**Programming Exercise: Business P9.23**

**Problem statement:** Design and implement a class *Country* that stores the name of the country, its population, and its area. Then write a program that reads in a set of countries and prints: the country with the largest area, largest population, and the largest population density. The class should contain methods to answer the 3 questions using list classes, and then dictionary classes. In total there would be 6 methods: 3 for list classes, and 3 for dictionary classes.

**Step 1** Get an informal list of the responsibilities of the objects

The following tasks are to be accomplished:

**Reads and finds the country with the largest area (in a list and in a dictionary)**

**Reads and finds the country with the largest population (in a list and in a dictionary)**

**Reads and finds the country with the largest density (in a list and in a dictionary)**

**Step 2** Specify the public interface

The class has multiple class lists and class dictionaries to enable the storing of the country, population, area, and density.

\_countrylist = []

    \_arealist = []

    \_poplist = []

    \_denselist = []

    \_countrydictarea = {}

    \_countrydictpop = {}

    \_countrydictdense = {}

The constructor should accept three parameters: the country, its population, and its area.

def \_\_init\_\_(self, country, population, area):

        self.\_country = country

        self.\_population = population

        self.\_area = area

        self.\_density = self.\_population / self.\_area

        Country.\_arealist.append(self.\_area)

        Country.\_countrylist.append(self.\_country)

        Country.\_poplist.append(self.\_population)

        Country.\_denselist.append(self.\_density)

        Country.\_countrydictarea[self.\_country] = self.\_area

        Country.\_countrydictpop[self.\_country] = self.\_population

        Country.\_countrydictdense[self.\_country] = self.\_density

The class lists and class dictionaries are being accessed in the constructor.

There are multiple accessors in the class which will enable the class to read and find the country with the largest area, population, and density both in lists and dictionaries.

def largestArea\_list (cls)

def largestArea\_dict (cls):

def largestPop\_list (cls):

def largestPop\_dict (cls):

def largestDen\_list (cls):

def largestDen\_dict (cls):

**Step 3** Document the public interface

# Creates a class Country that stores the name of a country, its populations, and its area.

# This will have three parameters: country, population, and area

class Country:

    #Constructs a country class with three parameters.

    def \_\_init\_\_(self, country, population, area):

#a class method that will find the country with the largest area in a list

    @classmethod

    def largestArea\_list (cls):

#a class method that will find the country with the largest area in a dictionary

    @classmethod

    def largestArea\_dict (cls):

#a class method that will find the country with the largest population in a list

    @classmethod

    def largestPop\_list (cls):

#a class method that will find the country with the largest population in a dictionary

    @classmethod

    def largestPop\_dict (cls):

#a class method that will find the country with the largest density in a list

    @classmethod

    def largestDen\_list (cls):

#a class method that will find the country with the largest density in a dictionary

    @classmethod

    def largestDen\_dict (cls):

**Step 4** Determine instance variables

We need to store the country, population, area, and calculated density as instance variables.

def \_\_init\_\_(self, country, population, area):

        self.\_country = country

        self.\_population = population

        self.\_area = area

        self.\_density = self.\_population / self.\_area

**Step 5** Implement the constructor and methods

The constructor takes the three parameters and stores them in the instance variables for each of them. Class lists and dictionaries have been created too to enable storing of the parameters in either lists or dictionaries.

class Country:

    \_countrylist = []

    \_arealist = []

    \_poplist = []

    \_denselist = []

    \_countrydictarea = {}

    \_countrydictpop = {}

    \_countrydictdense = {}

def \_\_init\_\_(self, country, population, area):

        self.\_country = country

        self.\_population = population

        self.\_area = area

        self.\_density = self.\_population / self.\_area

        Country.\_arealist.append(self.\_area)

        Country.\_countrylist.append(self.\_country)

        Country.\_poplist.append(self.\_population)

        Country.\_denselist.append(self.\_density)

        Country.\_countrydictarea[self.\_country] = self.\_area

        Country.\_countrydictpop[self.\_country] = self.\_population

        Country.\_countrydictdense[self.\_country] = self.\_density

The constructor also accesses the class lists and dictionaries and appends each instance variable (area,population, country, and density) to the lists and dictionaries.

There are 6 methods that have been created for this class. Three of them reads and finds the country with the largest area, population, and density in a list. For methods concerning lists, the lists are zipped together to create a single list of the combination of country and area/population/density and will return the name of the country with the largest parameter.

@classmethod

    def largestArea\_list (cls):

        \_zipped = zip(cls.\_countrylist,cls.\_arealist)

        return max(\_zipped, key = lambda x: x[1])

@classmethod

    def largestPop\_list (cls):

        \_zipped = zip(cls.\_countrylist,cls.\_poplist)

        return max(\_zipped, key = lambda x: x[1])

@classmethod

    def largestDen\_list (cls):

        \_zipped = zip(cls.\_countrylist,cls.\_denselist)

        return max(\_zipped, key = lambda x: x[1])

The other three methods are the same but is reading and finding the largest parameters in a dictionary. This returns the **Country as the key**, and the **corresponding parameter as the value**.

@classmethod

    def largestArea\_dict (cls):

        return max(cls.\_countrydictarea, key = lambda x: cls.\_countrydictarea[x])

@classmethod

    def largestPop\_dict (cls):

        return max(cls.\_countrydictpop, key = lambda x: cls.\_countrydictpop[x])

#a class method that will find the country with the largest density in a dictionary

    @classmethod

    def largestDen\_dict (cls):

        return max(cls.\_countrydictdense, key = lambda x: cls.\_countrydictdense[x])

**Step 6** Test your class

Here is a simple tester program using argparse that exercises all methods.

if args.run\_demo:

        canada = Country("Canada",1200000,1500)

        philippines = Country("Philippines",1000000,1200)

        us = Country("USA",20000000,1000)

        largest\_area\_country\_list = Country.largestArea\_list()

        largest\_area\_country\_dict = Country.largestArea\_dict()

        print(largest\_area\_country\_list,largest\_area\_country\_dict)

        print("Expected: Canada")

        largest\_pop\_country\_list = Country.largestPop\_list()

        largest\_pop\_country\_dict = Country.largestPop\_dict()

        print(largest\_pop\_country\_list,largest\_pop\_country\_dict)

        print("Expected: USA")

        largest\_dense\_country\_list = Country.largestDen\_list()

        largest\_dense\_country\_dict = Country.largestDen\_dict()

        print(largest\_dense\_country\_list,largest\_dense\_country\_dict)

        print("Expected: USA")

**Program run**

A screen shot of a computer

Description automatically generated