Міністерство освіти і науки України

Національний університет “Львівська політехніка”

Кафедра ЕОМ

Звіт

з лабораторної роботи №4

з дисципліни: «Кросплатформенні засоби програмування»

на тему: «Спадкування та інтерфейси»

Виконав: ст.гр. КІ-34

Степанов А. О.

Прийняв:

викл. каф. ЕОМ

Іванов Ю. С.

Львів 2022

Мета роботи: ознайомитися з спадкуванням та інтерфейсами у мові Java.

Завдання:

1. Написати та налагодити програму на мові Java, що розширює клас, що реалізований у лабораторній роботі №3, для реалізації предметної області заданої варіантом. Суперклас, що реалізований у лабораторній роботі №3, зробити абстрактним. Розроблений підклас має забезпечувати механізми свого коректного функціонування та реалізовувати мінімум один інтерфейс. Програма має розміщуватися в пакеті Група.Прізвище.Lab4 та володіти коментарями, які дозволять автоматично згенерувати документацію до розробленого пакету.
2. Автоматично згенерувати документацію до розробленої програми.
3. Скласти звіт про виконану роботу з приведенням тексту програми, результату її виконання та фрагменту згенерованої документації.
4. Дати відповідь на контрольні запитання.

Варіант 21

21. Водяний Пістолет

Лістинг програми:

Файл GunApp.java

```
} /**
 * lab 4 package
 */
package KI34.Stepanov.Lab4;
import java.io.*;
/**
 * Gun Application class implements main method for Gun class possibilities
 demonstration
 * @author Andriy Stepanov
 * @version 1.0
 */
public class GunApp {
    /**
     * @param args function parameter
     * @throws FileNotFoundException throw about non-existent file
     */
    public static void main(String[] args) throws FileNotFoundException {
        WaterGun gun = new WaterGun("Watergun", 15, 5, 25, 75, "green", 50,
100);
        gun.takeGun();
        gun.printInfo();
        gun.aim();
        gun.shoot();
        gun.shoot();
        gun.aim();
        gun.shoot();
        gun.reload();
        gun.aim();
        gun.aim();
        gun.shoot();
        gun.aim();
        gun.shoot();
        gun.increaseMaxAmmo(40);
        gun.printInfo();
```

```
        gun.decreaseMaxAmmo(15);
        gun.repair();
        gun.increaseMaxAmmo(30);
        gun.aim();
        gun.shoot();
        gun.putGun();
        gun.printInfo();
        gun.drawWaterGun("червоний", 75);
        gun.takeGun();
        gun.aim();
        gun.doubleShoot();
        gun.putGun();
        gun.printInfo();
        gun.deleteInfo();
        gun.printInfo();
        gun.dispose();
    }
}
```

