# множества на основе дерева объектов

В данной лабораторной работе мы будем строить множества с использованием объектно-ориентированного подхода. Элементами множества в данной работе будут текстовые сообщения, твиты, каждый из которых описывается тремя свойствами: автором (пользователем), текстом сообщения (телом) и количеством ретвитов.

class Tweet(val user: String, val text: String, val retweets: Int) {

override def toString: String =

"User: " + user + "\n" +

"Text: " + text + " [" + retweets + "]"

}

Создаваемое множество организуется в виде бинарного дерева поиска. Сортировка твитов осуществляется по тексту сообщения (телу) с использованием стандартного лексикографического сравнения. Твиты, текст которых лексикографически меньше чем текст текущего узла дерева, помещаются в левое поддерево, твиты с большим значения тела – в правое.

Обратите внимание, что в лабораторной работе реализуется модель неизменяемых (иммутабельных) объектов: любые операции, изменяющие множество не должны менять текущий объект, а вместо этого должны возвращать ссылки на новое множество.

## Задание к лабораторной работе

Завершите описание классов в файле «TweetSet.scala». В файле определены абстрактный класс «TweetSet» и два его потомка:

* Empty, представляющего пустое множество;
* NonEmpty, непустой элемент двоичного дерева.

### Этап 1: Фильтрация данных

В первую очередь опишем функцию «filter», позволяющую осуществлять фильтрацию множеств. Она получает на вход предикат (функцию типа «Tweet => Boolean») и возвращает подмножество исходного множества, содержащее только твиты, удовлетворяющие условию предиката.

Реализация функции «filter» станет проще, если вначале вы завершите определение вспомогательной функции «filterAcc», получающей на вход тот же фильтрующий предикат и аккумулирующее множество, накапливающее результат фильтрации.

def filter(p: Tweet => Boolean): TweetSet

def filterAcc(p: Tweet => Boolean, acc: TweetSet): TweetSet

### Этап 2: Объединение множеств

Реализуйте объединение двух множеств, завершив описание функции «union» в классе «TweetSet». Метод получает на вход множество «that» и должен вернуть новое множество, являющееся объединением элементов текущего множества и множества «that».

def union(that: TweetSet): TweetSet

Прежде чем приступать к реализации метода, подумайте, в каком или в каких классах необходимо реализовывать данный метод. Должен он быть реализован в базовом классе «TweetSet» или он должен остаться там абстрактным?

### Этап 3: Сортировка на основе количества ретвитов

Количество ретвитов сообщения – это количество раз, которое другие пользователи (не исходный автор) повторили данное сообщение, возможно дополнив его некоторыми комментариями.

Целью данного этапа является построение на основе имеющегося множества линейного списка сообщений, отсортированных по убыванию количества ретвитов. Для этого нужно завершить описание метода «descendingByRetweet», описанного как

def descendingByRetweet: TweetList

Основная проблема построения такого списка в том, что исходное множество сообщений при хранении в дереве отсортировано не по количеству ретвитов, а по своему телу (тексту сообщения).

Идея построения списка заключается в том, что мы, начиная с пустого списка (класса Nil, наследника TweetList) ищем сообщение с максимальным количеством ретвитов (функция «mostRetweeted»), удаляем его из множества и добавляем в создаваемый список путем создания нового элемента списка типа «Cons». После чего повторяем алгоритм рекурсивно, но с множеством меньшего размера.

### Этап 4: Собирая воедино

Имея на руках все реализованные на предыдущих этапах функции можно приступить к основному заданию: выявлению наиболее значимых сообщений, появившихся в сети за последние дни. Исходные данные для исследования нам предоставит класс «TweetReader» (файла «TweetReader.scala») и его свойство «allTweets» с несколькими сотнями сообщений за несколько дней от самых популярных новостных ресурсов.

Вся дальнейшая работа по реализации этапа будет проходить внутри объекта «GoogleVsApple» (файл «TweetSet»). На первом шаге определим два множества сообщений: «googleTweets» и «appleTweets», – каждое из которых должно содержать только те сообщения, в которых содержаться ключевые слова, связанные с компаниями Google и Apple соответственно. Списки таких слов заданы заранее через значения (*val*) «google» и «apple» класса «GoogleVsApple».

Вторым шагом будет определение значения «trending» как сортированного по популярности (количеству ретвитов) списка, составленного из объединения множеств «googleTweets» и «appleTweets».

По полученному списку можно в конечном счете судить, какая из фирм давала тон новостям в последние дни.