prev;   if (prev == null) {  first = null;  } else {  prev.next = null;  }   size--;   return element;  }   */\*\*  \* Removes the specified node from the list.  \** ***@param*** *current the node to be removed.  \*/* private void remove(Node<E> current) {  Node<E> next = current.next;  Node<E> prev = current.previous;   if (prev == null) {  first = next;  } else {  prev.next = next;  current.previous = null;  }   if (next == null) {  last = prev;  } else {  next.previous = prev;  current.next = null;  }   current.data = null;  size--;  }   */\*\*  \* Removes and returns the element at the specified index in the list.  \** ***@param*** *index the index of the element to be removed.  \** ***@return*** *the removed element, or {****@code*** *null} if the index is invalid.  \*/* public E removeAtIndex(int index) {  if (!verifyIndex(index)) {  System.*out*.print("Error: invalid index. ");   return null;  }   Node<E> current = findByIndex(index);  E element = current.data;  remove(current);   return element;  }   */\*\*  \* Finds the index of the first occurrence of the specified element in the list.  \** ***@param*** *element the element to find in the list.  \** ***@return*** *the index of the first occurrence, or {****@code*** *-1} if not found.  \*/* public int indexOfData(E element) {  int index = 0;   if (element == null) {  for (Node<E> i = first; i != null; i = i.next) {  if (i.data == null) {  return index;  }   index++;  }  } else {  for (Node<E> i = first; i != null; i = i.next) {  if (element.equals(i.data)) {  return index;  }   index++;  }  }   return -1;  }   */\*\*  \* Removes the first occurrence of the specified element from the list.  \** ***@param*** *element the element to be removed.  \** ***@return*** *{****@code*** *true} if the element is removed successfully, {****@code*** *false} otherwise.  \*/* public boolean removeFirstOccurrence(E element) {  int index = indexOfData(element);   if (index >= 0) {  Node<E> current = findByIndex(index);  remove(current);   return true;  }   return false;  }   */\*\*  \* Gets the size of the list.  \** ***@return*** *the size of the list.  \*/* public int listSize() {  return size;  }   */\*\*  \* Gets the data of the element at the specified index in the list.  \** ***@param*** *index the index of the element.  \** ***@return*** *the data of the element, or {****@code*** *null} if the index is invalid.  \*/* public E getDataByIndex(int index) {  if (!verifyIndex(index)) {  System.*out*.print("Error: invalid index. ");   return null;  }   return findByIndex(index).data;  }   */\*\*  \* Clears the list by removing all elements and resetting size to 0.  \*/* public void clearList() {  for (Node<E> i = first; i != null; ) {  Node<E> next = i.next;  i.data = null;  i.next = null;  i.previous = null;  i = next;  }   first = last = null;  size = 0;  }   */\*\*  \* Iterator for traversing the elements of the list.  \*/* private class DoublyLinkedListIterator implements Iterator<E> {  */\*\* The current node in the iteration.  \*/* private Node<E> current = first;   */\*\*  \* Checks if there is a next element in the iteration.  \** ***@return*** *true if there is a next element, false otherwise.  \*/* @Override  public boolean hasNext() {  return current != null;  }   */\*\*  \* Returns the next element in the iteration.  \** ***@return*** *the next element.  \*/* @Override  public E next() {  E data = current.data;  current = current.next;   return data;  }  }   */\*\*  \* Returns an iterator over the elements of the list.  \** ***@return*** *an iterator over the elements of the list.  \*/* @Override  public Iterator<E> iterator() {  return new DoublyLinkedListIterator();  }   */\*\*  \* Provides the string representing of the list.  \** ***@return*** *a string representation of the list.  \*/* @Override  public String toString() {  StringBuilder result = new StringBuilder("List size:\t" + listSize() + "\n");   if (listSize() > 0) {  result.append("Elements:\t[");   for (E data : this) {  result.append(data);  result.append(", ");  }   result.replace(result.length() - 2, result.length(), "]");  } else {  result.append("List is empty.\n");  }   return result.toString();  }   */\*\*  \* Perform testing the {****@code*** *DoublyLinkedList} functionality with Integer elements.  \*/* private static void testCreateIntegerList() {  DoublyLinkedList<Integer> list = new DoublyLinkedList<>();  list.addLast(18);  list.addLast(25);  list.addLast(37);  list.addLast(5);  list.addFirst(1);   System.*out*.println("Initial list of Integers:\n" + list + "\n");   int index = 1;  System.*out*.println("Remove an element from the list at index " + index + ":\t");  list.remove(list.findByIndex(index));  System.*out*.println(list);  System.*out*.println();   System.*out*.println("Verify that the element at index " + index + " exists in the list:\t"  + list.verifyIndex(index));  index = 7;  System.*out*.println("Verify that the element at index " + index + " exists in the list:\t"  + list.verifyIndex(index));  }   */\*\*  \* Main method to execute the test for creating a {****@code*** *DoublyLinkedList} with Integer elements.  \** ***@param*** *args the command-line arguments (not used).  \*/* public static void main(String[] args) {  *testCreateIntegerList*();  } } |

