Лабораторна робота №5

Тема: Інтерфейси в Java.

Мета: Вивчити особливості роботи з інтерфейсами у Java.

Завдання: Здобути навички створення та відлагодження проектів обробки даних з використанням інтерфейсів мовою Java.

(max: 10 балів)

Теоретичні відомості:

5.1. Інтерфейси

Механізм успадкування дуже зручний, але має обмеження. Зокрема можна наслідувати тільки від одного класу, на відміну, наприклад, від мови C++, де є множинне спадкування. У мові Java подібну проблему частково дозволяють вирішити інтерфейси. **Інтерфейси** визначають деякий функціонал, який не має конкретної реалізації і який потім реалізують класи, які наслідують ці інтерфейси. Один клас може наслідуватись від багатьох інтерфейсів. Щоб визначити інтерфейс, використовується **ключове слово *interface***, наприклад:

```
interface Printable { void print(); }
```

Даний інтерфейс називається *Printable*. Інтерфейс може визначати константи і методи, які можуть мати, а можуть і не мати реалізації. Методи без реалізації схожі на абстрактні методи абстрактних класів. В даному випадку оголошений один метод, який не має реалізації.

Всі методи інтерфейсу не мають модифікаторів доступу, за замовчуванням модифікатор *public*, тому що мета інтерфейсу – визначення функціоналу для реалізації його класом. Тому весь функціонал повинен бути відкритий для реалізації. Щоб клас реалізовував інтерфейс використовується **ключове слово *implements***:

```
public class Program {
   public static void main(String[] args) {
      Book b1 = new Book("Java", "H. Shildt");
      b1.print();
   }
}
interface Printable { void print(); }
class Book implements Printable {
   String name;
   String author;
   Book (String name, String author) {
      this.name = name; this.author = author;
   }
   public void print() {
      System.out.printf(" %s (%s) \n", name, author);
   }
}
```

В даному випадку клас *Book* реалізує інтерфейс *Printable*. При цьому треба враховувати, що якщо клас реалізовує інтерфейс, то він повинен реалізувати всі методи інтерфейсу, як у прикладі вище, реалізований метод *print*. Це дає можливість, в методі *main* створити об'єкт класу *Book* і викликати його метод *print*.

Якщо клас не реалізує якісь методи інтерфейсу, то такий клас повинен бути визначений як абстрактний, а його не абстрактні класи-спадкоємці повинні будуть реалізувати ці методи.

У той же час не можна створювати об'єкти інтерфейсів, тому наступний код не буде працювати:

```
Printable p = new Printable();
p.print();
```

Однією з переваг використання інтерфейсів є те, що вони дозволяють додати в додаток гнучкості. Наприклад, крім класу *Book* визначимо ще клас, який також буде реалізовувати інтерфейс *Printable*:

```
class Journal implements Printable {
```

```
    private String name;
    String getName() { return name; }
    Journal(String name) { this.name = name; }
    public void print() { System.out.println(name); }
}
```

Клас *Book* і клас *Journal* пов'язані тим, що вони реалізують інтерфейс *Printable*. Тому можна динамічно в програмі створити об'єкти Printable як екземпляри обох класів:

```
public class Program {
    public static void main (String[] args) {
        Printable printable = new Book("Java", "H. Shildt");
        printable.print(); // Java (H. Shildt)
        printable = new Journal ( "Foreign Policy");
        printable.print(); // Foreign Policy
    }
}
```

Крім цього, існує така функціональність, як **методи за замовчуванням**. Це означає, що інтерфейси, крім оголошення методів можуть мати їх реалізацію за замовчуванням, яка використовується, якщо клас, який реалізує даний інтерфейс, не реалізує метод. Наприклад, створимо метод за замовчуванням в інтерфейсі *Printable*:

```
interface Printable {
    default void print() { System.out.println("Не визначено"); }
}
```

Метод за замовчуванням – це звичайний метод без модифікаторів, який позначається ключовим словом *default*. Тепер в класі *Journal* не обов'язково цей метод реалізувати, хоча його можна і перевизначити.

