Лабораторна робота № 2

Застосування технологій об'єктно-орієнтованого програмування у програмах на Java

Мета роботи: Навчитися проєктувати класи, створювати об'єкти класів, використовувати успадкування класів та поліморфізм у вигляді перезавантажених та перевизначених методів. Вивчити технології введення абстракцій поведінки об'єктів у вигляді абстрактних класів і інтерфейсів.

1. Виконання роботи

BlueJ являє собою вільне для некомерційного використання середовище розробки програм на мові Java, створене в університеті Монаша в Мельбурні (Австралія) і в університеті Кент в Кентербері (Великобританія) для навчання програмуванню на Java. Установчий файл *BlueJ* можна скопіювати за адресою http://www.bluej.org. *BlueJ* поширюється для різних платформ: для систем Windows, для MacOS, і для всіх інших систем. Для використання *BlueJ* на Вашому комп'ютері повинен бути встановлений JDK. Інсталяція середовища здійснюється стандартним чином з інсталяційного файлу. Існує можливість вибору мови інтерфейсу в меню *Tools-Preferences*, вкладка *Interface*.

Для створення проекту в середовищі *BlueJ* виберіть з *Головного меню* команду *Проект-Новий проект*, вкажіть місцезнаходження проекту та його ім'я. Після натискання на кнопку *Створити* в стандартному діалоговому вікні роботи з файлами буде створений проект і відкриється вікно середовища для роботи з проектом (Рис. 1). Єдина піктограма в робочій області середовища являє собою текстовий файл Readme, у якому можна внести інформацію про проект та його автора, включаючи назву проекту, призначення проекту, версію або дату створення проекту, команду запуску проекту, інструкції користувача, тощо.

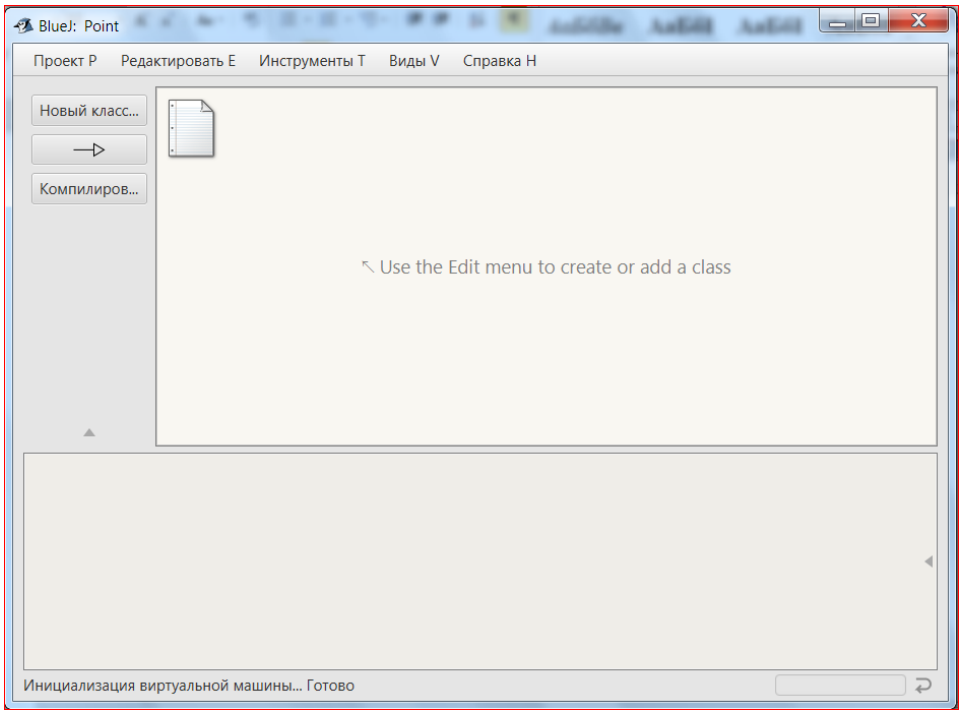

Рис. 1. Вікно середовища BlueJ для роботи з проектом

Для початку роботи необхідно створити клас і визначити в ньому необхідні змінні і методи, які будуть визначати властивості і поведінку об'єктів цього класу. Для прикладу створимо клас Point (точка), що описує властивості точки на площині і застосовні до цього об'єкта методи. Для цього в середовищі *BlueJ* необхідно клацнути на кнопці *Новий клас* і в діалоговому вікні вказати ім'я класу і його тип (поки виберемо тип *Клас*, що характеризує звичайний клас).

Після натискання на кнопку *ОК* в робочій області з'явиться піктограма класу з штрихуванням всередині. Штрихування позначає, що клас ще не скомпільовано. Виконайте клацання правою кнопкою миші на класі Point і виберіть команду *Відкрити редактор*. Відкриється вікно *Редактора* вихідного коду з шаблоном коду класу Point (Рис. 3). Відредагуйте шаблон відповідно до наведеного нижче коду, що визначає атрибути класу Point – змінні цілого типу `x` і `y`, які зазначають координати точки на площині, і двома конструкторами, що використовуються для створення об'єктів класу Point: конструктор без параметрів, який встановлює координати точки в нуль і конструктор, до якого координати точки передаються як аргументи. Зверніть увагу на наявність документованих коментарів класу і конструкторів.

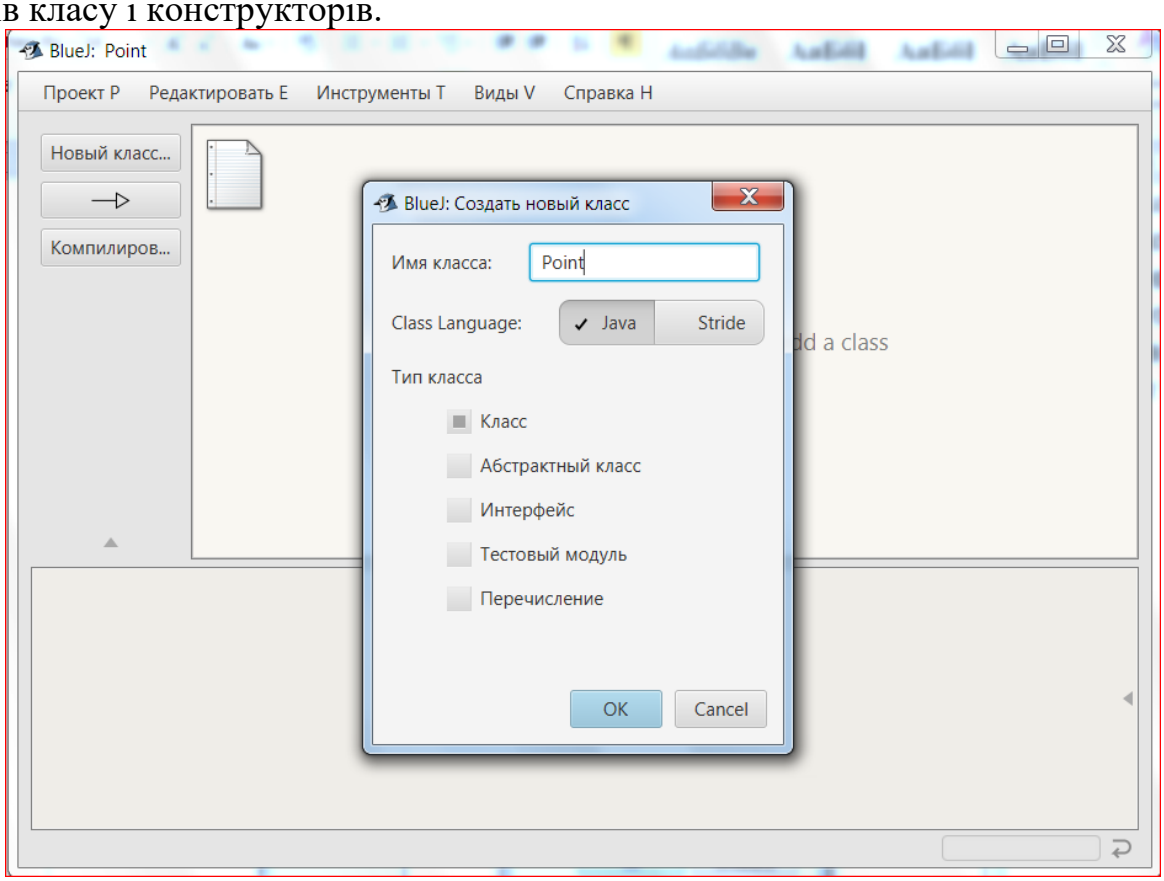

Рис. 2. Створення нового класу в середовищі BlueJ

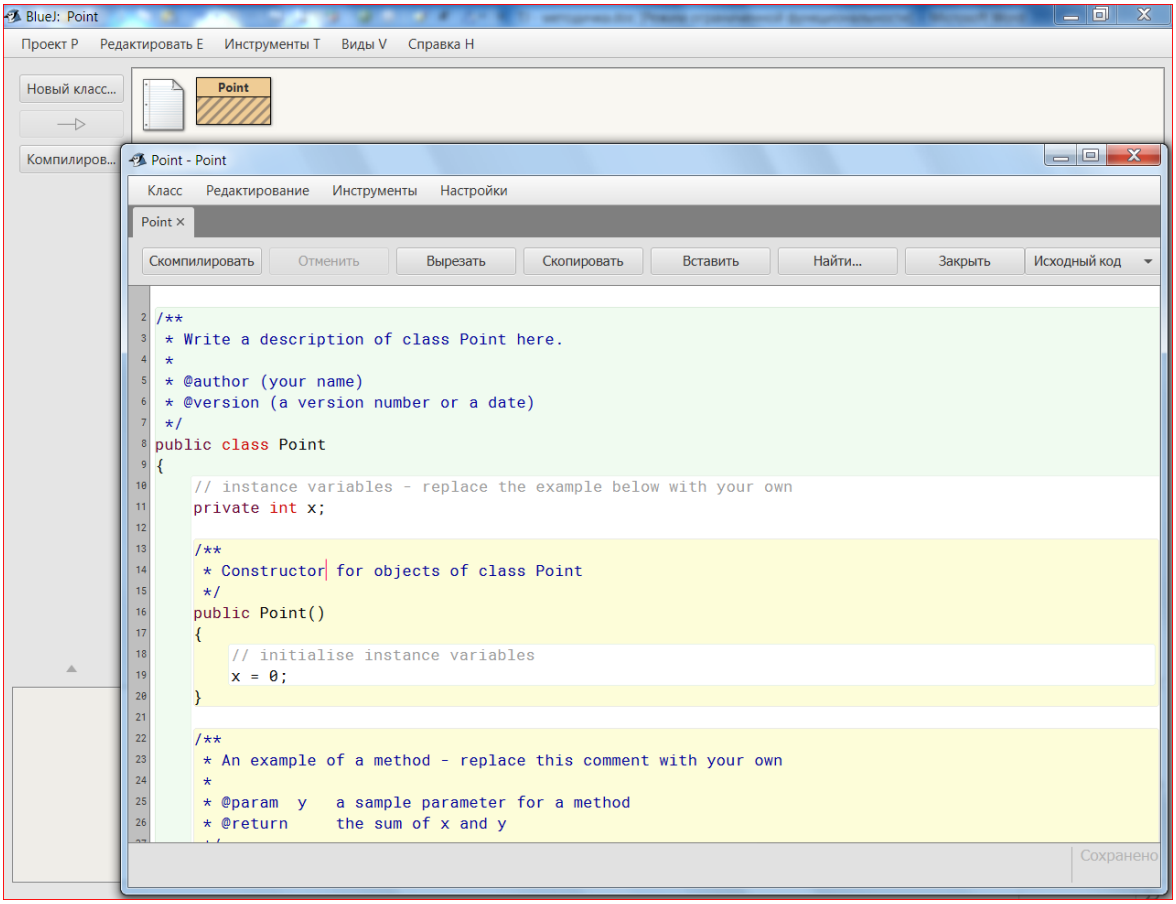

Рис. 3. Вікно Редактора вихідного коду

```
/**
 * Клас, що описує точку на площині
 * @author student
 * @version 1.01
 */
public class Point
{
    /**
     * Координати точки на площині.
     */
    private int x;
    private int y;

    /**
     * Конструктор об'єктів класу Point без параметрів.
     */
    public Point()
    {
        //ініціалізація змінних об'єкта
        x = 0;
        y = 0;
    }
    /**
     * Конструктор об'єктів класу Point з параметрами
     */
```

```
    public Point(int x, int y)
    {
        //ініціалізація змінних об'єкта
        this.x = x;
        this.y = y;
    }
}
```

Після редагування вихідного коду класу Point виконайте клацання по кнопці *Скомпілювати* у вікні *Редактора* вихідного коду. Тепер штрихування на піктограмі класу зникло і можна створювати об'єкт класу – точку на площині з координатами `x` і `y`. Для цього виконайте клацання правою кнопкою миші по піктограмі класу і виберіть з контекстного меню команду створення об'єкта за допомогою конструктора без параметрів `new Point()`. У діалоговому вікні задайте ім'я об'єкта (екземпляра класу Point), наприклад `point1`, і натисніть *ОК* (Рис. 4). На *Панелі об'єктів* знизу робочого вікна середовища з'явиться червона піктограма об'єкту `point1` класу Point (Рис. 5**).** Аналогічно, створіть об'єкт `point2` класу Point, але вже за допомогою параметризованого конструктора, вказавши в діалоговому вікні цілі значення координат `x` і `y` другої точки.

Виконайте інспекцію створених об'єктів командою *Перевірити* з контекстного меню об'єктів. Відкриється вікно *Інспектора об'єктів*, в якому будуть відображені значення властивостей екземпляра класу (тобто координат `x` і `y` для нашого прикладу). Для об'єктів `point1` і `point2` координати будуть відрізнятися.