Лістинг 8.2 - Програмний код реалізований у файлі “DoublyLinkedListTest.java”

|  |
| --- |
| package task8;  */\*\*  \* The {****@link*** *DoublyLinkedListTest} class that contains test scenarios for the {****@link*** *DoublyLinkedList} class.  \*/* public class DoublyLinkedListTest {  */\*\*  \* Tests the functionality of the {****@link*** *DoublyLinkedList} class with the specified element data.  \** ***@param*** *list the DoublyLinkedList to be tested;  \** ***@param*** *elementData the element data to be used in the test.  \** ***@param*** <*T*> *the type of elements stored in the list.  \*/* private static <T> void testFunctionality(DoublyLinkedList<T> list, T elementData) {  System.*out*.println("Initial list:\n" + list + "\n");   int index = 1;  System.*out*.println("Add list item data \'" + elementData + "\' by index " + index + ": \t"  + list.addByIndex(index, elementData));  index = 3;  System.*out*.println("Add list item data \'" + elementData + "\' by index " + index + ": \t"  + list.addByIndex(index, elementData));  index = -4;  System.*out*.println("Add list item data \'" + elementData + "\' by index " + index + ": \t"  + list.addByIndex(index, elementData));  System.*out*.println("List after adding items:\n" + list + "\n");   System.*out*.println("Remove first list item:\t\t\t" + list.removeFirst());  System.*out*.println("Remove last list item:\t\t\t" + list.removeLast());  index = 1;  System.*out*.println("Remove list item by index " + index + ":\t" + list.removeAtIndex(index));  System.*out*.println("List after removing items:\n" + list + "\n");   System.*out*.println("Find list item index by data \'" + elementData + "\':\t"  + list.indexOfData(elementData));  System.*out*.println("Find list item index by data \'" + null + "\':\t"  + list.indexOfData(null) + "\n");   System.*out*.println("Remove the first occurrence of data \'" + elementData + "\' from the list: \t"  + list.removeFirstOccurrence(elementData));  System.*out*.println("Remove the first occurrence of data \'" + null + "\' from the list:\t\t"  + list.removeFirstOccurrence(null));  System.*out*.println("List after removing items data:\n" + list + "\n");   index = 1;  System.*out*.println("Get list item data by index " + index + ":\t\t" + list.getDataByIndex(index));  index = 28;  System.*out*.print("Get list item data by index " + index + ":\t\t");  System.*out*.println(list.getDataByIndex(index) + "\n");   System.*out*.println("Clear all items from list:");  list.clearList();  System.*out*.println("List after cleaning items:\n" + list + "\n");  }   */\*\*  \* Tests the {****@link*** *DoublyLinkedList} class with Integer elements.  \*/* private static void testIntegerList() {  System.*out*.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  System.*out*.println("Test the DoublyLinkedList class with type Integer:");  System.*out*.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  DoublyLinkedList<Integer> list = new DoublyLinkedList<>();  list.addFirst(-1);  list.addFirst(8);  list.addFirst(12);  list.addFirst(15);  list.addLast(21);  list.addLast(29);  *testFunctionality*(list, 8);  }   */\*\*  \* Tests the {****@link*** *DoublyLinkedList} class with String elements.  \*/* private static void testStringList() {  System.*out*.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  System.*out*.println("Test the DoublyLinkedList class with type String:");  System.*out*.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  DoublyLinkedList<String> list = new DoublyLinkedList<>();  list.addLast("Apple");  list.addLast("Orange");  list.addLast("Banana");  list.addLast("Grape");  list.addFirst("Plum");  list.addFirst("Kiwi");  *testFunctionality*(list, "Banana");  }   */\*\*  \* Main method to execute the test scenarios for the {****@link*** *DoublyLinkedList} class.  \** ***@param*** *args the command-line arguments (not used).  \*/* public static void main(String[] args) {  *testIntegerList*();  *testStringList*();  } } |

