REFACTORING

Clase QuestionRetriever >>retrieveQuestions:aUser

```
retrieveQuestions: aUser

        | qRet temp followingCol topicsCol newsCol popularTCol averageVotes|

        qRet := OrderedCollection new.

        option = #social ifTrue:[

                                followingCol := OrderedCollection new.

                                aUser following do:[ :follow | followingCol addAll: follow questions ].

                                temp := followingCol asSortedCollection:[ :a :b | a positiveVotes size > b
positiveVotes size ].

                                qRet := temp last: (100 min: temp size).

                        ].


        option = #topics ifTrue:[

                                topicsCol := OrderedCollection new.

                                aUser topics do:[ :topic | topicsCol addAll: topic questions ].

                                temp := topicsCol asSortedCollection:[ :a :b | a positiveVotes size > b
positiveVotes size ].

                                qRet := temp last: (100 min: temp size).

                        ].


        option = #news ifTrue:[

                                newsCol := OrderedCollection new.
                                cuoora questions do:[:q | (q timestamp asDate = Date today) ifTrue:
[newsCol add: q]].
                                temp := newsCol asSortedCollection:[ :a :b | a positiveVotes size > b
positiveVotes size ].
                                qRet := temp last: (100 min: temp size).
                        ].

        option = #popularToday ifTrue:[
                                popularTCol := OrderedCollection new.
                                cuoora questions do:[:q | (q timestamp asDate = Date today) ifTrue:
[popularTCol add: q]].
                                averageVotes := (cuoora questions sum: [:q | q positiveVotes size ]) /
popularTCol size.
                                temp := (popularTCol select:[:q | q positiveVotes size >= averageVotes ])
asSortedCollection:[ :a :b | a positiveVotes size > b positiveVotes size ].
                                qRet := temp last: (100 min: temp size).
                        ].

        ^qRet reject:[:q | q user = aUser].
```

Bad smells: Long Method, Código duplicado, demasiados Condicionales, demasiadas variables locales.

Solución: reemplazar condicionales con polimorfismo creando clase Padre QuestionRetriever y clases hijas con comportamientos respectivos.

Se realiza Repleace conditional with Polymorphism

Clase QuestionRetriever >>retrieveQuestions:aUser

1) Create Class SocialRetriever que hereda de QuestionRetriever
2) Create Class TopicsRetriever que hereda de QuestionRetriever

3) Create Class NewsRetriever que hereda de QuestionRetriever

4) Create Class PopularRetriever que hereda de QuestionRetriever

5) Pull up del Método retrieveQuestions:aUser de las clases hijas. A Clase padre QuestionRetriever. Convertir método retrieveQuestions:aUser a método abstracto.

Pull up Mehod

Clase QuestionRetriever >> retrieveQuestions:aUser

```
	^ self subclassResponisibility.
```

Clase SocialRetriever >> retrieveQuestions:aUser

Reemplazar temp por un query

```
	| followingQuestions |

	followingQuestions := aUser following flatCollect: [ :ea | ea questions ].

	^ (followingQuestions asSortedCollection: [ :q1 :q2  | q1 positiveVotes size > q2 positiveVotes
size]) last: (100 min: followingQuestions size).
```

Clase TopicsRetriever >> retrieveQuestions:aUser

Reemplazar temp por un query

```
	| topicQuestions |

	topicQuestions := aUser topics flatCollect: [ :t | t questions ].

	^ (topicQuestions asSortedColleection: [ :q1 :q2 | q1 positiveVotes size > q2 positiveVotes size])
last: (100 min: topicQuestions size).
```

Clase NewsRetriever >> retrieveQuestions:aUser

Reemplazar temp por un query

```
	| todayNews |

	todayNews := cuoora questions select :[:q | q timestamp asDate = Date today ].
```

```
    ^ (todayNews asSortedCollection: [:q1 :q2 | q1 positiveVotes size > q2 positiveVotes size]) last:
(100 min: todayNews size).
```

Clase PopularRetriever >> retrieveQuestions:aUser

Renombrar variable local temp -> aboveAverage

Renombrar variable local popularTCol -> todayNews

```
    | todayNews  averageVotes aboveAverage |

    todayNews :=  cuoora questions select :[ :q | q timestamp asDate = Date today ].

    averageVotes := (cuoora questions sum: [ :q | q positiveVotes size ]) / todayNews size.

    aboveAverage := (todayNews select:[:q | q positiveVotes size >= averageVotes ]).

    ^ (aboveAverage asSortedCollection: [:q1 :q2 | q1 positiveVotes size > q2 positiveVotes size ])
last: (100 min: aboveAverage size).
```

