

Додаток 1
Міністерство освіти і науки України
Національний технічний університет України «Київський політехнічний інститут імені Ігоря
Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

Звіт
з лабораторної роботи № 4 з дисципліни «Основи програмування»
«Успадкування та поліморфізм»

Варіант 12

Виконав студент: ІП-13 Дойчев Костянтин Миколайович

Перевірила: Вечерковська Анастасія Сергіївна

Київ 2021

Лабораторна робота №4

Тема: Успадкування та поліморфізм

Постановка задачі

12. Спроекувати клас TEquation, який представляє рівняння і містить віртуальні методи для знаходження коренів рівняння та перевірки, чи є деяке значення коренем рівняння. На основі цього класу створити класи-нащадки, які представляють лінійні та квадратні рівняння. Створити n лінійних рівнянь та m квадратних рівнянь, згенерувавши дані для них випадковим чином. Знайти суму коренів для кожного із видів рівнянь (за умови, що вони існують). Перевірити, чи є задане значення коренем вказаного рівняння

Код:

Файл - main.py

```
from QuadraticEquation import QuadraticEquation
from LinearEquation import LinearEquation
from random import randint

def generate_random_equation(Equation, min_coeff, max_coeff,
                             number_of_instances, number_of_coefficients):
    try:
        equations = []
        for i in range(number_of_instances):
            coefficients = []
            for j in range(number_of_coefficients):
                coefficients.append(randint(min_coeff, max_coeff))
            eq = Equation(*coefficients)
            equations.append(eq)
        return equations
    except ValueError as e:
        print(str(e))

def main():
```

```

n = int(input("Number of linear equations: "))
m = int(input("Number of quadratic equations: "))
min_coeff = int(input("Minimum coefficient: "))
max_coeff = int(input("Maximum coefficient: "))

linear_equations = generate_random_equation(LinearEquation, min_coeff,
max_coeff, n, 2)
quadratic_equations = generate_random_equation(QuadraticEquation,
min_coeff, max_coeff, m, 3)

for i in range(len(linear_equations)):
    print("Linear equation {}: \nRoot: {}; Coefficients: {}; \n".format(i +
1, *linear_equations[i].find_roots(),

linear_equations[i].get_coefficients()))

print('\n')

for i in range(len(quadratic_equations)):
    print("Quadratic equation {}: \nRoots: {}; Coefficients: {}; \n".format(i
+ 1, quadratic_equations[i].find_roots(),

quadratic_equations[i].get_coefficients()))

linear_roots_sum = [eq.calculate_roots_sum() for eq in linear_equations]
quadratic_equations_roots = [eq.calculate_roots_sum() for eq in
quadratic_equations]

linear_sum = sum(linear_roots_sum)
quadratic_sum = 0

for i in range(len(quadratic_equations_roots)):
    if quadratic_equations_roots[i] is not None:
        quadratic_sum += quadratic_equations_roots[i]

print("Linear equations sum:", linear_sum)
print("Quadratic equations sum:", quadratic_sum)

possible_root = float(input("Possible root: "))
is_linear = True if input('Check linear equation? (y/n): ') == 'y' else
False
equation_index = int(input("Equation index: "))

equations = linear_equations if is_linear else quadratic_equations
is_root_correct = False
for index in range(len(equations)):
    if index == equation_index and equations[index].is_root(possible_root):
        is_root_correct = True
        break

```

```
print('Root is correct' if is_root_correct else 'Root is incorrect')
```

```
if __name__ == '__main__':  
    main()
```

