МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

“ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”

КАФЕДРА «Програмна інженерія та інтелектуальні технології управління»

ЗВІТ

з лабораторної роботи № 3

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**

“Використання успадкування та поліморфізму в Java”.

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

Створити ієрархію класів, які представляють сутності відповідно до індивідуального завдання. Варіант завдання, який слід реалізувати у програмі, визначається залежно від номеру студента у списку групи.

Слід окремо здійснити тестування кожного з класів, після чого окремими методами першого класу реалізувати основне завдання. У функції main() створити необхідний об'єкт та викликати для нього методи, які реалізують основне завдання. Вивести результати у консольне вікно.

Умова завдання 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

Клас, який представляє другу сутність індивідуального завдання, повинен містити методи доступу, перевизначення функцій toString(), equals() та hashCode(), а також реалізацію функцій, визначених індивідуальним завданням (за необхідністю). Цей клас також повинен реалізовувати інтерфейс Comparable для природного порівняння об'єктів під час сортування за однією з ознак.

Базовий абстрактний клас, який представляє першу з сутностей індивідуального завдання (умовно AbstractFirstEntity), не повинен містити даних про послідовність елементів типу другого класу. Слід визначити такі функції:

* функції для доступу до даних;
* абстрактні функції для доступу до послідовності елементів типу другого класу;
* перевизначення функції toString() для виведення даних про об'єкти;
* перевизначення методу equals() для перевірки еквівалентності об'єктів;
* перевизначення методу hashCode() для отримання хеш-кодів об'єктів;
* реалізацію методів пошуку за визначеними ознаками;
* реалізацію функції сортування елементів послідовності за першою ознакою індивідуального завдання методом бульбашки;
* реалізацію функції сортування елементів послідовності за другою ознакою індивідуального завдання методом сортування включенням;
* реалізацію методу тестування функціональності класів.

Ознаки сортування визначаються залежно від номера студента у списку групи. Функції пошуку повинні повертати масиви об'єктів (або null, якщо пошук не дав результатів). Індивідуальні ознаки сортування вказані в таблиці.

Умова завдання для варіанту 24 (номер 24 за порядком у списку групи) наведено нижче.

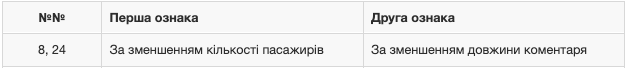


Рисунок 1.2 – Умова 2 завдання № 1 варіант 24

Від створеного абстрактного класу необхідно створити похідний клас (умовно FirstEntityWithArray), який повинен містити поля конкретних типів, зокрема, у класі FirstEntityWithArray послідовність елементів другої сутності повинна бути представлена у вигляді масиву.

Створити ще один клас (умовно FirstEntityWithSorting), похідний від попередньо створеного класу FirstEntityWithArray. Цей клас повинен перевизначати методи сортування – замість сортування бульбашкою і включенням застосувати стандартну функцію Arrays.sort(). Одне з сортувань повинне бути забезпечене реалізацією інтерфейсу Comparable для сутності, об'єкти якої зберігаються в масиві. Друге сортування забезпечується створенням окремого класу, який реалізує інтерфейс Comparator. Рекомендовано скористатися лямбда-виразом.

Тестування програми повинно включати основне завдання, а також сортування за певними критеріями.

Примітка: слід створити класи зі змістовними іменами, які відображають фізичну сутність індивідуального завдання, а не FirstEntity, SecondEntity тощо.

Слід додати коментарі Javadoc до сирцевого коду.

**1.2 Набір тестових даних завдання № 1**

**1.2.1 Тестовий випадок № 1** **“Тестування класу Hour”**

Опис: Тестування функціональності класу Hour.

Тестові дані:

Таблиця 1.1 – Тестові дані для класу Hour

|  |  |  |
| --- | --- | --- |
| Індекс | Пасажиропотік | Коментар |
| Тестування конструктора з параметрами – валідні дані | | |
|  | 100 | "Low ridership" |
| Тестування конструктора з параметрами – невалдіні дані | | |
|  | -200 | null |
| Тестування функцій встановлення значень пасажиропотоку та коментаря | | |
|  | 200 | "Medium ridership" |
| Масив для тестування рівності обʼєктів, отримання hashcode, порівняння, сортування за зменшенням пасажиропотоку | | |
| 0 | 200 | "Medium ridership" |
| 1 | 50 | "Very low ridership" |
| 2 | 200 | "Medium ridership" |
| 3 | 100 | "Low ridership" |
| 4 | 700 | "High ridership" |
| 5 | 1200 | "Very high ridership" |
| 6 | -200 | null |

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

|  |
| --- |
| Create Hour with default constructor:  Hour { ridership = 0, comment = '' }  Length of comment: 0  Count of words in comment: 0  Create Hour with parameterized constructor:  Valid data for hour:  Hour { ridership = 100, comment = 'Low ridership' }  Invalid data for hour:  Hour { ridership = 0, comment = '' }  Set values for the Hour:  Hour { ridership = 200, comment = 'Medium ridership' }  Get values for the Hour:  Hour { ridership = 200, comment = 'Medium ridership' }  Get length of comment: 16  Get count of words in comment: 2  Array of hours:  Hour { ridership = 200, comment = 'Medium ridership' }  Hour { ridership = 50, comment = 'Very low ridership' }  Hour { ridership = 200, comment = 'Medium ridership' }  Hour { ridership = 100, comment = 'Low ridership' }  Hour { ridership = 700, comment = 'High ridership' }  Hour { ridership = 1200, comment = 'Very high ridership' }  Hour { ridership = 0, comment = '' }  Check for equal values of Hours at index 0 and 1: false  Hour at index 0: Hour { ridership = 200, comment = 'Medium ridership' }  Hour at index 1: Hour { ridership = 50, comment = 'Very low ridership' }  Check for equal values of Hours at index 0 and 2: true  Hour at index 0: Hour { ridership = 200, comment = 'Medium ridership' }  Hour at index 2: Hour { ridership = 200, comment = 'Medium ridership' }  Comparison of Hours at index 0 and 1: -1  Hashcode of Hour at index 0: -2138105960  Hashcode of Hour at index 1: -1270411456  Comparison of Hours at index 0 and 2: 0  Hashcode of Hour at index 0: -2138105960  Hashcode of Hour at index 2: -2138105960  Comparison of Hours at index 1 and 2: 1  Hashcode of Hour at index 1: -1270411456  Hashcode of Hour at index 2: -2138105960  Sort array of Hours by descending ridership:  Array of hours:  Hour { ridership = 1200, comment = 'Very high ridership' }  Hour { ridership = 700, comment = 'High ridership' }  Hour { ridership = 200, comment = 'Medium ridership' }  Hour { ridership = 200, comment = 'Medium ridership' }  Hour { ridership = 100, comment = 'Low ridership' }  Hour { ridership = 50, comment = 'Very low ridership' }  Hour { ridership = 0, comment = '' } |

**1.2.2 Тестовий випадок № 2** **“Тестування класу MetroStationWithArray”**

Опис: Тестування функціональності класу MetroStationWithArray.

Тестові дані:

Таблиця 1.2 – Тестові дані для класу MetroStationWithArray

|  |  |  |
| --- | --- | --- |
| Індекс | Пасажиропотік | Коментар |
| Тестування конструктора з параметрами – валідні дані станції 1, – тестування отримання даних про назву та даних про рік відкриття, перевірка станцій метро на рівність:   * назва станції: "Sportyvna" * рік відкриття: 1975 | | |
| Тестування конструктора з параметрами – валідні дані станції 2, – тестування отримання даних про назву та даних про рік відкриття, перевірка станцій метро на рівність:   * назва станції: "Oleksiivska" * рік відкриття: 2010 * Масив годин роботи: | | |
| 0 | 23 | "Very low ridership" |
| 1 | 345 | "Medium ridership" |
| 2 | 87 | "Low ridership" |
| Тестування функції встановлення значень масиву, підрахунку кількості годин в масиві | | |
| 0 | 256 | "Medium ridership" |
| 1 | 817 | "High ridership" |
| 2 | 1115 | "Very high ridership" |
| Тестування функції встановлення години роботи у масиві за індексом | | |
| 0 | 88 | "Very low ridership" |
| Тестування функції отримання години роботи з масиву за індексом:   * індекс: 1 | | |
| 0 | 88 | "Very low ridership" |
| Дані масиву 2 для дублювання даних за масиву 2 у масив 1:   * назва станції: "Oleksiivska" * рік відкриття: 2010 * Масив годин роботи: | | |
| 0 | 88 | "Very low ridership" |
| 1 | 817 | "High ridership" |
| 2 | 1115 | "Very high ridership" |
| Тестування на рівність масивів, отримання hashcode станцій метро:  Два масиви із однаковими даними:   * назва станції: "Oleksiivska" * рік відкриття: 2010 * Масив годин роботи: | | |
| 0 | 88 | "Very low ridership" |
| 1 | 817 | "High ridership" |
| 2 | 1115 | "Very high ridership" |

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

|  |
| --- |
| Create Metro Station with default constructor:  Station: Name: 'null'. Opened: 0. Hours:  There are no hours for this station.  Create Metro Station with parameterized constructor without Operating Hours:  Station: Name: 'Sportyvna'. Opened: 1975. Hours:  There are no hours for this station.  Create Metro Station with parameterized constructor with 3 Operating Hours:  Station: Name: 'Oleksiivska'. Opened: 2010. Hours:  Hour { ridership = 23, comment = 'Very low ridership' }  Hour { ridership = 345, comment = 'Medium ridership' }  Hour { ridership = 87, comment = 'Low ridership' }  Reset all Operating Hours for 'Oleksiivska' Metro Station:  Station: Name: 'Oleksiivska'. Opened: 2010. Hours:  Hour { ridership = 256, comment = 'Medium ridership' }  Hour { ridership = 817, comment = 'High ridership' }  Hour { ridership = 1115, comment = 'Very high ridership' }  Set new Operating Hour at index 0 for Metro Station 'Oleksiivska':  Operating Hours for Metro Station 'Oleksiivska':  Hour { ridership = 88, comment = 'Very low ridership' }  Hour { ridership = 817, comment = 'High ridership' }  Hour { ridership = 1115, comment = 'Very high ridership' }  Get Operating Hour for Metro Station 'Oleksiivska' at index 1:  Hour { ridership = 817, comment = 'High ridership' }  Count Operating Hours of Metro Station 'Oleksiivska': 3  Get info about Metro Station 1: Name: 'Sportyvna' Opened: 1975  Get info about Metro Station 2: Name: 'Oleksiivska' Opened: 2010  Check for equal Metro Station 1 and 2: false  Hashcode of Metro Station 1: -1008436996  Hashcode of Metro Station 2: -1277635050  Duplicate all the data of Metro Station 2 into Metro Station 1.  Check for equal Metro Station 1 and 2 when its data are identical: true  Hashcode of Metro Station 1: -1277635050  Station: Name: 'Oleksiivska'. Opened: 2010. Hours:  Hour { ridership = 88, comment = 'Very low ridership' }  Hour { ridership = 817, comment = 'High ridership' }  Hour { ridership = 1115, comment = 'Very high ridership' }  Hashcode of Metro Station 2: -1277635050  Station: Name: 'Oleksiivska'. Opened: 2010. Hours:  Hour { ridership = 88, comment = 'Very low ridership' }  Hour { ridership = 817, comment = 'High ridership' }  Hour { ridership = 1115, comment = 'Very high ridership' } |

**1.2.3 Тестовий випадок № 3** **“Тестування класу MetroStationWithSorting з методами класу CommentComparator”**

Опис: Тестування функціональності класу MetroStationWithSorting із сортуванням даних з перевизначеним методом класу CommentComparator.

