Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт перспективной инженерии
Департамент цифровых, робототехнических систем и электроники

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №6
дисциплины
«Объектно-ориентированное программирование»
Вариант № 3

Выполнил:
Левашев Тимур Рашидович
3 курс, группа ИВТ-б-о-23-2,
09.03.01 «Информатика и вычислительная техника»,
направленность (профиль) «Программное обеспечение средств вычислительной техники и автоматизированных систем», очная форма обучения

_____
(подпись)

Проверил:
Доцент департамента цифровых, робототехнических систем и электроники института перспективной инженерии Воронкин Р.А

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2025 г.

Тема работы: “Работа с исключениями в языке Python”

Цель работы: приобретение навыков по работе с исключениями при написании программ с помощью языка программирования Python версии 3.x.

Порядок выполнения работы:

Ссылка на Git репозиторий: https://github.com/mazy99/oop_pract_6

1. Простой пример использования исключения при делении числа на 0.

Листинг программы:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-


print("start")

try:
    val = int(input("Enter a number: "))
    tmp = 10 / val
    print(f"10 divided by {val} is {tmp}")

except Exception as e:
    print(f"Error: {e}")
print("end")
```

```
(oop-pract-6) PS C:\учеба\ООП\6 лаба\oop_pract_6> uv run examples\simple_except.py
start
Enter a number: 2
10 divided by 2 is 5.0
end
(oop-pract-6) PS C:\учеба\ООП\6 лаба\oop_pract_6> uv run examples\simple_except.py
start
Enter a number: 0
Error: division by zero
end
(oop-pract-6) PS C:\учеба\ООП\6 лаба\oop_pract_6>
```

Рисунок 1 – Результат работы программы

2. Использование нескольких исключений в прогармме.

Листинг программы:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
```

```python
print("start")

try:
    val = int(input("Enter a number: "))
    tmp = 10 / val
    print(f"10 divided by {val} is {tmp}")

except ValueError as ve:
    print("ValueError: {0}".format(ve))

except ZeroDivisionError as zde:
    print("ZeroDivisionError: {0}".format(zde))

except Exception as ex:
    print("Error: {0}".format(ex))

print("stop")
```

```
(oop-pract-6) PS C:\учеба\ООП\6 лаба\oop_pract_6> uv run examples\set_except_2.py
start
Enter a number: 2
10 divided by 2 is 5.0
stop
(oop-pract-6) PS C:\учеба\ООП\6 лаба\oop_pract_6> uv run examples\set_except_2.py
start
Enter a number: 0
ZeroDivisionError: division by zero
stop
(oop-pract-6) PS C:\учеба\ООП\6 лаба\oop_pract_6> uv run examples\set_except_2.py
start
Enter a number: a
ValueError: invalid literal for int() with base 10: 'a'
stop
(oop-pract-6) PS C:\учеба\ООП\6 лаба\oop_pract_6>
```

Рисунок 2 – Результат работы программы

3. Пример пользовательского исключения.

Листинг программы:

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-


class NegValException(Exception):
    pass


try:
    val = int(input("Enter a positive number: "))
```

```
    if val < 0:
        raise NegValException(f"Negative value entered {val}")
    print(f"val + 10 = {val+10}")
except NegValException as nve:
    print(f"NegValException: {nve}")
```

```
(oop-pract-6) PS C:\учеба\ООП\6 лаба\oop_pract_6> uv run examples\user_except.py
Enter a positive number: -1
NegValException: Negative value entered -1
(oop-pract-6) PS C:\учеба\ООП\6 лаба\oop_pract_6>
```

Рисунок 3 – Результат работы программы

4. **Задание 1:** написать программу, которая запрашивает ввод двух значений. Если хотя бы одно из них не является числом, то должна выполняться конкатенация, т.е соединение строк. В остальных случаях введённые числа суммируются.

