# Ministerul Educației al Republicii Moldova Universitatea Tehnică a Moldovei

# Departamentul Ingineria Software si Automatica

# Дисциплина: Programare Declarativă

# Отчёт

# по лабораторной работе №4

тема:

**GUI**

# Выполнил: учащийся гр. TI-217

А. Выртикуш

# Проверил: преподаватель по специальности

# Leah

Chișinău, 2023

**Цель работы:** Научиться работать с GUI.

**Требование:**

1. Лабораторная работа выполняется в виде отдельного python-скрипта.
2. Каждая лабораторная работа должна быть загружена в отдельный git-репозиторий.
3. Взаимодействие с репозиторием должно производиться посредством работы с IDE, а не через сайт.
4. Репозитории с одним коммитом к проверке не принимаются.
5. Сообщения коммитов должны быть осмысленными и отражать процесс выполнения задания.
6. Код должен соответствовать требованиям соглашения PEP8.

**Задание для варианта 3–10**

1. Приложение должно иметь кнопки для получения следующего экземпляра класса из датасета. Например, если у вас датасет из кошек и собак, то должна быть кнопка "Следующая кошка" и кнопка "Следующая собака". После нажатия на них должен быть получен следующий путь при помощи итератора, а затем отображена картинка новой кошки или собаки из вашего датасета в интерфейсе вашей программы. В случае варианта с текстом отображаем текст.

**Решение**

import os

import sys

import random

import csv

from PyQt5 import QtWidgets, QtGui

# Папка, куда будет создан файл-аннотации

annotation\_folder = 'D:\\UTM\\PD\\PD\_git\_lab\\lab4\\'

annotation\_file = os.path.join(annotation\_folder, 'annotation.csv')

# Функция для создания аннотации

def create\_annotation\_file(source\_folder, class\_name):

    with open('annotations.csv', 'w', newline='') as csv\_file:

        writer = csv.writer(csv\_file)

        writer.writerow(['D:\\UTM\\PD\\PD\_git\_lab\\lab\_3\\dataset3', '..\\lab\_2\\dataset', 'zebra'])

        for root, dirs, files in os.walk(source\_folder):

            for file in files:

                if file.endswith('.jpg'):

                    absolute\_path = os.path.join(root, file)

                    relative\_path = os.path.relpath(absolute\_path, annotation\_folder)

                    label = os.path.basename(absolute\_path)

                    writer.writerow([absolute\_path, relative\_path, label])

def get\_next\_instance(class\_label, dataset\_path):

    class\_files = [f for f in os.listdir(dataset\_path) if f.startswith(class\_label)]

    if class\_files:

        random.shuffle(class\_files)

        return os.path.join(dataset\_path, class\_files.pop())

    else:

        return None

class DatasetApp(QtWidgets.QWidget):

    def \_\_init\_\_(self):

        super().\_\_init\_\_()

        self.initUI()

    def initUI(self):

        # Определите интерфейс

        self.setWindowTitle('Dataset App')

        self.setGeometry(100, 100, 1300, 550)

        self.folderpath = None  # Переменная для хранения пути к папке с исходным датасетом

        # Кнопка для выбора папки с исходным датасетом

        self.browse\_button = QtWidgets.QPushButton('Выбрать папку  датасетом', self)

        self.browse\_button.setGeometry(20, 20, 200, 30)

        self.browse\_button.clicked.connect(self.browse\_dataset\_folder)

        # Кнопка для создания файла аннотации

        self.create\_annotation\_button = QtWidgets.QPushButton('Создать аннотацию', self)

        self.create\_annotation\_button.setGeometry(20, 60, 200, 30)

        self.create\_annotation\_button.setEnabled(False)  # По умолчанию отключена

        self.create\_annotation\_button.clicked.connect(self.create\_annotation)