Файл Gun.java

```
} /**
 * lab 4 package
 */
package KI34.Stepanov.Lab4;
import java.io.*;
/**
 * Class <code>Gun</code> implements gun
 * @author Andriy Stepanov
 * @version 1.0
 */
public abstract class Gun {
    protected String gunName;
    protected int damage;
    protected int ammo;
    protected int maxAmmo;
    protected boolean isAim;
    protected boolean isTaken;
    protected int exploitation;
    protected PrintWriter fout;

    /**
     * Constructor
     * @param gunName Gun's Name
     * @param damage Received damage
     * @param ammo The count of ammo
     * @param maxAmmo Maximum count of ammo
     * @param exploitation Exploitation
     * @throws FileNotFoundException throw about non-existent file
     */
    public Gun(String gunName, int damage, int ammo, int maxAmmo, int
            exploitation) throws FileNotFoundException {
        this.gunName = gunName;
        this.damage = damage;
        this.ammo = ammo;
        this.maxAmmo = maxAmmo;
        this.isAim = false;
        this.isTaken = false;
        this.exploitation = exploitation;
        fout = new PrintWriter(new File("Log.txt"));
    }
    /**
     * Method returns gun's name
     * @return Gun's name
     */
    public String getGunName() {
```

```
        return gunName;
    }
    /**
     * Method sets the new gun's name
     * @param gunName The name of the gun
     */
    public void setGunName(String gunName) {
        this.gunName = gunName;
    }
    /**
     * Method returns gun's damage
     * @return Gun's damage
     */
    public int getDamage() {
        return damage;
    }
    /**
     * Method sets the new gun's damage
     * @param damage The damage of the gun
     */
    public void setDamage(int damage) {
        this.damage = damage;
    }
    /**
     * Method returns the count of the ammo
     * @return the count of the ammo
     */
    public int getAmmo() {
        return ammo;
    }
    /**
     * Method sets the count of the ammo
     * @param ammo The count of the ammo
     */
    public void setAmmo(int ammo) {
        this.ammo = ammo;
    }
    /**
     * Method returns the count of the ammo
     * @return the count of the ammo
     */
    public int getMaxAmmo() {
        return maxAmmo;
    }
    /**
     * Method sets maximum count of ammo
     * @param maxAmmo Maximum count of ammo
     */
    public void setMaxAmmo(int maxAmmo) {
        this.maxAmmo = maxAmmo;
    }
    /**
     * Method returns aiming the gun
     * @return aiming the gun
     */
    public boolean isAim() {
        return isAim;
    }
    /**
     * Method sets aiming the gun
     * @param aim Aiming the gun
     */
    public void setAim(boolean aim) {
        isAim = aim;
    }
    /**
```

```java
     * Method returns exploitation
     * @return exploitation
     */
    public int getExploitation() {
        return exploitation;
    }
    /**
     * Method sets exploitation the gun
     * @param exploitation Exploitation the gun
     */
    public void setExploitation(int exploitation) {
        this.exploitation = exploitation;
    }
    /**
     * Method returns taking gun
     * @return taking gun
     */
    public boolean isTaken() {
        return isTaken;
    }
    /**
     * Method sets taking gun
     * @param taken Taking gun
     */
    public void setTaken(boolean taken) {
        isTaken = taken;
    }
    /**
     * Method simulates gun's shoot
     */
    public void shoot() {
        if (isTaken) {
            if (isAim) {
                if (ammo == 0) {
                    System.out.println("Немає патронів");
                    fout.print("Немає патронів\n");
                    isAim = false;
                } else {
                    System.out.println("Вистріл");
                    fout.print("Вистріл\n");
                    ammo -= 1;
                    exploitation -= 1;
                    isAim = false;
                }
            } else {
                System.out.println("Спершу потрібно прицілитися");
                fout.print("Спершу потрібно прицілитися\n");
            }
        } else {
            System.out.println("Візьміть пістолет");
            fout.print("Візьміть пістолет\n");
        }
    }
    /**
     * Method simulates gun's reload
     */
    public void reload() {
        ammo = maxAmmo;
        System.out.println("Перезарядка");
        fout.print("Перезарядка\n");
    }
    /**
     * Method simulates gun's aim
     */
    public void aim() {
        if (isAim) {
```

```
            System.out.println("Ви і так прицілені");
            fout.print("Ви і так прицілені\n");
        } else {
            System.out.println("Ви прицілилися");
            fout.print("Ви прицілилися\n");
            isAim = true;
        }
    }
    /**
     * Method simulates gun's repair
     */
    public void repair() {
        if (exploitation == 100) {
            System.out.println("Пістолет повністю робочий");
            fout.print("Пістолет повністю робочий\n");
        } else {
            exploitation = 100;
            System.out.println("Пістолет відремонтовано");
            fout.print("Пістолет відремонтовано\n");
        }
    }
    /**
     * Method simulates gun's taking
     */
    public void takeGun() {
        System.out.println("Ви взяли пістолет до рук");
        fout.print("Ви взяли пістолет до рук\n");
        isTaken = true;
    }
    /**
     * Method simulates gun's putting
     */
    public void putGun() {
        System.out.println("Ви положили пістолет");
        fout.print("Ви положили пістолет\n");
        isTaken = false;
    }
    /**
     * Method simulates gun's increasing maximum count of the ammo
     * @param count the count of the maximum ammo
     */
    public void increaseMaxAmmo(int count) {
        if (count <= maxAmmo) {
            System.out.println("Магазин менший для збільшення");
            fout.print("Магазин менший для збільшення\n");
        } else {
            maxAmmo = count;
            ammo = count;
            System.out.println("Магазин збільшений до " + maxAmmo);
            fout.print("Магазин збільшений до " + maxAmmo + "\n");
        }
    }
    /**
     * Method simulates gun's decreasing maximum count of the ammo
     * @param count the count of the maximum ammo
     */
    public void decreaseMaxAmmo(int count) {
        if (count >= maxAmmo) {
            System.out.println("Магазин більший для зменшення");
            fout.print("Магазин більший для зменшення\n");
        } else {
            maxAmmo = count;
            ammo = count;
            System.out.println("Магазин зменшений до " + maxAmmo);
            fout.print("Магазин зменшений до " + maxAmmo + "\n");
        }
```

```
    }
    /**
     * Method simulates deleting information about gun
     */
    public void deleteInfo() {
        this.gunName = "None";
        this.damage = 0;
        this.ammo = 0;
        this.maxAmmo = 0;
        this.isAim = false;
        this.isTaken = false;
        this.exploitation = 0;
        System.out.println("Інформація про пістолет видалена");
        fout.print("Інформація про пістолет видалена");
    }
    /**
     * Method simulates printing information about gun
     */
    public void printInfo() {
        System.out.println("\n");
        System.out.println("Імя " + gunName);
        System.out.println("Шкода " + damage);
        System.out.println("Боєприпаси " + ammo);
        System.out.println("Максимальна кількість боєприпасів " + maxAmmo);
        System.out.println("Експлуатація " + exploitation + "%" + "\n\n");

        fout.print("\n");
        fout.print("Імя " + gunName + "\n");
        fout.print("Шкода " + damage + "\n");
        fout.print("Боєприпаси " + ammo + "\n");
        fout.print("Максимальна кількість боєприпасів " + maxAmmo + "\n");
        fout.print("Експлуатація " + exploitation + "%" + "\n\n");
    }
    /**
     * Method releases used recourses
     */
    public void dispose()
    {
        fout.flush();
        fout.close();
    }
}
```