```
$ python3 test.py
Первое значение: 4
Второе значение: 5
Результат: 9.0
$ python3 test.py
Первое значение: a
Второе значение: 9
Результат: a9
```

Рисунок 4 – Ожидаемый вывод программы

Листинг программы:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-


from typing import Union


class InputReader:

    @staticmethod
    def read(prompt: str) -> str:
        try:
            value = input(prompt)
        except Exception as e:
            print(f"Ошибка ввода: {e}")

        if value.strip() == "":
```

```
            raise ValueError("Пустое значение недопустимо")
        return value


class ValueParser:

    @staticmethod
    def parse_int(value: str) -> Union[int, float, str]:
        try:
            if "." in value:
                return float(value)
            return int(value)
        except ValueError:
            return value


class Calculator:

    a: Union[int, float, str]
    b: Union[int, float, str]

    def __init__(self, a: Union[float, int, str], b: Union[float, int, str]) ->
None:
        self.a = a
        self.b = b

    def calculate(self) -> Union[float, str]:
        if isinstance(self.a, (int, float)) and isinstance(self.b, (int, float)):
            return self.a + self.b
        return str(self.a) + str(self.b)

    def __str__(self) -> str:
        return f"Результат сложения: {self.calculate()}"
```

```
(oop-pract-6) PS C:\учеба\ООП\6 лаба\oop_pract_6> uv run tasks\task_calculator.py
Введите первое значение: 2
Введите второе значение: 2
Результат сложения: 4
(oop-pract-6) PS C:\учеба\ООП\6 лаба\oop_pract_6> uv run tasks\task_calculator.py
Введите первое значение: a
Введите второе значение: a
Результат сложения: aa
(oop-pract-6) PS C:\учеба\ООП\6 лаба\oop_pract_6> uv run tasks\task_calculator.py
Введите первое значение: a
Введите второе значение: 2
Результат сложения: a2
(oop-pract-6) PS C:\учеба\ООП\6 лаба\oop_pract_6>
```

Рисунок 5 – Результат выполнения программы

5. **Задание 2:** написать программу, которая будет генерировать матрицу из случайных целых чисел. Пользователь может указать число строк и столбцов, а также диапазон целых чисел. Произвести обработку ошибок ввода пользователей.

Листинг программы:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-


import random
from typing import List


class MatrixGenerator:
    def __init__(self, rows: int, cols: int, min_value: int, max_value: int) ->
None:
        self.rows = rows
        self.cols = cols
        self.min_value = min_value
        self.max_value = max_value

    def generate_matrix(self) -> List[List[int]]:
        matrix = []
        for _ in range(self.rows):
            row = [
                random.randint(self.min_value, self.max_value) for _ in
range(self.cols)
            ]
            matrix.append(row)
        return matrix

    def __str__(self) -> str:
        matrix = self.generate_matrix()
        return "Сгенерированная матрица:\n" + "\n".join(str(row) for row in
matrix)
```

Листинг вызова программы:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

from matrix_generate import MatrixGenerator


def main():
```

```
    try:
        rows = int(input("Введите количество строк матрицы: "))
        cols = int(input("Введите количество столбцов матрицы: "))

        if rows <= 0 or cols <= 0:
            raise ValueError("Количество строк и столбцов должно быть
положительным.")

        min_value = int(input("Введите минимальное значение элемента матрицы: "))
        max_value = int(input("Введите максимальное значение элемента матрицы:
"))

        if min_value > max_value:
            raise ValueError("Минимальное значение не может быть больше
максимального.")

        generator = MatrixGenerator(rows, cols, min_value, max_value)
        print(generator)
    except ValueError as ve:
        print(f"Ошибка ввода: {ve}")
        return


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

