Лабораторна робота №3

Тема: Створення класів

Мета: Вивчити особливості роботи з власними класами у Java.

Завдання: Здобути навички створення, відлагодження та тестування проектів обробки даних з використанням класів мовою ***Java***.

(max: 5 балів)

Теоретичні відомості:

3.1. Основні поняття ООП: клас, об’єкт, метод, модифікатор доступу

Java є об'єктно-орієнтованою мовою, тому такі поняття як "клас" і "об'єкт" мають ключову роль. Будь-яку програму на Java можна уявити як набір взаємодіючих між собою об'єктів. Шаблоном або описом об'єкта є **клас**, а **об'єкт** являє собою екземпляр цього класу. Можна ще провести наступну аналогію. У нас у всіх є деяке уявлення про людину – наявність двох рук, двох ніг, голови, тулуба і т.д. Є деякий шаблон – цей шаблон можна назвати класом. Реально ж існуюча людина (фактично екземпляр даного класу) є об'єктом цього класу. Клас визначається за допомогою ключового слова *class*.

Будь-який об'єкт може володіти двома основними характеристиками: стан – деякі дані, які зберігає об'єкт, і поведінка – дії, які може здійснювати об'єкт. Для зберігання стану об'єкта в класі застосовуються **поля або змінні класу**. Для визначення поведінки об'єкта в класі використовуються **методи**. Наприклад, клас *Person*, який представляє людину, може мати таке визначення:

```
class Person {
    int a; // вік
    String n; // ім'я
    void PrintPerson() {
        System.out.printf( "Ім’я:%s \t Вік: %d \n", n, a);
    }
}
```

У класі *Person* визначені два поля: name представляє ім'я людини, a age – його вік. І також визначено метод *PrintPerson*, який нічого не повертає і просто виводить ці дані на консоль. Тепер використаємо даний клас. Для цього створимо наступну програму:

```
public class Program {
    public static void main(String[] args) { Person Vasia; }
}
```

Як правило, класи визначаються в окремих файлах. У даному випадку, для простоти, визначимо два класи в одному файлі. Варто зазначити, що в цьому випадку тільки один клас може мати модифікатор *public* (в даному випадку це клас *Program*), а назва файлу з програмним кодом повинна співпадати з іменем класу, який в ньому міститься, тобто, в даному випадку, файл повинен називатися *Program.java*.

Клас представляє новий тип, тому ми можемо визначати змінні, які представляють даний тип. Так, в методі *main* визначена змінна *Vasia*, яка представляє клас *Person*. Але поки що ця змінна не вказує на жодний об'єкт і за замовчуванням має значення *null*, тому ми не можемо її використовувати, спочатку необхідно створити об'єкт класу *Person*.

Крім звичайних методів класи можуть містити спеціальні методи, які називаються **конструкторами**. Конструктори викликаються при створенні нового об'єкта даного класу. Конструктори виконують ініціалізацію об'єкта. Якщо в класі не визначено жодного конструктора, то для цього класу автоматично створюється конструктор без параметрів.

Вище наведений клас *Person* немає жодних конструкторів. Тому для нього автоматично створюється конструктор за замовчуванням, який можна використати для створення об'єкта *Person*. Створимо об'єкт класу *Person*:

```
public class Program {
    public static void main (String[] args) {
        Person vasia = new Person(); // створення об'єкта
        vasia.PrintPerson();
```

```
        vasia.name = "Vasia"; // визначаємо ім'я та вік
        vasia.age = 45;
        vasia.PrintPerson();
    }
}
```

Для створення об'єкта *Person* використовується вираз *new Person()*. Оператор *new* виділяє пам'ять для об'єкта *Person*, і потім викликається конструктор за замовчуванням, який не має жодних параметрів. В результаті після виконання цього виразу в пам'яті буде виділено комірки пам’яті, де будуть зберігатися всі дані об'єкта *Person*, а змінна *vasia* отримає посилання на створений об'єкт.

Якщо конструктор не ініціалізує значення змінних об'єкта, то вони отримують значення по замовчуванню. Для змінних числових типів це число 0, а для типу string та класів – це значення *null* (тобто, фактично, відсутність значення).