Файл WaterGun.java

```
/**
 * lab 4 package
 */
package KI34.Stepanov.Lab4;

import java.io.FileNotFoundException;

/**
 * Class <code>WaterGun</code> extends class Gun
 * @author Andriy Stepanov
 * @version 1.0
 */
public class WaterGun extends Gun {
    public String color;
    public int saturation;
    public int power;

    /**
     * Constructor
     * @param color Watergun's color
     * @param saturation The saturation of the color
     * @param power Watergun's power
     * @throws FileNotFoundException throw about non-existent file
```

```
     */
    public WaterGun(String gunName, int damage, int ammo, int maxAmmo, int
exploitation, String color, int saturation, int power) throws
FileNotFoundException {
        super(gunName, damage, ammo, maxAmmo, exploitation);
        this.color = color;
        this.saturation = saturation;
        this.power = power;
    }

    /**
     * Method simulates drawing a watergun
     * @param color color to draw
     * @param saturation saturation to add
     */
    void drawWaterGun(String color, int saturation) {
        this.color = color;
        this.saturation = saturation;

        System.out.println("Ви покрасили ваш пістолет у " + color + " колір із
насиченістю " + saturation + "%\n");
        fout.print("Ви покрасили ваш пістолет у " + color + " колір із
насиченістю " + saturation + "%\n");
    }

    /**
     * Method simulates double shoot
     */
    void doubleShoot() {
        ammo -= 2;

        System.out.println("Ви вистрілили подвійним снарядом");
        fout.print("Ви вистрілили подвійним снарядом\n");
    }

    /**
     * Method simulates deleting information about gun
     */
    public void deleteInfo() {
        this.gunName = "None";
        this.damage = 0;
        this.ammo = 0;
        this.maxAmmo = 0;
        this.isAim = false;
        this.isTaken = false;
        this.exploitation = 0;
        this.color = "None";
        this.saturation = 0;
        this.power = 0;
        System.out.println("Інформація про пістолет видалена");
        fout.print("Інформація про пістолет видалена");
    }
    /**
     * Method simulates printing information about gun
     */
    public void printInfo() {
        System.out.println("\n");
        System.out.println("Імя " + gunName);
        System.out.println("Шкода " + damage);
        System.out.println("Боєприпаси " + ammo);
        System.out.println("Максимальна кількість боєприпасів " + maxAmmo);
        System.out.println("Експлуатація " + exploitation + "%");
        System.out.println("Колір " + color);
        System.out.println("Насиченість " + saturation + "%");
        System.out.println("Сила " + power + "%" + "\n\n");
```

```
        fout.print("\n");
        fout.print("Ім'я " + gunName + "\n");
        fout.print("Шкода " + damage + "\n");
        fout.print("Боєприпаси " + ammo + "\n");
        fout.print("Максимальна кількість боєприпасів " + maxAmmo + "\n");
        fout.print("Експлуатація " + exploitation + "%" + "\n");
        fout.print("Колір " + color + "\n");
        fout.print("Насиченість " + saturation + "%" + "\n");
        fout.print("Сила " + power + "%" + "\n\n");
    }
}
```