Тестові дані:

Таблиця 1.3 – Тестові дані для класу MetroStationWithSorting

|  |  |  |
| --- | --- | --- |
| Індекс | Пасажиропотік | Коментар |
| Тестування конструктора з параметрами – валідні дані станції 1:   * назва станції: "Kholodna Hora" * рік відкриття: 1975 | | |
| Тестування конструктора з параметрами – валідні дані станції 2:   * назва станції: "Armiiska" * рік відкриття: 1978 * Масив годин роботи: | | |
| 0 | 23 | "Very low ridership" |
| 1 | 345 | "Medium ridership" |
| 2 | 87 | "Low ridership" |
| **Тестування сортування даних класу MetroStationWithSorting** | | |
| 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, 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. Виведення у консоль повідомлення з результатами:

|  |
| --- |
| Create Metro Station with default constructor:  Station: Name: 'null'. Opened: 0. Hours:  There are no hours for this station.  Create Metro Station with parameterized constructor without Operating Hours:  Station: Name: 'Kholodna Hora'. Opened: 1975. Hours:  There are no hours for this station.  Create Metro Station with parameterized constructor with 3 Operating Hours:  Station: Name: 'Armiiska'. Opened: 1978. Hours:  Hour { ridership = 23, comment = 'Very low ridership' }  Hour { ridership = 345, comment = 'Medium ridership' }  Hour { ridership = 87, comment = 'Low ridership' }  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 true  Add one Operating Hour with invalid data at Metro Station: false  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 = 470, comment = 'Medium ridership' }  Hour { ridership = 105, comment = 'Low ridership' }  Hour { ridership = 688, comment = 'High ridership' }  Hour { ridership = 1234, comment = 'Very high ridership' }  Sort Operating Hours of by decreasing ridership:  Station: Name: 'Derzhprom'. Opened: 1995. Hours:  Hour { ridership = 1234, comment = 'Very high ridership' }  Hour { ridership = 688, comment = 'High ridership' }  Hour { ridership = 470, comment = 'Medium ridership' }  Hour { ridership = 320, comment = 'Medium ridership' }  Hour { ridership = 105, comment = 'Low ridership' }  Hour { ridership = 88, comment = 'Very low ridership' }  Sort Operating Hours by descending comment length:  Station: Name: 'Derzhprom'. Opened: 1995. Hours:  Hour { ridership = 1234, comment = 'Very high ridership' }  Hour { ridership = 88, comment = 'Very low ridership' }  Hour { ridership = 470, comment = 'Medium ridership' }  Hour { ridership = 320, comment = 'Medium ridership' }  Hour { ridership = 688, comment = 'High ridership' }  Hour { ridership = 105, comment = 'Low ridership' } |

**1.2.4 Тестовий випадок № 4** **“Тестування класу MetroStationWithArray з методами класу AbstractMetroStation”**

Опис: Тестування функціональності класу MetroStationWithArray з використанням методів пошуку даних та сортування абстрактного класу AbstractMetroStation.

Тестові дані:

Таблиця 1.4 – Тестові дані для класу MetroStationWithArray

|  |  |  |
| --- | --- | --- |
| Індекс | Пасажиропотік | Коментар |
| **Тестування пошуку даних класу MetroStationWithArray** | | |
| Тестування встановлення значень назви та року відкриття станції метро 1, видалення даних масиву годин роботи, пошуку загальної кількості пасажиропотоку по станції, пошуку годин із мінімальним пасажиропотоком та пошуку годин із максимальної кількістю слів в коментарі при відсутності даних масиву годин роботи:   * назва станції: " Universytet" * рік відкриття: 1984 * Масив годин роботи: відсутній | | |
| Тестування створення масиву годин роботи станції метро 1 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 1 з послідовним додаванням даних окремих годин роботи в кінець масиву годин роботи – невалідні дані: | | |
|  | -1 | null |
| Тестування створення масиву годин роботи станції метро 1 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 1 пошуку загальної кількості пасажиропотоку по станції, пошуку годин із мінімальним пасажиропотоком та пошуку годин із максимальної кількістю слів в коментарі | | |
| 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" |
| Тестування пошуку загальної кількості пасажиропотоку по станції, пошуку годин із мінімальним пасажиропотоком та пошуку годин із максимальної кількістю слів в коментарі після послідовного додавання нових окремих годин роботи із мінімальним пасажиропотоком і із максимальною кількісю слів в коментарі: | | |
|  | 75 | "Very low ridership" |
|  | 2000 | "Maximum possible ridership for station" |
| **Тестування сортування даних класу MetroStationWithArray** | | |
| Тестування встановлення значень назви та року відкриття станції метро 2, 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 2 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 2 з послідовним додаванням даних окремих годин роботи в кінець масиву годин роботи – невалідні дані: | | |
|  | -1 | null |
| Тестування створення масиву годин роботи станції метро 2 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 2 сортування за зменшенням пасажиропотоку та сортування 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. Виведення у консоль повідомлення з результатами:

|  |
| --- |
| SEARCHING RESULTS:  Search data for Metro Station without Operating Hours:  Total ridership for station: There is no ridership hours.  Hours with minimal ridership: There is no ridership hours.  Hours with the maximum word count in a comment: There is no ridership hours.  Create the Metro Station:  Add 6 valid Operating Hours at Metro Station:  true true true true true true  Add one Operating Hour with invalid data at Metro Station: false  Add one Operating Hour with duplicate data at Metro Station: false  Station: Name: 'Universytet'. Opened: 1984. Hours:  Hour { ridership = 320, comment = 'Medium ridership' }  Hour { ridership = 88, comment = 'Very low ridership' }  Hour { ridership = 470, comment = 'Medium ridership' }  Hour { ridership = 105, comment = 'Low ridership' }  Hour { ridership = 688, comment = 'High ridership' }  Hour { ridership = 1234, comment = 'Very high ridership' }  Total ridership for station: 2905  Hours with minimal ridership:  Hour { ridership = 88, comment = 'Very low ridership' }  Hours with the maximum word count in a comment:  Hour { ridership = 88, comment = 'Very low ridership' }  Hour { ridership = 1234, comment = 'Very high ridership' }  Add new two Operating Hours with min ridership and max word count in comment for searching:  true  true  Station: Name: 'Universytet'. Opened: 1984. Hours:  Hour { ridership = 320, comment = 'Medium ridership' }  Hour { ridership = 88, comment = 'Very low ridership' }  Hour { ridership = 470, comment = 'Medium ridership' }  Hour { ridership = 105, comment = 'Low ridership' }  Hour { ridership = 688, comment = 'High ridership' }  Hour { ridership = 1234, comment = 'Very high ridership' }  Hour { ridership = 75, comment = 'Very low ridership' }  Hour { ridership = 2000, comment = 'Maximum possible ridership for station' }  Total ridership for station: 4980  Hours with minimal ridership:  Hour { ridership = 75, comment = 'Very low ridership' }  Hours with the maximum word count in a comment:  Hour { ridership = 2000, comment = 'Maximum possible ridership for station' }  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 true  Add one Operating Hour with invalid data at Metro Station: false  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 = 470, comment = 'Medium ridership' }  Hour { ridership = 105, comment = 'Low ridership' }  Hour { ridership = 688, comment = 'High ridership' }  Hour { ridership = 1234, comment = 'Very high ridership' }  Sort Operating Hours of by decreasing ridership:  Station: Name: 'Derzhprom'. Opened: 1995. Hours:  Hour { ridership = 1234, comment = 'Very high ridership' }  Hour { ridership = 688, comment = 'High ridership' }  Hour { ridership = 470, comment = 'Medium ridership' }  Hour { ridership = 320, comment = 'Medium ridership' }  Hour { ridership = 105, comment = 'Low ridership' }  Hour { ridership = 88, comment = 'Very low ridership' }  Sort Operating Hours by descending comment length:  Station: Name: 'Derzhprom'. Opened: 1995. Hours:  Hour { ridership = 1234, comment = 'Very high ridership' }  Hour { ridership = 88, comment = 'Very low ridership' }  Hour { ridership = 470, comment = 'Medium ridership' }  Hour { ridership = 320, comment = 'Medium ridership' }  Hour { ridership = 688, comment = 'High ridership' }  Hour { ridership = 105, comment = 'Low ridership' } |