Файл: Tequation.py

```
class TEquation:

    _number_of_coefficients = 0

    _coefficients = []

    _roots = []

    def __init__(self, *coefficients):

        if self.are_coefficients_valid(coefficients):

            self._coefficients = coefficients

        else:

            raise ValueError("Invalid coefficients: {}".format(coefficients))

    def are_coefficients_valid(self, coefficients):

        is_valid_quantity = len(self.get_coefficients()) ==
self.get_number_of_coefficients()

        return is_valid_quantity and coefficients[0] != 0

    def find_roots(self):

        pass

    def get_roots(self):

        return self._roots
```

```
def get_coefficients(self):

    return self._coefficients


def get_number_of_coefficients(self):

    return self._number_of_coefficients


def is_root(self, root):

    if self.get_roots() is None:

        return False


    if len(self.get_roots()) == 0:

        self.find_roots()


    for r in self.get_roots():

        if r == root:

            return True


def calculate_roots_sum(self):

    if self.get_roots() is None:

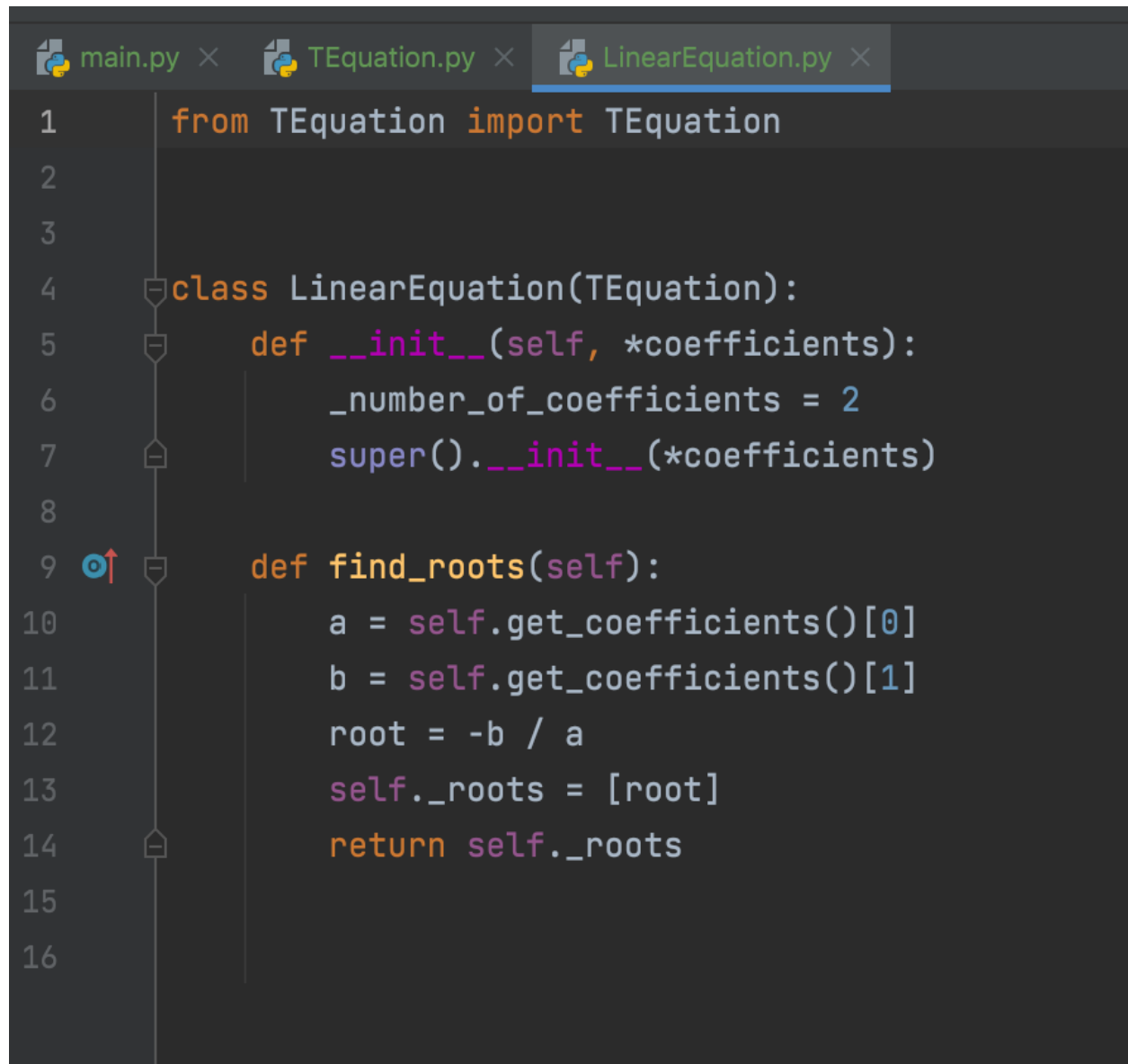
        return None


    if len(self.get_roots()) == 0:
```

```
self.find_roots()
```

```
return sum(self.get_roots())
```

LinearEquation.py



```
1 from TEquation import TEquation
2
3
4 class LinearEquation(TEquation):
5     def __init__(self, *coefficients):
6         _number_of_coefficients = 2
7         super().__init__(*coefficients)
8
9     def find_roots(self):
10         a = self.get_coefficients()[0]
11         b = self.get_coefficients()[1]
12         root = -b / a
13         self._roots = [root]
14         return self._roots
15
16
```

QuadraticEquation.py

```
main.py x TEquation.py x LinearEquation.py x QuadraticEquation.py x
1  from TEquation import TEquation
2
3
4  class QuadraticEquation(TEquation):
5      def __init__(self, *coefficients):
6          _number_of_coefficients = 3
7          super().__init__(*coefficients)
8
9      def find_roots(self):
10         a = self.get_coefficients()[0]
11         b = self.get_coefficients()[1]
12         c = self.get_coefficients()[2]
13         discriminant = b ** 2 - 4 * a * c
14         if discriminant < 0:
15             self._roots = None
16         elif discriminant == 0:
17             self._roots = [-b / (2 * a)]
18         else:
19             self._roots = [(-b + discriminant ** 0.5) / (2 * a), (-b - discriminant ** 0.5) / (2 * a)]
20         return self._roots
21
22
```

Дані і консоль:


```
/usr/local/bin/python3.9 /Users/Kostia/Documents/Programming/kpi/programming-basics/Python/bachelor/year-1/semester-2/lab-4/main.py
Number of linear equations: 2
Number of quadratic equations: 2
Minimum coefficient: -20
Maximum coefficient: 20
Linear equation 1:
Root: -2.375; Coefficients: (-8, -19);

Linear equation 2:
Root: 19.0; Coefficients: (-1, 19);

Quadratic equation 1:
Roots: None; Coefficients: (-15, -14, -13)

Quadratic equation 2:
Roots: [0.9540659228538015, -3.3540659228538017]; Coefficients: (5, 12, -16)

Linear equations sum: 16.625
Quadratic equations sum: -2.4000000000000004
Possible root: 19.0
Check linear equation? (y/n): y
Equation index: 1
Root is correct

Process finished with exit code 0
|
```