Файл Draw.java

```
/**
 * lab 4 package
 */
package KI34.Stepanov.Lab4;

// оголошуємо інтерфейс Draw
public interface Draw {
    void drawWaterGun(String color, int saturation); // прототип методу
}
```

Результат виконання програми:

```
Ви взяли пістолет до рук


Імя Watergun
Шкода 15
Боєприпаси 5
Максимальна кількість боєприпасів 25
Експлуатація 75%
Колір green
Насиченість 50%
Сила 100%


Ви прицілилися
Вистріл
Спершу потрібно прицілитися
Ви прицілилися
Вистріл
Перезарядка
Ви прицілилися
Ви і так прицілені
Вистріл
Ви прицілилися
Вистріл
Магазин збільшений до 40


Імя Watergun
Шкода 15
Боєприпаси 40
Максимальна кількість боєприпасів 40
Експлуатація 71%
Колір green
Насиченість 50%
Сила 100%
```

Протокол діяльності в консолі

```
Магазин зменшений до 15
Пістолет відремонтовано
Магазин збільшений до 30
Ви прицілилися
Вистріл
Ви положили пістолет


Імя Watergun
Шкода 15
Боєприпаси 29
Максимальна кількість боєприпасів 30
Експлуатація 99%
Колір green
Насиченість 50%
Сила 100%


Ви покрасили ваш пістолет у червоний колір із насиченістю 75%

Ви взяли пістолет до рук
Ви прицілилися
Ви вистрілили подвійним снарядом
Ви положили пістолет


Імя Watergun
Шкода 15
Боєприпаси 27
Максимальна кількість боєприпасів 30
Експлуатація 99%
Колір червоний
Насиченість 75%
Сила 100%


Інформація про пістолет видалена
```

