**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра CАПР**

отчет

**по лабораторной работе №2**

**по дисциплине «ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ KOTLIN»**

Тема: **Консольная программа для работы с CSV и XML файлами.**

|  |  |  |
| --- | --- | --- |
| Студент гр. 0302 |  | Кузнецов К.Е. |
| Студент гр. 0302 |  | Головатюк К.А. |
| Преподаватель |  | Кулагин М.В. |

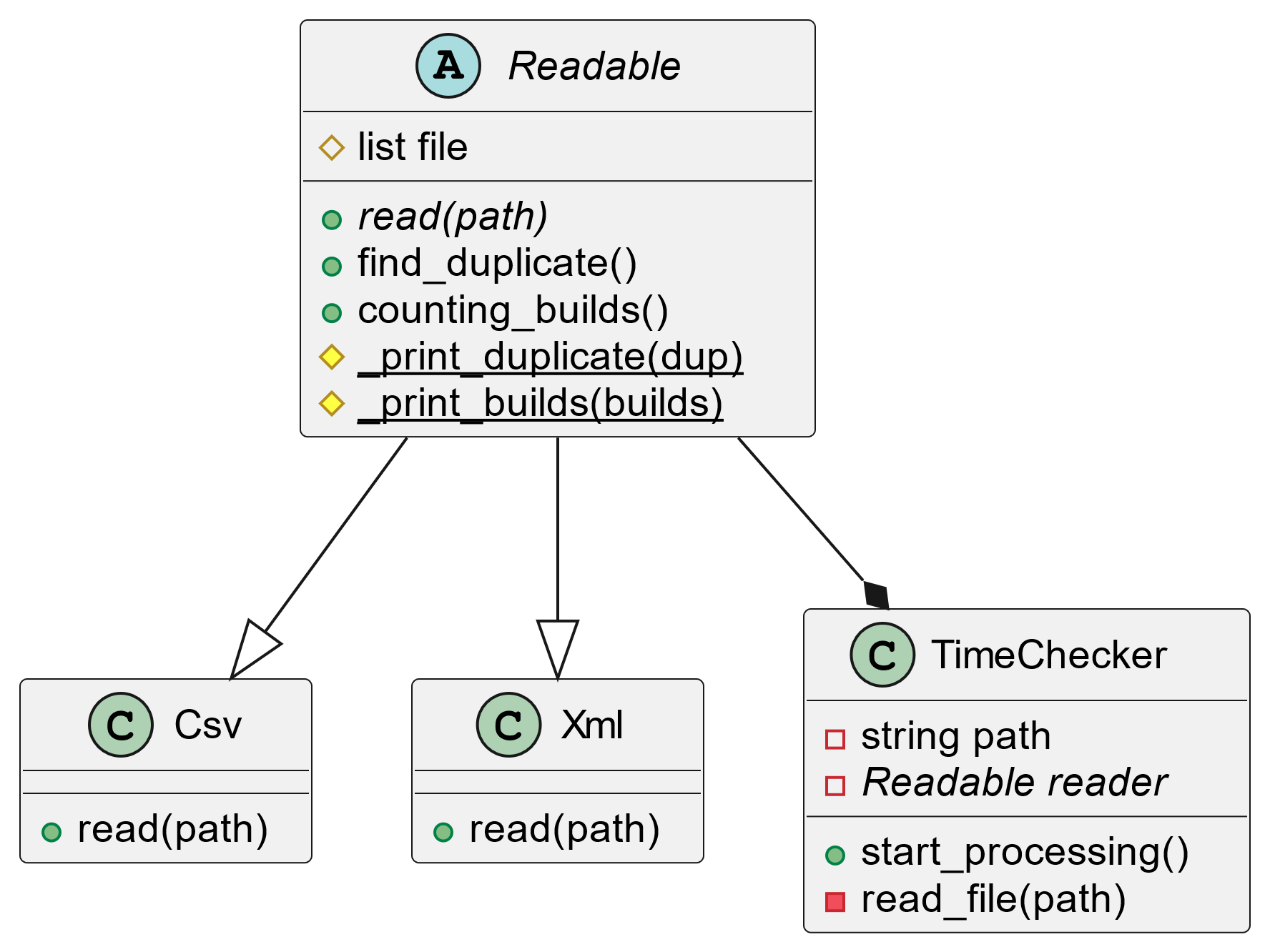
Санкт-Петербург

2022

# Цель работы

Написать программу, которая обрабатывает XML и CSV файлы.

# Спецификация программы



# Описание интерфейса пользователя программы

Пользователю выводится информация о действиях. Пользователь может выйти из программы, написав “exit”, или ввести путь до файла. После ввода пути начинается обработка файла, с выводом затраченного времени. При обработке печатается количество дублирующихся строк и количество зданий в каждом городе.