**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()[1 + i].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 hour = getHours()[i];  int j;   for (j = i - 1; j >= 0  && Integer.*compare*(getHours()[j].getCommentLength(), hour.getCommentLength()) < 0; j--) {  getHours()[j + 1] = getHours()[j];  }   getHours()[j + 1] = hour;  }  }   */\*\*  \* 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(470, "Medium ridership") + "\t"  + addHour(105, "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 testSortigData() {  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 of 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 “MetroStationWithArray.java”

|  |
| --- |
| package task1;  */\*\*  \* Represents metro station data with an array of operating hours.  \* This class is inherited from the abstract {****@link*** *AbstractMetroStation} class.  \*/* public class MetroStationWithArray extends AbstractMetroStation {  */\*\* Name of the metro station. \*/* private String name;   */\*\* Opened year of the metro station. \*/* private int opened;   */\*\* Operating hours of the metro station. \*/* private Hour[] hours;   */\*\*  \* The constructor initialises the metro station object with the default values.  \*/* public MetroStationWithArray() {}   */\*\*  \* 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 MetroStationWithArray(String name, int opened) {  this(name, opened, new Hour[0]);  }   */\*\*  \* 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 array of operating hours of metro station.  \*/* public MetroStationWithArray(String name, int opened, Hour[] hours) {  this.name = name;  this.opened = opened;  this.hours = hours;  }   */\*\*  \* 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;  }   */\*\*  \* Gets the array of operating hours for the metro station.  \** ***@return*** *the array of hours.  \*/* @Override  public Hour[] getHours() {  return hours;  }   */\*\*  \* Sets the array of operating hours for the metro station.  \** ***@param*** *hours the array of hours to be set.  \*/* @Override  public void setHours(Hour[] hours) {  this.hours = hours;  }   */\*\*  \* Gets the {****@code*** *hour} with index {****@code*** *i} from the hours array.  \** ***@return*** *the object of class {****@code*** *Hour} with index {****@code*** *i}.  \*/* @Override  public Hour getHour(int i) {  if (getHours() == null  || getHours().length == 0  || i < 0  || i >= getHours().length) {  return null;  }   return getHours()[i];  }   */\*\*  \* Sets the {****@code*** *hour} with index {****@code*** *i} to hours array.  \** ***@param*** *i index of {****@code*** *hour} in hours array;  \** ***@param*** *hour the object of class {****@code*** *Hour} with index {****@code*** *i} to be set.  \*/* @Override  public void setHour(int i, Hour hour) {  if (getHours() == null  || getHours().length == 0  || i < 0  || i >= getHours().length) {  return;  }   getHours()[i] = hour;  }   */\*\*  \* Adds a link to the new {****@code*** *hour} at the end of the hours array.  \** ***@param*** *hour the object of class {****@code*** *Hour} to be added to the hours array;  \** ***@return*** *{****@code*** *true}, if the link was added successfully, {****@code*** *false} otherwise.  \*/* @Override  public boolean addHour(Hour hour) {  if (getHours() != null) {  if (hour.getRidership() < 0  || hour.getComment() == null) {  return false;  }   for (Hour h : getHours()) {  if (h.equals(hour)) {  return false;  }  }  }   setHours(*addHourToArray*(getHours(), hour));   return true;  }   */\*\*  \* Creates a new {****@code*** *hour} and adds a link to it at the end of the sequence at the hours array.  \** ***@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) {  if (ridership < 0  || comment == null) {  return false;  }   Hour newHour = new Hour(ridership, comment);   return addHour(newHour);  }   */\*\*  \* Counts the number of hours in the sequence at hours array.  \** ***@return*** *the number of hours.  \*/* @Override  public int countHours() {  if (getHours() == null) {  return 0;  }   return getHours().length;  }   */\*\*  \* Removes the sequence of hours from hours array.  \*/* @Override  public void removeHours() {  setHours(null);  }   */\*\*  \* Performs testing of the functionality of the {****@code*** *MetroStationWithArray} class.  \*/* public void testMetroStationWithArray() {  System.*out*.println("Create Metro Station with default constructor:");  MetroStationWithArray metroStation = new MetroStationWithArray();  System.*out*.println(metroStation);   System.*out*.println("Create Metro Station with parameterized constructor without Operating Hours:");  MetroStationWithArray metroStation1 = new MetroStationWithArray("Sportyvna", 1975);  System.*out*.print(metroStation1);   System.*out*.println("Create Metro Station with parameterized constructor with 3 Operating Hours:");  Hour[] hours1 = {  new Hour(23, "Very low ridership"),  new Hour(345, "Medium ridership"),  new Hour(87, "Low ridership")  };  MetroStationWithArray metroStation2 = new MetroStationWithArray("Oleksiivska", 2010, hours1);  System.*out*.println(metroStation2);   System.*out*.println("Reset all Operating Hours for \'" + metroStation2.getName() + "\' Metro Station:");  Hour[] hours2 = {  new Hour(256, "Medium ridership"),  new Hour(817, "High ridership"),  new Hour(1115, "Very high ridership")  };  metroStation2.setHours(hours2);  System.*out*.println(metroStation2);   System.*out*.println("Set new Operating Hour at index 0 for Metro Station \'" + metroStation2.getName() + "\':");  metroStation2.setHour(0, new Hour(88, "Very low ridership"));  System.*out*.println("Operating Hours for Metro Station \'" + metroStation2.getName() + "\':");  for (Hour h : metroStation2.getHours()) {  System.*out*.println(h);  }  System.*out*.println();   int index = 1;  System.*out*.println("Get Operating Hour for Metro Station \'" + metroStation2.getName()  + "\' at index " + index + ":");  System.*out*.println(metroStation2.getHour(index));  System.*out*.println();   System.*out*.println("Count Operating Hours of Metro Station \'" + metroStation2.getName() + "\':\t"  + metroStation2.countHours());  System.*out*.println();   System.*out*.println("Get info about Metro Station 1:\t" + "Name:\t\'" + metroStation1.getName() + "\'\tOpened:\t"  + metroStation1.getOpened());  System.*out*.println("Get info about Metro Station 2:\t" + "Name:\t\'" + metroStation2.getName() + "\'\tOpened:\t"  + metroStation2.getOpened());  System.*out*.println();   System.*out*.println("Check for equal Metro Station 1 and 2:\t" + metroStation1.equals(metroStation2));  System.*out*.println("Hashcode of Metro Station 1:\t" + metroStation1.hashCode());  System.*out*.println("Hashcode of Metro Station 2:\t" + metroStation2.hashCode());  System.*out*.println();   System.*out*.println("Duplicate all the data of Metro Station 2 into Metro Station 1.");  metroStation1.setName(metroStation2.getName());  metroStation1.setOpened(metroStation2.getOpened());  metroStation1.setHours(metroStation2.getHours());  System.*out*.println("Check for equal Metro Station 1 and 2 when its data are identical:\t"  + metroStation1.equals(metroStation2));  System.*out*.println("Hashcode of Metro Station 1:\t" + metroStation1.hashCode());  System.*out*.println(metroStation1);  System.*out*.println("Hashcode of Metro Station 2:\t" + metroStation2.hashCode());  System.*out*.println(metroStation2);  } } |

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

|  |
| --- |
| package task1;  import java.util.Comparator;  */\*\*  \* The class that implements the Comparator interface for sorting hours by descending comment length.  \* This class implements the {****@code*** *Comparator} interface for objects of type {****@code*** *Hour}.  \*/* public class CommentComparator implements Comparator<Hour> {  */\*\*  \* Compares two hours based on comment length.  \** ***@param*** *firstHour the first hour to compare;  \** ***@param*** *secondHour the second hour to compare.  \** ***@return*** *negative number, if {****@code*** *secondHour} length is larger than {****@code*** *firstHour},  \* zero, if their length are equal, positive number, if {****@code*** *secondHour} length is smaller than {****@code*** *firstHour}.  \*/* @Override  public int compare(Hour firstHour, Hour secondHour) {  return Integer.*compare*(secondHour.getCommentLength(), firstHour.getCommentLength());  } } |

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

|  |
| --- |
| package task1;  import java.util.Arrays;  */\*\*  \* Represents metro station data with an array of operating hours and overridden sorting methods.  \* This class extends (inherits from) the {****@link*** *MetroStationWithArray} class.  \* Also, this class is inherited from the abstract {****@link*** *AbstractMetroStation} class.  \*/* public class MetroStationWithSorting extends MetroStationWithArray {  */\*\*  \* The constructor initialises the metro station with the default values.  \*/* public MetroStationWithSorting() {}   */\*\*  \* The constructor initialises the metro station with the specified values with {****@code*** *name} and {****@code*** *opened} year.  \** ***@param*** *name the name;  \** ***@param*** *opened the opened year.  \*/* public MetroStationWithSorting(String name, int opened) {  super(name, opened);  }   */\*\*  \* The constructor initialises the metro station object with the specified values with {****@code*** *name},  \* {****@code*** *opened} year and operating {****@code*** *hour}.  \** ***@param*** *name the name;  \** ***@param*** *opened the opened year;  \** ***@param*** *hours the array of hours.  \*/* public MetroStationWithSorting(String name, int opened, Hour[] hours) {  super(name, opened, hours);  }   */\*\*  \* Overridden decreasing ridership sorting method using the standard sort function of class {****@code*** *Arrays}.  \* This method is provided by the implementation of the Comparable interface for objects of type {****@code*** *Hour}.  \*/* @Override  public void sortByDecreasingRidership() {  if (getHours() == null) {  return;  }   Arrays.*sort*(getHours());  }   */\*\*  \* Overridden descending comment length sorting method using the standard sort function of class {****@code*** *Arrays}  \* and {****@code*** *Comparator}. This method is provided by creating a separate class {****@code*** *CommentComparator}  \* that implements the {****@code*** *Comparator} interface for objects of type {****@code*** *Hour}.  \*/* @Override  public void sortByDescendingCommentLength() {  if (getHours() == null) {  return;  }   Arrays.*sort*(getHours(), new CommentComparator());  }   */\*\*  \* Performs testing of the functionality of the {****@code*** *MetroStationWithSorting} class.  \*/* public void testMetroStationWithSorting() {  System.*out*.println("Create Metro Station with default constructor:");  MetroStationWithSorting metroStation = new MetroStationWithSorting();  System.*out*.println(metroStation);   System.*out*.println("Create Metro Station with parameterized constructor without Operating Hours:");  MetroStationWithSorting metroStation1 = new MetroStationWithSorting("Kholodna Hora", 1975);  System.*out*.println(metroStation1);   System.*out*.println("Create Metro Station with parameterized constructor with 3 Operating Hours:");  Hour[] hours1 = {  new Hour(23, "Very low ridership"),  new Hour(345, "Medium ridership"),  new Hour(87, "Low ridership")  };  MetroStationWithSorting metroStation2 = new MetroStationWithSorting("Armiiska", 1978, hours1);  System.*out*.println(metroStation2);  metroStation2.testSortigData();  } } |

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

|  |
| --- |
| package task1;  */\*\*  \* The {****@code*** *Main} class performs testing of classes functionality.  \*/* public class Main {  */\*\*  \* Calls the test methods for testing the functionality of the {****@link*** *Hour} class.  \*/* public static void testingHour() {  System.*out*.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  System.*out*.println("TESTING HOUR:");  System.*out*.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  Hour hour = new Hour();  hour.testHour();  System.*out*.println("\n");  }   */\*\*  \* Calls the test methods for testing the functionality of the {****@link*** *MetroStationWithArray} class.  \*/* public static void testingMetroStationWithArray(){  System.*out*.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  System.*out*.println("TESTING METRO STATION WITH ARRAY:");  System.*out*.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  MetroStationWithArray metroStation1 = new MetroStationWithArray();  metroStation1.testMetroStationWithArray();  System.*out*.println();  }   */\*\*  \* Calls the test methods for testing the functionality of the {****@link*** *MetroStationWithSorting} class.  \*/* public static void testingMetroStationWithSorting() {  System.*out*.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  System.*out*.println("TESTING METRO STATION WITH SORTING:");  System.*out*.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  MetroStationWithSorting metroStation = new MetroStationWithSorting();  metroStation.testMetroStationWithSorting();  System.*out*.println();  }   */\*\*  \* Calls the test methods which inherit from the abstract {****@link*** *AbstractMetroStation} class  \* for testing the functionality of the {****@link*** *MetroStationWithArray} class.  \*/* public static void testingMetroStationWithArrayWithSearchAndSort() {  System.*out*.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  System.*out*.println("TESTING METRO STATION WITH ARRAY FOR SEARCHING AND SORTING:");  System.*out*.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  MetroStationWithArray metroStation2 = new MetroStationWithArray();  metroStation2.testSearchData();  System.*out*.println("\n");  metroStation2.testSortigData();  System.*out*.println();  }   */\*\*  \* Performs testing of classes functionality.  \* {****@code*** *args} are not used;  \** ***@param*** *args the command-line arguments.  \*/* public static void main(String[] args) {  *testingHour*();  *testingMetroStationWithArray*();  *testingMetroStationWithSorting*();  *testingMetroStationWithArrayWithSearchAndSort*();  } } |

