МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний аерокосмічний університет ім. М. Є. Жуковського

«Харківський авіаційний інститут»

факультет програмної інженерії та бізнесу

кафедра інженерії програмного забезпечення

**Розрахункова-практична робота**

з дисципліни « ООП на Java »

на тему: « Класи, об’єкти та успадкування у мові Java »

Виконав: студент 2 курсу групи № 621п

121 «Інженерія програмного забезпечення»

(код спеціальності)

Кононов Э.В

(ПІБ студента)

Прийняв: доц. кафедри 603

Симбірський Г. Д.

(ПІБ викладача)

Національна шкала:

Кількість балів:

Харків – 2024

**Зміст**

[**ВСТУП** 3](file:///C:\Users\amilb\OneDrive\Рабочий%20стол\621п%20Гордеюк%20ргр.docx#_Toc167729066)

[Постановка задачі 3](file:///C:\Users\amilb\OneDrive\Рабочий%20стол\621п%20Гордеюк%20ргр.docx#_Toc167729067)

[Порядок виконання роботи 3](file:///C:\Users\amilb\OneDrive\Рабочий%20стол\621п%20Гордеюк%20ргр.docx#_Toc167729068)

[**РОЗДІЛ 1 ТЕОРЕТИЧНА ЧАСТИНА** 5](file:///C:\Users\amilb\OneDrive\Рабочий%20стол\621п%20Гордеюк%20ргр.docx#_Toc167729069)

[**РОЗДІЛ 2 КОНЦЕПТУАЛЬНА МОДЕЛЬ ТА ДІАГРАМА КЛАСІВ** 17](file:///C:\Users\amilb\OneDrive\Рабочий%20стол\621п%20Гордеюк%20ргр.docx#_Toc167729070)

[**ВИСНОВОК**](file:///C:\Users\amilb\OneDrive\Рабочий%20стол\621п%20Гордеюк%20ргр.docx#_Toc167729074) 22

**ВСТУП**

**Мета роботи:** отримання теоретичних знань та практичних навичок із застосування класів та методів при проектуванні об'єктно-орієнтованих програмних додатків мовою Java.

**Досліджуваний матеріал:** класи, об’єкти, механізми реалізації успадкування класів у мові Java, абстрактні класи та методи.

**Постановка задачи**

Розробити та реалізувати мовою Java програмний додаток для моделювання властивостей, поведінки та взаємодії об'єктів. Предметною областю для цього додатка можуть бути об'єкти, наведені нижче. Для заданої предметної області створити ієрархію класів. У додатку продемонструвати використання механізмів успадкування, абстрактних класів та методів, перевизначення методів у підкласах, а також застосування механізмів висхідних і низхідних перетворень типів об'єктів та поліморфізму.

**Порядок виконання роботи**

1. Вивчити теоретичний матеріал, пов'язаний з темою розрахунково-практичної роботи, використовуючи конспект лекцій, літературу, що рекомендується, і документацію, що входить до складу пакету Java SE JDK.

2. Розробити концептуальну модель для заданої категорії об'єктів у вигляді словесного опису їх основних властивостей та поведінки.

3. Спроектувати класи, в тому числі абстрактний, в яких будуть визначені загальні та характерні властивості об'єктів даної предметної області.

4. Визначити найбільш характерні риси поведінки цих об'єктів і подати їх у вигляді набору методів, визначених в абстрактному класі.

5. Відобразити структурні відносини між спроектованими класами та інтерфейсами за допомогою діаграми класів UML.

6. Реалізувати програмний код класів на мові Java. При реалізації програмного коду програми обов'язково використовувати коментарі та дескриптори документування для опису класів, конструкторів, методів та полів.

8. Відкомпілювати розроблені класи та інтерфейси програми за допомогою компілятора javac.exe.

9. Протестувати розроблений додаток.

12. Отримати документацію на розроблений додаток за допомогою утиліти javadoc.exe.

13. Зробити висновки щодо роботи.

14. Оформити звіт з розрахунково-практичної роботи відповідно до існуючих стандартів.

15. Захистити роботу.