**8.4 Екранні форми за результатами роботи програмного коду завдання № 8**

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 8.1 - Результати роботи програмного коду відповідно до тестового випадку № 1 “Тестування приватних методів класу DoublyLinkedList”

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

Рисунок 8.2 - Результати роботи програмного коду відповідно до тестового випадку № 2 “Тестування з типом Integer”

Изображение выглядит как текст, снимок экрана, документ, чек

Автоматически созданное описание

Рисунок 8.3 - Результати роботи програмного коду відповідно до тестового випадку № 3 “Тестування з типом String”

## **9.1 Завдання № 9 “Реалізація функції видалення елемента з дерева (додаткове завдання)”**

Додати до прикладу 3.9 функцію видалення визначеного елемента з дерева.

**9.2 Набір тестових даних завдання № 9**

**9.2.1 Тестовий випадок № 1** **“Тестування видалення вузла з бінарного дерева для класу “BinaryTree”**

Опис: Тестування класу BinaryTree із функціями видаленння вузла.

Тестові дані:

Таблиця 9.1 – Тестові дані

|  |  |  |
| --- | --- | --- |
| № з/п | Ключ | Значення |
| Тестування бінарного дерева з додаванням наступних даних вузлів: | | |
| 1 | 1 | "Europe" |
| 2 | 2 | "Africa" |
| 3 | 3 | "Australia" |
| 4 | 4 | "America" |
| 5 | 5 | "Asia" |
| 6 | 6 | "Antarctica" |
| Видалення вузла за невалідним ключем:   * Ключ: 9 | | |
| Видалення вузла за валідним ключем без дочірніх вузлів та виведення даних бінарного дерева після операції:   * Ключ: 6 | | |
| Видалення вузла за валідним ключем з одним дочірнім вузлом праворуч та виведення даних бінарного дерева після операції:   * Ключ: 5 | | |
| Видалення вузла за валідним ключем з одним дочірнім вузлом ліворуч та виведення даних бінарного дерева після операції:   * Ключ: 4 | | |
| Видалення вузла за валідним ключем з двома дочірніми вузлами та виведення даних бінарного дерева після операції:   * Ключ: 3 | | |
| Отримання значення по ключу:   * Ключ: 4 | | |
| Отримання значення по ключу:   * Ключ: 2 | | |

