Министерство Просвещения, Культуры и Исследований Республики Молдова

Технический университет Молдовы

Факультет Вычислительной Техники, Информатики и Микроэлектроники

Департамент Микроэлектроники и Биоинженерии

**Отчет**

**По предмету Programarea Declarativă**

**По лабораторной работе №3**

**Тема:** «Обработка данных.»

Выполнил Cuțar Alexandr (TI-218)

Проверил A.Leah

Кишинев 2023

# **Задания к самостоятельной работе**

**Задание**

Создать приложение с графическим интерфейсом повторяющее функционал лабораторной работы 3. Необходимо создать файл main\_window.py в котором будет реализован графический интерфейс вашего приложения. Здесь не должно быть реализации алгоритмов, все алгоритмы должны быть **импортированы** как функции или объекты из других файлов. Возможно вам потребуется выполнить **рефакторинг** вашего кода из лабораторной работы 3, чтобы было удобнее интегрировать его в графический интерфейс.

1. Приложение должно запрашивать у пользователя путь к папке исходного датасета:

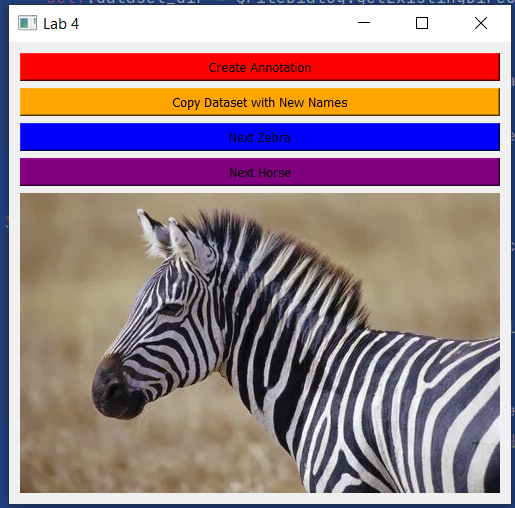
folderpath = QtWidgets.QFileDialog.getExistingDirectory(self, 'Select Folder')

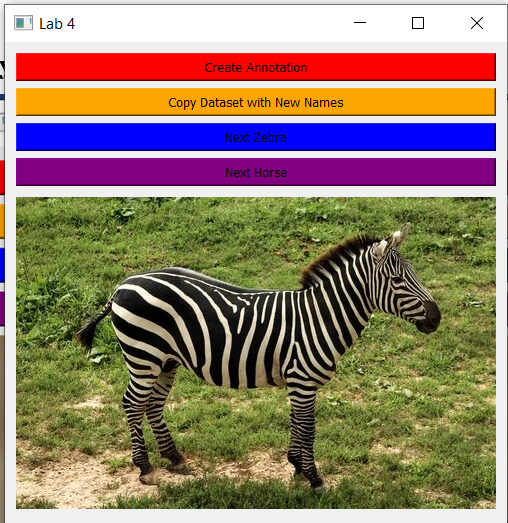
1. Приложение должно иметь кнопку для создания файла аннотации исходного датасета. Для этого потребуется запросить у пользователя путь к файлу назначения.
2. **Вариант 3-10**
3. 1. Приложение должно иметь кнопки для получения следующего экземпляра класса из датасета. Например, если у вас датасет из кошек 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.

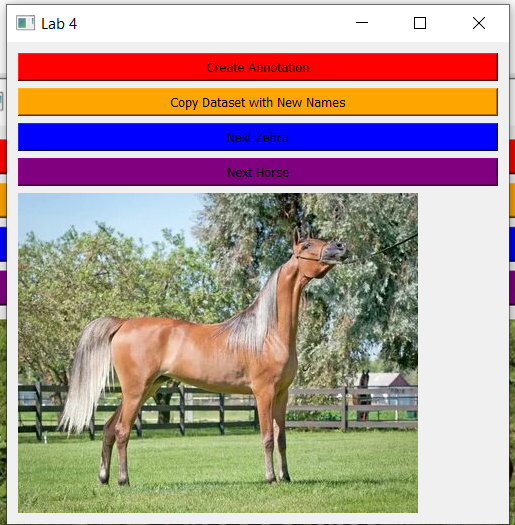
**Код main\_window.py**

import sys  
import os  
from PyQt5.QtWidgets import QApplication, QMainWindow, QWidget, QVBoxLayout, QPushButton, QLabel, QFileDialog  
from PyQt5.QtGui import QPixmap  
  