Інтерфейси можуть мати статичні методи, які повністю аналогічні статичним методам звичайних класів:

```
interface Printable {
    void print();
    static void staticMethod() { System.out.println("статичний метод"); }
}
```

Для звернення до статичного методу інтерфейсу так само як і у випадку з класами, пишуть назву інтерфейсу та метод:

```
public static void main(String [] args) { Printable. staticMethod (); }
```

Всі методи в інтерфейсі, фактично, мають модифікатор *public*. Однак можна визначати в інтерфейсі методи з модифікатором *private*. Вони можуть бути статичними і не статичними, але вони не можуть мати реалізації за замовчуванням. Подібні методи можуть використовуватися тільки всередині самого інтерфейсу, в якому вони визначені. Наприклад, якщо потрібно виконувати в інтерфейсі деякі повторювані дії, і в цьому випадку такі дії можна виокремити у приватні методи.

Крім методів в інтерфейсах можуть бути визначені статичні константи. Константи також не мають модифікаторів, але за замовчуванням вони мають модифікатор доступу *public static final*, і тому їх значення є доступним з будь-якого місця програми.

При потребі успадкування класу від кількох інтерфейсів, інтиерфейси перераховуються через кому після слова *implements*:

```
interface Printable { // методи інтерфейсу }
interface Searchable { // методи інтерфейсу }
class Book implements Printable, Searchable { // реалізація класу }
```

Клас *Book* повинен буде реалізувати як методи інтерфейсу *Printable*, так і методи інтерфейсу *Searchable*.

Інтерфейси, як і класи, можуть успадковуватись:

```
interface BookPrintable extends Printable { void paint(); }
```

5.2. Узагальнені типи та методи

Узагальнення або *generics* (узагальнені типи і методи) дозволяють відійти від жорсткого визначення використовуваних типів.

Припустимо, є клас *Account* для подання банківського рахунку, який має наступний вигляд:

```
class Account {
    private int id;
    private int sum;
    Account(int id, int sum) { this.id = id;  this.sum = sum; }
    public int getId() { return id; }
    public int getSum() { return sum; }
    public void setSum(int sum) {this.sum = sum; }
}
```

Клас *Account* має два поля: *id* – унікальний ідентифікатор рахунку і *sum* – сума на рахунку. В даному випадку ідентифікатор заданий як цілочисельне значення, наприклад, 1, 2, 3, 4 і так далі. Але для ідентифікатора можуть використовуватися і рядкові значення. І числові, і рядкові значення мають свої переваги і недоліки. На момент написання коду, як правило, точно не відомо, що краще вибрати для збереження ідентифікатора – рядки або числа. Або, можливо, цей клас буде використовуватися іншими розробниками, які можуть мати свою думку з даного питання. Наприклад, в якості типу *id* захочуть використовувати якийсь свій клас.

На перший погляд можна вирішити цю проблему наступним чином: поставити *id* як поле типу *Object*, який є універсальним і базовим суперкласом для всіх інших типів. При цьому можна буде створити об'єкт класу *Account* і з рядковим і з числовим значенням *id*:

```
Account acc1 = new Account(2334, 5000); // id - число
Account acc2 = new Account("sid5523", 5000); // id - рядок
```

Однак виникає проблема безпеки типів. Наприклад, в наступному випадку буде помилка:

```
Account acc1 = new Account("2345", 5000);
int acc1Id = (int)acc1.getId(); // java.lang.ClassCastException
```

Проблема може здаватися штучною, оскільки в даному випадку видно, що в конструктор передається рядок, тому навряд чи будемо намагатися перетворити його до типу *int*. Однак в процесі розробки можлива ситуація, коли невідомо, який саме тип представляє значення в *id*, і при спробі отримати число в даному випадку отримаємо виключенням *java.lang.ClassCastException*.