6) Crear método sortByPositives en clase padre QuestionRetriever

Clase QuestionRetriever >> sortByPositives:aCollection

```
    ^ aCollection asSortedCollection: [ :q1 :q2 | q1 positiveVotes size > q2 positiveVotes size ]
```

7) Crear método getLast100 en clase padre QuestionRetriever

Clase QuestionRetriever >> getLast100:aCollection

```
    ^ aCollection last: (100 min: aCollection size).
```

8) Crear método getFromToday en clase padre QuestionRetriever

Clase QuestionRetriever >> getFromToday:aCollection

```
    ^ aCollection select:[:ea | ea timestamp asDate = Date today].
```

9) Método reject:aUser from:aCollection para filtrar usuario de colección.

Clase QuestionRetriever>> reject:aUser from:aCollection

```
        ^ aCollection reject: [ :ea | ea user = aUser ].
```

9) User métodos **sortByPositives:aCollection** y el método **getLast100:aCollection** en clases hijas. (Replace inline code with function call).

Clase SocialRetriever >> retrieveQuestions:aUser

```
        | followingQuestions |

        followingQuestions := aUser following flatCollect: [ :ea | ea questions ].

        ^ (followingQuestions asSortedCollection: [ :q1 :q2  | q1 positiveVotes size > q2 positiveVotes
size]) last: (100 min: followingQuestions size).

        | followingQuestions |

        followingQuestions := aUser following flatCollect: [ :ea | ea questions ].

        ^ self getLast100: ( self sortByPositives: followingQuestions).
```

Clase TopicsRetriever >> retrieveQuestions:aUser

```
        | topicQuestions |

        topicQuestions := aUser topics flatCollect: [ :t | t questions ].

        ^ (topicQuestions asSortedColleection: [ :q1 :q2 | q1 positiveVotes size > q2 positiveVotes size])
last: (100 min: topicQuestions size).

        | topicQuestions |

        topicQuestions := aUser topics flatCollect: [ :t | t questions ].

        ^ self getLast100: (self sortByPositives: topicQuestions).
```

10) Usar además el método getFromToday:aCollection en las clases herederas que lo necesitan. (Replace inline code with function call).

Clase NewsRetriever >> retrieveQuestions:aUser

```
        | todayNews |
        todayNews := cuoora questions select :[:q | q timestamp asDate = Date today ].

        ^ (todayNews asSortedCollection: [:q1 :q2 | q1 positiveVotes size > q2 positiveVotes size]) last:
(100 min: todayNews size).

        | todayNews |

        todayNews := self getFromToday: cuoora questions.

        ^ self getLast100: (self sortByPositives: todayNews).
```

Clase PopularRetriever >> retrieveQuestions:aUser

```
        | todayNews  averageVotes aboveAverage |

        todayNews :=  cuoora questions select :[ :q | q timestamp asDate = Date today ].

        averageVotes := (cuoora questions sum: [ :q | q positiveVotes size ]) / todayNews size.

        aboveAverage := (todayNews select:[:q | q positiveVotes size >= averageVotes ]).

        ^ (aboveAverage asSortedCollection: [:q1 :q2 | q1 positiveVotes size > q2 positiveVotes size ])
last: (100 min: aboveAverage size).


        | todayNews  averageVotes aboveAverage |

        todayNews :=  self getFromToday: cuoora questions.

        averageVotes := (cuoora questions sum: [ :q | q positiveVotes size ]) / todayNews size.

        aboveAverage := (todayNews select:[:q | q positiveVotes size >= averageVotes ]).

        ^self  getLast100: ( self sortByPositives: aboveAverage).
```

11) Crear método **get:aCollection above:anAverage** en clase **PopularRetriever**

Clase PopularRetriever >> get:aCollection above:anAverage

```
        ^ aCollection select: [:q | q positiveVotes size >= average ].
```

12) Reemplazar en método retrieveQuestions:aUser (Replace inline code with function call)

Clase PopularRetriever >> retrieveQuestions:aUser

```
        | averageVotes |

        averageVotes := (cuoora questions sum: [:q | q positiveVotes size ]) / (self getFromToday: cuoora
questions) size.

        ^ self getLast100:( self sortByPositives: (self get: (cuoora questions) aboveAverage:
averageVotes) ).
```

13) Hay código duplicado entre dos métodos de la clase “Question” y “Answer”. Entonces se realiza Extract superclass a una clase padre Post, y se suben los métodos negativeVotes y positiveVotes.