# Импортируйте функции из предыдущих заданий  
from main import create\_annotation\_file as create\_annotation\_file\_func  
from main import copy\_dataset\_with\_new\_names  
from main import get\_next\_instance\_by\_class  
  
class MainWindow(QMainWindow):  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
  
 self.setWindowTitle("Lab 4")  
  
 self.central\_widget = QWidget(self)  
 self.layout = QVBoxLayout(self.central\_widget)  
  
 self.annotation\_button = QPushButton("Create Annotation")  
 self.annotation\_button.setStyleSheet("background-color : red")  
 self.annotation\_button.clicked.connect(self.create\_annotation)  
  
 self.copy\_button = QPushButton("Copy Dataset with New Names")  
 self.copy\_button.setStyleSheet("background-color : orange")  
 self.copy\_button.clicked.connect(self.copy\_dataset)  
  
 self.next\_zebra\_button = QPushButton("Next Zebra")  
 self.next\_zebra\_button.setStyleSheet("background-color : blue")  
 self.next\_zebra\_button.clicked.connect(self.get\_next\_zebra)  
  
 self.next\_horse\_button = QPushButton("Next Horse")  
 self.next\_horse\_button.setStyleSheet("background-color : purple")  
 self.next\_horse\_button.clicked.connect(self.get\_next\_horse)  
  
 self.image\_label = QLabel()  
  
 self.layout.addWidget(self.annotation\_button)  
 self.layout.addWidget(self.copy\_button)  
 self.layout.addWidget(self.next\_zebra\_button)  
 self.layout.addWidget(self.next\_horse\_button)  
 self.layout.addWidget(self.image\_label)  
  
 self.setCentralWidget(self.central\_widget)  
  
 self.dataset\_dir = "D:\\Ucheba\\3 course\\PD\\lab2\\dataset"  
  
 def create\_annotation(self):  
 self.dataset\_dir = QFileDialog.getExistingDirectory(self, "Select Dataset Folder")  
  
 if self.dataset\_dir:  
 annotation\_file\_path, \_ = QFileDialog.getSaveFileName(self, "Save Annotation File", "", "CSV Files (\*.csv)")  
 if annotation\_file\_path:  
 create\_annotation\_file\_func(self.dataset\_dir, annotation\_file\_path) # Вызываем версию функции с двумя аргументами  
  
  
 def copy\_dataset(self):  
 self.dataset\_dir = QFileDialog.getExistingDirectory(self, "Select Dataset Folder")  
  
 if self.dataset\_dir:  
 new\_dataset\_dir = QFileDialog.getExistingDirectory(self, "Select Destination Folder")  
  
 if new\_dataset\_dir:  
 copy\_dataset\_with\_new\_names(self.dataset\_dir, new\_dataset\_dir) # Вызываем функцию с нужными аргументами  
 # annotation\_file\_path, \_ = QFileDialog.getSaveFileName(self, "Save Annotation File", "", "CSV Files (\*.csv)")  
 #  
 # if annotation\_file\_path:  
 # create\_annotation\_file\_func(annotation\_data, annotation\_file\_path) # Вызываем функцию с правильными аргументами  
  
 def get\_next\_zebra(self):  
 if self.dataset\_dir:  
 zebra\_generator = get\_next\_instance\_by\_class("zebra", self.dataset\_dir)  
 zebra\_instance = next(zebra\_generator, None)  
  
 if zebra\_instance:  
 pixmap = QPixmap(zebra\_instance)  
 self.image\_label.setPixmap(pixmap)  
 self.image\_label.show()  
 else:  
 self.image\_label.clear()  
 self.image\_label.hide()  
  