```
(oop-pract-6) PS C:\учеба\ООП\6 лаба\oop_pract_6> uv run tasks\task_matrix.py
Введите количество строк матрицы: 3
Введите количество столбцов матрицы: 3
Введите минимальное значение элемента матрицы: 1
Введите максимальное значение элемента матрицы: 10
Сгенерированная матрица:
[4, 2, 5]
[9, 5, 3]
[5, 9, 6]
(oop-pract-6) PS C:\учеба\ООП\6 лаба\oop_pract_6> uv run tasks\task_matrix.py
Введите количество строк матрицы: 0
Введите количество столбцов матрицы: 1
Ошибка ввода: Количество строк и столбцов должно быть положительным.
(oop-pract-6) PS C:\учеба\ООП\6 лаба\oop_pract_6> uv run tasks\task_matrix.py
Введите количество строк матрицы: 2
Введите количество столбцов матрицы: 2
Введите минимальное значение элемента матрицы: 10
Введите максимальное значение элемента матрицы: 5
Ошибка ввода: Минимальное значение не может быть больше максимального.
(oop-pract-6) PS C:\учеба\ООП\6 лаба\oop_pract_6>
```

Рисунок 6 – Результат работы программы

6. Выполнение индивидуального задания 1 (Проверка корректность e-mail).

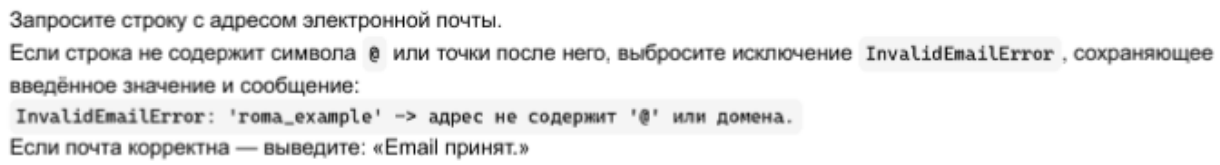
Запросите строку с адресом электронной почты.
Если строка не содержит символа `@` или точки после него, выбросите исключение `InvalidEmailError`, сохраняющее введённое значение и сообщение:
`InvalidEmailError: 'roma_example' -> адрес не содержит '@' или домена.`
Если почта корректна — выведите: «Email принят.»

Рисунок 7 – Индивидуальное задание 1

Листинг программы:

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-


class InvalidEmailError(Exception):

    def __init__(
        self, email: str, message: str = "адрес не содержит '@' или домена"
    ) -> None:
        self.email = email
        self.message = message
        super(InvalidEmailError, self).__init__(self.message)

    def __str__(self) -> str:
        return f"{self.email} -> {self.message}"


class EmailValidator:

    def validate(self, email: str) -> str:

        if "@" not in email:
            raise InvalidEmailError(email)

        name, _, domain = email.partition("@")

        if "." not in domain:
            raise InvalidEmailError(email)

        if not name or not domain:
            raise InvalidEmailError(email)

        return f"Ваш {email} принят"
```

```
(oop-pract-6) PS C:\учеба\ООП\6 лаба\oop_pract_6> uv run tasks\task_email.py
Введите email: maz1882@mail.ru
Ваш maz1882@mail.ru принят
(oop-pract-6) PS C:\учеба\ООП\6 лаба\oop_pract_6> uv run tasks\task_email.py
Введите email: mazzxz111.ru
mazzxz111.ru -> адрес не содержит '@' или домена
(oop-pract-6) PS C:\учеба\ООП\6 лаба\oop_pract_6> uv run tasks\task_email.py
Введите email: mmasdada199@
mmasdada199@ -> адрес не содержит '@' или домена
(oop-pract-6) PS C:\учеба\ООП\6 лаба\oop_pract_6>
```

Рисунок 8 – Результат работы программы

7. Выполнение индивидуального задания 2 (Учет студентов с отбором по неудовлетворительным оценкам).

Листинг исключений программы:

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-


class UnknownCommandError(Exception):

    def __init__(self, command: str, message: str = "Unknown command") -> None:
        self.command = command
        self.message = message
        super(UnknownCommandError, self).__init__(self.message)

    def __str__(self) -> str:
        return f"{self.command} -> {self.message}"


class InvalidGradeError(Exception):

    def __init__(self, grade: str, message: str = "Invalid grade") -> None:
        self.grade = grade
        self.message = message
        super(InvalidGradeError, self).__init__(self.message)