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 “lab3”

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

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

Автоматически созданное описание

Рисунок 1.4 - Результати роботи програмного коду відповідно до тестового випадку № 1 “Тестування класу Hour” частина 1

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

Автоматически созданное описание

Рисунок 1.5 - Результати роботи програмного коду відповідно до тестового випадку № 1 “Тестування класу Hour” частина 2

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

Автоматически созданное описание

Рисунок 1.6 - Результати роботи програмного коду відповідно до тестового випадку № 2 “Тестування класу MetroStationWithArray” частина 1

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

Автоматически созданное описание**

Рисунок 1.7 - Результати роботи програмного коду відповідно до тестового випадку № 2 “Тестування класу MetroStationWithArray” частина 2

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

Автоматически созданное описание

Рисунок 1.8 - Результати роботи програмного коду відповідно до тестового випадку № 3 “Тестування класу MetroStationWithSorting з методами класу CommentComparator” частина 1

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

Автоматически созданное описание

Рисунок 1.9 - Результати роботи програмного коду відповідно до тестового випадку № 3 “Тестування класу MetroStationWithSorting з методами класу CommentComparator” частина 2

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

Автоматически созданное описание

Рисунок 1.10 - Результати роботи програмного коду відповідно до тестового випадку № 4 “Тестування класу MetroStationWithArray з методами класу AbstractMetroStation” частина 1

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

Автоматически созданное описание

Рисунок 1.11 - Результати роботи програмного коду відповідно до тестового випадку № 4 “Тестування класу MetroStationWithArray з методами класу AbstractMetroStation” частина 2

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

Автоматически созданное описание

Рисунок 1.12 - Результати роботи програмного коду відповідно до тестового випадку № 4 “Тестування класу MetroStationWithArray з методами класу AbstractMetroStation” частина 3

**2.1 Завдання № 2 “Ієрархія класів”**

Реалізувати класи "Людина", "Громадянин", "Студент", "Співробітник". Створити масив посилань на різні об'єкти ієрархії. Для кожного об'єкта вивести на екран рядок даних про нього.

Примітка: слід створити класи зі змістовними іменами.

**2.2 Набір тестових даних завдання № 2**

**2.2.1 Тестовий випадок № 1 “****Тестові дані для класів «Людина», «Громадянин», «Студент», «Співробітник»”**

Тестові дані для класу «Людина»:

* Ім’я: "Inessa",
* Прізвище: "Repeshko",
* Стать: "Female",
* Вік: 25.

Тестові дані для класу «Громадянин»:

* Ім’я: "Inessa",
* Прізвище: "Repeshko",
* Стать: "Female",
* Вік: 25,
* Країна: "Ukraine",
* Громадянство: "Ukrainian".

Тестові дані для класу «Студент»:

* Ім’я: "Inessa",
* Прізвище: "Repeshko",
* Стать: "Female",
* Вік: 25,
* Країна: "Ukraine",
* Громадянство: "Ukrainian",
* Університет: "NTU \"KhPI\"",
* Ідентифікатор спеціальності: 121
* Номер групи: "KN-222a"
* Рік вступу: 2020.

Тестові дані для класу «Працівник»:

* Ім’я: "Inessa",
* Прізвище: "Repeshko",
* Стать: "Female",
* Вік: 25,
* Країна: "Ukraine",
* Громадянство: "Ukrainian",
* Компанія: "\"Software\"",
* Департамент: "Development",
* Позиція: "Junior Developer",
* Стаж роботи: 1.

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

|  |
| --- |
| Human personal data:  First Name: Inessa  Last Name: Repeshko  Sex: Female  Age: 25  Citizen personal data:  First Name: Inessa  Last Name: Repeshko  Sex: Female  Age: 25  Country: Ukraine  Citizenship: Ukrainian  Student personal data:  First Name: Inessa  Last Name: Repeshko  Sex: Female  Age: 25  Country: Ukraine  Citizenship: Ukrainian  University: NTU "KhPI"  Speciality ID: 121  Group ID: KN-222a  Entering Year: 2020  Employee personal data:  First Name: Inessa  Last Name: Repeshko  Sex: Female  Age: 25  Country: Ukraine  Citizenship: Ukrainian  Company: "Software"  Department: Development  Position: Junior Developer  Service(years): 1 |

**2.3 Програмний код завдання № 2**

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

|  |
| --- |
| package task2;  class Human {  private String firstName;  private String lastName;  private String sex;  private int age;   public Human(  String firstName,  String lastName,  String sex,  int age) {  this.firstName = firstName;  this.lastName = lastName;  this.sex = sex;  this.age = age;  }   public String getFirstName() { return firstName; }   public String getLastName() { return lastName; }   public String getSex() { return sex; }   public int getAge() { return age; }   @Override  public String toString() {  return "First Name:\t\t" + getFirstName() + "\n" +  "Last Name:\t\t" + getLastName() + "\n" +  "Sex:\t\t\t" + getSex() + "\n" +  "Age:\t\t\t" + getAge() + "\n";  }   public void printData() {  System.*out*.println(this);  } } |

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

|  |
| --- |
| package task2;  class Citizen extends Human {  private String country;  private String citizenship;   public Citizen(  String firstName,  String lastName,  String sex,  int age,  String country,  String citizenship) {  super(firstName, lastName, sex, age);  this.country = country;  this.citizenship = citizenship;  }   public String getCountry() { return country; }   public String getCitizenship() { return citizenship; }   @Override  public String toString() {  return super.toString() +  "Country:\t\t" + getCountry() + "\n" +  "Citizenship:\t" + getCitizenship() + "\n";  }   @Override  public void printData() {  System.*out*.println(this);  } } |

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

|  |
| --- |
| package task2;  class Student extends Citizen {  private String university;  private int specialityID;  private String groupID;  private int enteringYear;   public Student(  String firstName,  String lastName,  String sex,  int age,  String country,  String citizenship,  String university,  int specialityID,  String groupID,  int enteringYear) {  super(firstName, lastName, sex, age, country, citizenship);  this.university = university;  this.specialityID = specialityID;  this.groupID = groupID;  this.enteringYear = enteringYear;  }   public String getUniversity() { return university; }   public int getSpecialityID() { return specialityID; }   public String getGroupID() { return groupID; }   public int getEnteringYear() { return enteringYear; }   @Override  public String toString() {  return super.toString() +  "University:\t\t" + getUniversity() + "\n" +  "Speciality ID:\t" + getSpecialityID() + "\n" +  "Group ID:\t\t" + getGroupID() + "\n" +  "Entering Year:\t" + getEnteringYear() + "\n";  }   @Override  public void printData() {  System.*out*.println(this);  } } |

Лістинг 2.4 - Програмний код реалізований у файлі “ Employee.java”

|  |
| --- |
| package task2;  class Employee extends Citizen {  private String company;  private String department;  private String position;  private int yearsOfService;   public Employee(  String firstName,  String lastName,  String sex,  int age,  String country,  String citizenship,  String company,  String department,  String position,  int yearsOfService) {  super(firstName, lastName, sex, age, country, citizenship);  this.company = company;  this.department = department;  this.position = position;  this.yearsOfService = yearsOfService;  }   public String getCompany() { return company; }   public String getDepartment() { return department; }   public String getPosition() { return position; }   public int getYearsOfService() { return yearsOfService; }   @Override  public String toString() {  return super.toString() +  "Company:\t\t" + getCompany() + "\n" +  "Department:\t\t" + getDepartment() + "\n" +  "Position:\t\t" + getPosition() + "\n" +  "Service(years):\t" + getYearsOfService() + "\n";  }   @Override  public void printData() {  System.*out*.println(this);  } } |

Лістинг 2.5 - Програмний код реалізований у файлі “ Main.java”

|  |
| --- |
| package task2;  public class Main {  public static void main(String[] args) {  Human human = new Human(  "Inessa",  "Repeshko",  "Female",  25);  Citizen citizen = new Citizen(  "Inessa",  "Repeshko",  "Female",  25,  "Ukraine",  "Ukrainian");  Student student = new Student(  "Inessa",  "Repeshko",  "Female",  25,  "Ukraine",  "Ukrainian",  "NTU \"KhPI\"",  121,  "KN-222a",  2020);  Employee employee = new Employee(  "Inessa",  "Repeshko",  "Female",  25,  "Ukraine",  "Ukrainian",  "\"Software\"",  "Development",  "Junior Developer",  1);  Human[] persons = {human, citizen, student, employee};   for (Human person : persons) {  System.*out*.println("Personal data:");  person.printData();  }  } } |

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

Изображение выглядит как текст, меню, снимок экрана

Автоматически созданное описание

Рисунок 2.1 - Результати роботи програмного коду відповідно до тестового випадку № 1 “ Тестові дані для класів «Людина», «Громадянин», «Студент», «Співробітник»” частина 1

Изображение выглядит как текст, меню, снимок экрана

Автоматически созданное описание

Рисунок 2.2 - Результати роботи програмного коду відповідно до тестового випадку № 1 “ Тестові дані для класів «Людина», «Громадянин», «Студент», «Співробітник»” частина 2

**3.1 Завдання № 3 “** **Мінімум функції”**

Реалізувати програму, що дозволяє знайти мінімум деякої функції на визначеному інтервалі. Алгоритм знаходження мінімуму полягає в послідовному переборі з певним кроком точок інтервалу і порівнянні значень функції в поточній точці з раніше знайденим мінімумом.

Реалізувати п'ять варіантів розв'язання:

* використання абстрактного та похідних класів
* опис інтерфейсу, створення класу, який використовує інтерфейс як тип параметру функції знаходження мінімуму, створення окремих класів, які реалізують інтерфейс
* використання попередньо описаного інтерфейсу і безіменних класів
* використання лямбда-виразів
* використання посилань на методи.

Перевірити роботу програми на двох різних функціях.