Після створення об'єкта можна звернутися до змінних об'єкта *Person* через змінну *vasia* та встановити або отримати їх значення, наприклад, *vasia.name = "Vasia"*.

Якщо потрібно щоб при створенні об'єкта спрацювувала певна логіка, наприклад, щоб поля класу отримували деякі значення, то можна визначити в класі конструктори. Наприклад:

```
public class Program {
    public static void main(String[] args) {
        Person george=new Person(); // виклик 1-го конструктора
        Person vasia=new Person("Vasia"); // виклик 2-го констр.
        Person petia=new Person("Petia", 25); // виклик 3-го констр.
    }
}
class Person {
    int age; // вік
    String name; // ім'я
    Person() { age=18; name="Undefined"; } //перший  конструктор
    Person(String n) { age=18; name=n; } //другий  конструктор
    Person(int a, String n,) { age=a; name=n; } //третій  конструктор
    void PrintPerson() {
        System.out.printf("Вік: %d \t Ім’я: %s \n ", age, name);
    }
}
```

Ключове слово *this* в Java являє собою посилання на поточний екземпляр класу, з його допомогою можна звертатися до змінних, методів об'єкта, а також викликати його конструктори. Наприклад:

```
class Person {
    int age; // вік
    String name; // ім'я
    Person() { this("Undefined", 18); }
    Person(String name) {  this(name, 18);  }
    Person(int age, String name) { this.age = age; this.name = name; }
}
```

У третьому конструкторі параметри називаються так само, як і поля класу. Щоб розмежувати поля класу і параметри, застосовується ключове слово *this*. Крім того, три конструктори виконують ідентичні дії: встановлюють значення полів *age* та *name*. Щоб уникнути повторів, за допомогою *this* можна викликати один з конструкторів класу і передати для його параметрів необхідні значення.

Крім конструктора початкову ініціалізацію об'єкта цілком можна проводити за допомогою **ініціалізатора об'єкта**. Останній виконується для будь-якого конструктора. Тобто в ініціалізатор можна помістити код, загальний для всіх конструкторів:

```
class Person {
    String name; // ім'я
```

```
    int age; // вік
    {
        name = "Undefined"; // ініціалізатор
        age = 18;
    }
    Person() { }
}
```

Як правило, в Java класи об'єднуються в пакети. **Пакети** дозволяють організувати класи в набори. За замовчуванням java вже має ряд вбудованих пакетів, наприклад, *java.lang, java.util, java.io* і т.д. Крім того, пакети можуть мати вкладені пакети.

Організація класів у вигляді пакетів дозволяє уникнути конфлікту імен між класами. Адже можливі випадки, коли розробники називають свої класи однаковими іменами. Належність до пакету дозволяє гарантувати однозначність імен.

Щоб вказати, що клас належить певному пакету, треба використовувати **директиву *package***, після якої вказується ім'я пакета:

package назва_пакету;

Як правило, назви пакетів відповідають фізичній структурі проекту, тобто організації каталогів, в яких знаходяться файли з вихідним кодом. А шлях до файлів всередині проекту відповідає назві пакета цих файлів. Наприклад, якщо класи належать пакету *mypack*, то ці класи поміщаються в проекті в папку *mypack*. Класи необов'язково визначати в пакети. Якщо для класу пакет не визначений, то вважається, що даний клас знаходиться в пакеті за замовчуванням, який немає імені. Наприклад:

```
package study;
public class Program {  ... }
class Person {  ... }
```

Директива *package study* на початку файлу вказує, що класи *Program* і *Person*, які тут визначені, належать пакету *study*.

При роботі в середовищі розробки, наприклад, в NetBeans, IDE бере на себе всі питання компіляції пакетів і файлів, які належать до них. Для цього достатньо натиснути на кнопку запуску.

Якщо потрібно використати класи з інших пакетів, то потрібно підключити ці пакети і класи. Виняток становлять класи з пакета java.lang (наприклад, *String*), які підключаються до програми автоматично.