    def __str__(self) -> str:
        return f"{self.grade} -> {self.message}"
```

Листинг программы:

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
```

```python
from dataclasses import dataclass, field
from typing import List

from exceptions import InvalidGradeError


@dataclass(frozen=True)
class Student:
    name: str
    group: int
    grades: List[int]

    def has_failing_grades(self) -> bool:
        return 2 in self.grades


@dataclass
class Staff:
    students: List[Student] = field(default_factory=lambda: [])

    def add(self, name: str, group: int, grades: str) -> None:

        try:
            grades_list = list(map(int, grades.split()))
            if len(grades_list) != 5:
                raise InvalidGradeError(grades, "Оценок должно быть ровно 5")
            for grade in grades_list:
                if grade < 2 or grade > 5:
                    raise InvalidGradeError(
                        grades, "Оценки должны быть в диапазоне от 2 до 5"
                    )
        except ValueError:
            raise InvalidGradeError(grades, "Оценки должны быть целыми числами")

        self.students.append(Student(name=name, group=group, grades=grades_list))
        self.students.sort(key=lambda student: student.name)

    def __str__(self) -> str:
        if not self.students:
            return "Список студентов пуст."

        table = []
        line = "+-{}-+-{}-+-{}-+-{}-+".format("-" * 4, "-" * 30, "-" * 20, "-" *
20)
        table.append(line)
        table.append(
            "| {:^4} | {:^30} | {:^20} | {:^20} |".format(
                "№", "Ф.И.О", "Группа", "Оценки"
            )
        )
        table.append(line)
```

```
        for idx, student in enumerate(self.students, 1):
            table.append(
                "| {:>4} | {:<30} | {:>20} | {:<20} |".format(
                    idx, student.name, student.group, " ".join(map(str,
student.grades))
                )
            )
        table.append(line)
        return "\n".join(table)

    def select(self) -> List[Student]:
        return [student for student in self.students if
student.has_failing_grades()]
```

Листинг вызова программы:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-


import logging
import sys

from exceptions import InvalidGradeError, UnknownCommandError
from models import Staff
from storage import StudentStorage


def show_help():
    help_text = """
    Список команд:

    add     - добавить студента;
    list    - вывести весь список студентов;
    select  - выполнить выборку по критерию (студенты с оценкой 2);
    save    - сохранить данные в XML файл;
    load    - загрузить данные из XML файла;
    help    - вывести справку;
    exit    - завершить программу.
    """
    print(help_text)


def main():
    logging.basicConfig(
        filename="student.log",
        level=logging.INFO,
        encoding="utf-8",
```

```
    )

    staff = Staff()

    try:
        students = StudentStorage.load("data.xml")
        staff.students = students
        staff.students.sort(key=lambda student: student.name)
        logging.info("Загружены данные из файла data.xml")
        print("Данные из файла data.xml успешно загружены.")
    except FileNotFoundError:
        logging.info("Файл data.xml не найден, начинаем с пустой базы")
    except Exception as e:
        logging.error(f"Ошибка при загрузке данных: {e}")
        print(f"Ошибка при загрузке данных: {e}")

    print("Программа для учета студентов")
    print("Введите 'help' для просмотра списка команд")

    while True:
        try:
            command = input(">>> ").lower()
            logging.info(f"Введена команда: {command}")

            if command == "exit":
                save_choice = (
                    input("Сохранить данные перед выходом? (y/n):
").strip().lower()
                )
                if save_choice == "y":
                    StudentStorage.save(staff.students, "students.xml")
                    logging.info("Данные сохранены перед выходом")
                    print("Данные сохранены в файл students.xml")

                print("Завершение работы программы...")
                logging.info("Программа завершена")
                break

            elif command == "add":
                name = input("Ф.И.О студента: ").strip()
                group = input("Номер группы: ").strip()
                print("Введите 5 оценок через пробел (например: 5 4 3 5 4):")
                grades = input("Оценки: ").strip()

                try:
                    staff.add(name, group, grades)
                    logging.info(
                        f"Добавлен студент: {name}, группа: {group}, оценки:
{grades}"
                    )
                    print(f"Студент {name} успешно добавлен!")
```

```
                except InvalidGradeError as e:
                    logging.error(f"Ошибка при добавлении студента: {e}")
                    print(f"Ошибка: {e}")

            elif command == "list":
                print(staff)
                logging.info("Выведен полный список студентов")

            elif command == "select":
                selected = staff.select()
                if selected:
                    print("\nСтуденты с неудовлетворительными оценками (2):")
                    print("=" * 60)
                    for idx, student in enumerate(selected, 1):
                        print(
                            f"{idx:>3}. {student.name}, группа: {student.group},
"
                            f"оценки: {student.grades}"
                        )
                    print("=" * 60)
                    logging.info(f"Найдено {len(selected)} студентов с оценкой
2")
                else:
                    print("Студентов с неудовлетворительными оценками (2) не
найдено.")
                    logging.info("Не найдено студентов с оценкой 2")