**3.2 Набір тестових даних завдання № 3**

**3.2.1 Тестовий випадок № 1 “****Валідний інтервал значень х”**

Опис: Тестові дані будуть використовуватись для тестування всіх методів обчислення. Очікуваний результат при використанні різних методів повинен бути однаковий для кожної математичної функції відповідно.

Тестові дані зі значеннями х на заданому інтервалі для пошуку значень функції ступеня x2 та функції експоненти ex:

* Початок інтервалу: -5
* Кінець інтервалу: 5.3
* Крок: 0.8

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

|  |
| --- |
| Interval for x:  start = -5.0;  end = 5.3;  step = 0.8.  First Method: using abstract class and derived classes.  The minimum value of a power function (x^2): 0.04000000000000016  The minimum value of a exponent function (e^x): 0.006737946999085467  Second Method: using interface and separate classes that implement it.  The minimum value of a power function (x^2): 0.04000000000000016  The minimum value of a exponent function (e^x): 0.006737946999085467  Third Method: using interface and anonymous classes.  The minimum value of a power function (x^2): 0.04000000000000016  The minimum value of a exponent function (e^x): 0.006737946999085467  Fourth Method: using lambda expressions.  The minimum value of a power function (x^2): 0.04000000000000016  The minimum value of a exponent function (e^x): 0.006737946999085467  Fifth Method: using references to methods.  The minimum value of a power function (x^2): 0.04000000000000016  The minimum value of a exponent function (e^x): 0.006737946999085467 |

**3.2.2 Тестовий випадок № 2 “****Невалідний інтервал значень х: end < start, step = 0”**

Опис: Тестові дані будуть використовуватись для тестування всіх методів обчислення. Очікуваний результат при використанні різних методів повинен бути однаковий для кожної математичної функції відповідно.

Тестові дані зі значеннями х на заданому інтервалі для пошуку значень функції ступеня x2 та функції експоненти ex:

* Початок інтервалу: 5.0
* Кінець інтервалу: -5.3
* Крок: 0

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

|  |
| --- |
| Incorrect interval values have been passed: step <= 0 and start > end. Try again! |

**3.2.3 Тестовий випадок № 3 “****Невалідний інтервал значень х: end < start”**

Опис: Тестові дані будуть використовуватись для тестування всіх методів обчислення. Очікуваний результат при використанні різних методів повинен бути однаковий для кожної математичної функції відповідно.

Тестові дані зі значеннями х на заданому інтервалі для пошуку значень функції ступеня x2 та функції експоненти ex:

* Початок інтервалу: 5.0
* Кінець інтервалу: -5.3
* Крок: 0.8

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

|  |
| --- |
| Incorrect interval values have been passed: start > end. Try again! |

**3.2.4 Тестовий випадок № 4 “****Невалідний інтервал значень х: step < 0”**

Опис: Тестові дані будуть використовуватись для тестування всіх методів обчислення. Очікуваний результат при використанні різних методів повинен бути однаковий для кожної математичної функції відповідно.

Тестові дані зі значеннями х на заданому інтервалі для пошуку значень функції ступеня x2 та функції експоненти ex:

* Початок інтервалу: -5.0
* Кінець інтервалу: 5.3
* Крок: -2.0

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

|  |
| --- |
| Incorrect interval values have been passed: step <= 0. Try again! |

**3.3 Програмний код завдання № 3**

Реалізація варіанту розв’язання з використанням абстрактного та похідних класів.

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

|  |
| --- |
| package task3;  abstract class FindFunctionValueAbstractClass {  abstract double findFunctionValue(double x); } |

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

|  |
| --- |
| package task3;  class PowFunctionWithAbstractClass extends FindFunctionValueAbstractClass {  @Override  double findFunctionValue(double x) {  return Math.*pow*(x, 2);  } } |

Лістинг 3.3 - Програмний код реалізований у файлі “ExpFunctionWithAbstractClass.java”

|  |
| --- |
| package task3;  class ExpFunctionWithAbstractClass extends FindFunctionValueAbstractClass {  @Override  double findFunctionValue(double x) {  return Math.*exp*(x);  } } |

Реалізація варіанту розв’язання з використанням:

* опис інтерфейсу, створення класу, який використовує інтерфейс як тип параметру функції знаходження мінімуму, створення окремих класів, які реалізують інтерфейс
* використання попередньо описаного інтерфейсу і безіменних класів
* використання лямбда-виразів
* використання посилань на методи.

Лістинг 3.4 - Програмний код реалізований у файлі “ Function.java”

|  |
| --- |
| package task3;  @FunctionalInterface interface Function {  double findFunctionValue(double x); } |

Лістинг 3.5 - Програмний код реалізований у файлі “PowFunctionWithInterface.java”

|  |
| --- |
| package task3;  class PowFunctionWithInterface implements Function {  @Override  public double findFunctionValue(double x) {  return Math.*pow*(x, 2);  } } |

Лістинг 3.6 - Програмний код реалізований у файлі “ExpFunctionWithInterface.java”

|  |
| --- |
| package task3;  class ExpFunctionWithInterface implements Function {  @Override  public double findFunctionValue(double x) {  return Math.*exp*(x);  } } |

Лістинг 3.7 - Програмний код реалізований у файлі “FindMinimumFunctions.java”

|  |
| --- |
| package task3;  class FindMinimumFunctions {  static void verifyInterval(double start, double end, double step) {  if (start > end && step <= 0) {  System.*err*.println("Incorrect interval values have been passed: step <= 0 and start > end. Try again!");  System.*exit*(1);  } else if (start > end) {  System.*err*.println("Incorrect interval values have been passed: start > end. Try again!");  System.*exit*(1);  } else if (step <= 0) {  System.*err*.println("Incorrect interval values have been passed: step <= 0. Try again!");  System.*exit*(1);  }  }   static double findMinFunctionValueUsingAbstractClass(double start, double end, double step, FindFunctionValueAbstractClass function) {  double minValue = function.findFunctionValue(start);   for (double x = start + step; x <= end; x += step) {  double value = function.findFunctionValue(x);   if (value < minValue) {  minValue = value;  }  }   return minValue;  }   static void usingAbstractClass(double start, double end, double step) {  *verifyInterval*(start, end, step);  PowFunctionWithAbstractClass powerFunction = new PowFunctionWithAbstractClass();  ExpFunctionWithAbstractClass exponentFunction = new ExpFunctionWithAbstractClass();  System.*out*.println("First Method: using abstract class and derived classes.");  System.*out*.println("The minimum value of a power function (x^2):\t\t"  + *findMinFunctionValueUsingAbstractClass*(start, end, step, powerFunction));  System.*out*.println("The minimum value of a exponent function (e^x):\t\t"  + *findMinFunctionValueUsingAbstractClass*(start, end, step, exponentFunction) + "\n");  }   static double findMinFunctionValueUsingInterface(double start, double end, double step, Function function) {  double minValue = function.findFunctionValue(start);   for (double x = start + step; x <= end; x += step) {  double value = function.findFunctionValue(x);   if (value < minValue) {  minValue = value;  }  }   return minValue;  }   static void usingInterface(double start, double end, double step) {  *verifyInterval*(start, end, step);  PowFunctionWithInterface powerFunction = new PowFunctionWithInterface();  ExpFunctionWithInterface exponentFunction = new ExpFunctionWithInterface();  System.*out*.println("Second Method: using interface and separate classes that implement it.");  System.*out*.println("The minimum value of a power function (x^2):\t\t"  + *findMinFunctionValueUsingInterface*(start, end, step, powerFunction));  System.*out*.println("The minimum value of a exponent function (e^x):\t\t"  + *findMinFunctionValueUsingInterface*(start, end, step, exponentFunction) + "\n");  }   static void usingAnonymousClasses(double start, double end, double step) {  *verifyInterval*(start, end, step);  System.*out*.println("Third Method: using interface and anonymous classes.");  System.*out*.println("The minimum value of a power function (x^2):\t\t"  + *findMinFunctionValueUsingInterface*(start, end, step, new Function() {  @Override  public double findFunctionValue(double x) {  return Math.*pow*(x, 2);  }  }));  System.*out*.println("The minimum value of a exponent function (e^x):\t\t"  + *findMinFunctionValueUsingInterface*(start, end, step, new Function() {  @Override  public double findFunctionValue(double x) {  return Math.*exp*(x);  }  }) + "\n");  }   static void usingLambdaExpressions(double start, double end, double step) {  *verifyInterval*(start, end, step);  System.*out*.println("Fourth Method: using lambda expressions.");  System.*out*.println("The minimum value of a power function (x^2):\t\t"  + *findMinFunctionValueUsingInterface*(start, end, step, x -> Math.*pow*(x, 2)));  System.*out*.println("The minimum value of a exponent function (e^x):\t\t"  + *findMinFunctionValueUsingInterface*(start, end, step, x -> Math.*exp*(x)) + "\n");  }   static double powerFunction(double x) {  return Math.*pow*(x, 2);  }   static double exponentFunction(double x) {  return Math.*exp*(x);  }   static void usingReferencesToMethods(double start, double end, double step) {  *verifyInterval*(start, end, step);  System.*out*.println("Fifth Method: using references to methods.");  System.*out*.println("The minimum value of a power function (x^2):\t\t"  + *findMinFunctionValueUsingInterface*(start, end, step, FindMinimumFunctions::*powerFunction*));  System.*out*.println("The minimum value of a exponent function (e^x):\t\t"  + *findMinFunctionValueUsingInterface*(start, end, step, FindMinimumFunctions::*exponentFunction*));  }   public static void main(String[] args) {  /\*  double start = 5.0,  end = -5.3,  step = 0;  System.out.println("INVALID DATA:\n" +  "Interval for x:\nstart =\t\t" + start + ";\nend =\t\t" + end + ";\nstep =\t\t" + step + ".\n");  \*/   /\*  double start = 5.0,  end = -5.3,  step = 0.8;  System.out.println("INVALID DATA:\n" +  "Interval for x:\nstart =\t\t" + start + ";\nend =\t\t" + end + ";\nstep =\t\t" + step + ".\n");  \*/   /\*  double start = -5.0,  end = 5.3,  step = -2;  System.out.println("INVALID DATA:\n" +  "Interval for x:\nstart =\t\t" + start + ";\nend =\t\t" + end + ";\nstep =\t\t" + step + ".\n");  \*/   double start = -5.0,  end = 5.3,  step = 0.8;  System.*out*.println("VALID DATA:\n" +  "Interval for x:\nstart =\t\t" + start + ";\nend =\t\t" + end + ";\nstep =\t\t" + step + ".\n");   *usingAbstractClass*(start, end, step);  *usingInterface*(start, end, step);  *usingAnonymousClasses*(start, end, step);  *usingLambdaExpressions*(start, end, step);  *usingReferencesToMethods*(start, end, step);  } } |

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

Изображение выглядит как текст, снимок экрана, Шрифт, меню

Автоматически созданное описание

Рисунок 3.1 - Результати роботи програмного коду відповідно до тестового випадку № 1 “ Валідний інтервал значень х”

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 3.2 - Результати роботи програмного коду відповідно до тестового випадку № 2 “Невалідний інтервал значень х: end < start, step = 0”

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 3.3 - Результати роботи програмного коду відповідно до тестового випадку № 3 “ Невалідний інтервал значень х: end < start”

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 3.4 - Результати роботи програмного коду відповідно до тестового випадку № 4 “ Невалідний інтервал значень х: step < 0”

## **4.1 Завдання № 4 “Реалізація масиву точок через масиви дійсних чисел”**

Реалізувати функціональність абстрактного класу AbstractArrayOfPoints, наведеного в прикладі 3.2, у два способи:

* через використання двовимірного масиву дійсних чисел: кожен рядок масиву має відповідати точці;
* через використання одновимірного масиву дійсних чисел: кожна пара чисел у масиві має відповідати точці.