Протокол діяльності в консолі

```
Ви взяли пістолет до рук

Імя Watergun
Шкода 15
Боєприпаси 5
Максимальна кількість боєприпасів 25
Експлуатація 75%
Колір green
Насиченість 50%
Сила 100%

Ви прицілилися
Вистріл
Спершу потрібно прицілитися
Ви прицілилися
Вистріл
Перезарядка
Ви прицілилися
Ви і так прицілені
Вистріл
Ви прицілилися
Вистріл
Магазин збільшений до 40

Імя Watergun
Шкода 15
Боєприпаси 40
Максимальна кількість боєприпасів 40
Експлуатація 71%
Колір green
Насиченість 50%
Сила 100%

Магазин зменшений до 15
Пістолет відремонтовано
Магазин збільшений до 30
Ви прицілилися
Вистріл
Ви положили пістолет
```

Протокол діяльності в консолі

```
Імя Watergun
Шкода 15
Боєприпаси 29
Максимальна кількість боєприпасів 30
Експлуатація 99%
Колір green
Насиченість 50%
Сила 100%

Ви покрасили ваш пістолет у червоний колір із насиченістю 75%
Ви взяли пістолет до рук
Ви прицілилися
Ви вистрілили подвійним снарядом
Ви положили пістолет

Імя Watergun
Шкода 15
Боєприпаси 27
Максимальна кількість боєприпасів 30
Експлуатація 99%
Колір червоний
Насиченість 75%
Сила 100%

Інформація про пістолет видалена
Імя None
Шкода 0
Боєприпаси 0
Максимальна кількість боєприпасів 0
Експлуатація 0%
Колір None
Насиченість 0%
Сила 0%
```

Протокол діяльності в консолі

Package KI34.Stepanov.Lab4

package KI34.Stepanov.Lab4

All Classes and Interfaces | Interfaces | Classes

Class	Description
Draw	
Gun	Class Gun implements gun
GunApp	Gun Application class implements main method for Gun class possibilities demonstration
WaterGun	Class WaterGun extends class Gun

Згенерована документація

Package KI34.Stepanov.Lab4

Class Gun

java.lang.Object
  KI34.Stepanov.Lab4.Gun

Direct Known Subclasses:

WaterGun

public abstract class **Gun**
extends Object

Class Gun implements gun

Field Summary

Fields

Modifier and Type	Field	Description
protected int	ammo	
protected int	damage	
protected int	exploitation	
protected PrintWriter	fout	
protected String	gunName	
protected boolean	isAim	
protected boolean	isTaken	
protected int	maxAmmo	

Constructor Summary

Constructors

Constructor	Description
Gun(String gunName, int damage, int ammo, int maxAmmo, int exploitation)	Constructor

Інформація про клас Gun

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method	Description
void	aim()	Method simulates gun's aim
void	decreaseMaxAmmo(int count)	Method simulates gun's decreasing maximum count of the ammo
void	deleteInfo()	Method simulates deleting information about gun
void	dispose()	Method releases used recourses
int	getAmmo()	Method returns the count of the ammo
int	getDamage()	Method returns gun's damage
int	getExploitation()	Method returns exploitation
String	getGunName()	Method returns gun's name
int	getMaxAmmo()	Method returns the count of the ammo
void	increaseMaxAmmo(int count)	Method simulates gun's increasing maximum count of the ammo
boolean	isAim()	Method returns aiming the gun
boolean	isTaken()	Method returns taking gun
void	printInfo()	Method simulates printing information about gun
void	putGun()	Method simulates gun's putting
void	reload()	Method simulates gun's reload
void	repair()	Method simulates gun's repair
void	setAim(boolean aim)	Method sets aiming the gun
void	setAmmo(int ammo)	Method sets the count of the ammo
void	setDamage(int damage)	Method sets the new gun's damage
void	setExploitation(int exploitation)	Method sets exploitation the gun
void	setGunName(String gunName)	Method sets the new gun's name
void	setMaxAmmo(int maxAmmo)	Method sets maximum count of ammo
void	setTaken(boolean taken)	Method sets taking gun
void	shoot()	Method simulates gun's shoot
void	takeGun()	Method simulates gun's taking