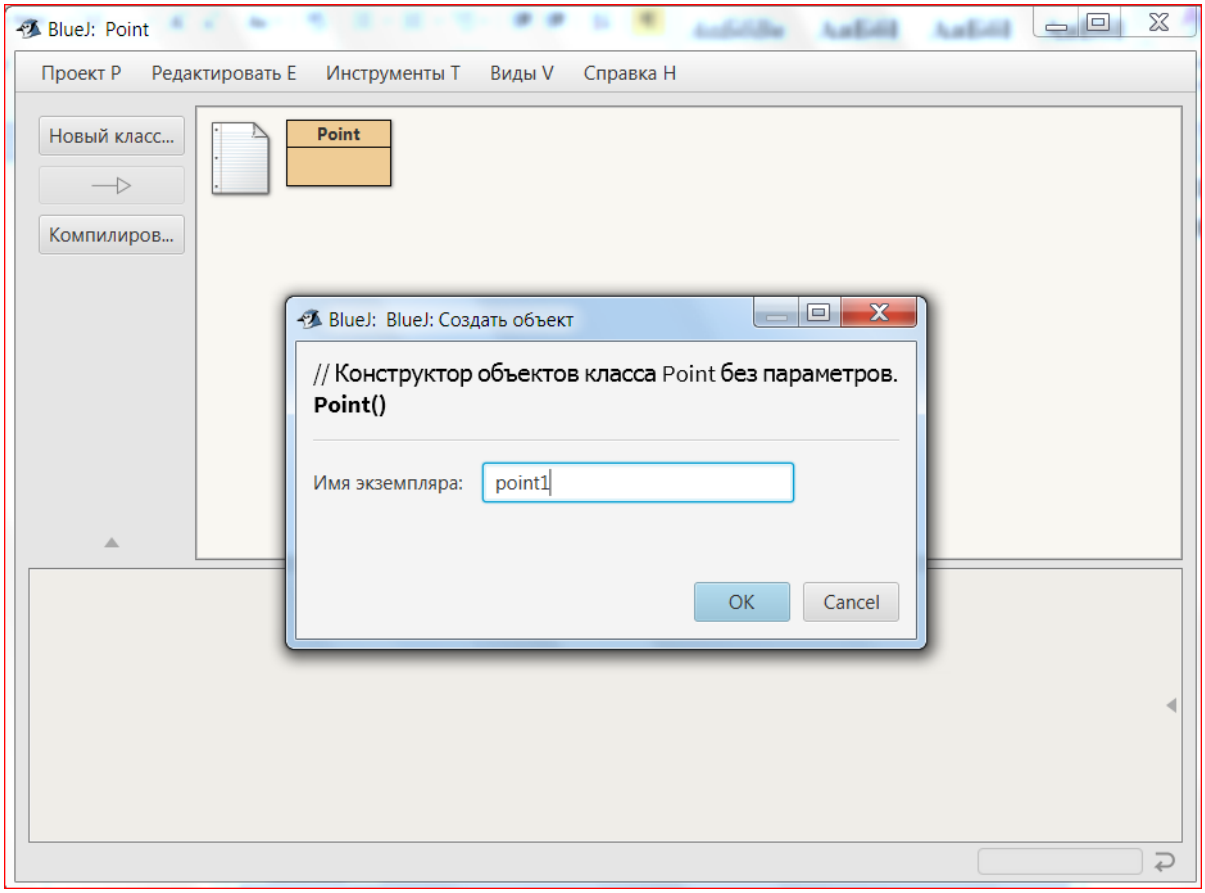

Рис. 4. Створення об'єкта (екземпляра класу) за допомогою конструктора без параметрів

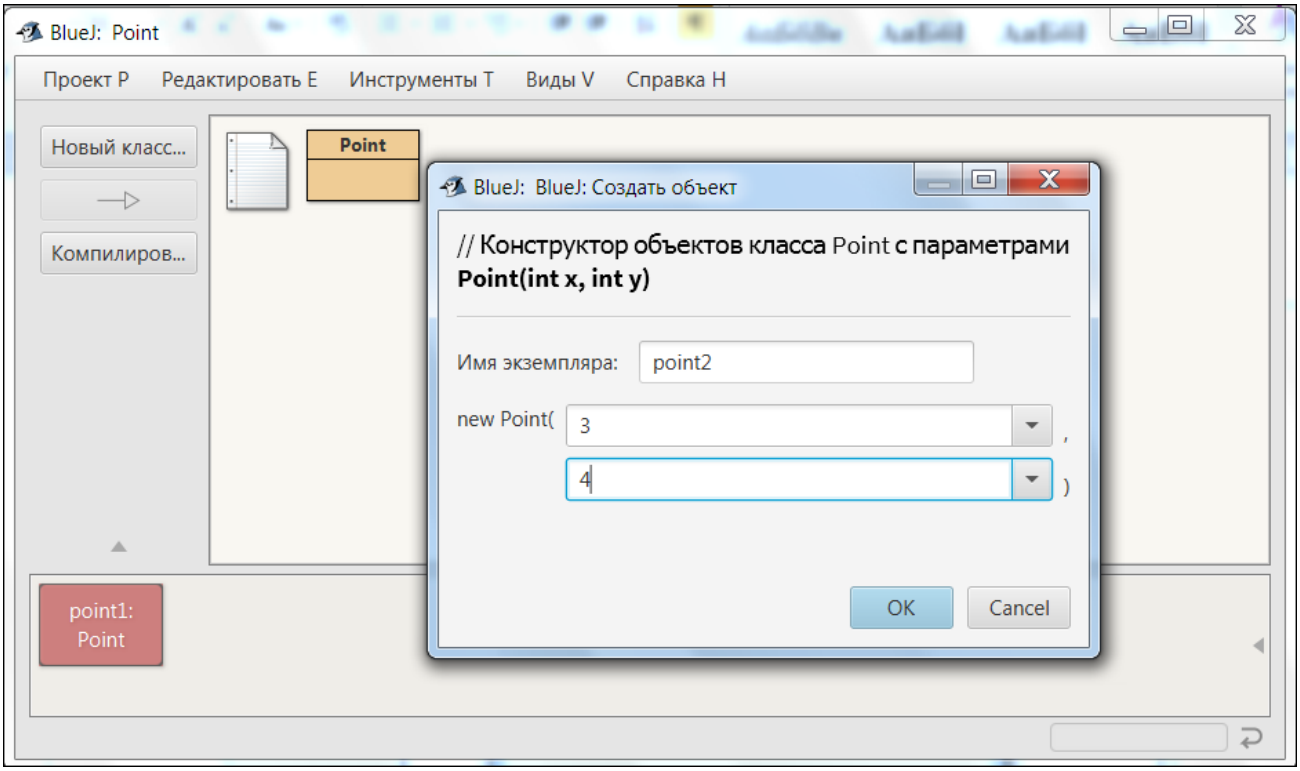

Рис. 5. Створення об'єкта (екземпляра класу) за допомогою конструктора з параметрами

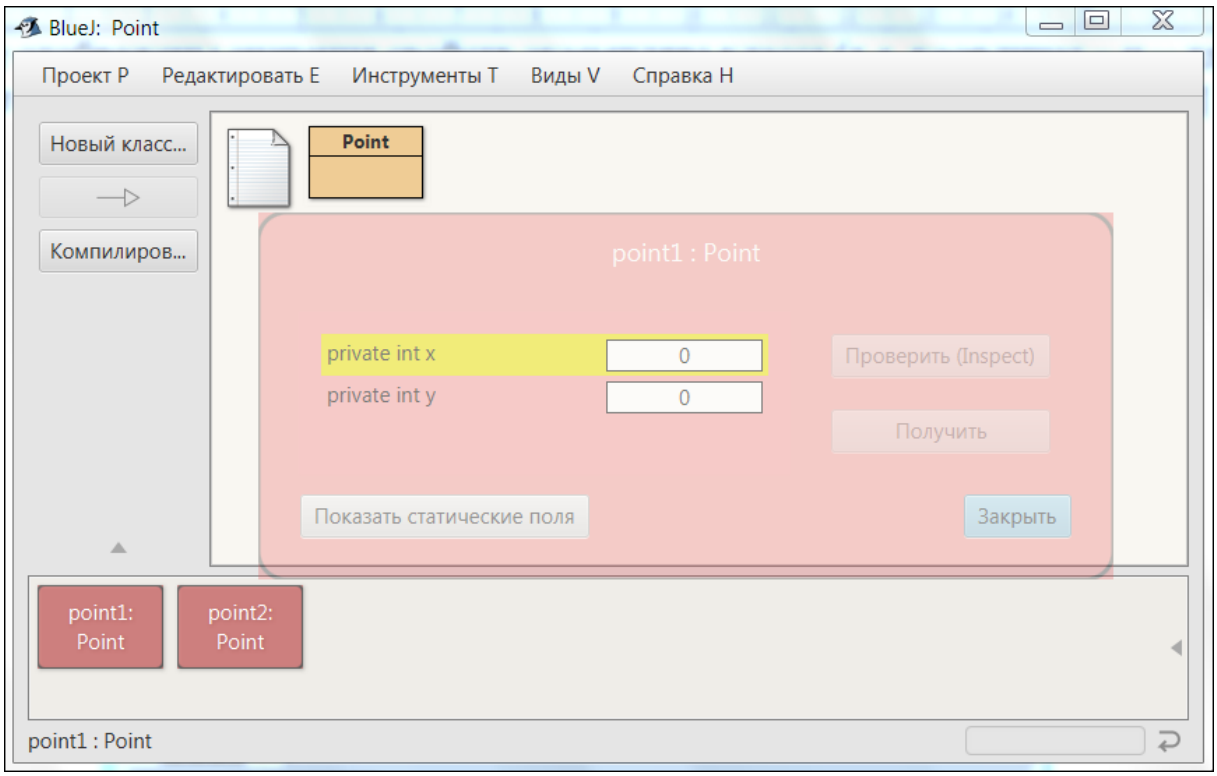

Рис. 6. Інспекція об'єкта (екземпляра класу)

Таким чином, ми навчилися визначати властивості об'єктів і задавати їм значення при створенні за допомогою конструктора. Додамо до визначення класу метод, який, наприклад, буде обчислювати відстань від нашого об'єкта (точки на площині) до іншої точки, координати якої будуть задаватися як аргументи

методу. Додайте наведений нижче код методу з коментарями після коду другого конструктора і перед закриває фігурної дужки, завершальній тіло класу Point.

```
/**
 * Метод, який обчислює відстань між точкою, визначеною класом
 * Point і довільної точкою на площині, заданою координатами,
 * переданими методу distance.
 * @param x, y – координати довільної точки
 * @return відстань між точкою, визначеної класом Point
 * і довільної точкою на площині
 */
public double distance(int x, int y)
{
    int dx = this.xx;
    int dy = this.yy;
    return Math.sqrt (dx * dx + dy * dy);
}
```

Зверніть увагу на використання методу `sqrt` бібліотечного класу з математичними функціями java.lang.Math. Відстань між визначеною нашим класом точкою і довільної точкою на площині розраховується як гіпотенуза прямокутного трикутника (Рис. 7). Виконайте компіляцію зміненого класу Point.

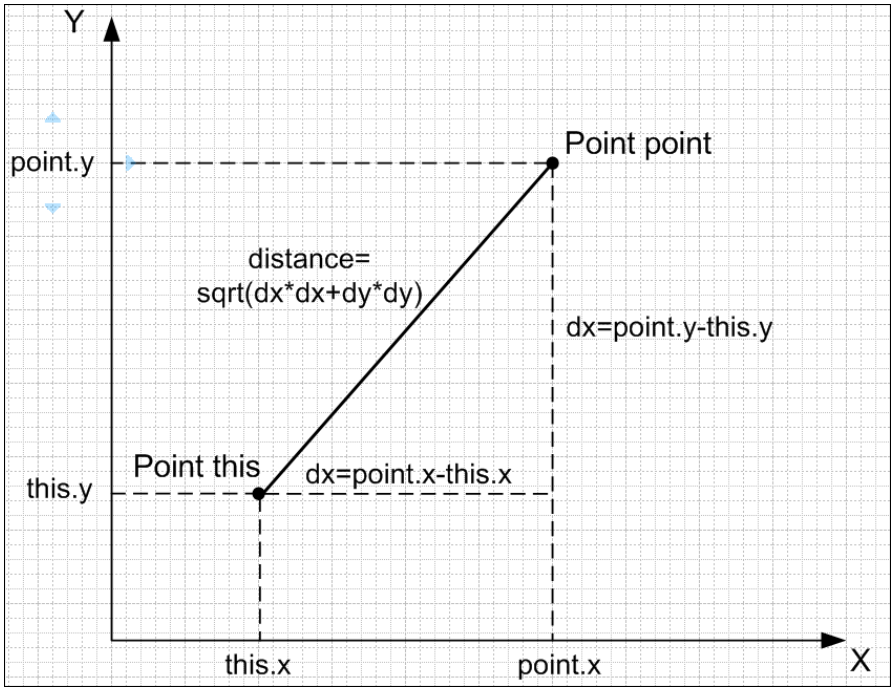

Рис. 7. Знаходження відстані між двома точками

Оскільки код класу змінився, то об'єкти `point1` і `point2` з *Панелі об'єктів* були вилучені. Створіть ще раз ці об'єкти: `point1` зі значеннями `x = 0, y = 0` і `point2` зі значеннями `x = 3, y = 4`. Викличте з об'єкта `point1` на *Панелі об'єктів* метод `distance(int x, int y)`, Передавши йому в якості значень координат точки, до якої буде обчислюватися відстань `x = 3` і `y = 4` **(**Рис. 8**).**