        # Кнопки для получения следующего экземпляра

        self.next\_class1\_button = QtWidgets.QPushButton('Следующий класс 1', self)

        self.next\_class1\_button.setGeometry(20, 100, 200, 30)

        self.next\_class1\_button.setEnabled(False)  # По умолчанию отключена

        self.next\_class1\_button.clicked.connect(self.get\_next\_class1)

        self.next\_class2\_button = QtWidgets.QPushButton('Следующий класс 2', self)

        self.next\_class2\_button.setGeometry(20, 140, 200, 30)

        self.next\_class2\_button.setEnabled(False)  # По умолчанию отключена

        self.next\_class2\_button.clicked.connect(self.get\_next\_class2)

        self.image\_label = QtWidgets.QLabel(self)

        self.image\_label.setGeometry(250, 20, 1000, 500)  # Изменили размеры здесь

        self.image\_label.setScaledContents(True)

    def browse\_dataset\_folder(self):

        self.folderpath = QtWidgets.QFileDialog.getExistingDirectory(self, 'Выберите папку  исходным датасетом')

        if self.folderpath:

            self.create\_annotation\_button.setEnabled(True)

            self.next\_class1\_button.setEnabled(True)

            self.next\_class2\_button.setEnabled(True)

    def create\_annotation(self):

        if self.folderpath:

            create\_annotation\_file(self.folderpath, 'zebra')

            create\_annotation\_file(self.folderpath, 'tulip')

            pass

    def get\_next\_class1(self):

        if self.folderpath:

            next\_instance = get\_next\_instance('zebra', self.folderpath)

            if next\_instance:

                pixmap = QtGui.QPixmap(next\_instance)

                self.image\_label.setPixmap(pixmap)

    def get\_next\_class2(self):

        if self.folderpath:

            next\_instance = get\_next\_instance('tulip', self.folderpath)

            if next\_instance:

                pixmap = QtGui.QPixmap(next\_instance)

                self.image\_label.setPixmap(pixmap)

def main():

    app = QtWidgets.QApplication(sys.argv)

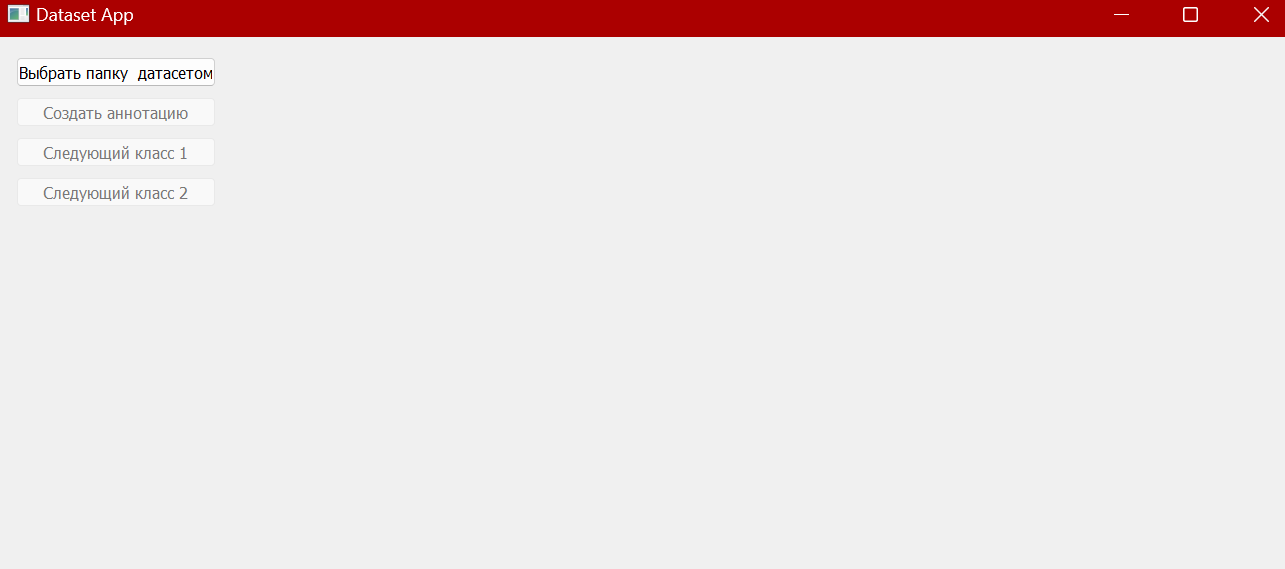
    window = DatasetApp()