# **РОЗДІЛ 1 ТЕОРЕТИЧНА ЧАСТИНА**

Java – це об’єктно-орієнтована мова програмування, але на початковому рівні використовується здебільшого процедурне програмування.

Процедурне програмування ‒ це написання процедур, які виконують операції з даними, тоді як об’єктно-орієнтоване програмування (ООП) ‒ це створення об’єктів, які містять і дані, і методи.

Основними поняттями об’єктно-орієнтованого програмування є класи та об’єкти.

Клас визначає абстрактні характеристики певної сутності, включаючи характеристики самої сутності (її атрибути або властивості) та дії, які вона здатна виконувати (її поведінки, методи або можливості).

Об’єкт – це окремий екземпляр класу (створюється після запуску програми та ініціалізації полів класу).

Під час створення окремих об’єктів вони успадковують усі змінні та методи класу.

Змінні чи атрибути відображають властивості та особливості класів та об’єктів.

Усе в Java пов’язано з класами та об’єктами разом із їх атрибутами та методами. Наприклад, у реальному житті автомобіль є предметом і має певні властивості – вага, колір та інші (**атрибути**), і виконує певні дії чи операції - водіння, гальмування, набір швидкості та інші (**методи**).

Сукупність значень атрибутів окремого об’єкта називається станом.

Методи– це можливості об’єкта. У межах програми використання методу має впливати лише на один об’єкт.

Подивіться на наступний приклад, щоб побачити різницю між класом та об’єктами.

Клас: фрукти.

Об’єкти: яблука, груши, персики.

Отже, клас – це шаблон для об’єктів, а об’єкт – екземпляр класу.

ООП – це метод програмування, заснований на поданні програми у вигляді сукупності взаємодіючих об’єктів, кожен з яких є екземпляром певного класу, а класи є членами певної ієрархії наслідування.

Об’єктно-орієнтований підхід полягає в наступному наборі основних принципів:

‒ усе (тобто оточуючий нас світ) складається з об’єктів;

‒ усі операції виконуються шляхом взаємодії (обміну даними) між об’єктами, при якій один об’єкт потребує, щоб інший виконав певну дію;

‒ об’єкти взаємодіють, надсилаючи і отримуючи повідомлення. Повідомлення – це запит на виконання дії, доповнений набором аргу-ментів, які можуть знадобитися під час виконання;

‒ кожен об’єкт має незалежну пам’ять, яка може складатись з інших об’єктів;

‒ кожен об’єкт є представником (екземпляром, примірником) класу, який виражає загальні властивості об’єктів;

‒ у класі задається поведінка (функціональність) об’єкта. Таким чином усі об’єкти, які є екземплярами одного класу, можуть виконувати одні й ті ж самі дії;

‒ класи організовані у єдину деревоподібну структуру з загальним корінням, яка називається ієрархією успадкування;

Таким чином, програма являє собою набір об’єктів, що мають певні стан та поведінку.

Загальна форма оголошення класу виглядає наступним чином:

модифікатори class Ім’я\_класу

{

// Оголошення змінних (полів)

модифікатори тип ім’я\_змінної1;

модифікатори тип ім’я\_змінної2 = ініц\_знач;

// Оголошення методів

модифікатори тип ім’я\_методу1(список\_параметрів)

{ тіло\_методу; }

}

Опис класу починається з ключового слова class, після якого записується ім'я класу. Загальноприйняті угоди (Code Conventions) рекомендують починати ім'я класу з великої літери, на відміну від імен методів і змінних, які починаються з маленької літери.

Перед словом class можуть знаходитись *модифікатори* класу (class modifiers) – ключові слова public, abstract, final, strictfp – які змінюють властивості класу. Ці модифікатори обговорюватимуться далі.

Тіло класу виділяється фігурними дужками і містить всі змінні (поля), методи, вкладені класи і інтерфейси

Щоб створити клас, потрібно використати ключове слово class.

Створення класу під назвою Main зі змінною x:

public class Main

{

int x = 5;

}

!!!Ім’я Класу завжди має починатися з великої першої літери та ім’я файлу Java має відповідати імені класу.