Створювати для кожного окремого типу свою версію класу *Account* теж не є раціональним рішенням, оскільки в цьому випадку буде багато повторів коду. Для вирішення подібних проблем були створені **узагальнення (*generics*)**. Узагальнення дозволяють не вказувати конкретний тип, який буде використовуватися. Тому визначимо клас *Account* як узагальнений:

```
class Account <T> {
    private T id;
    private int sum;
    Account(T id, int sum) { this.id = id; this.sum = sum; }
    public T getId() {return id; }
    public int getSum() {return sum; }
    public void setSum(int sum) {this.sum = sum; }
}
```

За допомогою літери *T* у визначенні класу *class Account <T>* ми показуємо, що даний тип *T* буде використовуватися цим класом. Параметр *T* в кутових дужках називається **універсальним параметром**, тому що замість нього можна підставити будь-який тип. При цьому не відомо, який саме це буде тип: *String*, int або якийсь інший. Причому буква *T* обрана умовно, це може і будь-яка інша буква або набір символів.

Після оголошення класу можна застосувати універсальний параметр *T*: так далі в класі оголошується змінна цього типу, якій потім присвоюється значення в конструкторі. Метод *getId()* повертає значення змінної *id*, але оскільки дана змінна представляє тип *T*, то даний метод також повертає об'єкт типу *T*: *public T getId()*.

Використаємо даний клас:

```
public class Program {
```

```
    public static void main(String[] args) {
        Account <String> acc1 = new Account <String> ("2345", 5000);
        String acc1Id = acc1.getId();
        System.out.println(acc1Id);
        Account <Integer> acc2 = new Account <Integer> (2345, 5000);
        Integer acc2Id = acc2.getId();
        System.out.println(acc2Id);
    }
}
```

При визначенні змінної даного класу і створенні об'єкта, після імені класу в кутових дужках потрібно вказати, який саме тип буде використовуватися замість універсального параметра. При цьому потрібно враховувати, що в якості типу *T* можуть виступати тільки вказівниковий тип, не примітивний. Тобто можна написати *Account <Integer>*, але не можна використовувати тип *int* або *double*, наприклад, *Account <int>*. Замість примітивних типів треба використовувати класи-обгортки: *Integer* замість *int*, *Double* замість *double* і т.д.

Наприклад, перший об'єкт буде використовувати тип *String*, тобто замість *T* буде підставлятися *String*:

```
Account <String> acc1 = new Account <String> ( "2345", 5000);
```

У цьому випадку в якості першого параметра в конструктор передається рядок. А другий об'єкт використовує тип int (Integer):

```
Account <Integer> acc2 = new Account <Integer> (2345, 5000);
```

Інтерфейси, як і класи, також можуть бути узагальненими. Створимо узагальнений інтерфейс *Accountable* і використаємо його в програмі:

```
public class Program {
    public static void main(String[] args) {
        Accountable <String> acc1 = new Account( "1235rwr", 5000);
        Account acc2 = new Account("2373", 4300);
        System.out.println(acc1.getId());
        System.out.println(acc2.getId());
    }
}
interface Accountable <T> {
    T getId();
    int getSum();
    void setSum(int sum);
}
class Account implements Accountable <String> {

    private String id;
    private int sum;
    Account(String id, int sum) { this.id = id; this.sum = sum; }
    public String getId() {return id; }
    public int getSum() {return sum; }
    public void setSum(int sum) {this.sum = sum; }
}
```

В даному випадку реалізована стратегія, коли при реалізації для універсального параметру інтерфейсу задається конкретний тип, як наприклад, в даному випадку це тип *String*. Тоді клас, який реалізує інтерфейс, жорстко прив'язаний до цього типу.

Крім узагальнених типів можна також створювати узагальнені методи, які так само будуть використовувати універсальні параметри. Наприклад:

```
public class Program {
    public static void main (String[] args) {
        Printer printer = new Printer();
        String [] people = { "George", "Vasia", "Yuzik", "Alex"};
```

```
        Integer [] numbers = {23, 4, 5, 2, 13, 456, 4};
        printer. <String> print(people);
        printer. <Integer> print(numbers);
    }
}
class Printer {
    public <T> void print (T [] items) {
        for (T item: items) {
            System.out.println(item);
        }
    }
}
```

Особливістю узагальненого методу є використання універсального параметру в оголошенні методу після всіх модифікаторів і перед типом значення, що повертається:

public <T> void print(T [] items)

Потім всередині методу всі значення типу *T* представлятимуть даний універсальний параметр. При виклику подібного методу перед його ім'ям в кутових дужках вказується, який тип буде передаватися на місце універсального параметру:

printer. <String> print(people);
printer. <Integer> print(numbers);

Можна також використовувати одночасно декілька універсальних параметрів. Конструктори як і методи також можуть бути узагальненими. В цьому випадку перед конструктором також вказуються в кутових дужках універсальні параметри.