Наприклад клас *Scanner* знаходиться в пакеті java.util, тому його можна отримати в наступний спосіб:

```
java.util.Scanner in = new java.util.Scanner(System.in);
```

Однак таке нагромадження назв пакетів не завжди зручно, і в якості альтернативи можна імпортувати пакети і класи в проект за допомогою **директиви *import***, яка вказується після директиви *package*:

```
package study;
import java.util.Scanner; // імпорт класу Scanner
public class Program {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
    }
}
```

Директива *import* вказується на самому початку коду, після чого слідує ім'я класу, що підключається (у даному випадку класу *Scanner*). В наведеному вище прикладі, підключений тільки один клас, хоча пакет *java.util* містить ще багато класів. Щоб не підключати окремо кожен клас, можна одразу підключити весь пакет:

```
import java.util. *; // імпорт всіх класів з пакета java.util
```

Можлива ситуація одночасного використання двох класів з однаковою назвою з різних пакетів, наприклад, клас *Date* є і в пакеті *java.util*, і в пакеті *java.sql*. Якщо потрібно одночасно використовувати два цих класи, то необхідно вказувати повний шлях до цих класів в пакеті:

```
java.util.Date utilDate = new java.util.Date();
java.sql.Date sqlDate = new java.sql.Date();
```

У java є також особлива форма імпорту – статичний імпорт. Для цього разом з директивою import використовується модифікатор *static*:

```
import static java.lang.System.*;
import static java.lang.Math.*;
```

Тут відбувається статичний імпорт класів *System* і *Math*. Ці класи мають статичні методи. Завдяки операції статичного імпорту можна використовувати ці методи без назви класу. Наприклад, писати не *Math.sqrt(20)*, а *sqrt(20)*, оскільки функція *sqrt()*, яка повертає квадратний корінь числа, є статичною.

Всі члени класу в мові Java – поля і методи – мають модифікатори доступу. **Модифікатори доступу** дозволяють задати допустиму область видимості для членів класу, тобто контекст, в якому можна використовувати цю змінну або метод.

В Java використовуються наступні модифікатори доступу:

- *public*: публічний, загальнодоступний клас або член класу; поля і методи, оголошені з модифікатором *public*, видно з інших класів з поточного пакета і з зовнішніх пакетів;
- *private*: закритий клас або член класу, протилежність модифікатору *public;* закритий клас або член класу доступний тільки з коду в тому ж класі;
- *protected*: такий клас або член класу доступний з будь-якого місця в поточному класі або пакеті або в похідних класах, навіть якщо вони знаходяться в інших пакетах;
- *модифікатор за замовчуванням*: відсутність модифікатора у поля або методу класу передбачає застосування до нього модифікатора за замовчуванням (такі поля або методи видно з усіх класів в поточному пакеті).

Розглянемо модифікатори доступу на прикладі наступної програми:

```
public class Program {
   public static void main (String [] args) {
     Person alex=new Person("alex", "Khreshchatyk Street",
    "+380991234567", 21);
     alex.printName(); // нормально, public
     alex.printAge(); // нормально, модифікатор за замовчуванням
     alex.printPhone(); // нормально, protected
       //kate.printAddress(); //! помилка, private
     System.out.println(alex.name); // нормально,
                                    // модифікатор за замовчуванням
     System.out.println(alex.address); // нормально, public
     System.out.println(alex.age); // нормально, protected
     //System.out.println(alex.phone); //! помилка, private
   }
}
class Person {
   String name;  public String address;
   private String phone;protected int age;
   public Person (String name, String address, String phone, int age) {
      this.name = name; this.address = address;
      this.phone = phone; this.age = age;
   }
   public void printName() { ...  }
   void printAge() { ... }
   private void printAddress() { ...  }
   protected void printPhone() { ... }
}
```

В даному випадку обидва класи розташовані в одному пакеті – пакеті за замовчуванням, тому в класі *Program* можна використовувати всі методи та змінні класу *Person*, які мають модифікатор за замовчуванням, public і protected. Поля та методи з модифікатором private в класі

Program стануть недоступними. Якби клас *Program* розташовувався в іншому пакеті, то йому були б доступні з класу *Person* тільки поля та методи з модифікатором *public*. Модифікатор доступу повинен передувати іншій частині визначення змінної або методу.