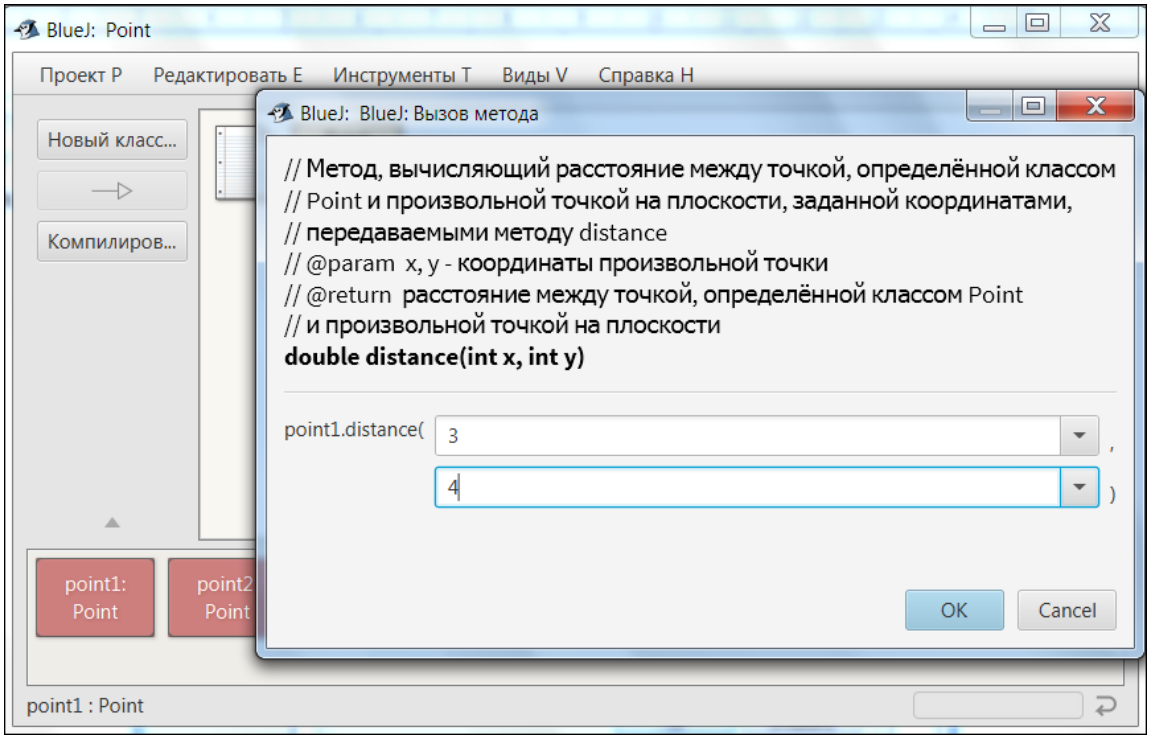

Рис. 8. Виклик методу point1.distance з передачею йому координат довільної точки на площині

Після натискання на кнопку *OK* з'являється вікно з результатом, який повертається методом – відстанню від `point1` до точки з координатами `x= 3, y= 4` (`= 5.0`).

Додамо до класу ще один метод, який обчислює за аналогією з попереднім відстань, але вже між двома об'єктами класу Point. Для цього додайте до вихідного коду класу Point оголошення ще одного методу `distance`, аргументом якого буде об'єкт класу Point:

```
/**
 * Метод, який обчислює відстань між двома об'єктами класу Point.
 * @param point – об'єкт класу Point, до якого обчислюється
 * відстань
 * @return відстань між поточним об'єктом класу Point
 * і іншим об'єктом класу Point, що задається як аргумент
 */
public double distance(Point point)
{
    return distance(point.x, point.y);
}
```

Даний метод викликає раніше визначений метод `distance`, передаючи йому в якості аргументів значення координат `x` і `y` об'єкта, відстань до якого потрібно обчислити. Виконайте компіляцію коду класу, а потім ще раз створіть два об'єкти: об'єкт `point1` з координатами`x = 0`, `y = 0`, і об'єкт `point2` з координатами `x = 3`, `y = 4`. Викличте метод `point1.distance(Point point)`, передавши

йому в якості аргументу ім'я об'єкта `point2` клацанням миші по цьому об'єкту на *Панелі об'єктів* (Рис. 9).

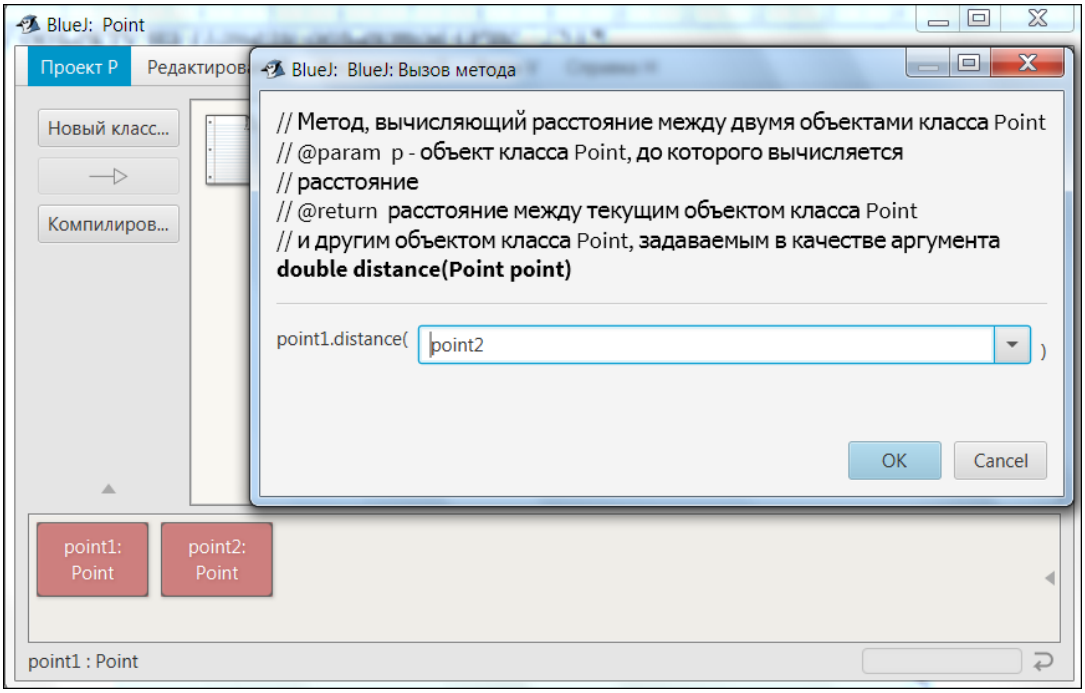

Рис. 9. Виклик методу point1.distance з передачею йому об'єкта Point point2

Результат розрахунку відстані повинен співпасти з попереднім, оскільки координати об'єктів мають ті ж значення.

BlueJ дозволяє генерувати документацію проекту за допомогою утиліти javadoc аналогічно тому. як це було описано в лабораторній роботі №1. Для генерування документації слід в *Головному вікні* BlueJ вибрати команду *Інструменти-Документація проекту*. Після її вибору по javadoc-коментарям буде згенерована документація і вона відкриється в браузері за замовчуванням (Рис. 10).

Для практичного вивчення технологій успадкування та поліморфізму створіть в навчальному проекті в *BlueJ* ще один клас Point3D, що представляє точку в тривимірному просторі. Оскільки властивості точки в просторі включають координати `x` і `y`, має сенс успадкувати цей клас від раніше створеного класу Point. Для організації успадкування класів виконайте клацання на кнопці зі стрілкою з наконечником у вигляді незаповненого трикутника, потім виконайте клацання по класі-спадкоємцю (Point3D) і, не відпускаючи кнопку миші, протягніть стрілку до бащового класу (Point). В результаті Ви повинні отримати діаграму класів, наведену на Рис. 11.

