ЗВІТ

з лабораторної роботи **№3**

з дисципліни ‘Кросплатформні засоби програмування’

Виконала:

Ст.гр. КІ-303

Онишкевич Тетяна

Перевірив:

Майдан М.В

Львів - 2023

**СПАДКУВАННЯ ТА ІНТЕРФЕЙСИ**

**Завдання:**

1. Написати та налагодити програму на мові Java, що розширює клас, що реалізований у лабораторній роботі №2, для реалізації предметної області заданої варіантом. Суперклас, що реалізований у лабораторній роботі №2, зробити абстрактним. Розроблений підклас має забезпечувати механізми свого коректного функціонування та реалізовувати мінімум один інтерфейс. Програма має розміщуватися в пакеті Група.Прізвище.Lab3 та володіти коментарями, які дозволять автоматично згенерувати документацію до розробленого пакету.

2. Автоматично згенерувати документацію до розробленого пакету.

3. Завантажити код на GitHub згідно методичних вказівок по роботі з GitHub.

4. Скласти звіт про виконану роботу з приведенням тексту програми, результату її виконання та фрагменту згенерованої документації та завантажити його у ВНС.

5. Дати відповідь на контрольні запитання.

**Варіант:**



**Код програми:**

from abc import ABC, abstractmethod

class TV(ABC):

"""Абстрактний клас"""

def \_\_init\_\_(self, brand: str, price: float, screen\_size: int):

self.COLOUR = ("neutral", "warm", "white", "cool", "native")

self.STATES = ("OFF", "ON")

self.BASE\_CHANNEL = 0

self.BASE\_VOLUME = 50

self.brand = brand

self.price = price

self.screen\_size = screen\_size

self.is\_on = True

self.volume = self.BASE\_VOLUME

self.scheme = self.COLOUR[0]

self.current\_channel = self.BASE\_CHANNEL

@abstractmethod

def state(self) -> None: ...

@abstractmethod

def set\_colour\_scheme(self, scheme: str) -> None: ...

@abstractmethod

def chanel\_switch(self, channel: int) -> None: ...

@abstractmethod

def volume\_control(self, volume: int) -> None: ...

@abstractmethod

def \_\_str\_\_(self) -> str: ...

class Tuner:

def \_\_init\_\_(self):

...

class TvWithTuner(TV, ABC):

"""Перевизначення класу"""

def \_\_init\_\_(self, brand: str, price: float, screen\_size: int, tuner: Tuner):

super().\_\_init\_\_(brand, price, screen\_size)

self.tuner = tuner

def state(self) -> None:

print(f"Turning off the {self.brand} TV...") if self.is\_on else print(f"Turning on the {self.brand} TV...")

self.is\_on = not self.is\_on

def set\_colour\_scheme(self, scheme: str) -> None:

self.scheme = scheme if scheme in self.COLOUR else self.COLOUR[0]

def chanel\_switch(self, channel: int) -> None:

self.current\_channel = channel if self.is\_on else self.BASE\_CHANNEL

print(f"Changed to channel {self.current\_channel}")

def volume\_control(self, volume: int) -> None:

self.volume = volume if volume in range(100) else self.BASE\_VOLUME

print(f"Changed on {self.volume} dB")

def \_\_str\_\_(self) -> str:

return (f"Model: {self.brand}\n"

f"Price: {self.price} $\n"

f"Screen size: {self.screen\_size} Inch \n"

f"Power: {self.STATES[self.is\_on]}\n"

f"Channel: {self.current\_channel}\n"

f"Volume: {self.volume}")

def main():

tv = TvWithTuner("Samsung", 130, 45, Tuner())

tv.chanel\_switch(10)

tv.set\_colour\_scheme("white")

print(tv)

if \_\_name\_\_ == '\_\_main\_\_':

main()

[Посилання на Github](https://github.com/totioti/CPPT_Onyshkevych_TI_KI-303_1" \o "Посилання на Github)

**Результат:**