Здавалося б, чому б не оголосити всі змінні та методи з модифікатором public, для того щоб вони були доступні в будь-якій точці програми незалежно від пакета або класу! Наприклад, поле *age*, яке представляє вік. Якщо інший клас має прямий доступ до цього поля, то є ймовірність, що в процесі роботи програми цьому полю буде передано некоректне значення, наприклад, від'ємне число. Така зміна даних не є бажаною. Іноді є потреба в тому, щоб деякі дані були безпосередньо доступні, напр., для виведу на консоль. У зв'язку з цим рекомендується якомога більше обмежувати доступ до даних, щоб захистити їх від несанкціонованого доступу ззовні (як для отримання значення, так і для його зміни). Використання різних модифікаторів гарантує, що дані не будуть спотворені або змінені не належним чином. Подібне приховання даних всередині деякої області видимості називається інкапсуляцією. Як правило, замість безпосереднього застосування полів використовують методи доступу. Наприклад, для поля *age* можна створити наступний метод:

```
public int getAge(){
   return this.age;
}
public void setAge(int age){
   if(age > 0 && age < 110) this.age = age;
}
```

При створенні об'єктів класу для кожного об'єкта створюється своя копія не статичного поля. А **статичні поля** є загальними для всього класу. Тому вони можуть використовуватися без створення об'єктів класу. Наприклад:

```
public class Program {
   public static void main (String[] args) {
      Person vasia = new Person();
      Person petia = new Person();
      vasia.printId(); // Id = 1
      petia.printId(); // Id = 2
      System.out.println(Person.counter); // 3
      Person.counter = 8;  // зміна Person.counter
      Person george = new Person();
      george.printId(); // Id = 8
   }
}
class Person {
   private int id;
   static int counter = 1;
   Person() { id = counter++; }
   public void printId() {
      System.out.printf( "Id: %d  \n", id);
   }
}
```

Клас *Person* містить статичну змінну *counter*, яка збільшується в конструкторі і її значення присвоюється змінній *id*. Тому при створенні кожного нового об'єкта *Person* змінна *counter* буде збільшуватися, і у кожного нового об'єкта *Person* значення поля *id* буде на одиницю більше ніж у попереднього. Оскільки змінна *counter* статична, то ми можемо звернутися до неї в програмі по імені класу.

Також статичними бувають константи, які є спільними для всього класу.

Статичний ініціалізатор призначений для ініціалізації статичних змінних, або для виконання дій, які виконуються при створенні найпершого об'єкта. Статичний ініціалізатор визначається як звичайний, але перед ним ставиться ключове слово *static*.

Статичні методи також відносяться до всього класу в цілому. Наприклад, в прикладі вище статична змінна counter була доступна ззовні, і її значення можна змінити поза класом *Person*. Зробимо її недоступною для зміни ззовні, але доступною для читання. Для цього використовуємо статичний метод:

```
public static void displayCounter() {
    System.out.printf("Counter: %d  \n", counter);
}
```

При використанні статичних методів треба враховувати обмеження: в статичних методах можна викликати тільки інші статичні методи і використовувати тільки статичні змінні. Взагалі, методи визначаються як статичні, коли методи не впливають на стан об'єкта, тобто його нестатичні поля і константи. Для виклику статичного методу немає сенсу створювати екземпляр класу. Наприклад:

```
public class Program {
   public static void main(String[] args) {
      System.out.println(Operation.subtract(45, 23)); // 22
      System.out.println(Operation.sum(45, 23)); // 68
      System.out.println(Operation.multiply(4, 23)); // 92
   }
}
class Operation {
   static int subtract(int a, int b) { return a - b; }
   static int sum(int a, int b) { return a + b; }
   static int multiply(int a, int b) { return a * b; }
}
```

В даному випадку для методів *subtract, sum, multiply* немає значення, який саме екземпляр класу *Operation* використовується. Ці методи працюють тільки з параметрами, не зачіпаючи стан класу. Тому їх можна визначити як статичні.

