На цьому етапі виконання курсової роботи, ми б хотіли описати абстрактні типи даних, які будуть потрібні нам під час вирішення обраної нами проблеми. Також ми представимо реалізацію описаних абстрактних типів даних та наведемо приклади їхнього використання з даними, які є максимально схожими до тих, які в подальшому будуть видобуванні з інтернету (це будуть коментарі, вручну скопійовані з інтернет ресурсів).

Першим абстрактним типом даних, який ми б хотіли використати – це коментар (Comment). Як поля, він буде мати текст самого відгука, число, яке буде характеризувати корисність цього повідомлення (як правило така інформація розташовується на інтернет сторінках, з яких ми і будемо видобувати всю інформацію), а також число, яке буде характеризувати оцінку якості шуканого продукту, така інформація за часту також знаходиться у мережі інтернет. За відсутності останньої інформації, ми спробуємо спираючись на коментарі, які мають заздалегідь визначену оцінку, визначати з деякою точністю оцінку якості коментаря, у якого ця інформація є невизначеною.

Методи, які будуть оголошені у цьому абстрактному класі, однак не будуть мати потрібної реалізації, а просто будуть збуджувати помилку під назвою NotImplementedError (тобто клас, який буде реалізовувати інтерфейс, який надає цей абстрактний тип даних повинен унаслідувати цей абстрактний тип даних та перевизначити всі його методи, для того щоб після створення екземпляра цього класу та виклику будь-якого доступного метода не відбувалася помилка NotImplementedError).

Перший метод при виконанні буде перевіряти чи оцінка якості товару, яка була залишена людиною, яка написала цей відгук, є близька до максимально. Якщо це так, то всі прикметники, які були використані у тексті будуть записані до спеціального сховища, яке ми опишемо пізніше. Другий метод, який буде оголошений у абстрактному типі даних, буде надзвичайно схожим до першого, однак перевірка буде відбуватися спираючись не на позитивність коментаря, а на його негативність, та відповідно так само всі прикметники будуть записуватися до спеціального сховища. І останнім буде метод, який при умові невизначеності оцінки продукту буде, опираючись на реалізацію другого описаного тут абстрактного класу, повертати число, яке б могло характеризувати відповідну ознаку.

Другий абстрактний тип даних, який ми будемо використовувати у нашій курсовій роботі – це такий собі контейнер, який буде в собі як поля мати послідовність позитивних прикметників, які були використанні при описах різних продуктів у коментарях, які мають дуже позитивну оцінку товару. Кожний прикметник буде зберігатися з відміткою часу, коли він був збережений у цей об'єкт, усі вони будуть збережені у чоловічому роді та до кожного прикметника буде прив'язано кількість його входжень у позитивні відгуки. Наголошуємо, що прикметники будуть видобуватися лише з коментарів, у яких рівень позитивності відгука є близьким до максимального.

Один з методів, який буде оголошений у цьому класі, це метод, реалізація якого повинна проходитися по всім позитивним прикметникам та негативним, шукати між ними однакові слова та в залежності від частоти появи цих слів у позитивних коментарях чи негативних, видаляти це слово або з двох списків або ж з одного. Другий метод буде приймати відгук та видобувати з нього усі прикметники (ми припускаємо, що коментарі пишуться українською або російською мовою).

Наступні два методи будуть приймати позитивні (негативні) прикметники та редагувати списки позитивних та негативних прикметників відповідно. Так як всі прикметники будуть розміщуватися у алфавітному порядку то перевірка на входження чи пошук місця для вставки елемента буде працювати за асимптотичною складністю О(logn). Зрозуміло, що такий підхід передбачає багаторазову вставку елементів не в кінець списку, тому було б доцільно скористатися зв’язною структурою даних, у якої ці операції є значно швидшими ніж у звичайному списку. Очевидно, що ці два методи є практично ідентичними, єдина різниця полягає в тому, що будуть опрацьовуватися різні списки. Тому, як на мене, доцільно було б у класі, який буде реалізовувати метод(скоріш за все статичний), який як вхід буде мати множину прикметників та список позитивних чи негативних прикметників, та буде або додавати прикметник, або збільшувати на один кількість його появ. Ще один метод буде приймати множину прикметників та повертати кортеж, перший елемент якого – це кількість прикметників, які є позитивними, а другий – кількість негативних (позитивність та негативність визначається присутністю прикметника у відповідному списку).

І останній абстрактний базовий клас як поле буде мати список коментарів та екземпляр класу, який реалізує вище згаданий абстрактний клас (“контейнер прикметників”). Буде мати метод, який повинен пройтися по всім необробленим коментарям та оновити контейнер та дати оцінку кожному коментарю, який має це поле невизначеним.