También se realiza pull up field ya que ambas clases tienen variables de instancia iguales y son: timestamp user votes description

Extract Superclass y pull up field

Clase Post

V.I: ‘timestamp user votes description’

Clase Question>> positiveVotes

```
        ^ votes select: [:vote | vote isLike=true ].
```

Clase Question>>NegativeVotes

```
        ^ votes select: [:vote | vote isLike=false ].
```

Clase Answer>>NegativeVotes

```
        ^ votes select: [:vote | vote isLike=false ].
```

Clase Answer>> positiveVotes

```
^ votes select: [:vote | vote isLike=true ].
```

↓

Clase Post>> positiveVotes

```
        ^ votes select: [:vote | vote isLike ].
```

Clase Post>> negativeVotes

```
^ votes select: [:vote | vote isLike=false ].
```

14) Se realizó un Pull Up method de addVote:aVote

Clase Question>> addVote:aVote

```
votes add:aVote.
```

Clase Answer >> addVote:aVote

```
votes add: aVote.
```

↓

Clase Post >> addVote:aVote

```
votes add: aVote.
```

15) Dentro de los tests también se podría aplicar una jerarquía de clases.

Creación de Clase SocialRetrieverTest. Hija de QuestionRetrieverTest.

Push down de method: Clase QuestionRetriever>>testSocialRetrieveQuestions.

Renombrar método a testRetrieveQuestions.

Clase SocialRetrieverTest

V.i: socialRetriever

<<setUp>>

```
super setUp.

socialRetriever := SocialRetriever new: cuoora.
```

>>testRetrieveQuestions

```
self assert: (socialRetriever retrieveQuestions: user1) size equals: 1.

self assert: (socialRetriever retrieveQuestions: user1) first equals: questionUser3TopicOO2.
```

```
	self assert: (socialRetriever retrieveQuestions: user2) size equals: 0.

	self assert: (socialRetriever retrieveQuestions: user3) size equals: 0.

	questionUser2TopicOO2 := Question newWithTitle: 'Which bad smell...?' description: '' user:
user2 topic: topicOO2.

	questionUser2TopicOO2 addVote: (Vote user: user2 dislikesPublication:
questionUser2TopicOO2).

	cuoora addQuestion: questionUser2TopicOO2 forUser:user2.

	self assert: (socialRetriever retrieveQuestions: user1) size equals: 2.

	self assert: (socialRetriever retrieveQuestions: user1) last equals: questionUser2TopicOO2.

	self assert: (socialRetriever retrieveQuestions: user3) size equals: 0.

	self assert: (socialRetriever retrieveQuestions: user2) size equals: 0.
```

Creación de Clase TopicsRetrieverTest. Hija de QuestionRetrieverTest.

Push down de method: Clase QuestionRetriever>>testTopicsRetrieveQuestions.

Renombrar método a testRetrieveQuestions.

Clase TopicsRetrieverTest

V.i: topicsRetriever

<<setUp>>

```
	super setUp.

	topicslRetriever := TopicsRetriever new: cuoora.
```

>>testRetrieveQuestions

```
	self assert: (topicsRetriever retrieveQuestions: user1) size equals: 0.

	  self assert: (topicsRetriever retrieveQuestions: user2) size equals: 1.

	  self assert: (topicsRetriever retrieveQuestions: user3) size equals: 0.

	  questionUser2TopicOO2 := Question newWithTitle: 'Which bad smell...?' description: '' user:
	user2 topic: topicOO2.
```

```
cuoora addQuestion: questionUser2TopicOO2 forUser: user2.

self assert: (topicsRetriever retrieveQuestions: user2) size equals: 1.

self assert: (topicsRetriever retrieveQuestions: user3) size equals: 1.

self assert: (topicsRetriever retrieveQuestions: user3) first equals: questionUser2TopicOO2.

self assert: (topicsRetriever retrieveQuestions: user1) size equals: 0
```

Creación de Clase NewsRetrieverTest. Hija de QuestionRetrieverTest.