3.2. Клас Object та його методи.

Хоча можна створити звичайний клас, який не є спадкоємцем, але, фактично, всі класи успадковуються від класу *Object*. Всі інші класи, навіть ті, які додані в проект, є неявно похідними від класу *Object*. Тому всі типи і класи можуть реалізувати ті методи, які визначені в класі *Object*. Розглянемо ці методи:

1. Метод *toString* служить для отримання представлення даного об'єкта у вигляді рядка. При спробі вивести рядкове подання об'єкта, як правило, буде виводитися повне ім'я класу. Наприклад:

```
public class Program {
   public static void main (String[] args) {
      Person yuzik = new Person("Yuzik");
      System.out.println(yuzik.toString());
     // Буде виводити щось типу Person@7960847b
   }
}
class Person {
   private String name;
   public Person(String n) { name = n; }
}
```

Отримане значення (у даному випадку *Person@7960847b*) навряд чи є хорошим рядковим описом об'єкта. Тому метод *toString()* прийнято перевизначати. Наприклад:

```
public class Program {
   public static void main (String[] args) {
      Person yuzik = new Person("Yuzik ");
      System.out.println(yuzik.toString()); // Person Yuzik
```

```
    }
}
class Person {
    private String name;
    public Person(String n) { name = n; }
    @Override
    public String toString() { return "Person " + name; }
}
```

2. Метод *hashCode* дозволяє задати деяке числове значення, яке буде відповідати даному об'єкту або його хеш-код. За цим числом, наприклад, можна порівнювати об'єкти. Наприклад, виведемо код вищестореного об'єкта:

```
Person yuzik = new Person("Yuzik");
System.out.println(yuzik.hashCode ()); // 2036368507
```

Можна використати свій алгоритм визначення хеш-коду об'єкта:

```
 @Override
    public int hashCode() {
        return 10 * name.hashCode() + 20456;
    }
```

3. Метод *getClass* дозволяє отримати тип об'єкта:

```
Person yuzik = new Person("Yuzik");
System.out.println (yuzik.getClass()); // class Person
```

4. Метод *equals* порівнює два об'єкти на рівність:

```
public class Program {
    public static void main (String [] args) {
        Person vasia = new Person("Vasia");
        Person petia = new Person("Petia");
        System.out.println (Vasia.equals(Petia)); // false
        Person vasia2 = new Person("Vasia");
        System.out.println(vasia.equals(vasia2)); // true
    }
}
class Person {
    private String name;
    public Person (String name) { this.name = name; }
    @Override
    public boolean equals(Object myObject) {
        if (!(myObject instanceof Person)) return false;
        Person per = (Person)myObject;
        return this.name.equals(per.name);
    }
}
```

Метод *equals* приймає як параметр об'єкт будь-якого типу, який порівнюється з поточним, якщо вони є об'єктами одного класу. **Оператор *instanceof*** дозволяє з'ясувати, чи є переданий в якості параметра об'єкт об'єктом певного класу, в даному випадку класу *Person*. Якщо об'єкти належать до різних класів, то їх порівняння не має сенсу, і повертається значення false. У протилежному випадку, об'єкти порівнюються по іменах, і якщо вони збігаються, то результат *true*, що означає, що об'єкти рівні.

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 класом згідно свого варіанту. Створити не менше 10 записів для виконання операцій, вказаних у завданні. Для ідентифікації спроби введення з клавіатури некоректних даних описати виключення.

(2 бала)

1. 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.
2. 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.
3. 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.
4. 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.
5. 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.
6. 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.
7. 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».
8. 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.
9. 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".
10. 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.
11. 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.
12. 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.

13. 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.

14. 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.

15. 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.

16. 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.

17. 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.

18. 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.

19. 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.

20. 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).

21. 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.

22. 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.

23. 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.

24. 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.

25. 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.

