ДОДАТОК А

ТЕСТОВІ ЗОБРАЖЕННЯ

Рисунок А.1 – Приклад вхідного зображення № 1

Рисунок А.2 – Приклад вхідного зображення № 2

Рисунок А.3 – Приклад вхідного зображення № 3

ДОДАТОК Б

НАЙВАЖЛИВІШІ ЧАСТИНИ КОДУ ПРОГРАМИ

Лістинг Б.1 – Програмний код методу класифікації з хешуванням

```
    def mostPosibleDescriptorsClassesSearching(self, res,borders):
        self.mostPossibleDescriptorClasses = [0] * len(self.descriptors)
        self.bordersPositionsSearching( res,borders )

        counter = 0
        for i in range(0, len(self.stringDesctiptors)):
            self.oneCounts.append(self.onesCounting(i))

            startId = self.idBorderSearching(self.oneCounts[i], borders)
            endId = startId + 1
            startIdBorder = self.bordersPositions[startId]

            endIdBorder = self.bordersPositions[endId]

            hemmingDistance = self.maxDistance
            for j in range(startIdBorder, endIdBorder):
                hD = self.hemmingDistanceCounting(res[j][self.idDescriptor],
self.stringDesctiptors[i])
                if hD < hemmingDistance and hD < self.criticalDistance:
                    hemmingDistance = hD
                    self.mostPossibleDescriptorClasses[i] =
res[j][self.idIdbread]
            if hemmingDistance == self.maxDistance:
                counter += 1

    def idBorderSearching(self,onesCount,borders):
        for i in range(0,len(borders)):
            if borders[i] > onesCount:
                return i - 1
            elif borders[i] > onesCount:
                return i

    def onesCounting(self,idDescr):
        counter = 0
        for i in range(0, len(self.stringDesctiptors[idDescr])):
            if self.stringDesctiptors[idDescr][i] == '1':
                counter += 1
        return counter

    def hemmingDistanceCounting(self, str1, str2):
        distance = 0
        for i in range(0, len(str1)):
            if str1[i] != str2[i]:
                distance += 1
        return distance

    def mostPossibleClassSearching(self, breeds, res,borders):
        self.mostPosibleDescriptorsClassesSearching(res,borders)
        for i in range(0, len(breeds)):
            self.classes.append(breeds[i][0])
            counter = 0
```

Лістинг Б.2 – Програмний код методу класифікації з хешуванням (продовження)

```
            for j in range(0, len(self.mostPossibleDescriptorClasses)):
                if self.mostPossibleDescriptorClasses[j] == breeds[i][0]:
                    counter += 1
            self.possibilities.append(counter)
        self.orderPossibilities(breeds)
        if self.possibilities[0] >= self.criticalBreedIds:
            self.isRecognized = True

    def orderPossibilities(self, breeds):
        for k in range(0, len(self.possibilities) - 1):
            for i in range(0, len(self.possibilities) - k - 1):
                if self.possibilities[i] < self.possibilities[i + 1]:
                    self.possibilities[i], self.possibilities[i + 1] =
self.possibilities[i + 1], self.possibilities[i]
                    self.classes[i], self.classes[i + 1] = self.classes[i +
1], self.classes[i]
                    breed = breeds[i]
                    breeds[i] = breeds[i + 1]
                    breeds[i + 1] = breed
```

Лістинг Б.3 – Програмний код віднесення зображення до певного класу

```
def recognizeHashed(self,unrecognizedDog):
    res = self.hashedDescriptorsFromDataBase()

    breeds = self.breedsFromDB()
    borders = self.bordersFromDB()
    startTime = time.time()
    unrecognizedDog.mostPossibleClassSearching(breeds,res,borders)
    endTime = time.time()
    print("Время распознавания: ",round(endTime - startTime,1))
    if unrecognizedDog.isRecognized:

        unrecognizedDog.idBreed = unrecognizedDog.classes[0]
        self.recognizedDog = Dog.Dog(unrecognizedDog.idBreed,True)

    return self.recognizedDog,breeds
```