Індивідуальні завдання:

(Для виконання індивідуальних завдань № варінта є для 532 групи порядковим номером прізвища студента в списку групи, для 531 – вказаний у додатку (окремий документ). Усі проекти та, за наявності, тести до них завантажити у власні репозиторії на Git Hub. Посилання на репозиторії проектів вказати у звіті та обов'язково долучати скріни успішного виконання)

1. Розробити консольний застосунок для роботи з базою даних, що зберігається у текстовому файлі (початковий масив не менше 10 записів). Структура бази даних описується класом згідно вашого варіанта. Передбачити роботу з довільною кількістю записів. Для ідентифікації спроби введення з клавіатури некоректних даних описати виключення. Реалізувати методи для:

– додавання записів;
– редагування записів;
– знищення записів;
– виведення інформації з файла на екран;
– обчислення підсумкових показників згідно вашого варіанта;
– сортування за різними полями.

Меню програми реалізувати по натисненню на певні клавіші: наприклад, Enter – вихід, п - пошук, р – редагування тощо.

Застосунок організувати таким чином, щоб в ньому існував інтерфейсний клас, якому належать усі методи (реалізація методів повинна залишитись у класі, що успадковує інтерфейс).

(4 бала)

1. Описати клас для бази зданих з інформацією про метеорологічні спостереження протягом місяця у форматі: дата, температура повітря, атмосферний тиск. Визначити дні з атмосферним тиском, більшим від середнього начення цього показника за весь період. Результат вивести на екран у формі таблиці.
2. Описати клас для бази зданих з інформацією про метеорологічні спостереження протягом року у форматі: дата (ДД:ММ), температура повітря. Знайти середню температуру повітря кожного місяця та вивести на екран таблицю: місяць, середня температура. Визначити назву місяця з найбільшою середньою температурою повітря.

3. Описати клас для бази зданих з інформацією про метеорологічні спостереження протягом місяця у форматі: дата, температура, атмосферний тиск. Впорядкувати дані у порядку зростання тиску та вивести результати на екран у формі таблиці. Визначити два дні з найбільшим перепадом тиску.

4. Описати клас для бази зданих з інформацією про метеорологічні спостереження протягом місяця у форматі: дата, температура, тиск. Визначити дні, температура яких є більшою від середнього значення температури. Результат вивести на екран у формі таблиці.

5. Описати клас для бази зданих з інформацією про власників авто з полями: марка автомобіля, номер автомобіля, колір, дані про власника. Вивести на екран у формі таблиці дані про власників автомобілів заданного кольору та заданої марки.

6. Описати клас для бази зданих з інформацією про успішність з полями: прізвище студента, оцінки з п'яти предметів. Знайти середній бал по кожному предмету. Вивести результати на екран у формі таблиці.

7. Описати два класи для бази зданих з інформацією про виробництво. Перший містить поля: назва виробу, вартість одиниці виробу. Другий про результати виробництва за звітний період: дата, назва виробу, кількість. Дані про один виріб можуть повторюватися в різні дати. Обчислити загальну вартість виготовленої за звітний період продукції за кожним видом виробів. Відсортувати утворений масив по зростанню вартості виробів та вивести його на екран у формі таблиці.

8. Описати клас для бази зданих з інформацією про автомобілі з полями: марка, колір, номер, рік випуску, дані про власника. Відсортувати масив даних по марках автомобілів та вивести його на екран у формі таблиці. Визначити кількість різних кольорів кожної марки.