Push down de method: Clase QuestionRetriever>>testNewsRetrieveQuestions.

Renombrar método a testRetrieveQuestions.

Clase NewsRetrieverTest

V.i: newsRetriever

<<setUp>>

```
super setUp.

newsRetriever := NewsRetriever new: cuoora.
```

>>testRetrieveQuestions

```
self assert: (newsRetriever retrieveQuestions: user1) size equals: 1.

self assert: (newsRetriever retrieveQuestions: user1) last equals: questionUser3TopicOO2.

self assert: (newsRetriever retrieveQuestions: user2) size equals: 2.

self assert: (newsRetriever retrieveQuestions: user3) size equals: 1.

self assert: (newsRetriever retrieveQuestions: user3) last equals: questionUser1TopicOO1.

questionUser2TopicOO2 := Question

        newWithTitle: 'Which bad smell...?'

        description: ''

        user: user2

        topic: topicOO2.

cuoora addQuestion: questionUser2TopicOO2 forUser: user2.
```

```
    self assert: (newsRetriever retrieveQuestions: user1) size equals: 2.

    self assert: (newsRetriever retrieveQuestions: user2) size equals: 2.

    self assert: (newsRetriever retrieveQuestions: user3) size equals: 2.
```

Creación de Clase PopularRetrieverTest. Hija de QuestionRetrieverTest.

Push down de method: Clase QuestionRetriever>>testPopularRetrieveQuestions.

Renombrar método a testRetrieveQuestions.

Clase NewsRetrieverTest

V.i: popularTodayRetriever

<<setUp>>

```
    super setUp.

    popularTodayRetriever := PopularRetriever new: cuoora.
```

>>testRetrieveQuestions

```
    self

        popularTodayNoLikesTwoQuestions;

        popularTodayNoLikesThreeQuestions;

        popularTodayAtLeastOneVote;

        popularTodayFourQuestionsAtLeastOneVote;

        popularTodayTwoLikesFourQuestionsAtLeastOneVote;

        popularTodayFourLikesFourQuestionsAtLeastOneVote;

        popularTodayFiveLikesFourQuestionsAtLeastOneVote
```

16) **Clase PopularRetrieverTest**. Renombrar métodos. Nombres demasiado largos y difíciles de entender de qué se trata.

    popularTodayAtLeastOneVote -> atLeastOneVote popularTodayFiveLikesFourQuestionsAtLeastOneVote -> fiveLikesFourQuestionsAtLeastOneVote

popularTodayFourLikesFourQuestionsAtLeastOneVote ->fourLikesFourQuestionsAtLeastOneVote

popularTodayFourQuestionsAtLeastOneVote -> fourQuestionsAtLeastOneVote

popularTodayNoLikesThreeQuestions -> noLikesThreeQuestions

popularTodayNoLikesTwoQuestions -> noLikesTwoQuestions

popularTodayTwoLikesFourQuestionsAtLeastOneVote -> twoLikesFourQuestionsAtLeastOneVote

17) Se realiza push down field de la variable de instancia "cuoora" de la clase QuestionRetriever a las subclases PopularRetriever y NewsRetriever que son las que utilizan la variable"cuoora".

Push down field

QuestionRetriever subclass: NewsRetriever

instanceVariableNames: '**cuoora**'

QuestionRetriever subclass: PopularRetriever

instanceVariableNames: '**cuoora**'

18)

ya que la variable de instancia '**cuoora**' la utilizan solamente las clases PopularRetriever y QuestionRetriever se realiza push down method de la superclase QuestionRetriever

Push down method

Metodo de clase de QuestionRetriever

QuestionRetriever >> new: cuoora

^ self new: cuoora;yourself.

```
PopularRetriever>>new:cuoora

        ^ self new: cuoora;yourself.
```

```
PopularRetriever>>new:cuoora

        ^ self new: cuoora;yourself.
```

19) Encontramos redundante la relación Vote - User.

Clase User

Eliminación de métodos #addVote:aVote y #votes.

Eliminación de variable de instancia: ‘votes’.

20) Nos pareció pertinente mencionar sobre un posible bad smell Speculative Generality en la clase User:

Existen setters y getters de un QuestionRetriever y una variable questionRetriever que no son utilizadas ni llamadas en ningún momento. Entendemos que podrían ser utilizadas en un futuro.

21) Clase QuestionRetreiver. Eliminación de variables ‘options’, incluyendo el setter.