Очікуваний результат: Успішне завершення виконання програми. Виведення у консоль повідомлення з результатами:

|  |
| --- |
| Initial binary tree:  Key: 1 Value: 'Europe'  Key: 2 Value: 'Asia'  Key: 3 Value: 'Africa'  Key: 4 Value: 'America'  Key: 5 Value: 'Australia'  Key: 6 Value: 'Antarctica'  Testing node removal by index:  Removing the node with an invalid index 9: false  Key: 1 Value: 'Europe'  Key: 2 Value: 'Asia'  Key: 3 Value: 'Africa'  Key: 4 Value: 'America'  Key: 5 Value: 'Australia'  Key: 6 Value: 'Antarctica'  Removing the node with a valid index 6 (has zero children): true  Key: 1 Value: 'Europe'  Key: 2 Value: 'Asia'  Key: 3 Value: 'Africa'  Key: 4 Value: 'America'  Key: 5 Value: 'Australia'  Removing the node with a valid index 5 (has only right children): true  Key: 1 Value: 'Europe'  Key: 2 Value: 'Asia'  Key: 3 Value: 'Africa'  Key: 4 Value: 'America'  Removing the node with a valid index 4 (has only left children): true  Key: 1 Value: 'Europe'  Key: 2 Value: 'Asia'  Key: 3 Value: 'Africa'  Removing the node with a valid index 3 (has both children): true  Key: 1 Value: 'Europe'  Key: 2 Value: 'Asia'  Continent with key 4: null  Continent with key 2:  Key: 2 Value: 'Asia' |

**9.4 Програмний код завдання № 9**

Лістинг 9.1 - Програмний код реалізований у файлі “BinaryTree.java”