9. Описати клас для бази зданих з інформацією про автомобілі з полями: марка, колір, номер, рік випуску, дані про власника. Визначити кількість автомобілів кожної марки та вивести на екран у формі таблиці впорядкувавши по спаданню кількості автомобілів.

10. Описати клас для бази зданих з інформацією про дні народження декількох студентів з полями: дата, прізвище та ініціали. Дата задається у форматі ДД:ММ:РР. З клавіатури ввести поточну дату. Визначити студента з найближчим днем народження. Вивести на екран дані про студентів, дні народження яких ще будуть у поточному році (відлік вести від поточної дати).

11. Описати клас для бази зданих з інформацією про про клієнтів банку з полями: № рахунку, прізвище та ім'я, сума вкладу, дата проведення операції. Визначити клієнтів банку з найбільшою кількістю банківських операцій та вивести дані про них на екран у формі таблиці.

12. Описати клас для бази зданих з інформацією про клієнтів банку з полями: дата проведення операції прізвище та ім'я, № рахунку, сума безготівкового отримання/переведення, отримано/видано готівкою, залишок вкладу. Вивести на екран у формі таблиці дані про клієнтів банку, які на протязі заданого періоду часу мають найбільшу суму безготівкового отримання коштів на рахунок.

13. Описати клас для бази зданих з інформацією про клієнтів банку з полями: дата проведення операції прізвище та ім'я, № рахунку, сума безготівкового отримання/переведення, отримано/видано готівкою, залишок вкладу. Вивести на екран у формі таблиці дані про клієнтів банку, які на протязі заданого періоду часу виконали безготівкове переведення на загальну суму, яка перевищує задане користувачем граничне значення.

14. Описати клас для бази зданих з інформацією про книги бібліотеки з полями: автор, назва книги, видавництво, рік видання. Відсортувати масив за прізвищами авторів та вивести на екран у формі таблиці. Підрахувати кількість книг від кожного видавництва.

15. Описати клас для бази зданих з інформацією про книги бібліотеки з полями: автор, назва книги, видавництво, рік видання, кількість штук. Дані про книги, видані після 2000 року вивести на екран у формі таблиці, порахувати загальну кількість таких книг.

16. Описати клас для бази зданих з інформацією про книги у формі: автор, назва книги, видавництво, рік видання, кількість штук. Вивести масив на екран у формі таблиці, згрупувавши книги з однаковим роком видання. Порахувати загальну кількість книг для кожного року видання та вивести цю інформацію у вигляді таблиці на екран.

17. Описати клас для бази зданих з інформацією про книги у формі: автор, назва книги, видавництво, рік видання. Вивести дані про книги з програмування (перевіряти, чи є частиною назви книги слово «програмування» з малої або великої літери) у порядку спадання років видань.

18. Описати клас для бази зданих з інформацією про конфігурацію комп'ютера з полями: тип процесора, тактова частота, обсяг оперативної пам'яті, обсяг дискової пам'яті, характеристика монітора та ін. Відсортувати записи по типу процесора та вивести на екран у формі таблиці. Визначити комп'ютери з найбільшим обсягом оперативної і дискової пам'яті.

19. Описати клас для бази зданих з інформацією про результати роботи підприємства протягом року у форматі: місяць, план випуску продукції, фактичний випуск продукції. изначити назви місяців з недовиконанням плану випуску продукції та вивести цю інформацію на екран у вигляді таблиці.

20. Описати клас для бази зданих з інформацією про результати роботи підприємства впродовж року з полями: місяць, план випуску продукції, фактичний випуск продукції. Відсортувати масив записів у порядку зростання відсотку виконання плану та вивести його на екран у формі таблиці. Визначити місяці з найбільшим та найменшим відсотком виконання плану.

21. Описати клас для бази зданих з інформацією про рейтинг студентів однієї групи: прізвище студента, № залікової книжки, рейтинг у 100 бальній шкалі. Впорядкувати записи по спаданню рейтингу та вивести його на екран у формі таблиці. Обчислити середній рейтинг групи та кількість студентів з рейтингом нижче середнього.