У Java об’єкт створюється з класу. Ми вже створили клас під назвою Main, тож тепер ми можемо використати його для створення об’єктів.

Щоб створити об’єкт класу Main, вкажіть ім’я класу, а потім ім’я об’єкту та використайте ключове слово new:

Main myObj = new Main();

Створення об’єкту під назвою myОbj і виведення значення x:

public class Main

{

int x = 5;

public static void main(String[] args)

{

Main myobj=new Main();

System.out.println(myobj.x);

}

}

Створення декількох об’єктів одного класу і виведення значення x:

public class Main

{

int x = 5;

public static void main(String[] args)

{

Main myobj1 = new Main(); //Об’єкт 1

Main myobj2 = new Main(); //Об’єкт 2

System.out.println(myobj1.x);

System.out.println(myobj2.x);

}

}

Можна створити об’єкт класу та отримати доступ до нього в іншому класі. Це часто використовується для кращої організації класів (один клас має всі атрибути та методи, а інший клас містить метод main() (код для виконання)).

Пам’ятайте, що ім’я файлу Java має відповідати імені класу. У наступному прикладі створені два файли в одному каталозі/папці.

Створення двух класів (Main.java та Second.java) і виведення значення x:

Main.java

public class Main {

int x = 5;

}

Second.java

class Second

{

public static void main(String[] args)

{

Main myObj = new Main();

System.out.println(myObj.x);

}

}

Коли обидва файли скомпільовані, маємо запис:

C:\Users\Your Name>javac Main.java

C:\Users\Your Name>javac Second.java

Запустіть файл Second.java:

C:\Users\Your Name>java Second

На екран буде виведене: 5

Атрибути класу

У попередньому розділі використано термін "змінна" для x в прикладах. Насправді, це - атрибут класу. Або можна сказати, що атрибути класу є змінними всередині класу.

Приклад 1.4

Створення класу під назвою Main із двома атрибутами: x та y:

public class Main

{

int x = 5;

int y = 3;

}

Іншим терміном для атрибуту класу є поле.

Доступ до атрибутів

Можна отримати доступ до атрибутів, створивши об’єкт класу та використовуючи синтаксис крапок (.):

Змінення атрибутів:

Встановлення значень атрибута x = 40:

public class Main

{

int x;

public static void main(String[] args)

{

Main myObj = new Main();

myObj.x = 40;

System.out.println(myObj.x);

}

}

Зміна існуючих значень:

Зміна значень атрибуту x:

public class Main

{

int x = 10;

public static void main(String[] args)

{

Main myObj = new Main();

myObj.x = 25; // x is now 25

System.out.println(myObj.x);

}

}

Якщо не потрібно мати можливість перевизначати існуючі значення, оголошують атрибут як final.

Встановлення незмінного значення атрибута х:

public class Main

{

final int x = 10;

public static void main(String[] args)

{

Main myObj = new Main();

myObj.x = 25; // спричинить помилку: неможливо

//змінити значення фінальної змінної

System.out.println(myObj.x);

}

}

Ключове слово final використовується у тих випадках, якщо ви хочете, щоб змінна завжди зберігала те саме значення, наприклад PI (3.14159...).

Ключове слово final називається "модифікатор". Більше про це далі.

Кілька об’єктів

Якщо ви створюєте кілька об’єктів одного класу, ви можете змінити значення атрибутів в одному об’єкті, не впливаючи на значення атрибутів в іншому.

Зміна значення x на 25 для myObj2 і залиште x в myObj1 без змін:

public class Main

{

int x = 5;

public static void main(String[] args)

{

Main myObj1 = new Main(); // Об’єкт 1

Main myObj2 = new Main(); // Об’єкт 2

myObj2.x = 25;

System.out.println(myObj1.x); // Виведення х для myObj1 (5)

System.out.println(myObj2.x); // Виведення х для myObj2 (25)

}

}

Можна вказати скільки завгодно атрибутів для одного об’єкту.

public class Main

{

String fname = "John";

String lname = "Doe";

int age = 24;

public static void main(String[] args)

[Методи класу в Java](https://w3schoolsua.github.io/java/java_methods.html)  оголошуються в класі та використовуються для виконання певних дій:

Створення методу під назвою myMethod():

Метод myMethod() друкує текст (дія), коли його викликають. Щоб викликати метод, напишіть назву методу, а потім дві дужки () та крапку з комою.