Здійснити тестування створених класів.

Примітка: не слід вносити зміни у код класу AbstractArrayOfPoints (за винятком імені пакету і реалізації методу sortByY()).

**4.2 Набір тестових даних завдання № 4**

**4.2.1 Тестовий випадок № 1 “Тестові дані координат точок”**

Опис: Тестування функціональності класів AbstractArrayOfPoints, TwoDimensionalArrayOfPoints та OneDimensionalArrayOfPoints.

Тестові дані:

Таблиця 4.1 – Тестові дані для тестування функціональності класів AbstractArrayOfPoints, TwoDimensionalArrayOfPoints та OneDimensionalArrayOfPoints

|  |  |  |
| --- | --- | --- |
| Індекс точки | Координата Xi | Координата Yi |
| 1 | 22 | 45 |
| 2 | 14 | 11 |
| 3 | 30 | 5.5 |
| 4 | -2 | 58 |
| 5 | -13 | -24 |

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

|  |
| --- |
| One-Dimensional Array Of Points:  Array of added points:  point0 (x = 22.0; y = 45.0)  point1 (x = 14.0; y = 11.0)  point2 (x = 30.0; y = 5.5)  point3 (x = -2.0; y = 58.0)  point4 (x = -13.0; y = -24.0)  Array sorted ascending by X:  point0 (x = -13.0; y = -24.0)  point1 (x = -2.0; y = 58.0)  point2 (x = 14.0; y = 11.0)  point3 (x = 22.0; y = 45.0)  point4 (x = 30.0; y = 5.5)  Array sorted ascending by Y:  point0 (x = -13.0; y = -24.0)  point1 (x = 30.0; y = 5.5)  point2 (x = 14.0; y = 11.0)  point3 (x = 22.0; y = 45.0)  point4 (x = -2.0; y = 58.0)  Array where the last point is removed:  point0 (x = -13.0; y = -24.0)  point1 (x = 30.0; y = 5.5)  point2 (x = 14.0; y = 11.0)  point3 (x = 22.0; y = 45.0)  Array where coordinates of the point with index 0 changed:  point0 (x = -6.5; y = -12.0)  point1 (x = 30.0; y = 5.5)  point2 (x = 14.0; y = 11.0)  point3 (x = 22.0; y = 45.0)  Two-Dimensional Array Of Points:  Array of added points:  point0 (x = 22.0; y = 45.0)  point1 (x = 14.0; y = 11.0)  point2 (x = 30.0; y = 5.5)  point3 (x = -2.0; y = 58.0)  point4 (x = -13.0; y = -24.0)  Array sorted ascending by X:  point0 (x = -13.0; y = -24.0)  point1 (x = -2.0; y = 58.0)  point2 (x = 14.0; y = 11.0)  point3 (x = 22.0; y = 45.0)  point4 (x = 30.0; y = 5.5)  Array sorted ascending by Y:  point0 (x = -13.0; y = -24.0)  point1 (x = 30.0; y = 5.5)  point2 (x = 14.0; y = 11.0)  point3 (x = 22.0; y = 45.0)  point4 (x = -2.0; y = 58.0)  Array where the last point is removed:  point0 (x = -13.0; y = -24.0)  point1 (x = 30.0; y = 5.5)  point2 (x = 14.0; y = 11.0)  point3 (x = 22.0; y = 45.0)  Array where coordinates of the point with index 0 changed:  point0 (x = -6.5; y = -12.0)  point1 (x = 30.0; y = 5.5)  point2 (x = 14.0; y = 11.0)  point3 (x = 22.0; y = 45.0) |

**4.3 Програмний код завдання № 4**

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

|  |
| --- |
| package task4;  public abstract class AbstractArrayOfPoints {  public abstract void setPoint(int i, double x, double y);   public abstract double getX(int i);   public abstract double getY(int i);   public abstract int countPoints();   public abstract void addPointToEnd(double x, double y);   public abstract void removeLastPoint();   public void sortByX() {  boolean unsorted = true;   while (unsorted) {  unsorted = false;   for (int i = 0; i < countPoints() - 1; i++) {  if (getX(i) > getX(i + 1)) {  double x = getX(i);  double y = getY(i);  setPoint(i, getX(i + 1), getY(i + 1));  setPoint(i + 1, x, y);  unsorted = true;  }  }  }  }   public void sortByY() {  boolean unsorted = true;   while (unsorted) {  unsorted = false;   for (int i = 0; i < countPoints() - 1; i++) {  if (getY(i) > getY(i + 1)) {  double x = getX(i);  double y = getY(i);  setPoint(i, getX(i + 1), getY(i + 1));  setPoint(i + 1, x, y);  unsorted = true;  }  }  }  }   @Override  public String toString() {  String string = "";   for (int i = 0; i < countPoints(); i++) {  string += "\tpoint" + i  + "\t(x = " + getX(i)  + ";\ty = " + getY(i)  + ")\n";  }   return string;  }   public void testArrayOfPoints() {  System.*out*.println("Array of added points:");  addPointToEnd(22, 45);  addPointToEnd(14, 11);  addPointToEnd(30, 5.5);  addPointToEnd(-2, 58);  addPointToEnd(-13, -24);  System.*out*.println(this);   System.*out*.println("Array sorted ascending by X:");  sortByX();  System.*out*.println(this);   System.*out*.println("Array sorted ascending by Y:");  sortByY();  System.*out*.println(this);   System.*out*.println("Array where the last point is removed:");  removeLastPoint();  System.*out*.println(this);   System.*out*.println("Array where coordinates of the point with index 0 changed:");  setPoint(0, getX(0) / 2, getY(0) / 2);  System.*out*.println(this);  } } |

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

|  |
| --- |
| package task4;  public class TwoDimensionalArrayOfPoints extends AbstractArrayOfPoints {  private double[][] points = new double[0][2];   @Override  public int countPoints() {  return points.length;  }   @Override  public void setPoint(int i, double x, double y) {  if (i >= 0 && i < countPoints()) {  points[i][0] = x;  points[i][1] = y;  }  }   @Override  public double getX(int i) {  if (i >= 0 && i < countPoints()) {  return points[i][0];  }   return 0;  }   @Override  public double getY(int i) {  if (i >= 0 && i < countPoints()) {  return points[i][1];  }   return 0;  }   @Override  public void addPointToEnd(double x, double y) {  double[][] newArray = new double[countPoints() + 1][2];  System.*arraycopy*(points, 0, newArray, 0, countPoints());  newArray[countPoints()][0] = x;  newArray[countPoints()][1] = y;  points = newArray;  }   @Override  public void removeLastPoint() {  if (countPoints() > 0) {  double[][] newArray = new double[countPoints() - 1][2];  System.*arraycopy*(points, 0, newArray, 0, countPoints() - 1);  points = newArray;  }  }   public static void main(String[] args) {  TwoDimensionalArrayOfPoints arrayOfPoints1 = new TwoDimensionalArrayOfPoints();  System.*out*.println("Two-Dimensional Array Of Points:");  arrayOfPoints1.testArrayOfPoints();  } } |

Лістинг 4.3 - Програмний код реалізований у файлі “OneDimensionalArrayOfPoints.java”

|  |
| --- |
| package task4;  public class OneDimensionalArrayOfPoints extends AbstractArrayOfPoints {  private double[] points = new double[0];   @Override  public void setPoint(int i, double x, double y) {  if (i >= 0 && i \* 2 + 1 < points.length) {  points[i \* 2] = x;  points[i \* 2 + 1] = y;  }  }   @Override  public double getX(int i) {  if (i >= 0 && i \* 2 + 1 < points.length) {  return points[i \* 2];  }   return 0;  }   @Override  public double getY(int i) {  if (i >= 0 && i \* 2 + 1 < points.length) {  return points[i \* 2 + 1];  }   return 0;  }   @Override  public int countPoints() {  return points.length / 2;  }   @Override  public void addPointToEnd(double x, double y) {  double[] newArray = new double[points.length + 2];  System.*arraycopy*(points, 0, newArray, 0, points.length);  newArray[points.length] = x;  newArray[points.length + 1] = y;  points = newArray;  }   @Override  public void removeLastPoint() {  if (countPoints() > 0) {  double[] newArray = new double[points.length - 2];  System.*arraycopy*(points, 0, newArray, 0, points.length - 2);  points = newArray;  }  }   public static void main(String[] args) {  OneDimensionalArrayOfPoints arrayOfPoints = new OneDimensionalArrayOfPoints();  System.*out*.println("One-Dimensional Array Of Points:");  arrayOfPoints.testArrayOfPoints();  } } |

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

Изображение выглядит как текст, снимок экрана, дизайн

Автоматически созданное описание

Рисунок 4.1 - Результати роботи програмного коду відповідно до тестового випадку № 1 “Тестові дані координат точок” частина 1

Изображение выглядит как текст, снимок экрана, меню, дизайн

Автоматически созданное описание

Рисунок 4.2 - Результати роботи програмного коду відповідно до тестового випадку № 1 “Тестові дані координат точок” частина 2

## **5.1 Завдання № 5 “Реалізація інтерфейсу Comparable”**

Створити клас Circle, який реалізує інтерфейс Comparable. Більшим вважається коло з більшим радіусом. Здійснити сортування масиву об'єктів типу Circle за допомогою функції Arrays.sort().