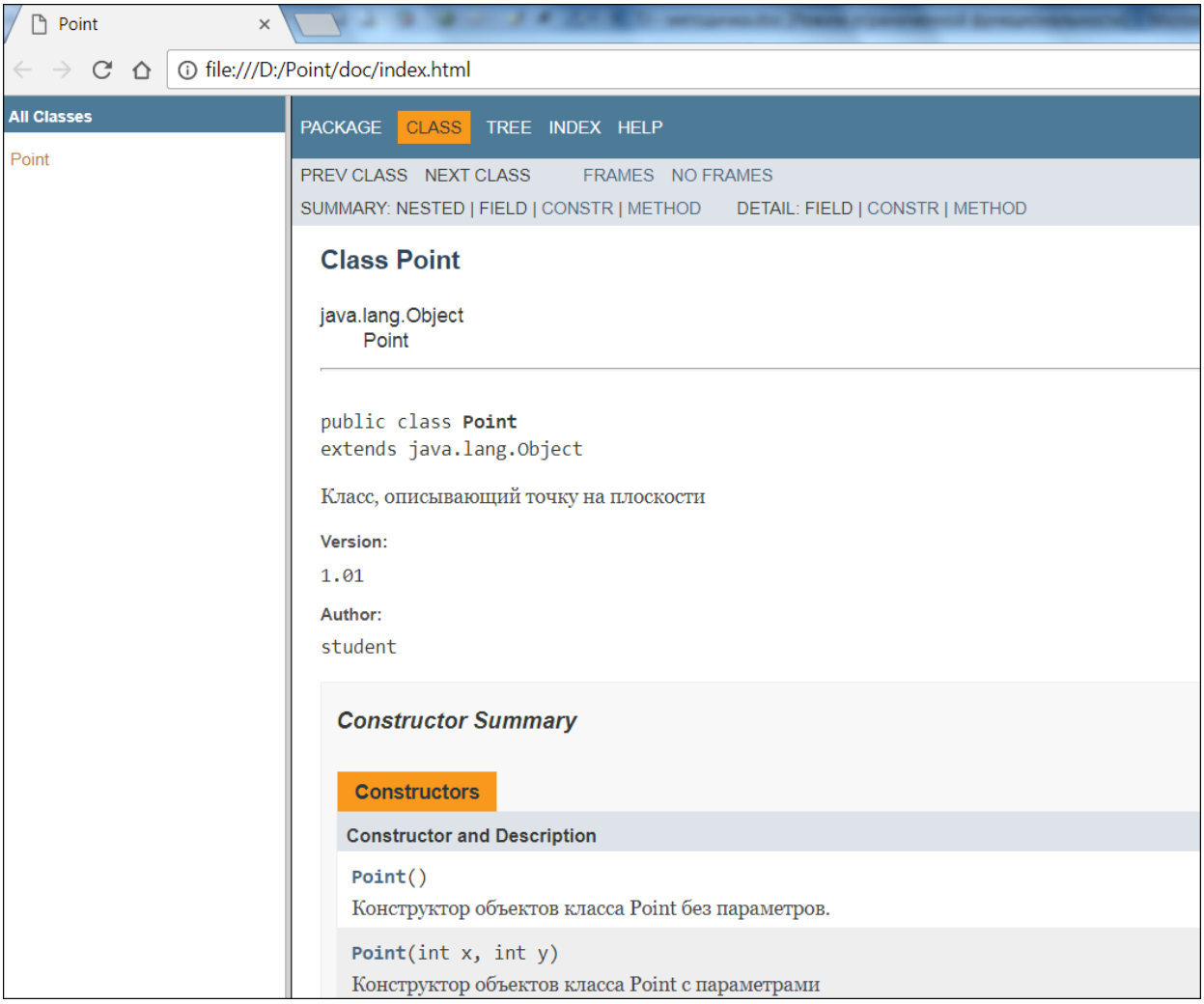

Рис. 10. Згенерована документація проекту

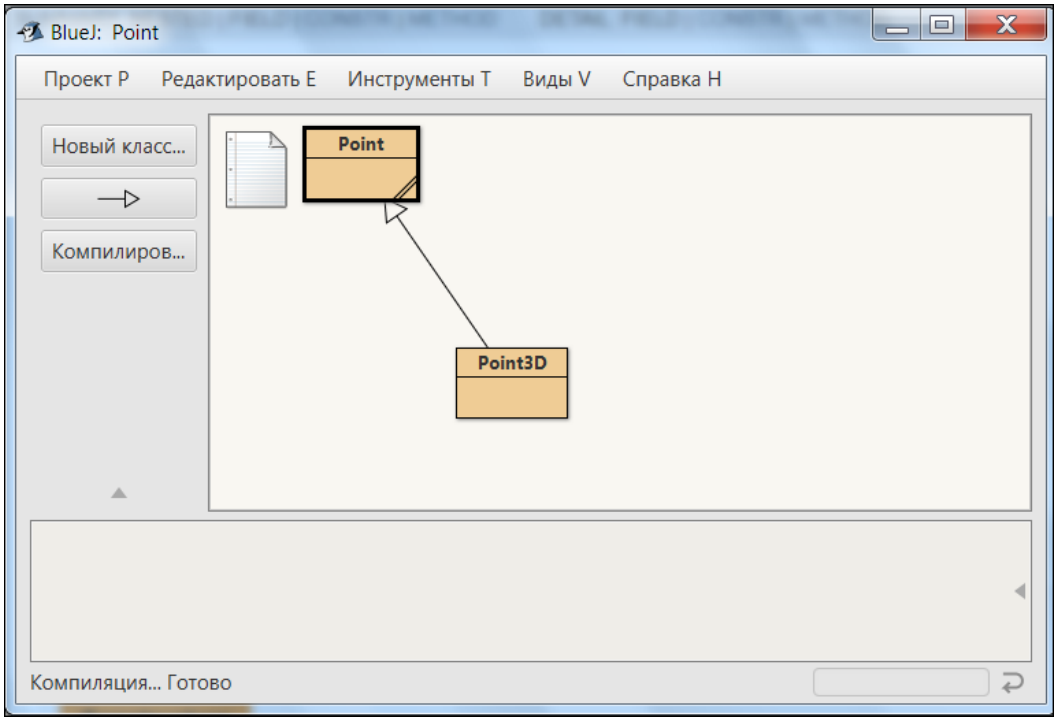

Рис. 11. Організація успадкування класів в BlueJ

Оскільки клас-спадкоємець Point3D повинен мати доступ до властивостей класу Point, що визначає координати `x` і `y` точки, змініть модифікатор доступу цього поля з `private` на `protected` (відкритий для методів самого класу і всіх його спадкоємців). Виконайте компіляцію класу Point. Відкрийте в редакторі вихідного коду клас Point3D і додайте до нього наступний код (зверніть увагу на те, що *BlueJ* в оголошенні класу Point3D додала оператор `extends`, що визначає успадкування від класу Point):

```
/**
 * Клас, який представляє точку в тривимірному просторі.
 * Успадковує клас точки на площині Point.
 * @author student
 * @version 1.01
 */
public class Point3D extends Point
{
    /**
     * Третя координата точки, додається до x і y суперкласу.
     */
    private int z;  //ця змінна не використовується методами
                    //інших класів, тому може бути private

    /**
     * Конструктор об'єктів класу Point3D з інтерактивним введенням
     * координат.
     */
    public Point3D(int x, int y, int z)
    {
        super(x, y); //виклик конструктора суперкласу
        this.z = z;
    }
    /**
     * Конструктор об'єктів класу Point3D, що виконує ініціалізацію
     * координат за замовчуванням.
     */
     public Point3D()
     {
         this(0, 0, 0);
     }

    /**
     * Метод, який обчислює відстань між точкою, визначеної класом
     * Point3D, і довільною точкою в просторі, заданою
     * координатами, переданими методу distance.
     * @param x, y, z – координати довільної точки
     * @return відстань між точкою, визначеної класом Point3D
     * і довільної точкою в просторі
     */
    public double distance(int x, int y, int z)
    {
        int dx = this.x-x;
        int dy = this.y-y;
        int dz = this.z-z;
        return Math.sqrt (dx*dx + dy*dy + dz*dz);
    }
```

```
    /**
     * Метод, який обчислює відстань між двома об'єктами класу
     * Point3D.
     * @param point – об'єкта класу Point3D, до якого
     * обчислюється відстань від поточного об'єкта
     * класу Point3D
     * @return відстань між двома об'єктами класу Point3D
     */
    public double distance(Point3D point)
    {
        return distance(point.x, point.y, point.z);
    }

    /**
     * Метод, який обчислює відстань між об'єктом класу Point3D і
     * довільною точкою на площині – заміщення методу
     * суперкласу.
     * @param x, y – координати точки на площині, до якої
     * обчислюється відстань від поточного об'єкта класу Point3D
     * @return відстань між точкою на площині
     * і поточним об'єктом класу Point3D
     */
    @Override
    public double distance(int x, int y)
    {
        double dx = this.x/z – x;
        double dy = this.y/z – y;
        return Math.sqrt(dx*dx + dy*dy);
    }


    /**
     * Метод, який обчислює відстань між об'єктом класу Point3D і
     * об'єктом класу Point на площині – заміщення методу
     * суперкласу.
     * @param p – об'єкт класу Point на площині, до якого
     * обчислюється відстань від поточного об'єкта класу Point3D
     * @return відстань між точкою на площині
     * і поточним об'єктом класу Point3D
     */
    @Override
    public double distance(Point point)
    {
        double dx = this.x/z – point.x;
        double dy = this.y/z – point.y;
        return Math.sqrt(dx*dx + dy*dy);
    }
}
```

У наведеному коді класу Point3D додається третя координата цілого типу (`z`), А координати `x` і `y` успадковуються від Point. Далі наведені два конструктора: один з передачею координат точки в якості аргументів, а інший – без параметрів, виконує ініціалізацію всіх трьох координат нульовими значеннями. Зверніть увагу на те, що перший конструктор класу-спадкоємця викликає конструктор

батьківського класу за допомогою зарезервованого слова `super`, це дозволяє повторно використовувати код конструктора суперкласу для завдання координат точок `x` і `y`. Далі наведені два методи, що обчислюють відстань між двома точками в тривимірному просторі, їх алгоритми аналогічні алгоритмам методів, що обчислюють відстань між двома точками на площині в класі Point. Оскільки кількість аргументів у цих методах різниться, то можна говорити про перевантаження (overloading) методів в цьому випадку. Метод `distance(int x, int y)` класу Point3D має однакові кількості і типи аргументів з однойменним методом базового класу, тому тут можна говорити про заміщення (overriding) методу базового класу (додана анотація `@Override` сигналізує компілятору про це і страхує від помилок в імені методу, а також типах аргументів). Аналогічно, метод `distance(Point point)` заміщує відповідний метод базового класу.

Виконайте перевірку роботи класів, створивши два об'єкти класу Point3D з іменами `point3D1` і `point3D2` і координатами (`x = 0, y = 0, z = 0`) і (`x = 1, y = 4, z = 8`) (для створення першого об'єкта можна використовувати конструктор без параметрів). Виконайте метод `distance`, що обчислює відстань в просторі від об'єкта `point3D1` до точки з координатами `x = 1, y = 4, z = 8`. Результат, що повертається методом, показаний на Рис. 12. Очевидно, що такий же результат повинен повернути і перевантажений метод `point3D1.distance(point3D2)`, в який в якості аргументу буде переданий об'єктpoint3D2 з координатами `x = 1, y = 4, z = 8`. Перевірте це.

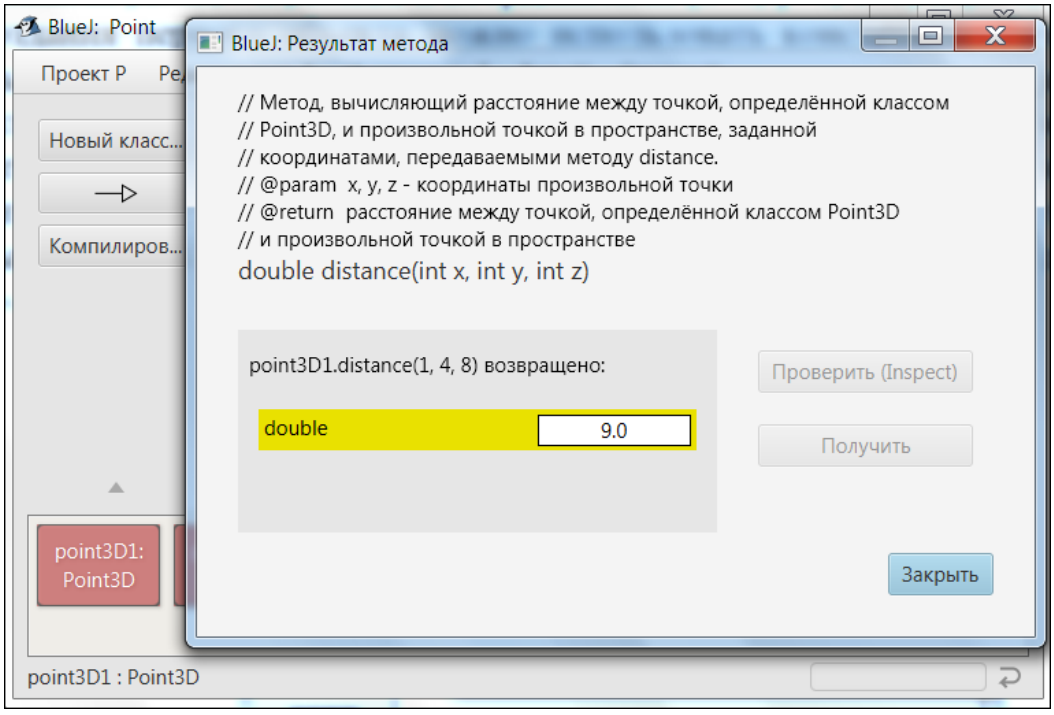

Рис. 12. Розрахунок відстані між двома точками в просторі

Додайте на *Панель об'єктів* об'єкт класу Point3D з ім'ям `point3D3` і координатами (`x = 1, y = 1, z = 1`). Виконайте розрахунок відстані між просторовим об'єктом `point3D1` і точкою на площині з координатами `x = 4, y = 5,`

викликавши метод `p1.distance(int x, int y)`. Повторіть розрахунок відстані між просторовим об'єктом `point3D3` і точкою на площині `point` з координатами `x = 4, y = 5`, викликавши метод `point3D1.distance(point)`. В обох випадках повинно отриматись відстань, рівна `5.0`. Зауважте, що вибір реалізації методу визначається аргументами, які ми передаємо методу (статичний поліморфізм) (Рис. 13), а також типом об'єкта, з якого викликається метод (динамічний поліморфізм).

Незважаючи на те, що *BlueJ* забезпечує виконання коду шляхом вибору команд з графічних уявлень класів і об'єктів, він також дозволяє створити головний клас з методом `main`, який містить код, що працює зі створеними об'єктами. Уявімо, що задача, яку необхідно вирішити – створити масив двовимірних і тривимірних точок, розташованих в ньому в довільному порядку, і вивести на екран відстані між сусідніми точками-елементами масиву, для сусідніх 2-мірної і 3-мірної точки потрібно вивести відстань між двовимірної точкою і проекцією тривимірної точки на площину. Таке завдання нетривіальне, але за допомогою поліморфізму його вдається вирішити, причому використовуючи тільки один прохід по масиву.

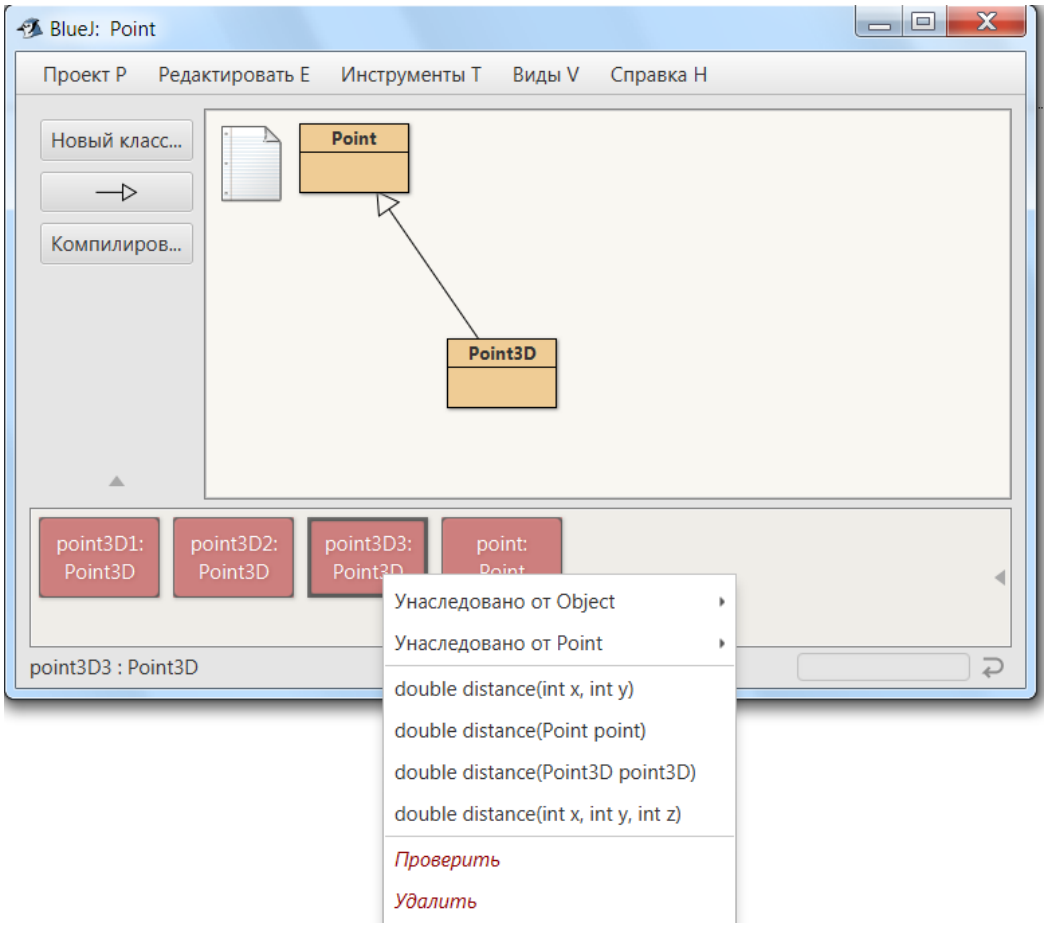

Рис. 13. Вибір поліморфних методів

Створіть клас Main, у який введіть наступний код:

```
/**
 * Клас, що демонструє динамічний поліморфізм (пізнє зв'язування).
 */
public class Main {

    private Main () {
    }

    public static void main(String[] args) {
        Point[] pointSet = {new Point3D(1, 1, 1), new Point3D(2, 5, 9),
                            new Point(1, 1), new Point(4, 5)};
        for (int i = 0; i < pointSet.length-1; i++) {
            System.out.println("Distance between " + pointSet[i]
                               + " and " + pointSet[i + 1] + " is "
                               + PointSet[i].distance(pointSet[i + 1]));
        }
    }
}
```

У наведеному коді створюється масив об'єктів класу Point з ім'ям `pointSet[]`. В цей масив можна включати і об'єкти класів-спадкоємців Point, що є перетворенням типів, що підвищує (upcasting), оскільки існують загальні для об'єктів класів Point і Point3D властивості і методи. У масив `pointSet[]` додаються і об'єкти двовимірної точки. Далі організується цикл по всіх елементах масиву (зверніть увагу на використання змінної створеного нами екземпляра масиву з ім'ям `length` – вона зберігає кількість елементів масиву). Завдання вирішене, але при виведенні оператором `System.out.println` елементів масиву виведення виглядає приблизно так:

```
Distance between Point @ 931c15 and Point @ 3f836b is 5.0
```

Для виведення замість імені класу і хеш-коду об'єкта інформації, яка має сенс для користувача, наприклад, координат об'єктів двовимірної і тривимірної точок, в класах Point і Point3D потрібно перевизначити метод `toString()` класу java.lang.Object (нагадаємо, що від цього класу неявно успадковуються всі інші класи Java). Додайте наступний код в класи Point і Point3D, відповідно:

```
    /**
     * Метод, який повертає координати двовимірної точки.
     */
    public String toString()
    {
        return "x=" + this.x + ", y=" + this.y;
    }

    /**
     * Метод, який повертає координати тривимірної точки.
     */
    @Override
    public String toString()
    {
```

```
        return "x=" + this.x + ", y=" + this.y + ", z=" + this.z;
    }
```

Виконайте компіляцію всіх класів командою натиснувши на кнопку *Компілювати ...* в *Головному вікні* програми. Оскільки метод `main` класу Main статичний, то він належить до класу і не буде доступний в об'єктах цього класу (перевірте це, створивши об'єкт). Для запуску методу `main` виконайте клацання правою кнопкою по класу Main і виберіть з меню метод `main`. У вікні для введення параметрів виберіть `{}`, що видається за замовчуванням (Нагадаємо, що в метод `main` можуть передаватися параметри, якщо вони в ньому обробляються, в нашому методі параметрів немає). Вікно, яке з'явиться після запуску методу, покаже виведення, що генерується методом `main`.