|  |
| --- |
| package task9;  */\*\*  \* The {****@link*** *BinaryTree} class representing ordinary (unbalanced) binary tree.  \*/* public class BinaryTree {  */\*\* The {****@link*** *Node} class representing a node in the binary tree. \*/* public static class Node {  */\*\* The node key of a binary tree. \*/* int key;   */\*\* The node data of a binary tree. \*/* String value;   */\*\* The left descendant(child) of the node of the binary tree. \*/* Node leftChild;   */\*\* The right descendant(child) of the node of the binary tree. \*/* Node rightChild;   */\*\*  \* Constructor for creation of a node with a key and value.  \** ***@param*** *key the node key;  \** ***@param*** *name the value of the node.  \*/* Node(int key, String name) {  this.key = key;  this.value = name;  }   */\*\*  \* Provides the string representing the object.  \** ***@return*** *the string representing the object.  \*/* @Override  public String toString() {  return "Key:\t" + key + "\tValue:\t\'" + value + "\'";  }  }   */\*\*  \* The root node of a binary tree.  \*/* private Node root;   */\*\*  \* Add the node to the binary tree.  \** ***@param*** *key the node key;  \** ***@param*** *value the value of the node.  \*/* public void addNode(int key, String value) {  Node newNode = new Node(key, value);   if (root == null) {  root = newNode;  } else {  Node currentNode = root;  Node parent;   while (true) {  parent = currentNode;   if (key < currentNode.key) {  currentNode = currentNode.leftChild;   if (currentNode == null) {  parent.leftChild = newNode;   return;  }  } else {  currentNode = currentNode.rightChild;   if (currentNode == null) {  parent.rightChild = newNode;   return;  }  }  }  }  }   */\*\*  \* Traverse nodes in ascending key order.  \** ***@param*** *currentNode the node from which its child nodes are traversed.  \*/* public void traverseTree(Node currentNode) {  if (currentNode != null) {  traverseTree(currentNode.leftChild);  System.*out*.println(currentNode);  traverseTree(currentNode.rightChild);  }  }   */\*\*  \* Traverse nodes in ascending key order starting from the root of the binary tree.  \*/* public void traverseTree() {  traverseTree(root);  }   */\*\*  \* Search for a node in a binary tree by its key.  \** ***@param*** *key the node key;  \** ***@return*** *the found node.  \*/* public Node findNode(int key) {  Node focusNode = root;   while (focusNode.key != key) {  if (key < focusNode.key) {  focusNode = focusNode.leftChild;  } else {  focusNode = focusNode.rightChild;  }   if (focusNode == null) {  return null;  }  }   return focusNode;  }   */\*\*  \* Process the case when the node has zero children at removal.  \** ***@param*** *currentNode the node to be removed;  \** ***@param*** *parent the parent node for {****@code*** *currentNode}.  \*/* private void removeNodeWithZeroChildren(Node currentNode, Node parent) {  if (parent == null) {  root = null;  } else if (parent.leftChild == currentNode) {  parent.leftChild = null;  } else {  parent.rightChild = null;  }  }   */\*\*  \* Process the case when the node has only the right child at removal.  \** ***@param*** *currentNode the node to be removed;  \** ***@param*** *parent the parent node for {****@code*** *currentNode}.  \*/* private void removeNodeWithRightChild(Node currentNode, Node parent) {  if (parent == null) {  root = currentNode.rightChild;  } else if (parent.leftChild == currentNode) {  parent.leftChild = currentNode.rightChild;  } else {  parent.rightChild = currentNode.rightChild;  }  }   */\*\*  \* Process the case when the node has only the left child at removal.  \** ***@param*** *currentNode the node to be removed;  \** ***@param*** *parent the parent node for {****@code*** *currentNode}.  \*/* private void removeNodeWithLeftChild(Node currentNode, Node parent) {  if (parent == null) {  root = currentNode.leftChild;  } else if (parent.leftChild == currentNode) {  parent.leftChild = currentNode.leftChild;  } else {  parent.rightChild = currentNode.leftChild;  }  }   */\*\*  \* Handling the case when a node has both children.  \** ***@param*** *currentNode the node to be removed;  \** ***@param*** *parent the parent node for {****@code*** *currentNode}.  \*/* private void removeNodeWithBothChildren(Node currentNode, Node parent) {  Node parentOfChild = currentNode;  Node child = currentNode.rightChild;   while (child.leftChild != null) {  parentOfChild = child;  child = child.leftChild;  }   if (parentOfChild != currentNode) {  parentOfChild.leftChild = child.rightChild;  child.rightChild = currentNode.rightChild;  }   child.leftChild = currentNode.leftChild;   if (parent == null) {  root = child;  } else if (parent.leftChild == currentNode) {  parent.leftChild = child;  } else {  parent.rightChild = child;  }  }   */\*\*  \* Remove the node with processing based on the existence of children.  \** ***@param*** *currentNode the node to be removed;  \** ***@param*** *parent the parent node for {****@code*** *currentNode}.  \*/* private void removeNodeWithChildren(Node currentNode, Node parent) {  if (currentNode.leftChild == null  && currentNode.rightChild == null) {  removeNodeWithZeroChildren(currentNode, parent);  } else if (currentNode.leftChild == null) {  removeNodeWithRightChild(currentNode, parent);  } else if (currentNode.rightChild == null) {  removeNodeWithLeftChild(currentNode, parent);  } else {  removeNodeWithBothChildren(currentNode, parent);  }  }   */\*\*  \* Remove the node by key.  \** ***@param*** *key the node key;  \** ***@return*** *{****@code*** *true} if the node was found and successfully removed, {****@code*** *false} otherwise.  \*/* public boolean removeNode(int key) {  Node currentNode = root;  Node parent = null;   while (currentNode != null) {  if (key < currentNode.key) {  parent = currentNode;  currentNode = currentNode.leftChild;  } else if (key > currentNode.key) {  parent = currentNode;  currentNode = currentNode.rightChild;  } else {  removeNodeWithChildren(currentNode, parent);   return true;  }  }   return false;  }   */\*\*  \* Performs testing of the {****@link*** *BinaryTree} class functionality. The {****@code*** *args} are not used.  \** ***@param*** *args the command-line arguments.  \*/* public static void main(String[] args) {  System.*out*.println("Initial binary tree:");  BinaryTree continents = new BinaryTree();  continents.addNode(1, "Europe");  continents.addNode(3, "Africa");  continents.addNode(5, "Australia");  continents.addNode(4, "America");  continents.addNode(2, "Asia");  continents.addNode(6, "Antarctica");  continents.traverseTree();  System.*out*.println();   System.*out*.println("Testing node removal by index:");  int key = 9;  System.*out*.println("Removing the node with an invalid index " + key + ":\t" + continents.removeNode(key));  continents.traverseTree();  System.*out*.println();   key = 6;  System.*out*.println("Removing the node with a valid index " + key + " (has zero children):\t\t\t"  + continents.removeNode(key));  continents.traverseTree();  System.*out*.println();   key--;  System.*out*.println("Removing the node with a valid index " + key + " (has only right children):\t"  + continents.removeNode(key));  continents.traverseTree();  System.*out*.println();   key--;  System.*out*.println("Removing the node with a valid index " + key + " (has only left children):\t"  + continents.removeNode(key));  continents.traverseTree();  System.*out*.println();   key--;  System.*out*.println("Removing the node with a valid index " + key + " (has both children):\t\t\t"  + continents.removeNode(key));  continents.traverseTree();  System.*out*.println();   key++;  System.*out*.println("Continent with key " + key + ":\t" + continents.findNode(key) + "\n");   key = 2;  System.*out*.println("Continent with key " + key + ":\n" + continents.findNode(key) + "\n");  } } |