# Текст программы

main.py:

from XmlReader import Xml

from CsvReader import Csv

from AbstractReadable import Readable

import time

class TimeChecker:

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

self.\_\_reader = Readable

def start\_processing(self, path: str):

timer = time.time()

if self.\_\_read\_file(path):

print(f"\nВремя потраченное на чтение - {time.time() - timer}\n")

prev = time.time() - timer

self.\_\_reader.find\_duplicate()

print(f"\nВремя потраченное на подсчет дубликатов - {time.time() - timer - prev}\n")

prev = time.time() - timer

self.\_\_reader.counting\_builds()

print(f"\nВремя потраченное на подсчет зданий - {time.time() - timer - prev}\n")

print(f"Общее время на обработку - {time.time() - timer}\n")

return

print("Неверный путь к файлу или неверный формат!!!")

def \_\_read\_file(self, path):

if len(path) <= 3:

return False

if path[-3:] == "xml":

self.\_\_reader = Xml()

elif path[-3:] == "csv":

self.\_\_reader = Csv()

else:

return False

return self.\_\_reader.read(path)

def loop():

data = TimeChecker()

while True:

print("Введите 'exit' для завершения")

print("Введите путь к файлу")

path = str(input())

if path == "exit":

break

data.start\_processing(path)

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

loop()

AbstractReadable.py:

from abc import ABC, abstractmethod

class Readable(ABC):

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

self.\_file = []

@property

def file(self):

return self.\_file

@abstractmethod

def read(self, path):

"""Чтение файла"""

def find\_duplicate(self):

rows = set()

dup = dict()

for row in self.\_file:

if str(row) in rows:

if dup.get(str(row)) is None:

dup[str(row)] = 0

dup[str(row)] += 1

rows.add(str(row))

self.\_print\_duplicate(dup)

return dup

def counting\_builds(self):

rows = set()

builds = dict()

for row in self.\_file:

if str(row) in rows:

continue

rows.add(str(row))

if builds.get(row["city"]) is None:

builds[row["city"]] = {'1': 0, '2': 0, '3': 0, '4': 0, '5': 0}

builds[row["city"]][row["floor"]] += 1

self.\_print\_builds(builds.items())

@staticmethod

def \_print\_duplicate(dup):

for i in dup.items():

str\_dict = eval(i[0])

print(f'"{str\_dict["city"]}";'

f'"{str\_dict["street"]}";'

f'{str\_dict["house"]};'

f'{str\_dict["floor"]}'

f' - duplicate amount : {i[1]}')

@staticmethod

def \_print\_builds(builds):

for i in builds:

print(f'{i[0]} :\n'

f'Количество этажей\t-\t\t1\t\t2\t\t3\t\t4\t\t5\n'

f'Количество домов\t-\t{i[1]["1"]}\t{i[1]["2"]}\t{i[1]["3"]}\t{i[1]["4"]}\t{i[1]["5"]}')

CsvReader.py:

import csv

from AbstractReadable import Readable

class Csv(Readable):

def read(self, path):

try:

reader = csv.DictReader(open(path, 'r', encoding='utf-8'), delimiter=';')

for row in reader:

self.\_file.append(row)

except FileNotFoundError:

return False

return True

XmlReader.py:

import xml.etree.ElementTree as ElementTree

from AbstractReadable import Readable

class Xml(Readable):

def read(self, path):

try:

tree = ElementTree.parse(path)

for row in tree.getroot():

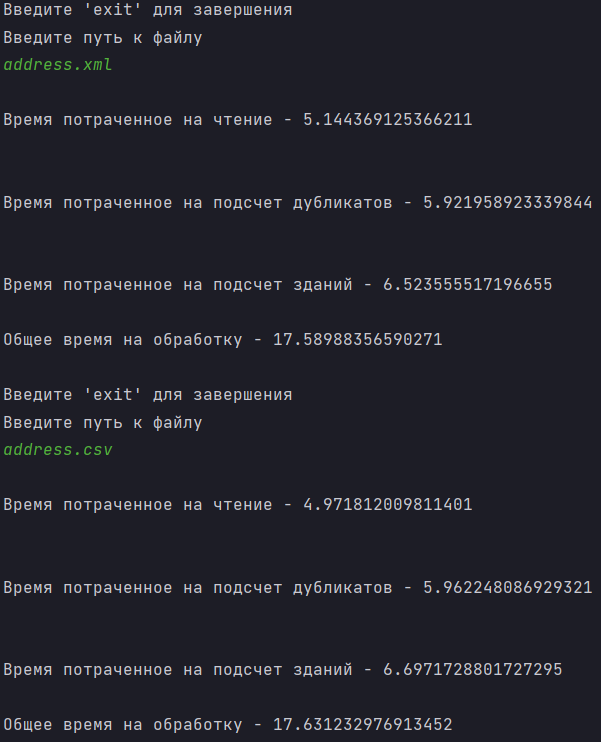
self.\_file.append(row.attrib)

except FileNotFoundError:

return False

return True

# Пример работы



# Выводы

В ходе данной работы были изучены структуры форматов CSV и XML, а также получен опыт обработки этих форматов.