Всередині main викликається myMethod():

public class Main {

static void myMethod() {

System.out.println("Hello, World!");

}

public static void main(String[] args) {

myMethod();

}

}

Буде виведений напис "Hello World!"

Статичні (static) та публічні (public) методи

Ви часто бачите програми Java, які мають атрибути та методи static або public.

У прикладі вище ми створили static метод, що означає, що до нього можна отримати доступ без створення об’єкта класу, на відміну від методу public, до якого можуть отримати доступ лише об’єкти:

Приклад 2

Приклад для демонстрації відмінностей між static та public методів:

public class Main {

// Статичний метод

static void myStaticMethod() {

System.out.println("Статичні методи можна викликати без створення об’єктів");

}

// Публічний метод

public void myPublicMethod() {

System.out.println("Публічні методи необхідно викликати під час створення об’єктів");

}

// Main метод

public static void main(String[] args) {

myStaticMethod(); // Виклик статичного методу

// myPublicMethod(); Це призведе до компіляції помилки

Main myObj = new Main(); // Створення об’єкту Main

myObj.myPublicMethod(); // Виклик публічного методу для об’єкта

}

}

Методи доступу до об’єкту

Створення об’єкту класа Car під назвою myCar. Виклик методів fullThrottle() та speed() для myCar:

// Створення класу Main

public class Main {

// Створення методу fullThrottle()

public void fullThrottle() {

System.out.println("Машина їде так швидко, як тільки може!");

}

// Створення методу speed() і додайте параметр

public void speed(int maxSpeed) {

System.out.println("Максимальна швидкість: " + maxSpeed);

}

// Всередині main виклик методів об’єкта myCar

public static void main(String[] args) {

Main myCar = new Main(); // Створення об’єкту myCar

myCar.fullThrottle(); // Виклик методу fullThrottle()

myCar.speed(200); // Виклик методу speed()

}

}

Результат:

Машина їде так швидко, як тільки може!

Максимальна швидкість: 200

Пояснення прикладу 3

1) Ми створили спеціальний клас Main із ключовим словом class.

2) Ми створили методи fullThrottle() та speed() в Main класі.

3) Методи fullThrottle() та speed() виводять деякий текст, коли їх викликають.

4) Метод speed() приймає параметр int під назвою maxSpeed).

5) Щоб використовувати клас Main і його методи, нам потрібно створити об’єкт клас Main.

6) Потім перейдіть до методу main(), який, як ви вже знаєте, є вбудованим методом Java, який запускає вашу програму (будь-який код у main виконується).

7) Використовуючи ключове слово new, ми створили об’єкт із назвою myCar.

8) Потім ми викликаємо методи fullThrottle() та speed() в myCar і запускаємо програму, використовуючи ім’я об’єкта (myCar), а потім крапку (.), після чого йде назва методу (fullThrottle(); та speed(200);). Зверніть увагу, що ми додаємо параметр int 200 у метод speed().

Пам’ятайте, що:

1. Крапка (.) використовується для доступу до атрибутів і методів об’єкта.
2. Щоб викликати метод у Java, напишіть ім’я методу, а потім у дужках () і крапку з комою (;).
3. Клас повинен мати відповідну назву файлу (клас Main та файл Main.java).

# **РОЗДІЛ 2 КОНЦЕПТУАЛЬНА МОДЕЛЬ** **ТА ДІАГРАМА КЛАСІВ**

Клас «Main»

Властивості

* scanner: для зчитування введення з консолі.
* tv: базовий об'єкт телевізора.
* watchAuto: флаг для керування основним циклом програми.
* choice: змінна для збереження вибору користувача.
* setting: змінна для збереження налаштування, яке потрібно змінити.