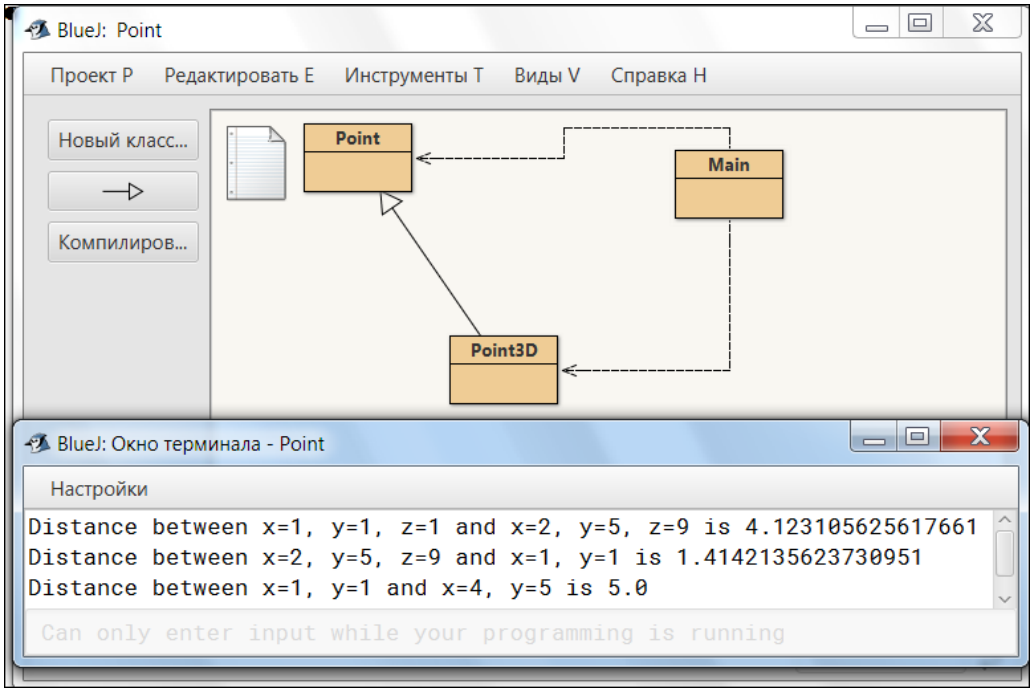

Рис. 14. Запуск статичного методу main і виведення результату

Для вивчення абстрактних класів і інтерфейсів пропонується використовувати навчальне середовище *BlueJ*, з яким Ви познайомилися при виконанні попередньої лабораторної роботи.

Створіть проект WindowAbstractClassApp і в ньому абстрактний клас Window (вікно) і його спадкоємців OneSashWindow і TwoSashWindow (одностулкове і двостулкове вікно), як показано на Рис. 15:

Для створення абстрактного класу необхідно у вікні, що відкрилося після натискання кнопки *Новый класс ...* вибрати *Абстрактный класс* (Рис. 16):

Абстрактний клас Window може містити властивості, характерні для всіх вікон, наприклад, висоту і ширину разом з методами їх отримання і установки (геттерами і сетерами), конструкторами, які зможуть використовуватись при створенні об'єктів класів-спадкоємців, і, обов'язково, абстрактні методи, характерні для всіх об'єктів класів-спадкоємців, але такі, що по-різному в них реалізуються (в прикладі це метод відкриття і метод закриття вікна).

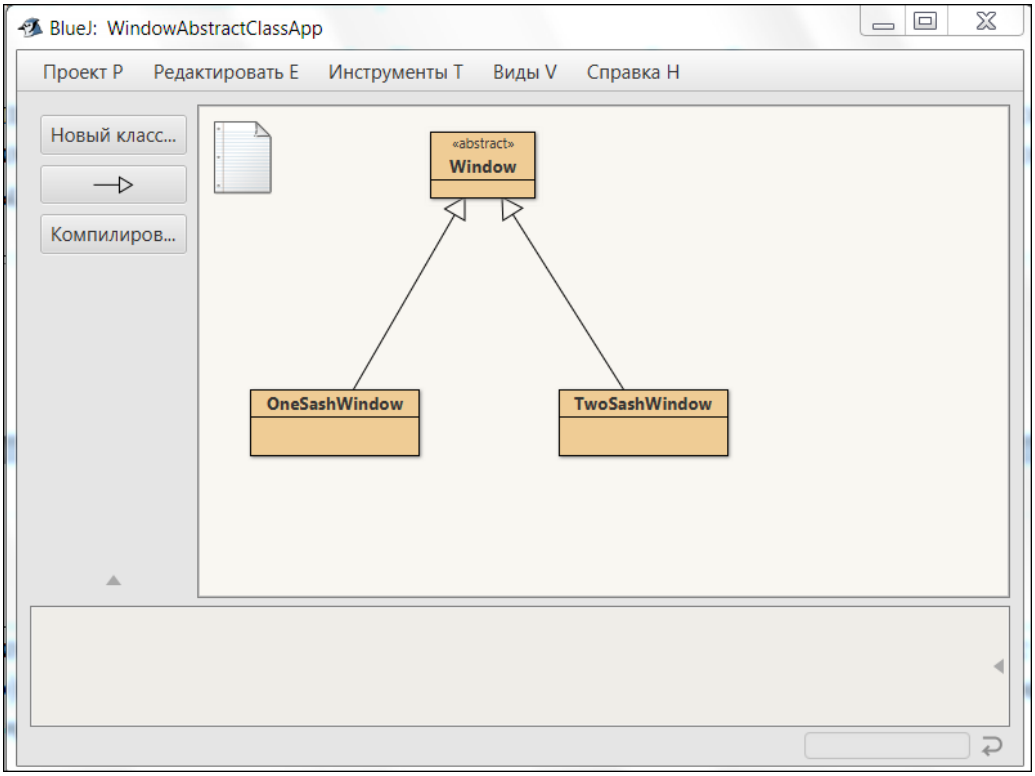

Рис. 15. Проект з абстрактним класом

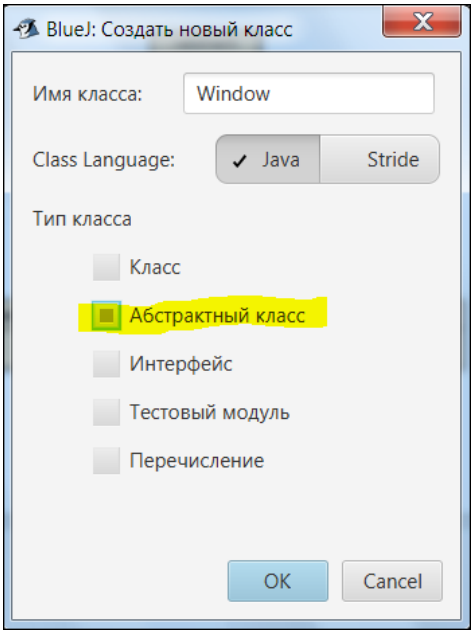

Рис. 16. Вікно вибору створюваної структури

```
/**
 * Абстрактний клас вікна з абстрактними методами відкриття
 * і закриття вікна.
 *
 * @author student
 */
public abstract class Window {

    private double width;
```

```
    private double height;

    public Window() {
    }

    public Window(double width, double height) {
        this.width = width;
        this.height = height;
    }

    /**
     * Відкрити вікно.
     */
    public abstract void open();

    /**
     * Закрити вікно.
     */
    public abstract void close();

    public double getWidth() {
        return width;
    }

    public void setWidth(double width) {
        this.width = width;
    }

    public double getHeight() {
        return height;
    }

    public void setHeight(double height) {
        this.height = height;
    }
}
```

Додайте до проекту клас OneSashWindow (представляє одностворчатое вікно), що успадковує абстрактний клас Window, з таким кодом:

```
/**
 * Конкретний клас вікна з однією стулкою.
 *
 * @author student
 */
public class OneSashWindow extends Window {

    private double sashWidth;
    private double sashHeight;
    /*Ознака відкритої стулки*/
    private boolean open;
```

```
    public OneSashWindow() {
    }

    public OneSashWindow(double sashWidth, double sashHeight,
                         double width, double height) {
        super(width, height);
        this.sashWidth = sashWidth;
        this.sashHeight = sashHeight;
    }

    @Override
    public void open() {
        if(open == false) {
            System.out.println("Відкриваю стулку вікна.");
            open = true;
        } else {
            System.out.println("Вікно вже відкрите.");
        }
    }

    @Override
    public void close() {
        if(open == true) {
            System.out.println("Закриваю стулку вікна.");
            open = false;
        } else {
            System.out.println("Вікно вже закрите.");
        }
    }

    public double getSashWidth() {
        return sashWidth;
    }

    public void setSashWidth(double sashWidth) {
        this.sashWidth = sashWidth;
    }

    public double getSashHeight() {
        return sashHeight;
    }


    public void setSashHeight(double sashHeight) {
        this.sashHeight = sashHeight;
    }
}
```

Клас OneSashWindow додатково до успадкованих містить свої властивості - ширину і висоту стулки, а також змінну-ознаку логічного типу відкриття-закриття вікна, що використовується в реалізаціях абстрактних методів.

Додайте до проекту ще один клас TwoSashWindow (представляє двостулкове вікно), що також успадковує абстрактний клас Window, з таким кодом:

```
/**
 * Конкретний клас вікна з двома стулками.
 *
 * @author student
 */
public class TwoSashWindow extends Window {

    private double leftSashWidth;
    private double leftSashHeight;
    private double rightSashWidth;
    private double rightSashHeight;
    /*Ознака відкритої лівої стулки*/
    private boolean leftSashOpen;
    /*Ознака відкритої правої стулки*/
    private boolean rightSashOpen;

    public TwoSashWindow() {
    }

    public TwoSashWindow(double leftSashWidth,
         double leftSashHeight, double rightSashWidth,
         double rightSashHeight, double width, double height) {
        super(width, height);
        this.leftSashWidth = leftSashWidth;
        this.leftSashHeight = leftSashHeight;
        this.rightSashWidth = rightSashWidth;
        this.rightSashHeight = rightSashHeight;
    }

    @Override
    public void open() {
        if(leftSashOpen == false) {
           System.out.println("Відкриваю ліву стулку вікна.");
           leftSashOpen = true;
        }
        if(rightSashOpen == false) {
           System.out.println("Відкриваю праву стулку вікна.");
           rightSashOpen = true;
        } else {
           System.out.println("Вікно вже відкрите.");
        }     }
    @Override
    public void close() {
        if(leftSashOpen == true) {
            System.out.println("Закриваю ліву стулку вікна.");
            leftSashOpen = false;
        }
        if(rightSashOpen == true) {
            System.out.println("Закриваю праву стулку вікна");
```

```
            rightSashOpen = false;
        } else {
            System.out.println("Вікно вже закрите.");
        }
    }

    public double getLeftSashWidth() {
        return leftSashWidth;
    }

    public void setLeftSashWidth(double leftSashWidth) {
        this.leftSashWidth = leftSashWidth;
    }

    public double getLeftSashHeight() {
        return leftSashHeight;
    }

    public void setLeftSashHeight(double leftSashHeight) {
        this.leftSashHeight = leftSashHeight;
    }

    public double getRightSashWidth() {
        return rightSashWidth;
    }

    public void setRightSashWidth(double rightSashWidth) {
        this.rightSashWidth = rightSashWidth;
    }

    public double getRightSashHeight() {
        return rightSashHeight;
    }

    public void setRightSashHeight(double rightSashHeight) {
        this.rightSashHeight = rightSashHeight;
    }
}
```

Клас TwoSashWindow містить свої властивості - ширину і висоту лівої і правої стулок вікна разом з булевими змінними-ознаками їх відкриття-закриття, методи отримання-установки властивостей (геттери-сеттери) і свою реалізацію абстрактних методів базового класу. Зверніть увагу на використання конструктора абстрактного класу в конструкторах класів-спадкоємців за допомогою ключового слова `super`;

Тепер, використовуючи середу *BlueJ*, створіть об'єкти одностулкового і двостулкового вікна з довільними розмірами, як показано на Рис. 17.

Викликаючи довільно методи `open()` і `close()` з об'єктів і контролюючи значення булевої ознаки відкриття-закриття командою *Проверить*, обраною з

контекстного меню об'єктів, переконайтеся, що реалізації абстрактних методів працюють правильно.