2. 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. Всі поля класу зробити закритими, а доступ до них реалізувати через `get`і `set`. Реалізувати конструктори з параметрами та без параметрів, а ініціалізацію полів виконати через властивості. Реалізувати методи для:

– додавання записів;

– редагування записів;

– знищення записів;

– виведення інформації з файла на екран;

– пошук потрібної інформації за конкретною ознакою (поле Параметр пошуку);

– сортування за різними полями (поле Параметр сортування).

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

(3 бала)

№ варіанта	Клас	Поля	Параметр сортування	Параметр пошуку
1.	Бібліотека	Інвентарний номер, автор, назва, кількість сторінок, рік видання	Рік видання	Автор
2.	Телефонний довідник	Прізвище, ім'я, по батькові, адреса, телефон.	Телефон	Прізвище
3.	Розклад руху літаків	Номер рейсу, тип літака, напрямок руху, періодичність вильоту.	Номер рейсу	Тип літака
4.	Колекція компакт-дисків	Інвентарний номер, назва альбому, об'єм диску, тип, дата запису.	Дата запису	Назва альбому
5.	Записна книжка	Прізвище, ім'я, асса м адреса, телефон, електронна пошта.	Прізвище	Електронна пошта
6.	Предметний покажчик	Слово; номера сторінок, де це слово зустрічається; кількість цих слів на даній сторінці	Номер сторінки	Слово
7.	Розклад пар	Номер пари, день тижня, предмет, прізвище викладача, форма заняття.	День тижня	Прізвище викладача
8.	Список файлів	ім'я файла, розширення, розмір, дата створення, атрибути.	Розширен-ня	Дата створення
9.	Архів програмного забезпечення	Назва програми, операційна система, розмір програми, дата запису	Назва програми	Операційна система
10.	Рахунки банку	Прізвище, ім'я, дата останньої операції, сума вкладу	Сума вкладу	Дата операції

№ варіанта	Клас	Поля	Параметр сортування	Параметр пошуку
11.	Користувачі локальної мережі	Прізвище, група, обліковий запис, тип облікового запису.	Тип облікового запису	Прізвище
12.	Камера схову	Прізвище, дата здачі, термін зберігання, інвентарний номер та назва предмета	Інвентарний номер	Дата здачі
13.	Склад товарів	Інвентарний номер, назва товару, вага, ціна, кількість	Вага	Назва товару
14.	Каса продажу квитків	Назва пункту, час відправлення, дата відправлення, час прибуття, дата прибуття, ціна квитка	Час від-правлення	Назва пункту
15.	Успішність студентів	Прізвище, номер групи, оцінки з трьох предметів	Прізвище	Номер групи
16	Асортимент виробів	Номер виробу, назва, вага, собівартість виробництва, ціна продажу	Назва	Ціна продажу
17	Метеорологічні дані	Дата, місто, атмосферний тиск, температура повітря, швидкість вітру	Дата	Місто
18	Склад автозапчастин	Порядковий номер, назва, марка авто, ціна, кількість	Назва	Марка авто
19	Пацієнти сімейного лікаря	Номер карти, Прізвище, ім’я, по-батькові, рік народження, стать	Прізвище	Стать
20	Викладачі університету	Номер посвідчення, Прізвище, ім’я, по-батькові, кафедра, стать, науковий ступінь	Прізвище	Кафедра
21	Власники квартир ОСББ	Прізвище, номер будинку, номер квартири, стать, сума боргу	Прізвище	Сума боргу

№ варіанта	Клас	Поля	Параметр сортування	Параметр пошуку
22	Колекція музичних треків	Назва пісні, виконавець, альбом, рік випуску, тривалість пісні	Назва пісні	Виконавець
23	Колекція фільмів	Назва, прізвище режисера, рік випуску, кіностудія, тривалість фільму	Назва	Рік випуску
24	ЖРЕПи міста	Назва, адреса, прізвище начальника, кількість підзвітних будинків, район міста	Назва	Район міста
25	Абоненти кабельної мережі	Номер договору, Прізвище, ім’я, адреса, телефон	Номер договору	Прізвище