**5.2 Набір тестових даних завдання № 5**

**5.2.1 Тестовий випадок № 1 “****Валідні дані радіусу кола”**

Тестові дані масиву кіл:

Таблиця 5.1 – Валідні тестові дані для масиву кіл

|  |  |
| --- | --- |
| Індекс | Радіус |
| 0 | 12.34 |
| 1 | 23.45 |
| 2 | 0.99 |
| 3 | 10 |
| 4 | 20.2 |
| 5 | 3.78 |
| 6 | 9.12 |

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

|  |
| --- |
| Initial array of circles:  Circle radius is 12.34 cm.  Circle radius is 23.45 cm.  Circle radius is 0.99 cm.  Circle radius is 10.0 cm.  Circle radius is 20.2 cm.  Circle radius is 3.78 cm.  Circle radius is 9.12 cm.  Sorted array of circles ascending by radius:  Circle radius is 0.99 cm.  Circle radius is 3.78 cm.  Circle radius is 9.12 cm.  Circle radius is 10.0 cm.  Circle radius is 12.34 cm.  Circle radius is 20.2 cm.  Circle radius is 23.45 cm. |

**5.2.2 Тестовий випадок № 2 “****Невалідні дані радіусу кола: radius < 0”**

Тестові дані масиву кіл:

Таблиця 5.2 – Невалідні тестові дані для масиву кіл

|  |  |
| --- | --- |
| Індекс | Радіус |
| 0 | -0.9 |
| 1 | 10 |

Очікуваний результат: Завершення виконання програми зі помилкою. Виведення у консоль повідомлення з помилкою:

|  |
| --- |
| Initial array of circles:  Invalid circle radius passed. The radius must be greater than zero. |

**5.2.3 Тестовий випадок № 3 “****Невалідні дані радіусу кола: radius = 0”**

Тестові дані масиву кіл:

Таблиця 5.3 – Невалідні тестові дані для масиву кіл

|  |  |
| --- | --- |
| Індекс | Радіус |
| 0 | 9 |
| 1 | 0 |

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

|  |
| --- |
| Initial array of circles:  Invalid circle radius passed. The radius must be greater than zero. |

**5.3 Програмний код завдання № 5**

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

|  |
| --- |
| package task5;  import java.util.Arrays;  public class Circle implements Comparable<Circle>{  private double radius;   public Circle(double radius) {   if (radius <= 0) {  System.*err*.println("Invalid circle radius passed.\n" +  "The radius must be greater than zero. Try again!");  System.*exit*(1);  }   this.radius = radius;  }   public double getRadius() {  return radius;  }   @Override  public int compareTo(Circle circle) {  return Double.*compare*(this.getRadius(), circle.getRadius());  }   @Override  public String toString() {  return "Circle radius is\t" + getRadius() + "\tcm.";  }   public static void printCirclesArray(Circle[] circles) {  for (Circle circle : circles) {  System.*out*.println(circle);  }  }   public static void main(String[] args) {  System.*out*.println("Initial array of circles:");  /\*  Circle[] circles = {  new Circle(-0.9),  new Circle(10)  };  \*/   /\*  Circle[] circles = {  new Circle(9),  new Circle(0)  };  \*/   Circle[] circles = {  new Circle(12.34),  new Circle(23.45),  new Circle(0.99),  new Circle(10),  new Circle(20.2),  new Circle(3.78),  new Circle(9.12)  };  *printCirclesArray*(circles);   System.*out*.println("\nSorted array of circles ascending by radius:");  Arrays.*sort*(circles);  *printCirclesArray*(circles);  } } |

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

Изображение выглядит как текст, снимок экрана, меню, Шрифт

Автоматически созданное описание

Рисунок 5.1 - Результати роботи програмного коду відповідно до тестового випадку № 1 “Валідні дані радіусу кола”

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

Рисунок 5.2 - Результати роботи програмного коду відповідно до тестового випадку № 2 “Невалідні дані радіусу кола: radius < 0”

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

Рисунок 5.3 - Результати роботи програмного коду відповідно до тестового випадку № 3 “Невалідні дані радіусу кола: radius = 0”

## **6.1 Завдання № 6 “Реалізація інтерфейсу Comparator”**

Створити клас Triangle. Трикутник визначати довжинами сторін. Площа трикутника в цьому випадку може бути обчислена за формулою Герона:

Heron

Рисунок 6.1 – Формула Герона

де a, b і c – довжини сторін трикутника. За допомогою функції Arrays.sort()здійснити сортування масиву трикутників за зменшенням площі. Для визначення ознаки сортування використовувати об'єкт, який реалізує інтерфейс Comparator.

**6.2 Набір тестових даних завдання № 6**

**6.2.1 Тестовий випадок № 1 “****Валідні дані сторін трикутників”**

Тестові дані масиву трикутників:

Таблиця 6.1 – Валідні тестові дані для масиву трикутників

|  |  |  |  |
| --- | --- | --- | --- |
| Індекс | Сторона a | Сторона b | Сторона c |
| 0 | 4 | 3 | 5 |
| 1 | 5.9 | 5.2 | 7.3 |
| 2 | 6 | 13 | 12 |
| 3 | 10.1 | 11.2 | 12.3 |
| 4 | 8.5 | 8.5 | 8.5 |
| 5 | 5 | 12 | 13 |

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

|  |
| --- |
| Initial array of triangles:  Triangle area is 2.4494 cm with edges: a = 4.0 b = 3.0 c = 5.0.  Triangle area is 5.3704 cm with edges: a = 5.9 b = 5.2 c = 7.3.  Triangle area is 16.0526 cm with edges: a = 6.0 b = 13.0 c = 12.0.  Triangle area is 15.9141 cm with edges: a = 10.1 b = 11.2 c = 12.3.  Triangle area is 10.7307 cm with edges: a = 8.5 b = 8.5 c = 8.5.  Triangle area is 12.2474 cm with edges: a = 5.0 b = 12.0 c = 13.0.  Sorted array of triangles descending by area:  Triangle area is 16.0526 cm with edges: a = 6.0 b = 13.0 c = 12.0.  Triangle area is 15.9141 cm with edges: a = 10.1 b = 11.2 c = 12.3.  Triangle area is 12.2474 cm with edges: a = 5.0 b = 12.0 c = 13.0.  Triangle area is 10.7307 cm with edges: a = 8.5 b = 8.5 c = 8.5.  Triangle area is 5.37047 cm with edges: a = 5.9 b = 5.2 c = 7.3.  Triangle area is 2.44948 cm with edges: a = 4.0 b = 3.0 c = 5.0. |

**6.2.2 Тестовий випадок № 2 “****Невалідні дані сторін трикутників: сторона < 0”**

Тестові дані масиву трикутників:

Таблиця 6.2 – Невалідні тестові дані для масиву трикутників

|  |  |  |  |
| --- | --- | --- | --- |
| Індекс | Сторона a | Сторона b | Сторона c |
| 0 | 10.0 | 9 | 8 |
| 1 | -2 | 10 | 10 |

Очікуваний результат: Завершення виконання програми зі помилкою. Виведення у консоль повідомлення з помилкою:

|  |
| --- |
| Initial array of triangles:  Invalid edge of triangle passed. The edge must be greater than zero. |

**6.2.3 Тестовий випадок № 3 “****Невалідні дані сторін трикутників: сторона = 0”**

Тестові дані масиву трикутників:

Таблиця 6.3 – Невалідні тестові дані для масиву трикутників

|  |  |  |  |
| --- | --- | --- | --- |
| Індекс | Сторона a | Сторона b | Сторона c |
| 0 | 0.09 | 8 | 0 |
| 1 | 2 | 3 | 4 |

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

|  |
| --- |
| Initial array of triangles:  Invalid edge of triangle passed. The edge must be greater than zero. |

**6.2.4 Тестовий випадок № 4 “****Невалідні дані сторін трикутників: a + b = c”**

Тестові дані масиву трикутників:

Таблиця 6.4 – Невалідні тестові дані для масиву трикутників

|  |  |  |  |
| --- | --- | --- | --- |
| Індекс | Сторона a | Сторона b | Сторона c |
| 0 | 5 | 4 | 3 |
| 1 | 4 | 3 | 7 |

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

|  |
| --- |
| Initial array of triangles:  Invalid edge of triangle passed. The sum of lengths of two edges must be greater than length of the third edge. |

**6.3 Програмний код завдання № 6**

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

|  |
| --- |
| package task6;  import java.util.Arrays; import java.util.Comparator;  public class Triangle {  private double edgeA, edgeB, edgeC;   public Triangle(double edgeA, double edgeB, double edgeC) {  if (edgeA <= 0  || edgeB <= 0  || edgeC <= 0) {  System.*err*.println("Invalid edge of triangle passed.\n" +  "The edge must be greater than zero. Try again!");  System.*exit*(1);  }   if (edgeB + edgeC <= edgeA  || edgeA + edgeC <= edgeB  || edgeA + edgeB <= edgeC) {  System.*err*.println("Invalid edge of triangle passed.\n" +  "The sum of lengths of two edges must be greater than length of the third edge. Try again!");  System.*exit*(1);  }   this.edgeA = edgeA;  this.edgeB = edgeB;  this.edgeC = edgeC;  }   public double getEdgeA() {  return edgeA;  }   public double getEdgeB() {  return edgeB;  }   public double getEdgeC() {  return edgeC;  }   public double calculateArea() {  double a = getEdgeA();  double b = getEdgeB();  double c = getEdgeC();  return 0.25 \* Math.*sqrt*((a + b + c) \* (b + c - a) \* (a + b - c));  }   @Override  public String toString() {  return "Triangle area is\t" + calculateArea()  + "\tcm with edges:\t"  + "a = " + getEdgeA()  + " b = " + getEdgeB()  + " c = " + getEdgeC()  + ".";  }   public static void printTrianglesArray(Triangle[] triangles) {  for (Triangle triangle : triangles) {  System.*out*.println(triangle);  }  }   public static void main(String[] args) {  System.*out*.println("Initial array of triangles:");  /\*  Triangle[] triangles = {  new Triangle(10.0, 9, 8),  new Triangle(-2, 10, 10)  };  \*/   /\*  Triangle[] triangles = {  new Triangle(0.09, 8, 0),  new Triangle(2, 3, 4)  };  \*/   /\*  Triangle[] triangles = {  new Triangle(5, 4, 3),  new Triangle(4, 3, 7)  };  \*/   Triangle[] triangles = {  new Triangle(4, 3, 5),  new Triangle(5.9, 5.2, 7.3),  new Triangle(6, 13,12),  new Triangle(10.1, 11.2, 12.3),  new Triangle(8.5, 8.5, 8.5),  new Triangle(5, 12, 13)  };  *printTrianglesArray*(triangles);   System.*out*.println("\nSorted array of triangles descending by area:");  Arrays.*sort*(triangles, new Comparator<Triangle>() {  @Override  public int compare(Triangle triangle1, Triangle triangle2) {  return Double.*compare*(triangle2.calculateArea(), triangle1.calculateArea());  }  });  *printTrianglesArray*(triangles);  } } |

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

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

Автоматически созданное описание

Рисунок 6.2 - Результати роботи програмного коду відповідно до тестового випадку № 1 “Валідні дані сторін трикутників”

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

Рисунок 6.3 - Результати роботи програмного коду відповідно до тестового випадку № 2 “Невалідні дані сторін трикутників: сторона < 0”

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

Рисунок 6.4 - Результати роботи програмного коду відповідно до тестового випадку № 3 “Невалідні дані сторін трикутників: сторона = 0”

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 6.5 - Результати роботи програмного коду відповідно до тестового випадку № 4 “Невалідні дані сторін трикутників: a + b = c”

## **7.1 Завдання № 7 “Обчислення визначеного інтегралу (додаткове завдання)”**

Створити інтерфейс Integrable, який містить опис абстрактної функції, що приймає аргумент типу double і повертає результат того ж типу. Інтерфейс повинен містити метод integral() з усталеною реалізацією (з модифікатором default) обчислення визначеного інтеграла. Метод повинен отримувати як параметри початок, кінець інтервалу і точність обчислень. Усталена реалізація обчислення інтегралу використовує метод прямокутників.