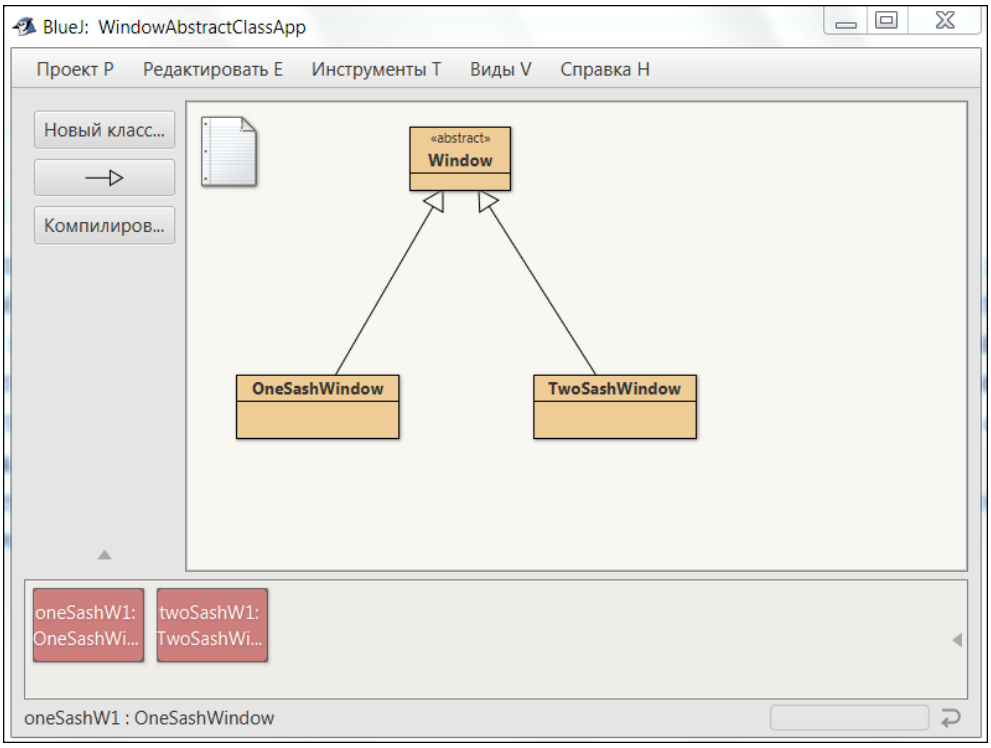

Рис. 17. Вікно проекту зі створеними об'єктами

Слід зазначити, що при додаванні завдання використання в системі вікна іншої конструкції (наприклад, розсувного), вона може бути легко вирішена розробкою класу такого вікна - спадкоємця абстрактного класу Window зі своєю реалізацією абстрактних методів.

З метою практичного вивчення інтерфейсів вирішимо завдання розробки класів для геометричних фігур кола і прямокутника. Будемо вважати вимогою підтримку методу малювання геометричних фігур різними кольорами, а також методу розрахунку їхньої площі.

Створіть в середовищі *BlueJ* новий проект ShapeApp. За допомогою кнопки *Новый класс* ... створіть інтерфейс Drawable так, як показано на *Рис. 18*.

Додайте наступний код в створений інтерфейс:

```
/**
 * Інтерфейс об'єктів, які можуть бути намальовані.
 */
public interface Drawable {
    void draw ();
}
```

Взагалі то, можна було б в інтерфейс додати і абстрактний метод обчислення площі `double calcArea()`, але це призведе до змішання в інтерфейсі різних функцій, що не рекомендується одним з принципів об'єктно-орієнтованого дизайну (SOLID) - принципом сегрегації інтерфейсів (ISP). Тому ми створимо абстрактний клас геометричної фігури Shape і помістимо цей метод до нього:

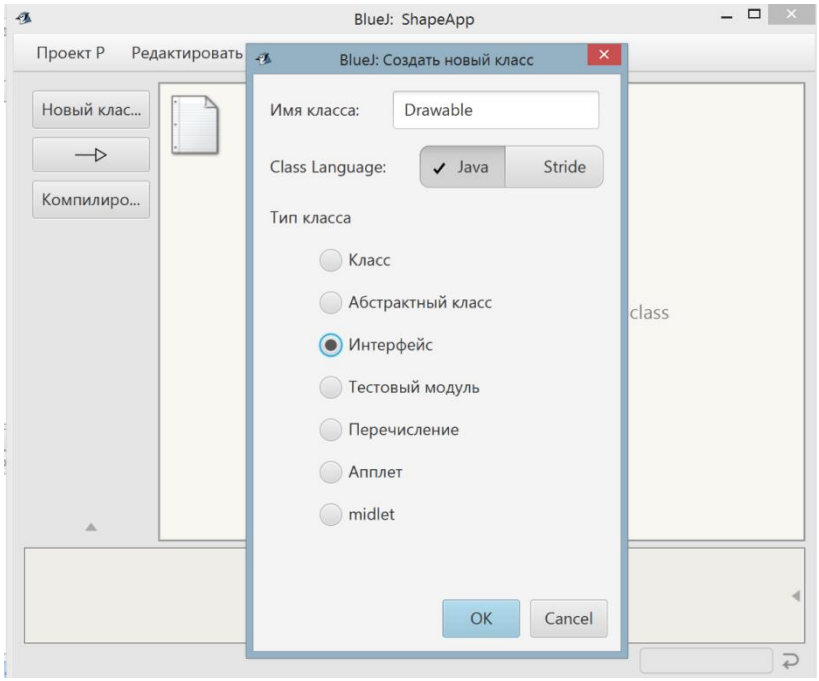

Рис. 18. Створення інтерфейсу в BlueJ

```
public abstract class Shape implements Drawable
{
    private String shapeColor;

    public Shape(String shapeColor) {
        this.shapeColor = shapeColor;
    }

    public abstract double calcArea();


    @Override
    public void draw() {
        System.out.println(toString());
    }

    public String getShapeColor() {
        return shapeColor;
    }
}
```

У цей клас також додано поле кольору фігури `shapeColor` разом з методом його отримання (геттером) і реалізація абстрактного методу інтерфейса.

Додайте до проекту клас Circle (коло) з наступним кодом:

```
public class Circle extends Shape {

    private double radius;

    public Circle(double radius, String shapeColor) {
        super(shapeColor);
```

```
        this.radius = radius;
    }

    @Override
    public double calcArea() {
        return Math.PI * radius * radius;
    }

    @Override
    public String toString() {
        return "This is Circle, color:" + getShapeColor ()
             + ", Radius=" + radius;
    }
}
```

Окрім своєї версії методу розрахунку площі, він містить перевизначений метод `toString()` класу java.lang.Object. Додайте до проекту клас Rectangle (прямокутник) з аналогічним кодом:

```
public class Rectangle extends Shape
{
    private double width;
    private double height;

    public Rectangle(double width, double height,
                     String shapeColor) {
        super(shapeColor);
        this.width = width;
        this.height = height;
    }

    @Override
    public double calcArea() {
        return width * height;
    }

    @Override
    public String toString() {
        return "This is Rectangle, color:" + getShapeColor ()
             + ", Width=" + width + ", height=" + height;
    }
}
```

Тепер, створіть об'єкти класів Circle і Rectangle і викличте з них по черзі методи, реалізовані в цих класах, і методи, успадковані від абстрактного класу Shape (Рис. 19).

Уявімо, що згодом знадобилося додати до проекту клас Triangle (трикутник) і забезпечити можливість порівняння і сортування фігур за значеннями їх площ. Перше завдання вирішується легко - додається ще один спадкоємець абстрактного класу Shape, в якому перевизначаються методи

`calcArea()` і `toString()`. Зауважте, що метод інтерфейсу `draw()` перевизначати не потрібно, тому що він використовує метод `toString()` об'єкта.

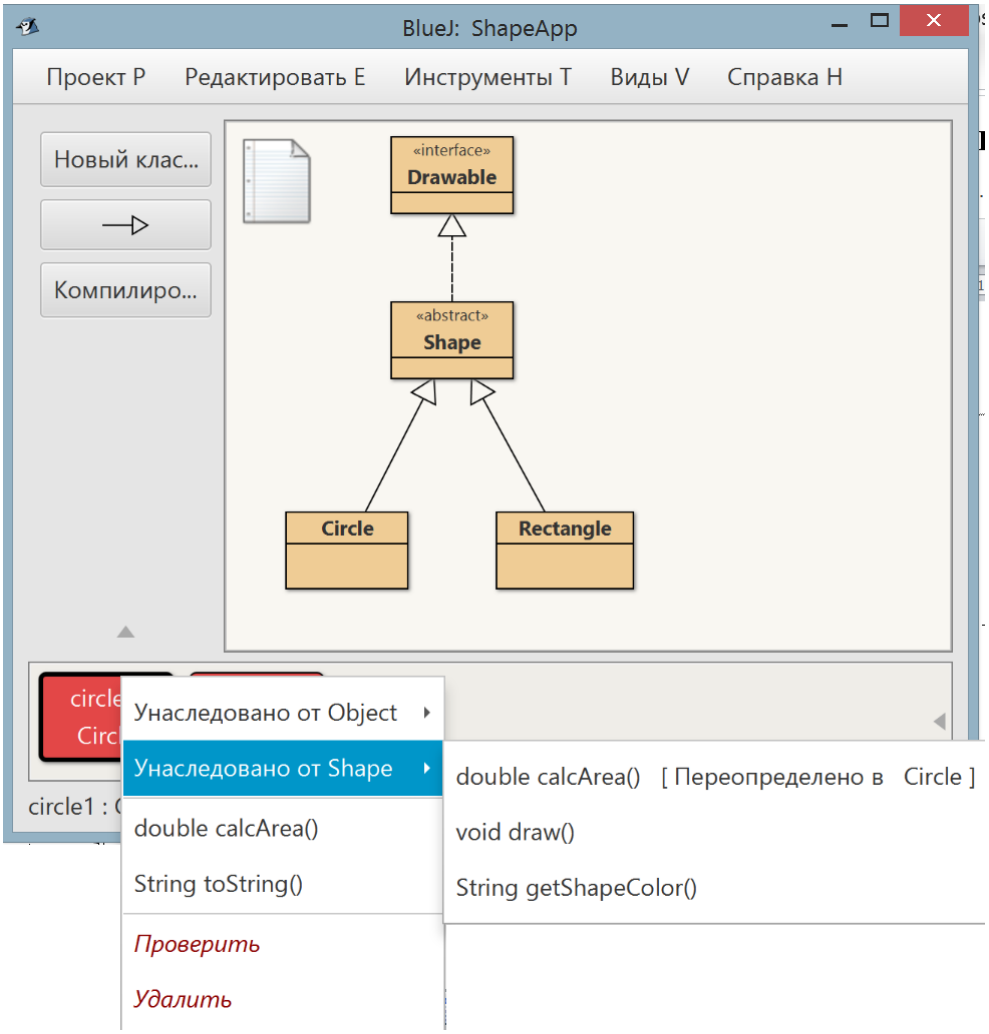

Рис. 19. Інтерфейс і класи проекту, методи, доступні з об'єктів

```
public class Triangle extends Shape {
    private double a;
    private double b;
    private double c;

    public Triangle(double a, double b, double c,
                    String shapeColor) {
        super(shapeColor);
        this.a = a;
        this.b = b;
        this.c = c;
    }

    @Override
    public double calcArea() {
        double s = (a + b + c) / 2;
        return Math.sqrt(s * (s - a) * (s - b) * (s - c));
    }
```

```
    @Override
    public String toString() {
        return "This is Triangle, color:" + getShapeColor()
             + ", A=" + a + ", b=" + b + ", c=" + c;
    }
}
```

Що стосується забезпечення порівняння, то на перший погляд потрібно використовувати інтерфейс java.lang.Comparable в кожному з дочірніх класів та реалізувати абстрактний метод `int compareTo(Object o)` для значень, одержуваних методом `calcArea()`, наприклад, для кола:

```
public class Circle extends Shape implements Comparable {
...
@Override
    public int compareTo(Object o) {
        Circle circle = (Circle) o;
        if (this.calcArea() > circle.calcArea()) {
            return 1;
        }
        if (this.calcArea() < circle.calcArea()) {
            return -1;
        }
        return 0;
    }
}
```

Однак раціональніше перенести реалізацію методу `int compareTo(Object o)` в абстрактний клас Shape, при цьому його потрібно буде визначати один раз, а не в кожному із спадкоємців:

```
public abstract class Shape implements Drawable, Comparable {
...
    @Override
    public int compareTo(Object o) {
        Shape shape = (Shape) o;
        return (int)(this.calcArea() - shape.calcArea());
    }
}
```

Для перевірки можливості сортування додайте до проекту клас Main з наступним кодом:

```
public class Main {
    public static void main(String[] args) {
        Shape[] shapes = {new Circle(10, "GREEN"),
                          new Rectangle(11, 22, "RED"),
                          new Triangle(5, 5, 5, "BLACK")};
        Arrays.sort(shapes);
        for Shape shape: shapes) {
            System.out.println(shape + "-" + Shape.calcArea());
        }     } }
```

Запустіть метод `main` і переконайтеся в правильності сортування (Рис. 20).