            elif command.startswith("save"):
                parts = command.split(maxsplit=1)
                if len(parts) > 1:
                    filename = parts[1]
                else:
                    filename = "students.xml"
                StudentStorage.save(staff.students, filename)
                logging.info(f"Сохранены данные в файл {filename}")
                print(f"Данные сохранены в файл {filename}")

            elif command.startswith("load"):
                parts = command.split(maxsplit=1)
                if len(parts) > 1:
                    filename = parts[1]
                else:
                    filename = "students.xml"
                try:
                    students = StudentStorage.load(filename)
                    staff.students = students
                    staff.students.sort(key=lambda student: student.name)
                    logging.info(f"Загружены данные из файла {filename}")
                    print(f"Данные из файла {filename} успешно загружены:")
                    print(staff)
                except FileNotFoundError:
```

```
                    logging.error(f"Файл {filename} не найден")
                    print(f"Файл {filename} не найден.")
                except Exception as e:
                    logging.error(f"Ошибка при загрузке из файла {filename}:
{e}")
                    print(f"Ошибка при загрузке файла: {e}")
            elif command == "help":
                show_help()
                logging.info("Выведена справка по командам")

            else:
                raise UnknownCommandError(command)

        except UnknownCommandError as e:
            logging.error(f"Неизвестная команда: {command}")
            print(f"Ошибка: {e}")
            print("Введите 'help' для просмотра списка команд")

        except KeyboardInterrupt:
            print("\nПрограмма прервана пользователем")
            logging.warning("Программа прервана пользователем")
            break

        except Exception as e:
            logging.error(f"Неожиданная ошибка: {e}")
            print(f"Произошла ошибка: {e}", file=sys.stderr)


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

```
Введите 'help' для просмотра списка команд
>>> help

    Список команд:

    add     - добавить студента;
    list    - вывести весь список студентов;
    select  - выполнить выборку по критерию (студенты с оценкой 2);
    save    - сохранить данные в XML файл;
    load    - загрузить данные из XML файла;
    help    - вывести справку;
    exit    - завершить программу.
```

Рисунок 9 – Основные команды программы

```
>>> add
Ф.И.О студента: Петров П.П
Номер группы: 999
Введите 5 оценок через пробел (например: 5 4 3 5 4):
Оценки: 5 5 5 5 4
Студент Петров П.П успешно добавлен!
>>> add
Ф.И.О студента: Борисов Б.Б
Номер группы: 777
Введите 5 оценок через пробел (например: 5 4 3 5 4):
Оценки: 2 2 2 2 2
Студент Борисов Б.Б успешно добавлен!
```

Рисунок 10 – Добавление студентов

```
>>> add
Ф.И.О студента: Петров П.П
Номер группы: 222
Введите 5 оценок через пробел (например: 5 4 3 5 4):
Оценки: 2
Ошибка: 2 -> Оценок должно быть ровно 5
```

Рисунок 11 – Обработка исключения для количества оценок

```
>>> list
+------+--------------------------------+----------------------+----------------------+
|  №   |            Ф.И.О               |        Группа        |        Оценки        |
+------+--------------------------------+----------------------+----------------------+
|    1 | Борисов Б.Б                    |                  777 | 2 2 2 2 2            |
|    2 | Петров П.П                     |                  999 | 5 5 5 5 4            |
+------+--------------------------------+----------------------+----------------------+
>>>
```

Рисунок 12 – Вывод добавленных студентов

```
>>> save
Данные сохранены в файл students.xml
>>>
```

Рисунок 13 – Сохранения данных о студентах в xml файл

```
>>> load
Данные из файла students.xml успешно загружены:
+------+--------------------------------+----------------------+----------------------+
|  №   |            Ф.И.О               |        Группа        |        Оценки        |
+------+--------------------------------+----------------------+----------------------+
|    1 | Борисов Б.Б                    |                  777 | 2 2 2 2 2            |
|    2 | Петров П.П                     |                  999 | 5 5 5 5 4            |
+------+--------------------------------+----------------------+----------------------+
>>>
```

Рисунок 14 – Загрузка данных из xml файла

Вывод: в ходе выполнения работы были приобретены навыки по работе с исключениями при написании программ с помощью языка программирования Python версии 3.x.