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Інформація про клас Gun

Package KI34.Stepanov.Lab4

Class WaterGun

java.lang.Object
  KI34.Stepanov.Lab4.Gun
    KI34.Stepanov.Lab4.WaterGun

```
public class WaterGun
extends Gun
```

Class WaterGun extends class Gun

Field Summary

Fields

Modifier and Type	Field	Description
String	color	
int	power	
int	saturation	

Fields inherited from class KI34.Stepanov.Lab4.Gun

ammo, damage, exploitation, fout, gunName, isAim, isTaken, maxAmmo

Constructor Summary

Constructors

Constructor	Description
WaterGun(String gunName, int damage, int ammo, int maxAmmo, int exploitation, String color, int saturation, int power)	Constructor

Інформація про клас WaterGun

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method	Description
void	deleteInfo()	Method simulates deleting information about gun
void	printInfo()	Method simulates printing information about gun

Methods inherited from class KI34.Stepanov.Lab4.Gun

aim, decreaseMaxAmmo, dispose, getAmmo, getDamage, getExploitation, getGunName, getMaxAmmo, increaseMaxAmmo, isAim, isTaken, putGun, reload, repair, setAim, setAmmo, setDamage, setExploitation, setGunName, setMaxAmmo, setTaken, shoot, takeGun

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Details

color

```
public String color
```

saturation

```
public int saturation
```

power

```
public int power
```

Інформація про клас WaterGun

Package KI34.Stepanov.Lab4

Interface Draw

```
public interface Draw
```

Method Summary

All Methods | Instance Methods | Abstract Methods

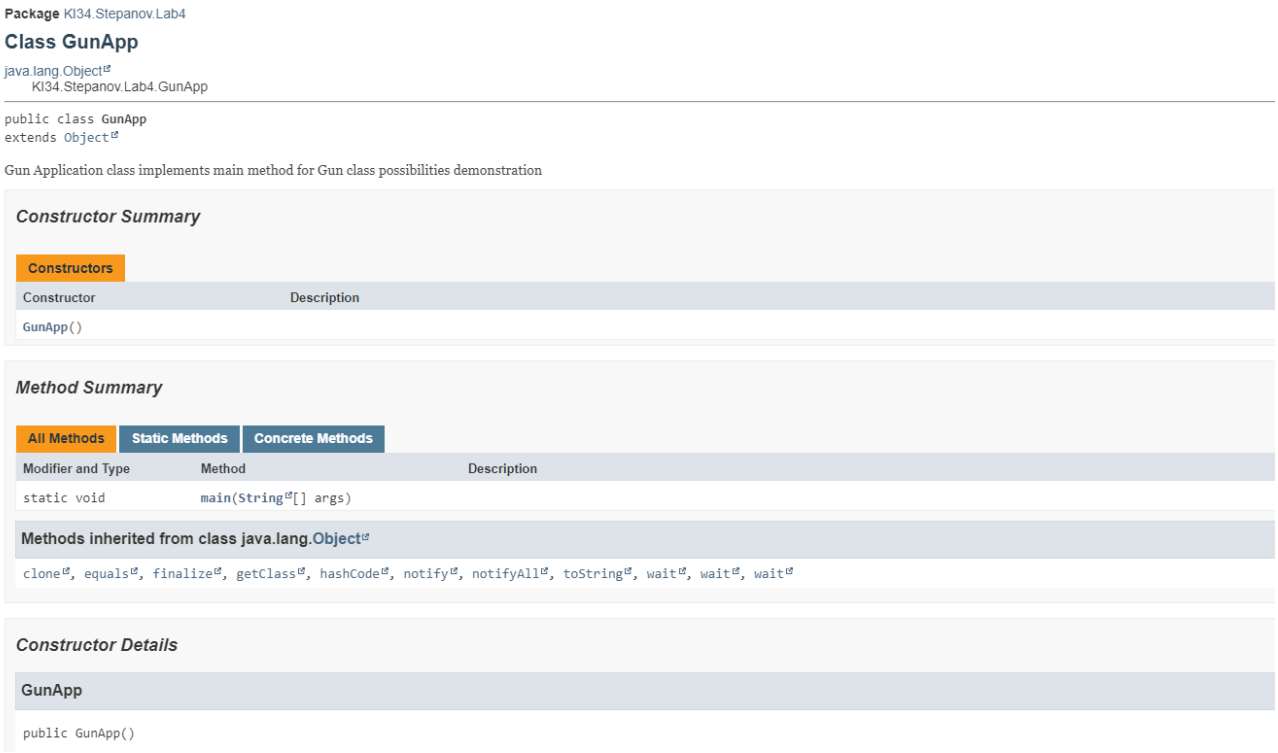
Modifier and Type	Method	Description
void	drawWaterGun(String color, int saturation)	