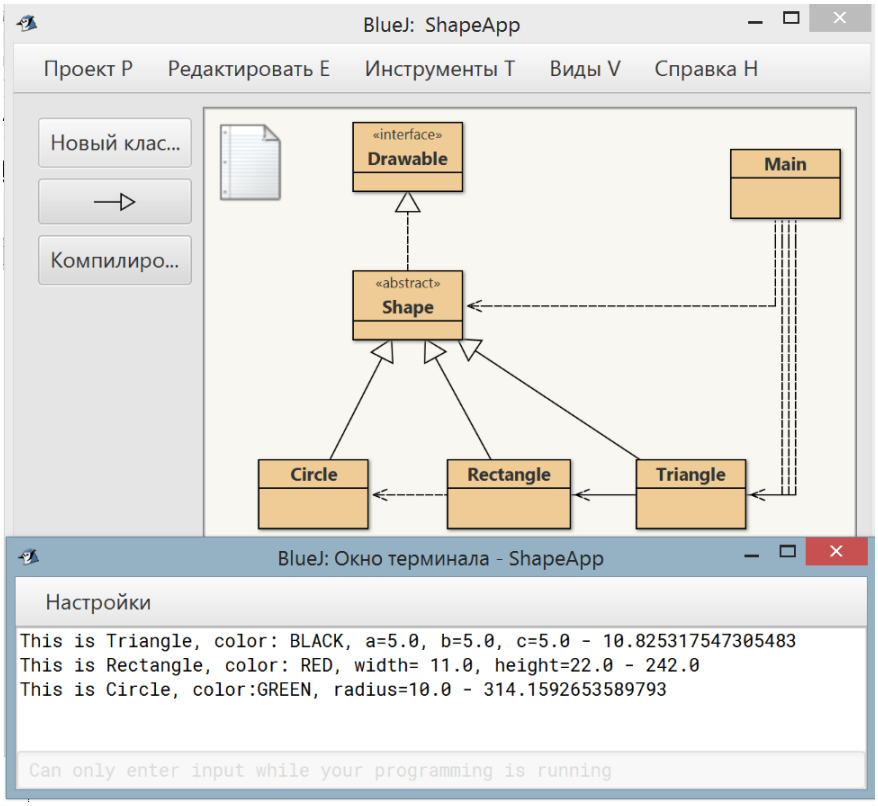

Рис. 20. Перевірка сортування фігур за значеннями площі

2. Завдання

1. Розробіть в *BlueJ* клас на мові Java відповідно до завдання в наведеній нижче таблиці (Номер варіанта обирайте за формулою V=(№ mod 16)+1, де № – Ваш порядковий номер в журналі академгрупи. Програма повинна містити конструктор без параметрів, конструктор з параметрами і необхідну кількість методів класу. Передбачте створення перевантажених методів. Додайте документовані коментарі до класу, його властивостей і методів по аналогії з розглянутим вище прикладом. Наведіть вихідний код класу у звіт

V	Завдання	V	Завдання
1	Клас, який представляє дійсне число, з методами розрахунку суми, різниці, добутку і частки двох чисел	9	Клас, який представляє множину цілих чисел, з методами визначення множин, що представляють перетин і об'єднання двох таких множин.
2	Клас, який представляє вектор на площині, заданий координатами початку і кінця вектора, з методами знаходження модуля вектора і геометричного складання двох векторів по правилу трикутника	10	Клас, який представляє банківський рахунок, з методами переказу коштів з іншого рахунку на даний рахунок і переказу коштів з цього рахунку на інший рахунок (при достатній кількості коштів рахунку-джерела) і виведенням стану рахунку

V	Завдання	V	Завдання
3	Клас, який представляє табурет, з методом, що виводить на екран його розміри і матеріал, з якого він виготовлений, а також з методом, що порівнює вагу двох табуретів з виведенням на екран інформації, який з них важче	11	Клас, який представляє записку, з методами виведення заголовку записки, тексту записки і кількості символів в ній
4	Клас, який представляє квадрат з методами розрахунку його периметра і площі	12	Клас, який представляє книгу, з методами виведення на екран автора книги, назви книги і кількості її сторінок
5	Клас, який представляє коло, з методами розрахунку його периметра і площі	13	Клас, який представляє лічильник електричної енергії, з методом розрахунку вартості енергії, спожитої за місяць з урахуванням диференційованого тарифу для розрахунків з населенням
6	Клас, який представляє персону, з методом виведення його прізвища і методом розрахунку кількості прожитих повних років, місяців і днів	14	Клас, який представляє прямокутник з методами розрахунку його периметра і площі
7	Клас, який представляє довільний трикутник, заданий координатами його вершин на площині, з методами розрахунку його периметра і площі	15	Клас, який представляє співробітника фірми, з методами виведення на екран його прізвища і посади
8	Клас, який представляє ціле число, з методами додавання, віднімання, множення і ділення двох цілих чисел	16	Клас, який представляє академгрупу з методами виведення на екран назви групи і кількості студентів в ній

2. Створіть два об'єкти виробленого Вами класу: один з використанням конструктора без параметрів, а інший з використанням параметризованого конструктора (значення параметрів вибирайте довільно).
3. Виконайте методи об'єктів, задавши довільні значення аргументів. Наведіть скріншот з результатами, що повертаються, у звіт.
4. Виконайте генерування документації класу і приведіть її в звіт.
5. Розробіть в BlueJ батьківський клас, один або більше класів-спадкоємців і головний клас відповідно до завдання в наведеній нижче таблиці (Номер варіанта вибирайте за формулою V=(№ mod 16)+1, де № – Ваш порядковий номер в журналі академгрупах). Батьківський клас і один або більше класів-спадкоємців повинні містити конструктор без параметрів, конструктор з параметрами і необхідну кількість методів класу. Головний клас повинен містити головний метод `main`, у якому реалізована функціональність з використанням динамічного поліморфізму. Додайте документовані коментарі до класу, його властивостей і методів. Наведіть вихідний код всіх класів у звіт.

V	Завдання	V	Завдання
1	Батьківський клас, який представляє дійсне число, з методами розрахунку суми, різниці, добутку і частки двох чисел. Дочірній клас, що представляє комплексне число, з методами додавання, віднімання, множення і ділення двох комплексних чисел. Головний клас з методом `main`, що виводить на екран значення і результати підсумовування і віднімання сусідніх елементів масиву з дійсними і комплексними числами.	9	Батьківський клас, що представляє множину цілих чисел, з методами визначення множин, що представляють перетин і об'єднання двох таких множин. Дочірній клас, що представляє собою множину комплексних чисел з цілими значеннями дійсної і уявної частини, з методами визначення множин, що представляють перетин і об'єднання двох таких множин (вважати рівними комплексні числа з рівними модулями). Головний клас з методом `main`, що виводить перетин і об'єднання елементів масиву, що представляють множини цілих або комплексних чисел
2	Батьківський клас, що представляє собою вектор на площині, заданий координатами початку і кінця вектора, з методами знаходження модуля вектора і геометрричного складання двох векторів по правилу трикутника. Дочірній клас, що представляє вектор в просторі з методами знаходження модуля вектора і геометрричного складання двох векторів по правилу трикутникаграма. Головний клас з методом main, що виводить на екран значення модулів двовимірних і тривимірних векторів, які є елементами одного масиву, і результатів додавань сусідніх елементів такого масиву	10	Батьківський клас, який представляє банківський рахунок, з методами переказу коштів з іншого рахунку на даний рахунок і переказу коштів з цього рахунку на інший рахунок (при достатній кількості коштів рахунку-джерела) і виведенням стану рахунку. Дочірній клас, що представляє депозитний рахунок з можливістю довнесення на нього коштів з іншого рахунку і переведення коштів з депозитного рахунку тільки після закінчення терміну депозиту (при достатній кількості коштів рахунку-джерела) і виведенням стану рахунку. Головний клас з методом `main`, який виконує перерахування деякої фіксованої суми грошей з попереднього на наступний рахунок-елемент масиву звичайних і депозитних рахунків і забезпечує виведення стану рахунків

V	Завдання	V	Завдання
3	Батьківський клас, що представляє табурет, з методом, що виводить на екран його розміри і матеріал, з якого він виготовлений, а також з методом, що порівнює вагу двох табуретів з виведенням на екран інформації, який з них важче. Дочірній клас, що представляє стілець, з методом порівняння висоти спинки сидіння двох стільців і методом порівняння ваги двох стільців. Головний клас з методом `main`, який одним оператором виводить на екран інформацію про порівняння ваги сусідніх елементів масиву з табуретами і стільцями.	11	Батьківський клас, що представляє представляє записку, з методами виведення заголовка записки і кількості символів в ній. Дочірній клас, що представляє лист з текстом записки і заголовком записки як темою з методами виведення теми листа і імен одержувача і відправника. Головний клас з методом `main`, що формує масив листів однакового змісту для 3 адресатів так, щоб не повторювалася пара "відправник-одержувач" і виводить пари адресатів на екран
4	Батьківський клас, що представляє квадрат, з методами розрахунку його периметра і площі. Дочірній клас, що представляє правильний багатокутник з методами розрахунку його периметра і площі. Головний клас з методом `main`, який одним оператором виводить на екран площі квадрата і довільного правильного багатокутника для елементів масиву з об'єктами обох класів	12	Батьківський клас, що представляє книгу, з методами виведення на екран автора книги, назви книги і кількості її сторінок. Дочірній клас, що представляє електронну книгу з методами виведення формату і розміру файлу. Головний клас з методом `main`, який одним оператором виводить на екран всі доступні властивості об'єктів обох класів, що представляють елементи загального масиву
5	Батьківський клас, що представляє коло, з методами розрахунку його периметра і площі. Дочірній клас, що представляє еліпс, з аналогічними методами. Головний клас з методом `main`, який одним оператором виводить на екран периметр і площу круга і еліпса для елементів масиву з об'єктами обох класів	13	Батьківський клас, що представляє лічильник електричної енергії, з методом розрахунку вартості енергії, спожитої за добу. Дочірній клас, що представляє двохтарифний лічильник електричної енергії, з методом розрахунку вартості енергії, спожитої за добу (денні 2/3 доби враховуються з коефіцієнтом 1, а нічна 1/3 доби – з коефіцієнтом 0,5). Головний клас з методом `main`, що обчислює різницю в оплаті при вимірюванні електроенергії лічильниками різних типів за 30 днів при випадковому значенні спожитої добової енергії від 7 до 10 кВт·год. Врахувати поточний диференційований тариф для розрахунків з населенням

V	Завдання	V	Завдання
6	Батьківський клас, що представляє персону, з методом виведення його прізвища та ініціалів і методом розрахунку кількості прожитих повних років, місяців і днів. Дочірній клас, який представляє співробітника фірми, з методом виведення його посади і часу роботи на ній. Головний клас з методом `main`, який одним оператором виводить на екран прізвище, ініціали та або кількість прожитих років, місяців і днів, або час роботи на посаді для елементів масиву з об'єктами обох класів	14	Головний клас, що представляє прямокутник з методами розрахунку його периметра і площі. Дочірній клас, що представляє прямокутник, що дозволяє задавати колір лінії і колір заповнення. Головний клас з методом `main`, який одним оператором виводить на екран площа, периметр прямокутників додатково колір лінії і заповнення для елементів масиву з об'єктами обох класів
7	Батьківський клас, що представляє довільний трикутник, заданий координатами його вершин на площині, з методами розрахунку його периметра і площі. Дочірній клас, що представляє довільний багатокутник, заданий координатами його вершин на площині, з методами розрахунку його периметра і площі. Головний клас з методом `main`, який одним оператором виводить на екран периметр і площу для елементів масиву з об'єктами обох класів	15	Батьківський клас, який представляє співробітника фірми, з методами виведення на екран його прізвища і посади. Дочірній клас, що представляє менеджера компанії з методом виведення на екран підлеглих йому співробітників. Головний клас з методом `main`, який одним оператором виводить на екран прізвище, посаду для всіх співробітників і додатково список підлеглих співробітників для менеджерів, які є елементами загального масиву
8	Батьківський клас, що представляє ціле число, з методами додавання, віднімання, множення і ділення двох цілих чисел. Дочірній клас, що представляє простий дріб, з методами додавання, віднімання, множення і ділення двох дробів. Головний клас з методом `main`, який одним оператором виводить на екран число або дріб, а також результат складання сусідніх елементів масиву з об'єктами обох класів	16	Батьківський клас, що представляє академгрупу з методами виведення на екран назви групи і числа студентів в ній. Дочірній клас, який представляє потік (об'єднання) декількох академгруп з методами, які виводять назву потоку і загальне число студентів в потоці. Головний клас з методом `main`, який одним оператором виводить назву і кількість студентів в академгрупі або потоці, які є елементами загального масиву

6. Створіть об'єкти розроблених Вами класів (значення параметрів вибирайте довільними).
7. Виконайте методи об'єктів, задавши довільні значення аргументів. Наведіть скріншот з результатами, що повертаються методами, у звіт.
8. Виконайте головний метод, який реалізує функціональність з використанням динамічного поліморфізму через пізнє зв’язування перевизначених методів. Наведіть скріншот з виведенням результатів роботи головного методу.
9. Виконайте генерування документації класів, приведіть її у звіт.

10. Розробіть в BlueJ клас на мові Java відповідно до завдання в наведеній нижче таблиці (номер варіанта вибирайте за формулою V=(№ mod 16)+1, де № - Ваш порядковий номер в журналі академгрупи. Створіть ієрархію з абстрактного батьківського класу і класів-спадкоємців. Наведіть вихідний код всіх класів до звіту.