Ось абстрактні класи:

class Comment:  
 *""" Represents an abstract class. """* def \_\_init\_\_(self, text, comment\_characteristic=0, constructive=0):  
 *""" Initialize three fields. """* self.text = text  
 self.comment\_characteristic = comment\_characteristic  
 self.constructive = constructive  
  
 def parse\_positive\_adjectives(self, adjectives):  
 *""" Abstract method. """* raise NotImplementedError("It is subclass responsibility.")  
  
 def parse\_negative\_adjectives(self, adjectives):  
 *""" Abstract method. """* raise NotImplementedError("It is subclass responsibility.")  
  
 def define\_comment\_characteristic(self, positives, negatives):  
 *""" Abstract method. """* raise NotImplementedError("It is subclass responsibility.")  
  
  
class Comments:  
 *""" Represents an abstract class. """* def \_\_init\_\_(self, adjectives\_container):  
 *""" Initialize three fields. """* self.adjective\_container = adjectives\_container  
 self.pointer = 0  
 self.comments = []  
  
 def add\_comments(self, comments):  
 *""" Abstract method. """* raise NotImplementedError("It is subclass responsibility.")  
  
 def process\_new\_comments(self):  
 *""" Abstract method. """* raise NotImplementedError("It is subclass responsibility.")  
  
  
class AdjectivesContainer:  
 *""" Represents an abstract class. """* def \_\_init\_\_(self):  
 *""" Initialize two empty lists. """* self.positive\_adjectives = []  
 self.negative\_adjectives = []  
  
 def process\_all\_adjectives(self):  
 *""" Abstract method. """* raise NotImplementedError("It is subclass responsibility.")  
  
 def get\_all\_adjectives(self, text):  
 *""" Abstract method. """* raise NotImplementedError("It is subclass responsibility.")  
  
 def process\_positive\_adjectives(self, positives):  
 *""" Abstract method. """* raise NotImplementedError("It is subclass responsibility.")  
   
 def process\_negative\_adjectives(self, negatives):  
 *""" Abstract method. """* raise NotImplementedError("It is subclass responsibility.")

Ось реалізація цих же абстрактних класів:

from main1 import AdjectivesContainer, Comment, Comments  
from string import punctuation  
import copy  
  
punctuation += " "  
  
  
class CommentImplemented(Comment):  
 *""" Represents a comment object. """* def parse\_adjectives(self):  
 *""" object -> list """* result, current = [], ""  
 for letter in self.text:  
 if letter not in punctuation:  
 current += letter  
 else:  
 if current.endswith("ий") or current.endswith("ій"):  
 result.append(current)  
 elif current.endswith("ая"):  
 result.append(current[:-2] + "ий")  
 elif current.endswith("яя"):  
 result.append(current[:-2] + "ій")  
 elif current.endswith("е"):  
 result.append(current[:-1] + "ий")  
 current = ""  
 return result  
  
 def parse\_positive\_adjectives(self):  
 *""" object -> list """* if self.comment\_characteristic < 8:  
 raise TypeError  
 return self.parse\_adjectives()  
  
 def parse\_negative\_adjectives(self):  
 *""" object -> list """* if self.comment\_characteristic > 3:  
 raise TypeError  
 return self.parse\_adjectives()  
  
 def define\_comment\_characteristic(self, positives, negatives):  
 *""" list(list) -> None """* current = 0  
 adjectives = self.parse\_adjectives()  
 for word in positives:  
 if word[0] in adjectives:  
 current += word[1]  
 for word in negatives:  
 if word[0] in adjectives:  
 current -= word[1]  
 if current > 10:  
 self.comment\_characteristic = 10  
 elif current < -10:  
 self.comment\_characteristic = 0  
 else:  
 self.comment\_characteristic = (10 - current % 10) % 10  
  
  
class CommentsImplemented(Comments):  
 *""" Represnt comments object. """* def add\_comments(self, comments):  
 *""" list -> None """* if isinstance(comments, list):  
 self.comments.extend(comments)  
 else:  
 self.comments.append(comments)  
  
 def process\_new\_comments(self):  
 *""" None -> None """* length = len(self.comments)  
 for i in range(self.pointer, length):  
 if self.comments[i].comment\_characteristic > 7:  
 for element in self.comments[i].parse\_positive\_adjectives():  
 if element in self.adjective\_container.positive\_adjectives.keys():  
 self.adjective\_container.positive\_adjectives[  
 element] += 1  
 else:  
 self.adjective\_container.positive\_adjectives[  
 element] = 1  
 elif self.comments[i].comment\_characteristic < 4 and self.comments[  
 i].comment\_characteristic > 0:  
 for element in self.comments[i].parse\_negative\_adjectives():  
 if element in self.adjective\_container.negative\_adjectives.keys():  
 self.adjective\_container.negative\_adjectives[  
 element] += 1  
 else:  
 self.adjective\_container.negative\_adjectives[  
 element] = 1  
 self.pointer = length  
  
  
class AdjectivesContainerImplemented(AdjectivesContainer):  
 *""" Represents adjectives container object. """* def process\_all\_adjectives(self):  
 *""" None -> None """* temp\_possitve = copy.deepcopy(self.positive\_adjectives)  
 for good\_word in temp\_possitve:  
 if good\_word in self.negative\_adjectives:  
 if self.negative\_adjectives[good\_word] >= \  
 self.positive\_adjectives[good\_word]:  
  
 self.positive\_adjectives[good\_word] += \  
 self.negative\_adjectives[good\_word]  
 del self.negative\_adjectives[good\_word]  
 else:  
 self.negative\_adjectives[good\_word] += \  
 self.positive\_adjectives[good\_word]  
 del self.positive\_adjectives[good\_word]