Створити клас, який перевизначає функцію integral(), реалізуючи метод трапецій обчислення визначеного інтеграла.

Обчислити визначений інтеграл за допомогою обох алгоритмів для різних математичних функцій класу java.lang.Math (див. приклад 3.3). Порівняти результати для різних алгоритмів і різних значень точності обчислення.

**7.2 Набір тестових даних завдання № 7**

**7.2.1 Тестовий випадок № 1 “****Тестування** **функції ступеня валідні дані”**

Опис: Дані для тестування двох методів обчислення визначеного інтеграла для функції ступеня . Знаходження значення функції на інтервалі з від’ємними значеннями.

Тестові дані інтервалу значень х:

* Початок інтервалу: -5
* Кінець інтервалу: 2
* Перше значення точності обчислень: 0.0001
* Друге значення точності обчислень: 0.00000001

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

|  |
| --- |
| Definite integral for the power function x^2 on the interval from -5.0 to 2.0 is equal to the area of the figure calculated:  1) with calculation accuracy 0.0001 by:  \* the rectangular method: S = 44.340683904994194;  \* the trapezoidal method: S = 44.333333904994234.  2) with calculation accuracy 0.00000001 by:  \* the rectangular method: S = 44.33333444698824;  \* the trapezoidal method: S = 44.33333371193688. |

**7.2.2 Тестовий випадок № 2 “****Тестування функції ступеня невалідні дані: start > end”**

Опис: Дані для тестування двох методів обчислення визначеного інтеграла для функції ступеня . Обчислення на інтервалі, де початок інтервалу більший за кінець інтервалу.

Тестові дані інтервалу значень х:

* Початок інтервалу: 2
* Кінець інтервалу: 0
* Перше значення точності обчислень: 0.0001
* Друге значення точності обчислень: 0.00000001

Очікуваний результат: Завершення виконання програми з помилкою. Виведення у консоль повідомлення про помилку:

|  |
| --- |
| Invalid interval passed. The start must be less than the end. |

**7.2.3 Тестовий випадок № 3 “****Тестування функції ступеня невалідні дані: start = end”**

Опис: Дані для тестування двох методів обчислення визначеного інтеграла для функції ступеня . Обчислення на інтервалі, де початок інтервалу дорівнює кінцю інтервалу.

Тестові дані інтервалу значень х:

* Початок інтервалу: 5
* Кінець інтервалу: 5
* Перше значення точності обчислень: 0.0001
* Друге значення точності обчислень: 0.00000001

Очікуваний результат: Завершення виконання програми з помилкою. Виведення у консоль повідомлення про помилку:

|  |
| --- |
| Invalid interval passed. The start must be less than the end. |

**7.2.4 Тестовий випадок № 4 “****Тестування функції кубічного кореня**   **валідні дані”**

Опис: Дані для тестування двох методів обчислення визначеного інтеграла для функції кубічного кореня  . Знаходження значення функції на інтервалі з додатними значеннями.

Тестові дані інтервалу значень х:

* Початок інтервалу: 1
* Кінець інтервалу: 10
* Перше значення точності обчислень: 0.0001
* Друге значення точності обчислень: 0.00000001

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

|  |
| --- |
| Definite integral for the cubic root function on the interval from 1.0 to 10.0 is equal to the area of the figure calculated:  1) with calculation accuracy 0.0001 by:  \* the rectangular method: S = 15.409679653196633;  \* the trapezoidal method: S = 15.410199177891199.  2) with calculation accuracy 0.00000001 by:  \* the rectangular method: S = 15.408260126288045;  \* the trapezoidal method: S = 15.40826017824237. |

**7.2.5 Тестовий випадок № 5 “****Тестування функції синуса валідні дані”**

Опис: Дані для тестування двох методів обчислення визначеного інтеграла для функції синуса . Знаходження значення функції на інтервалі з додатними значеннями.

Тестові дані інтервалу значень х:

* Початок інтервалу: 0
* Кінець інтервалу: (3,141593).
* Перше значення точності обчислень: 0.0001
* Друге значення точності обчислень: 0.00000001

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

|  |
| --- |
| Definite integral for the sine function sin(x) on the interval from 0.0 to 3.141592653589793 (PI) is equal to the area of the figure calculated:  1) with calculation accuracy 0.0001by:  \* the rectangular method: S = 1.9999999835503612;  \* the trapezoidal method: S = 1.999999983550358.  2) with calculation accuracy 0.00000001 by:  \* the rectangular method: S = 1.9999999964438926;  \* the trapezoidal method: S = 1.9999999964432238. |

**7.3 Програмний код завдання № 7**

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

|  |
| --- |
| package task7;  @FunctionalInterface public interface Integrable {  double someFunction(double x);   default double integral(double start, double end, double accuracy) {  if (start >= end) {  System.*err*.println("Invalid interval passed. The start must be less than the end. Try again!");  System.*exit*(1);  }   double step = (end - start) \* accuracy;  double area = 0;   for (double x = start; x < end; x += step) {  area += someFunction(x) \* step;  }   return area;  } } |

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

|  |
| --- |
| package task7;  public class IntegralWithRectangularMethod implements Integrable {  @Override  public double someFunction(double x) {  return Math.*pow*(x, 2);  /\*  return Math.cbrt(x);  \*/  /\*  return Math.sin(x);  \*/  } } |

Лістинг 7.3 - Програмний код реалізований у файлі “IntegralWithTrapezoidalMethod.java”

|  |
| --- |
| package task7;  public class IntegralWithTrapezoidalMethod implements Integrable {  @Override  public double someFunction(double x) {  return Math.*pow*(x, 2);  /\*  return Math.cbrt(x);  \*/  /\*  return Math.sin(x);  \*/  }   @Override  public double integral(double start, double end, double accuracy) {  if (start >= end) {  System.*err*.println("Invalid interval passed. The start must be less than the end. Try again!");  System.*exit*(1);  }   double step = (end - start) \* accuracy;  double area = 0;   for (double x = start; x < end; x += step) {  area += 0.5 \* step \* (someFunction(x) + someFunction(x + step));  }   return area;  } } |

Лістинг 7.4 - Програмний код реалізований у файлі “Main.java”

|  |
| --- |
| package task7;  public class Main {  public static void main(String[] args) {  IntegralWithRectangularMethod firstIntegrator = new IntegralWithRectangularMethod();  IntegralWithTrapezoidalMethod secondIntegrator = new IntegralWithTrapezoidalMethod();   /\*  double start = 2, end = 0, accuracy = 0.0001;  \*/  /\*  double start = 5, end = 5, accuracy = 0.0001;  \*/  double start = -5, end = 2, accuracy = 0.0001;  System.*out*.println("Definite integral for the power function x^2\n"  + "on the interval from " + start + " to " + end  + "\nis equal to the area of the figure calculated:"  + "\n1) with calculation accuracy " + accuracy + " by:");   /\*  double start = 1, end = 10, accuracy = 0.0001;  System.out.println("Definite integral for the cubic root function x^(-1/3)\n"  + "on the interval from " + start + " to " + end  + "\nis equal to the area of the figure calculated:"  + "\n1) with calculation accuracy " + accuracy + " by:");  \*/   /\*  double start = 0, end = Math.PI, accuracy = 0.0001;  System.out.println("Definite integral for the sine function sin(x)\n"  + "on the interval from " + start + " to " + end + " (PI)"  + "\nis equal to the area of the figure calculated:"  + "\n1) with calculation accuracy " + accuracy + " by:");  \*/   System.*out*.println("\t\* the rectangular method:\tS = " + firstIntegrator.integral(start, end, accuracy) + ";"  + "\n\t\* the trapezoidal method:\tS = " + secondIntegrator.integral(start, end, accuracy) + ".");  accuracy = 0.00000001;  System.*out*.println("2) with calculation accuracy " + accuracy + " by:");  System.*out*.println("\t\* the rectangular method:\tS = " + firstIntegrator.integral(start, end, accuracy) + ";"  + "\n\t\* the trapezoidal method:\tS = " + secondIntegrator.integral(start, end, accuracy) + ".");  } } |

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

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 7.1 - Результати роботи програмного коду відповідно до тестового випадку № 1 “Тестування функції ступеня  валідні дані”

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 7.2 - Результати роботи програмного коду відповідно до тестового випадку № 2 “Тестування функції ступеня невалідні дані: start > end”

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 7.3 - Результати роботи програмного коду відповідно до тестового випадку № 3 “Тестування функції ступеня невалідні дані: start = end”

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 7.4 - Результати роботи програмного коду відповідно до тестового випадку № 4 “Тестування функції кубічного кореня ∛x валідні дані”

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 7.5 - Результати роботи програмного коду відповідно до тестового випадку № 5 “Тестування функції синуса валідні дані”

**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 в галузях поліморфізму та успадкування. У процесі виконання завдань вдало використовувалися концепції успадкування та поліморфізму для створення ієрархії класів та ефективної реалізації необхідних функцій. В першому завданні використовувалися алгоритми бульбашкового сортування та сортування вставкою. Це додало реалізації роботи певний рівень складності та варіативності.

Покроково продемонстровано роботу з інтерфейсами, абстрактними класами, перевіркою еквівалентності та розробкою алгоритму для розрахунку хеш-кодів. До класу та окремих функцій були додані докладні 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. Отримані навички використання методів прямокутників та трапецій для розрахунку визначеного інтегралу дозволили додатково удосконалити знання у цьому напрямку.

У цілому, ця лабораторна робота є не тільки корисною для поглиблення розуміння поліморфізму, успадкування та вкладених класів у Java, але й сприяла розвитку високоорганізованих навичок програмування та розв'язання завдань. Використання цих можливостей дозволяє навчитись створювати більш функціональні, ефективні та зрозумілі програми на мові Java, що є важливим для розробки якісного програмного забезпечення та розв'язання різноманітних завдань.