V	Завдання	V	Завдання
1	Абстрактний батьківський клас, що представляє арифметичну операцію над двома цілими операндами з абстрактним методом, який повертає значення результату операції, і неабстрактним методом, що повертає назву операції та значення кожного з операндів. Дочірні класи, що представляють суму і різницю двох операндів. Головний клас з методом `main`, який одним оператором виводить на екран назву операції, операнди, і результат додання або віднімання для елементів масиву з об'єктами обох класів	9	Абстрактний батьківський клас, що описує тварину з абстрактними методами, які описують чим харчується тварина і повертають його розміри і вагу, а також з неабстрактними методами отримання назви тварини і назви загону, до якого вона відноситься (гризуни, хижі, парнокопитні і т.д.). Дочірні класи, що представляють собою слона і синього кита з методами отримання опису їжі тварини і її розмірів. Головний клас з методом `main`, який одним оператором виводить на екран назви тварини і її загону, а також описує, чим харчується тварина, і її розміри для елементів масиву з об'єктами обох класів
2	Абстрактний батьківський клас, що описує геометричне тіло з абстрактними методами обчислення об'єму та площі поверхні і неабстрактним методом отримання назви тіла. Дочірні класи, що представляють паралелепіпед і кулю з реалізаціями абстрактних методів. Головний клас з методом `main`, який одним оператором виводить на екран назви, значення площ поверхонь і об'єми множини геометричних тіл	10	Абстрактний батьківський клас, що представляє депозитний рахунок з абстрактним методом обчислення доходу від депозиту і неабстрактними методами задання суми депозиту, терміну депозиту і річних відсотків, Дочірні класи, що представляють депозит з виплатою відсотків у кінці терміну і депозит із щомісячною капіталізацією відсотків з методом обчислення суми на рахунку і доходу по закінченні терміну депозиту. Головний клас з методом `main`, який одним оператором виводить на екран характеристики депозитів різних типів і їх прибутковість в момент закінчення терміну депозиту для множини депозитів різних типів

V	Завдання	V	Завдання
3	Абстрактний батьківський клас, що представляє предмет меблів, з абстрактним методом, який повертає опис збірки меблів, і неабстрактним методом, що отримує назву меблів. Дочірні класи, що представляють стілець і шафу з реалізаціями абстрактних методів. Головний клас з методом `main`, який одним оператором виводить назву і опис збірки для елементів масиву з об'єктами обох класів	11	Абстрактний батьківський клас, що описує літак з абстрактним методом обчислення доходу від перевезення. Дочірні класи, що представляють пасажирський літак з кількістю пасажирів і вартістю квитка та транспортний літак з тоннажем вантажу і тарифом перевезення тонни вантажу і методом розрахунку доходу від рейсу. Головний клас з методом `main`, що підраховує сумарний дохід від рейсів декількох транспортних і пасажирських літаків
4	Абстрактний батьківський клас, що представляє геометричну фігуру з абстрактними методами обчислення її периметра і площі і неабстрактним методом отримання назви фігури. Дочірні класи, що представляють коло і трикутник з додатковими методами завдання і отримання радіусу кола і сторін трикутника, відповідно. Головний клас з методом `main`, який одним оператором виводить на екран довжини радіуса або сторін, периметр і площу для елементів масиву геометричних фігур	12	Абстрактний батьківський клас, що представляє валюту, з абстрактним методом переведення суми валюти в гривні і неабстрактним методом виведення назви валюти. Дочірні класи, що представляють долар, євро та англійський фунт з реалізаціями абстрактних методів. Головний клас з методом `main`, що виводить назви валют і вартість їх 100 одиниць в гривнях для набору валют
5	Абстрактний батьківський клас, що описує зброю з абстрактним методом отримання опису дії зброї і неабстрактним методом отримання назви зброї. Дочірні класи, які описують рушницю і кинджал з методами, що описують їх дію і повертають вагу зброї. Головний клас з методом `main`, який одним оператором виводить назву і дію зброї для елементів масиву різних видів зброї	13	Абстрактний батьківський клас, який представляє питання тесту з неабстрактними методами завдання і отримання питання тесту і абстрактним методом визначення правильності відповіді. Дочірні класи, що представляють питання тесту з декількома варіантами відповіді з методом введення варіантів і ознаки їх правильності і питання тесту на встановлення відповідності між двома частинами відповіді з методом введення частин відповіді і ознакою їх відповідності. Головний клас з методом `main`, який одним оператором виводить на екран питання і правильні відповіді для елементів масиву питань тесту

V	Завдання	V	Завдання
6	Абстрактний батьківський клас, що представляє рівняння з абстрактним методом обчислення його коренів і неабстрактним методом отримання назви типу рівняння. Дочірні класи, що представляють лінійне і квадратне рівняння з реалізаціями абстрактних методів. Головний клас з методом `main`, який одним оператором виводить на екран назви типів і значення коренів для множини рівнянь різних типів.	14	Абстрактний батьківський клас, що представляє друковане видання з абстрактним методом, що повертає періодичність випуску і неабстрактними методами отримання типу і назви видання. Дочірні класи, що представляють журнал і газету, з реалізаціями абстрактних методів. Головний клас з методом `main`, який одним оператором виводить тип, назву видання та періодичність випуску для елементів масиву з об'єктами обох класів
7	Абстрактний батьківський клас, що описує прогресію чисел з абстрактними методами обчислень і-го елемента прогресії і суми прогресії і неабстрактним методом, що повертає тип прогресії. Дочірні класи, що представляють арифметичну і геометричну прогресію з реалізацією абстрактних методів. Головний клас з методом `main`, який одним оператором виводить на екран характеристики прогресій для масиву прогресій різних типів.	15	Абстрактний батьківський клас, що представляє транспортний засіб, з абстрактним методом отримання способу переміщення в просторі і неабстрактними методами отримання його типу і швидкості. Дочірні класи, що представляють автомобіль, з методами отримання його моделі і виробника і теплохід з методами отримання його назви і водотоннажності. Головний клас з методом `main`, який одним оператором виводить на екран типи і характеристики транспортних засобів обох типів, які є елементами загального масиву
8	Абстрактний батьківський клас, що описує контейнер для зберігання множини об'єктів, з абстрактними методами сортування і поелементної обробки об'єктів і неабстрактним методом отримання значень елементів. Дочірні класи, що представляють контейнер цілих чисел з бульбашковим сортуванням і поелементною обробкою з поверненням суми елементів, і контейнер дійсних чисел з методом сортування вставками і поелементною обробкою з поверненням суми елементів. Головний клас з методом `main`, який одним оператором виводить на екран вихідний порядок елементів в контейнері і результати сортування і поелементної обробки для масиву контейнерів.	16	Абстрактний батьківський клас, що представляє персону, з абстрактним методом отримання інформації про персону і неабстрактними методами отримання ПІБ і віку персони. Дочірні класи, що представляють співробітника фірми, з методами отримання назви фірми і посади співробітника і студента університету, з методами отримання назви його групи і номера курсу, на якому він навчається. Головний клас з методом `main`, який одним оператором виводить на екран значення властивостей об'єктів-спадкоємців

11. Додайте (за можливості) ще одну реалізацію абстрактного класу. Виконайте рефакторинг коду головного класу, врахувавши нову реалізацію. Наведіть код класу-реалізації до звіту.

12. Розробіть в *BlueJ* інтерфейс на мові Java відповідно до завдання в наведеній нижче таблиці (номер варіанта вибирайте за формулою V=(№ mod 16)+1, де № - Ваш порядковий номер в журналі академгрупи. Розробіть класи-реалізації інтерфейсу. Наведіть вихідний код інтерфейсу і всіх класів у звіт.

V	Завдання	V	Завдання
1	Інтерфейс Awardable з методом `aword(boolean condition)`, що визначає виконання умов для заохочення персони. Класи-реалізації, що описують студента та співробітника	9	Інтерфейс Mammalable з методом `milkFeed()`, що визначає вік та особливості харчування ссавців. Класи-реалізації, що описують собаку та дельфіна
2	Інтерфейс Volumeable з методом `getVolume()`, що визначає обсяг об'єкта. Класи-реалізації, що описують паралелепіпед і кулю	10	Інтерфейс Incomeable з методом `getIncome()`, що визначає дохід від операцій. Класи-реалізації, що описують дохід від погодинної роботи та дохід від внеску до банку
3	Інтерфейс Colorable з методом `paint(String color)`, що визначає зміну кольору об'єкта. Класи-реалізації, що описують коло та лавку	11	Інтерфейс Flightable з методом `flight()`, що визначає здатність об'єкта до польоту. Класи-реалізації, що описують літак і голуба
4	Інтерфейс Scalable з методом `scale()`, що визначає масштабування правильних геометричних фігур. Класи-реалізації, що описують коло та квадрат	12	Інтерфейс Countable з методом `count()`, що визначає можливість кількісного обліку властивості об'єкта. Класи-реалізації, що описують навчання студента та довільний період часу
5	Інтерфейс Lightable з методом `light()`, що визначає можливість об'єкта випромінювати світло. Класи-реалізації, що описують ліхтарик і Місяць	13	Інтерфейс Searchable з методом `search()`, що визначає можливість пошуку в об'єкті. Класи-реалізації, що описують телефонний довідник і блог
6	Інтерфейс Weightable з методом `getWeight()`, що визначає можливість отримання ваги об'єкта. Класи-реалізації, що описують сталеву кулю та дерев'яний брусок	14	Інтерфейс Printable з методом `print()`, що визначає можливість друку властивостей об'єкта. Класи-реалізації, що описують журнал і газету
7	Інтерфейс Switchable з методом `switch()`, що визначає перемикання об'єкта. Класи-реалізації, що описують водопровідний кран та електричний вимикач	15	Інтерфейс Vehicle з методом `move()`, що описує переміщення транспортного засобу. Класи-реалізації, що описують автомобіль та катер
8	Інтерфейси Readable з методом `read()` та Writeable з методом `write()`, що визначають можливість читання і запису даних об'єкта, відповідно. Класи-реалізації, що описують диск CD-ROM і флеш-накопичувач	16	Інтерфейс Learnable з методом `learn()`, що визначає здатність до навчання об'єкта. Класи-реалізації, що описують учня школи і слухача курсів

13. Для одного з класів-реалізацій попереднього завдання реалізуйте метод інтерфейсу Comparable для порівняння за значенням властивості, що визначає

реалізацію абстрактного методу. Продемонструйте роботу реалізації на прикладі сортування декількох об'єктів.

14. Для класу-реалізації попереднього завдання розробіть компаратор за іншою, аніж у попередньому завданні, властивістю. Продемонструйте роботу реалізації на прикладі сортування декількох об'єктів.
15. Додайте до розробленого інтерфейсу довільний метод за замовчуванням та продемонструйте використання його та його перевизначених версій.
16. Задокументуйте вихідний код розроблених класів і інтерфейсів і приведіть його, а також скріншоти з результатами до звіту.

3. Контрольні питання

1. Назвіть принципи об'єктно-орієнтованого підходу до програмування і поясніть їх.
2. Опишіть поняття посилання на об'єкт і типа і наведіть приклади посилань і типів Java.
3. Наведіть приклад використання оператора `new` для створення об'єкта довільного класу.
4. Дайте визначення часу життя змінних і об'єктів Java.
5. Наведіть приклад виклику довільного метода з об'єкта довільного класу з передачею йому параметрів.
6. Поясніть механізм передачі параметрів методу за значенням для аргументів примітивного типу та аргументів-об'єктів, використовуваний Java.
7. Назвіть призначення і особливості конструкторів, чому в Java відсутні деструктори?
8. Для чого використовується посилання `this`?
9. Дайте визначення технологій успадкування та поліморфізму і опишіть переваги їх використання.
10. Чи можливе існування класів, які не мають батьківського класу?
11. Яким чином в конструкторі дочірнього класу можна викликати конструктор батьківського класу?
12. Чи може дочірній клас мати кілька батьківських класів?
13. Перерахуйте модифікатори доступу, що використовуються при визначенні полів і методів класу і опишіть їх застосування.
14. Наведіть приклади дерев спадкування для довільних об'єктів реального світу.
15. Наведіть приклади перетворення типів, що підвищує і що понижує, при успадкуванні класів.
16. Дайте визначення технології перевантаження методів і наведіть приклади її використання.
17. Дайте визначення технології заміщення методів і наведіть приклади її використання.
18. Як Ви розумієте, що таке динамічний поліморфізм методів (або пізнє зв'язування)? У чому полягає перевага використання цієї технології?

19. Що називають сигнатурою і контрактом методу? В якому з випадків поліморфізму методів враховується сигнатура, а в якому контракт методу?
20. Назвіть значення за замовченням для примітивних і довідкових типів, якими ініціалізуються змінні об'єкта.
21. Як визначається абстрактність предметів реального світу? Наведіть приклади абстрактних предметів і їх конкретних реалізацій.
22. Яким чином в Java визначаються абстрактні класи? Що є необхідною умовою для оголошення класу як абстрактного?
23. Чи можна створювати об'єкти з типами абстрактних класів, використовуючи конструктори абстрактних класів?
24. Назвіть переваги використання абстрактних методів абстрактних класів.
25. Чи можна закрити модифікатором `private` перевизначений `protected`-метод батьківського класу?
26. Назвіть призначення інтерфейсів. Перерахуйте члени класу, які може містити класичний інтерфейс Java (до JDK 8).
27. Назвіть модифікатори методів і констант інтерфейсів, що додаються за замовчуванням.
28. Наведіть приклад оголошення класу, що реалізовує інтерфейс. Чи може клас реалізовувати кілька інтерфейсів?
29. Поясніть, з чим пов'язано ослаблення залежності між класами при використанні інтерфейсів у якості типів членів класів, параметрів методів і значень, що повертаються методами.
30. Назвіть причини, за якими в JDK 8 з'явилися методи за-замовчуванням і статичні методи інтерфейсів. Дайте їм характеристику і назвіть відмінності одних від інших.
31. Яким чином вирішується проблема множинного спадкоємства при реалізації методів за-замовчуванням інтерфейсів Java 8?
32. Назвіть засоби, що дозволяють порівнювати об'єкти за довільною ознакою. Чим відрізняються інтерфейси Comparable і Comparator?
33. Наведіть приклад реалізації методу int compareTo (T o) інтерфейсу java.lang.Comparable для впорядкування персон за віком.
34. Наведіть приклад реалізації методу `int compare(T o1, To2)` інтерфейсу java.util.Comparator для впорядкування персон за віком і за прізвищем.

4. Література

1. Васильєв О. Програмування мовою Java. Тернопіль: Навчальна книга - Богдан, 2020. 696 с.
2. Horstmann C. S. Core Java, Volume I: Fundamentals. 12-th Ed. "Addison-Wesley", 2022. 1197 p.
3. Barnes D. J. and Kölling M. Objects First with Java™. A Practical Introduction Using BlueJ, Fifth Edition. – Pearson Education, Inc publishing as Prentice Hall, 2012.-578 p.