22. Описати клас для бази зданих з інформацією про студентів з полями: прізвище, дата народження, місце народження. Дата народження задається у вигляді ДД:ММ:РР. Відсортувати записи за зростанням дат народження та вивести його на екран у формі таблиці. Якщо є студенти з однаковою датою народження (співпадають день та місяць), то вивести записи про них окремо в таблиці «двійнята».

23. Описати клас для бази зданих з інформацією про успішність групи студентів з полями: прізвище та ім'я, № залікової книжки, оцінки за 100 бальною шкалою з п'яти предметів. Впорядкувати записи у порядку зростання зростання середнього балу і вивести їх на екран у формі таблиці. Визначити відсоток студентів, що мають незадовільні оцінки.

24. Описати клас для бази зданих з інформацією про успішність групи студентів з полями: прізвище та ім'я, № залікової книжки, оцінки з п'яти предметів за 100 бальною шкалою. Впорядкувати записи у порядку спадання середньої оцінки та вивести їх на екран у формі таблиці. Визначити відсоток студентів, середній бал яких відповідає оцінкам "добре" та "відмінно".

25. Описати клас для бази зданих з інформацією про метеорологічні спостереження протягом місяця у форматі: дата, температура повітря, атмосферний тиск. Впорядкувати записи у порядку спадання температури повітря та вивести його на екран у формі таблиці. Визначити два дні з найбільшим перепадом температури повітря.

2. Розробити консольний застосунок для роботи з базою даних, що зберігається у текстовому файлі (початковий масив не менше 5 записів). Структура бази даних описується ієрархією класів згідно вашого варіанта. Для ідентифікації спроби введення з клавіатури некоректних даних описати виключення. Реалізувати методи для:

– додавання записів;

– редагування записів;

– знищення записів;

– виведення інформації з файла на екран;

– обчислення та виведення на екран результатів згідно свого варіанта індивідуального завдання.

Меню програми реалізувати по натисненню на певні клавіші: наприклад, Enter – вихід, п - пошук, р – редагування тощо.

Для консольного застосунку організувати інтерфейсний клас з двома абстрактними методами (які з створених у проекті методів робити абстрактними – обирайте самі).

(6 балів)

Варіант №	Батьківський (базовий) клас		Похідний клас		Реалізувати з допомогою окремих методів обчислення та виведення на екран таких даних:
	Сутність	Обов'язкові поля	Сутність	Обов'язкові поля	
1.	Цілодобовий кіоск	Назва, адреса	Година	Кількість покупців, коментар	Загальна кількість покупців, година з найменшою кількістю покупців, коментарями з певними словами
2.	Виконавець	Прізвище, жанр	Концерт	Дата, кількість глядачів	Загальна кількість глядачів, концерт з максимальною кількістю глядачів, кількість слів у назві жанру
3.	Музичний гурт	Назва, прізвище керівника	Гастрольна поїздка	Місто, рік, кількість концертів	Гастрольна поїздка з максимальною кількістю концертів, список гастрольних поїздок у певне місто, остання літера в прізвищі керівника