Поведінка об’єктів:

* main: основний метод програми, який ініціалізує об'єкт телевізора на основі введення користувача і дозволяє взаємодіяти з ним.
* Type: визначає тип телевізора на основі введеного рядка.

Клас «TV»

Властивості

* modelType: властивість яка свідчить про тип телевізора (аналоговий, цифровий, кольоровий, чорно-білі, портативний).
* screenSize: властивість яка свідчить про розмір екрану в дюймах.
* numOfChannels: властивість яка свідчить про кількість доступних каналів.
* weight: властивість яка свідчить про вагу телевізора.
* price: властивість яка свідчить про ціну телевізора.
* condition: властивість яка свідчить про фізичний стан телевізора.
* color: властивість яка свідчить про колір телевізора.

Поведінка об’єктів:

* power: метод, що змінює стан на увімкнено або вімкнено.
* selectChannel: метод, який переключає канали.
* getCurrentChannel: метод, який виводить інформацію про поточний канал.
* GetChangeSettings: метод, що повертає поточні налаштування телевізора.
* SetChangeSettings: метод, що назначає поточні налаштування телевізора.

Клас «AnalogTV»

Властивості

* + power стан увімкнення телевізора (true - увімкнено, false - вимкнено).
  + channel поточний канал.
  + volume гучність звуку.
  + brightness: яскравість зображення.
  + contrast : контрастність зображення.

Поведінка об’єктів:

* + power(): змінює стан увімкнення телевізора.
  + selectChannel(int ch): вибирає канал за номером.
  + getCurrentChannel(): повертає поточний канал.
  + GetChangeSettings(int setting): повертає значення налаштування (гучність, яскравість або контрастність).
  + SetChangeSettings(int choice, int setting): змінює значення налаштування (гучність, яскравість або контрастність).

Клас «BlackAndWhiteTV»

Властивості

* + power стан увімкнення телевізора (true - увімкнено, false - вимкнено).
  + channel поточний канал.
  + volume гучність звуку.
  + brightness: яскравість зображення.
  + contrast : контрастність зображення.

Поведінка об’єктів:

* + power(): змінює стан увімкнення телевізора.
  + selectChannel(int ch): вибирає канал за номером.
  + getCurrentChannel(): повертає поточний канал.
  + GetChangeSettings(int setting): повертає значення налаштування (гучність, яскравість або контрастність).
  + SetChangeSettings(int choice, int setting): змінює значення налаштування (гучність, яскравість або контрастність).

Клас «ColoredTV»

Властивості

* + power стан увімкнення телевізора (true - увімкнено, false - вимкнено).
  + channel поточний канал.
  + volume гучність звуку.
  + brightness: яскравість зображення.
  + contrast : контрастність зображення.

Поведінка об’єктів:

* + power(): змінює стан увімкнення телевізора.
  + selectChannel(int ch): вибирає канал за номером.
  + getCurrentChannel(): повертає поточний канал.
  + GetChangeSettings(int setting): повертає значення налаштування (гучність, яскравість або контрастність).
  + SetChangeSettings(int choice, int setting): змінює значення налаштування (гучність, яскравість або контрастність).

Клас «DigitalTV»

Властивості

* + power стан увімкнення телевізора (true - увімкнено, false - вимкнено).
  + channel поточний канал.
  + volume гучність звуку.
  + brightness: яскравість зображення.
  + contrast : контрастність зображення.

Поведінка об’єктів:

* + power(): змінює стан увімкнення телевізора.
  + selectChannel(int ch): вибирає канал за номером.
  + getCurrentChannel(): повертає поточний канал.
  + GetChangeSettings(int setting): повертає значення налаштування (гучність, яскравість або контрастність).
  + SetChangeSettings(int choice, int setting): змінює значення налаштування (гучність, яскравість або контрастність).

Клас «PortableTV»

Властивості

* + power стан увімкнення телевізора (true - увімкнено, false - вимкнено).
  + channel поточний канал.
  + volume гучність звуку.
  + brightness: яскравість зображення.
  + contrast : контрастність зображення.