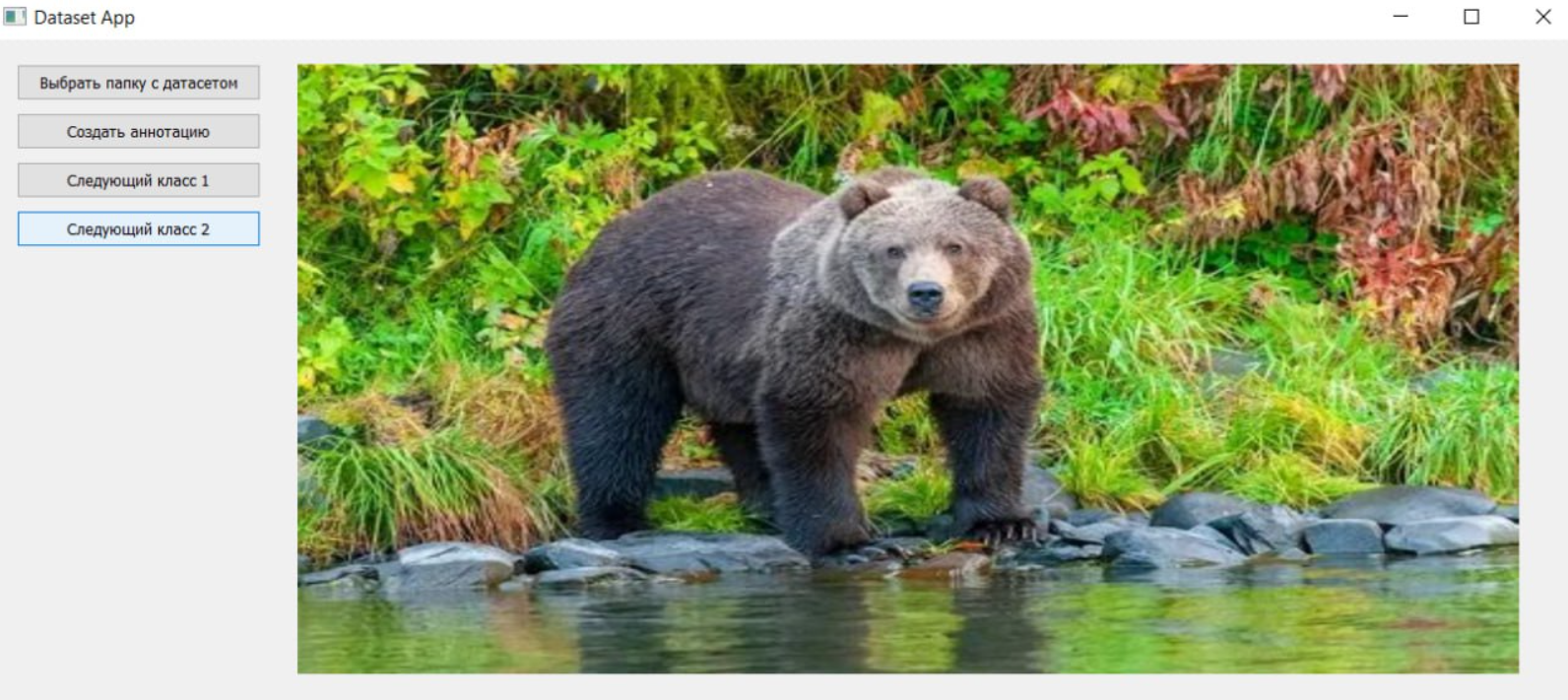
    window.show()

    sys.exit(app.exec\_())

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

    main()





**Вывод**:  
В процессе выполнения лабораторной работы был предоставлен код приложения, предназначенного для работы с изображениями в датасете. Этот код также иллюстрирует использование библиотеки PyQt5 для создания графического пользовательского интерфейса (GUI) на языке Python. Для преобразования изображения в объект QPixmap использовался метод QPixmap.fromImage.

Программа успешно обрабатывает запросы на получение следующего экземпляра класса в зависимости от выбранной кнопки и отображает соответствующую информацию в интерфейсе. В случае, если датасет содержит изображения, при каждом запросе отображается новая картинка белого или бурого медведя. В случае текстового датасета программа выводит новый текст.