4.	Навчальний курс	Назва курсу, назва кафедри	Лекція	Дата, група, кількість студентів	Середня кількість студентів, лекції з найбільшою кількістю студентів, кількість слів у назві кафедри
5.	Виставка	Назва, прізвище скульптора	День	Кількість відвідувачів, коментар	Сумарна кількість відвідувачів, день з найбільшою кількістю відвідувачів, день з найбільшою кількістю слів у коментарі
6.	Письменник	Прізвище, мова, кількість книжок	Виступ	Дата, місце, кількість слухачів	Сумарна кількість слухачів, день з найменшою кількістю слухачів, довжина прізвища
7.	Співробітник	ПІБ, посада	Робочій день	Дата, кількість годин, назва проекту	Середня кількість робочих годин за період; Кількість годин на проекті; Дні з максимальним навантаженням
8.	Лікар	ПІБ, спеціальність	Робочій день	Дата, кількість пацієнтів, час початку роботи	Середня кількість пацієнтів в день за період; Кількість днів з максимальним навантаженням; Дні, коли починав приймати після зазначеного часу
9.	Піцерія	Назва, адреса	Робочій день	Дата, кількість замовлень, піца дня	Середня кількість замовлень в день за період; Дні з максимальним відвідуванням; Сумарна кількість замовлень для днів з визначеною піцою дня
10.	Басейн	Назва, адреса	Робочій день	Дата, кількість відвідувачів, кількість доступних доріжок	Середня кількість відвідувань в день за період; Дні з мінімальною кількістю доступних доріжок; Кількість днів, коли було доступно не менше зазначеної кількості доріжок
11.	Бібліотека	Назва, адреса	Робочій день	Дата, кількість книг, що видано, кількість книг, що повернуто	Середній рух книжок в день за період; Кількість днів, коли було видано книг більше, ніж повернуто; Дні, коли видана парна кількість книг, а повернута – непарна
12.	Сайт	Назва, URL	Відвідування	Дата, кількість унікальних хостів, кількість завантажених сторінок	Середня кількість хостів в день за період; Дні з максимальною кількістю завантажених сторінок; Кількість днів, коли співвідношення хостів до сторінок перевищує задане значення
13.	Акаунт електронної пошти	E-mail, ПІБ володаря	Спам	Дата, кількість спам повідомлень, загальна кількість повідомлень	Середня кількість спаму в день за період; Кількість днів, коли відсоток спам повідомлень був менший за задане значення; Дні, коли кількість спаму збільшувалась
14.	Акція компанії на біржі	Назва компанії, код на біржі	Курс	Дата, курс відкриття, курс закриття	Середня вартість акцій по закриттю за період; Кількість днів, коли курс зростав протягом дня; Дні, коли зміна курсу за день перевищувала задане значення
15.	Телефонний номер	Номер, оператор	Дзвінки	Дата, кількість хвилин розмов, кошти, що використано на розмови	Середня платня в день за період; Кількість днів, коли вартість хвилини розмови перевищувала задане значення; Дні, коли кількість хвилин розмов була парна
16.	Навчальний курс	Назва, наявність іспиту	Практичне заняття	Дата, тема, кількість студентів	Середня кількість студентів, заняття з максимальною кількістю студентів, список тем з певним словом у назві

17.	Трамвайна зупинка	Назва, список номерів маршрутів	Година	Кількість пасажирів, коментар	Загальна кількість пасажирів, година з найменшою кількістю пасажирів, найдовший коментар
18.	Навчальний курс	Назва, прізвище викладача	Лекція	Дата, тема, кількість студентів	Лекція з мінімальною кількістю студентів, список тем з певним словом у назві, остання літера у прізвищі викладача
19.	Конференція	Назва, місце проведення	Засідання	Дата, тема, кількість учасників	Середня кількість учасників на засіданні, засідання з найбільшою кількістю учасників, довжина назви
20.	Виставка	Назва, прізвище художника	День	Кількість відвідувачів, коментар	Сумарна кількість відвідувачів, день з найменшою кількістю відвідувачів, список коментарів з певним словом
21.	Станція метрополітену	Назва, рік відкриття	Година	Кількість пасажирів, коментар	Сумарна кількість пасажирів, години з найменшою кількістю пасажирів та найбільшою кількістю слів у коментарі
22.	Лікар	Прізвище, фах	Прийом	День, зміна, кількість відвідувачів	Загальна кількість відвідувачів, прийом з мінімальною кількістю відвідувачів, довжина прізвища
23.	Поет	Прізвище, мова, кількість збірок	Виступ	Дата, місце, кількість слухачів	Сумарна кількість слухачів, день з найбільшою кількістю слухачів, довжина прізвища
24.	Лікар	Прізвище, стаж	Прийом	День, кількість відвідувачів, коментар	Середня кількість відвідувачів, прийом з мінімальною кількістю відвідувачів, найдовшим коментарем
25.	Трамвайний маршрут	Номер, середній інтервал руху	Зупинка	Назва, кількість пасажирів	Загальна кількість пасажирів, зупинки з найменшою кількістю пасажирів, найдовшою назвою