Поведінка об’єктів:

* + power(): змінює стан увімкнення телевізора.
  + selectChannel(int ch): вибирає канал за номером.
  + getCurrentChannel(): повертає поточний канал.
  + (гучність, яскравість або контрастність).
  + SetChangeSettings(int choice, int setting): змінює значення налаштування (гучність, яскравість або контрастність).

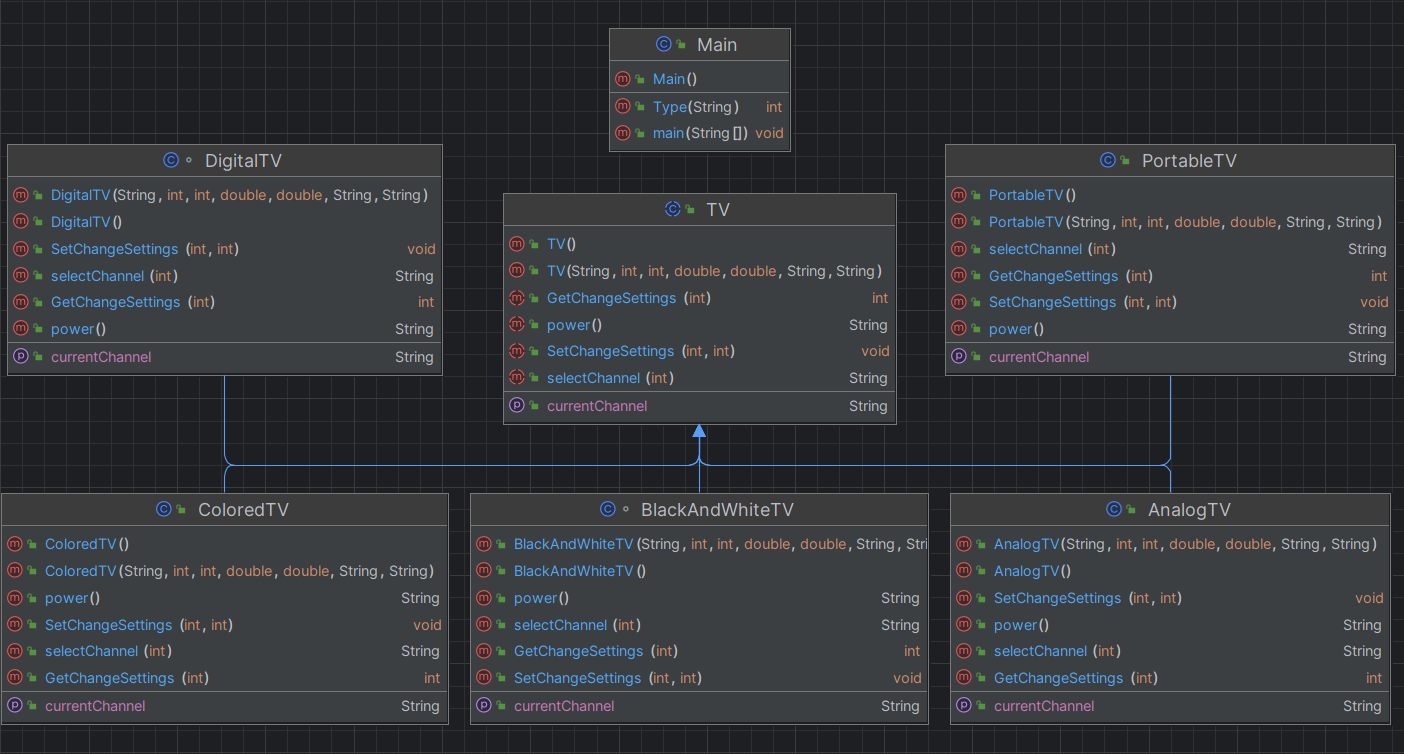


Рисунок 2.1 - UML діаграма створених класів

# **ВИСНОВОК**

В ході виконання розрахунково-практичної роботи було застосовано теоретичну частину та **КОНЦЕПТУАЛЬНУ МОДЕЛЬ ТА ДІАГРАМУ КЛАСІВ -** Клас «TV», Клас «Main», Клас «AnalogTV».