Method Details

drawWaterGun

```
void drawWaterGun(String color,
                  int saturation)
```

Інформація про інтерфейс Draw



Package KI34.Stepanov.Lab4

Class GunApp

java.lang.Object
KI34.Stepanov.Lab4.GunApp

```
public class GunApp
extends Object
```

Gun Application class implements main method for Gun class possibilities demonstration

Constructor Summary

Constructors

Constructor	Description
GunApp()	

Method Summary

All Methods | Static Methods | Concrete Methods

Modifier and Type	Method	Description
static void	main(String[] args)	

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Details

GunApp

```
public GunApp()
```

Інформація про клас GunApp

Відповіді на контрольні запитання:

1. Синтаксис реалізації спадкування.

Синтаксис реалізації спадкування:

class Підклас extends Суперклас

{

Додаткові поля і методи

}

2. Що таке суперклас та підклас??

В термінах мови Java базовий клас найчастіше називається суперкласом, а похідний клас – підкласом. Дана термінологія запозичена з теорії множин, де підмножина міститься у супермножині.

3. **Як звернутися до членів суперкласу з підкласу?**

Виклик методу суперкласу:

super.назваМетоду([параметри]);

Звертання до поля суперкласу:

super.назваПоля

4. **Коли використовується статичне зв'язування при виклику методу?**

Статичне зв'язування використовується коли метод є приватним, статичним, фінальним або конструктором.

5. **Як відбувається динамічне зв'язування при виклику методу?**

Віртуальна машина повинна викликати версію методу, що відповідає фактичному типу об'єкту на який посилається об'єктна змінна.

Оскільки на пошук необхідного методу потрібно багато часу, то віртуальна машина заздалегідь створює для кожного класу таблицю методів, в якій перелічуються сигнатури всіх методів і фактичні методи, що підлягають виклику. При виклику методу віртуальна машина просто переглядає таблицю методів.

6. **Що таке абстрактний клас та як його реалізувати?**

Абстрактний клас – це клас, для якого не можна створити об'єкти, призначений бути основою для розробки ієрархії класів.

Реалізується за допомогою ключового слова abstract.

7. **Для чого використовується ключове слово instanceof?**

Оператор instanceof дозволяє визначити, чи вказаний об'єкт належить до заданого типу.

8. **Як перевірити чи клас є підкласом іншого класу?**

Щоб перевірити чи клас є підкласом іншого класу, потрібно за допомогою instanceof порівняти, чи дійсно посилання на об'єкт супертипу посилається на об'єкт підтипу.

9. **Що таке інтерфейс?**

Це абстрактний тип, який використовується для визначення поведінки, яку класи повинні реалізовувати. Інтерфейси покликані компенсувати відсутність множинного спадкування у мові Java та гарантують визначення у класах оголошених у собі прототипів методів.

10. **Як оголосити та застосувати інтерфейс?**

Синтаксис оголошення інтерфейсів:

[public] interface НазваІнтерфейсу

{

    Прототипи методів та оголошення констант інтерфейсу

}

Висновок:

На цій лабораторній роботі я ознайомився з спадкуванням та інтерфейсами у мові Java.