  
 def get\_next\_horse(self):  
 if self.dataset\_dir:  
 horse\_generator = get\_next\_instance\_by\_class("bay\_horse", self.dataset\_dir)  
 horse\_instance = next(horse\_generator, None)  
  
 if horse\_instance:  
 pixmap = QPixmap(horse\_instance)  
 self.image\_label.setPixmap(pixmap)  
 self.image\_label.show()  
 else:  
 self.image\_label.clear()  
 self.image\_label.hide()  
  
  
def main():  
 app = QApplication(sys.argv)  
 main\_window = MainWindow()  
 main\_window.show()  
 sys.exit(app.exec\_())  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

**Результат выполнения:**







**Код main.py**

import os  
import shutil  
import random  
import csv  
  
  
def create\_annotation\_file(dataset\_dir, project\_dir):  
 with open('annotations.csv', 'w', newline='') as annotation\_file:  
 writer = csv.writer(annotation\_file)  
  
 for root, dirs, files in os.walk(dataset\_dir):  
 for filename in files:  
 class\_name = os.path.basename(root)  
 absolute\_path = os.path.join(root, filename)  
 relative\_path = os.path.relpath(absolute\_path, project\_dir)  
  
 writer.writerow([absolute\_path, relative\_path, class\_name])  
  
  
def copy\_dataset\_with\_new\_names(dataset\_dir, output\_dir):  
 os.makedirs(output\_dir, exist\_ok=True)  
  
 with open('annotations.csv', 'w', newline='') as annotation\_file:  
 writer = csv.writer(annotation\_file)  
  
 for root, dirs, files in os.walk(dataset\_dir):  
 for filename in files:  
 class\_name = os.path.basename(root)  
 old\_filepath = os.path.join(root, filename)  
 new\_filename = f'{class\_name}\_{filename}'  
 new\_filepath = os.path.join(output\_dir, new\_filename)  
  
 writer.writerow([new\_filepath, new\_filename, class\_name])  
 shutil.copy(old\_filepath, new\_filepath)  
  
  
def create\_dataset\_with\_random\_names(dataset\_dir, output\_dir):  
 random\_names\_dir = output\_dir+"\\random\_names"  
 os.makedirs(random\_names\_dir, exist\_ok=True)  
  
 random\_names = [str(random.randint(0, 10000)) for \_ in range(10000)]  
  
 for root, dirs, files in os.walk(dataset\_dir):  
 for filename in files:  
 class\_name = os.path.basename(root)  
 old\_filepath = os.path.join(root, filename)  
 random\_name = random\_names.pop()  
 new\_filename = f'{random\_name}.jpg'  
 new\_filepath = os.path.join(random\_names\_dir, new\_filename)  
 shutil.copy(old\_filepath, new\_filepath)  
  
  
def get\_next\_instance\_by\_class(class\_label, dataset\_dir):  
 class\_dir = os.path.join(dataset\_dir, class\_label)  
 if not os.path.exists(class\_dir):  
 return None  
  
 files = os.listdir(class\_dir)  
 random.shuffle(files)  
  
 while len(files) > 0:  
 next\_instance = os.path.join(class\_dir, files.pop())  
 print(f"Следующий экземпляр класса '{class\_label}': {next\_instance}")  
  
 print(f"Все экземпляры класса '{class\_label}' закончились")  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 dataset\_dir = "D:\\Ucheba\\3 course\\PD\\lab2\\dataset"  
 project\_dir = "D:\\Ucheba\\3 course\\PD\\lab2"  
 output\_dir = "D:\\Ucheba\\3 course\\\PD\\lab3\\dataset"  
  
 create\_annotation\_file(dataset\_dir, project\_dir)  
 copy\_dataset\_with\_new\_names(dataset\_dir, output\_dir)  
 create\_dataset\_with\_random\_names(dataset\_dir, output\_dir)  
  
 class\_label = 'bay\_horse'  
 get\_next\_instance\_by\_class(class\_label, dataset\_dir)

**Результат выполнения**