**9.4 Екранні форми за результатами роботи програмного коду завдання № 8**

Изображение выглядит как текст, снимок экрана, меню

Автоматически созданное описание

Рисунок 9.1 - Результати роботи програмного коду відповідно до тестового випадку № 1 “Тестування видалення вузла з бінарного дерева для класу “BinaryTree” частина 1

Изображение выглядит как текст, снимок экрана, меню

Автоматически созданное описание

Рисунок 9.2 - Результати роботи програмного коду відповідно до тестового випадку № 1 “Тестування видалення вузла з бінарного дерева для класу “BinaryTree” частина 2

**Висновки до роботи:**

У ході виконання даної лабораторної роботи використано різні засоби, методи та технології для ефективної реалізації поставлених завдань із використанням мови програмування Java.

Під час виконання даної лабораторної роботи, були отримані значущі навички програмування, зокрема робота з узагальненнями та колекціями. Також навички розширились за рахунок розробки ієрархії класів, де було додано похідні класи для зручності представлення даних у вигляді списку та множини, відповідно до вимог індивідуального завдання. Щоб переконатися у коректності та консистентності результатів, було виконано тестування реалізацій завдань.

Для оптимізації роботи з масивами та списками чисел, було розроблено класи із статичними узагальненими методами. Такі методи, як обмін елементів, знаходження індексу та робота зі списками чисел, були успішно реалізовані і протестовані на різних типів даних.

Узагальнені знання про роботу із списками та масивами чисел значно поглибили розуміння програмування на мові Java. Дана лабораторна робота зосереджена на оптимізації та ефективному використанні узагальнень, переліків та колекцій. Здобуті навички виявились дуже корисними при розв'язанні поставлених завдань.

Було продемонстровано покрокову роботу з інтерфейсами, абстрактними класами, перевірку еквівалентності та розробку алгоритму для розрахунку хеш-кодів. До класу та окремих функцій були додані коментарі JavaDoc, які детально визначають функціональність та призначення кожного елементу коду. Використання різних дескрипторів документації, таких як code, link, param та return, допомогло зрозуміло описати код.

Тому дана лабораторна робота є не тільки корисною для поглиблення розуміння узагальнень та колекцій у Java, але й сприяла розвитку високоорганізованих навичок програмування та розв'язання завдань. Використання цих можливостей дозволяє навчитись створювати більш функціональні, ефективні, масштабовані та зрозумілі програми на мові Java, що є важливим для розробки якісного програмного забезпечення та розв'язання різноманітних завдань.