TRABAJO PRÁCTICO 3 - REFACTORING

- QuestionRetriever>>retrieveQuestion: aUser

Encontramos el Bad Smells **Long Method**, ya que el método tiene muchas líneas de código.
Utilizamos el Refactoring **Extract Method** para simplificarlo y dividirlo en partes.

```
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].
```

Código nuevo.
Se extrajo el código resaltado para dividirlo en cuatro métodos nuevos, los cuales quedaron:

```
comprobarSocial: aUser
    | qRet temp followingCol |
    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).
    ^ qRet
```

```
comprobarTopics: aUser
    | qRet temp topicsCol |
    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).
    ^ qRet
```

```
comprobarNews: aUser
    | temp newsCol qRet |
    newsCol := OrderedCollection new.
    newsCol := cuoora questionsFromToday .
    temp := newsCol
            asSortedCollection: [ :a :b | a positiveVotes size > b positiveVotes size ].
    qRet := temp last: (100 min: temp size).
    ^ qRet
```

```
comprobarPopularToday: aUser
    | qRet popularTCol averageVotes temp |
    popularTCol := OrderedCollection new.
    popularTCol := cuoora questionsFromToday.
    averageVotes := cuoora sumAllPositiveVotes / popularTCol size.
    temp := (popularTCol
            select: [ :question | question positiveVotes size >= averageVotes ])
            asSortedCollection: [ :a :b | a positiveVotes size > b positiveVotes size ].
    qRet := temp last: (100 min: temp size).
    ^ qRet
```

- QuestionRetriever>>comprobarPopularToday: aUser

Tenemos código procedural.
Debemos usar métodos de la clase Collection (select:, collect:, sum:, etc)

```
cuoora questions
        do: [ :q |
                q timestamp asDate = Date today
                        ifTrue: [ popularTCol add: q ] ].
```

Código nuevo

```
popularTCol := cuoora questions
                select: [ :question | question timestamp asDate = Date today ].
```

- QuestionRetriever>>comprobarNews: aUser

Tenemos código procedural.
Debemos usar métodos de la clase Collection (select:, collect:, sum:, etc)

```
cuoora questions
        do: [ :q |
                q timestamp asDate = Date today
                        ifTrue: [ newsCol add: q ] ].
```

Código nuevo

```
newsCol := cuoora questions
                select: [ :question | question timestamp asDate = Date today ].
```

- QuestionRetriever>>comprobarNews: aUser

Responsabilidad mal asignada: QuestionRetriever le pide el atributo questions a CuOOra e itera sobre el.
Esto se conoce como envidia de atributos.
Además, es código duplicado (**Duplicate Code**) (en el método #comprobarPopularToday)
Mover el método a la clase CuOOra: Debemos delegar esta parte del código a CuOOra.

```
newsCol := cuoora questions
                select: [ :question | question timestamp asDate = Date today ].
```

- QuestionRetriever>>comprobarPopularToday: aUser

Responsabilidad mal asignada: QuestionRetriever le pide el atributo questions a CuOOra e itera sobre el.
Esto se conoce como envidia de atributos.
Además, es código duplicado (**Duplicate Code**) (en el método #comprobarNews)
Mover el método a la clase CuOOra: Debemos delegar esta parte del código a CuOOra.

```
popularTCol := cuoora questions
                select: [ :question | question timestamp asDate = Date today ].
```

Para solucionar ambos problemas, hacemos un extract method y move method a la clase CuOOra.

```
CuOOra >> questionsFromToday
        ^questions select: [ :question | question timestamp asDate = Date today ] .
```

Y lo usamos en ambos métodos.

```
QuestionRetriever >> comprobarPopularToday
        ...
        popularTCol := cuoora questionsFromToday.
        ...
```

```
QuestionRetriever >> comprobarNews
        ...
        newsCol := cuoora questionsFromToday .
        ...
```

- QuestionRetriever>>comprobarPopularToday: aUser

Responsabilidad mal asignada: QuestionRetriever le pide el atributo questions a CuOOra e itera sobre el.
Esto se conoce como envidia de atributos.

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

Debemos hacer un **extract method** y luego **move method** a la clase CuOOra.

```
CuOOra >> sumAllPositiveVotes
        ^questions sum: [ :q | q positiveVotes size ]
```

```
QuestionRetriever >> comprobarPopularToday
        ...
        averageVotes := cuoora sumAllPositiveVotes / popularTCol size.
        ...
```

- QuestionRetriever>>retrieveQuestions

Tenemos varios condicionales innecesarios.
Técnica: Reemplazar el condicional con polimorfismo (**Replace conditional with Polymorphism**).

```
QuestionRetriever >> retrieveQuestions: aUser
        | qRet |
        qRet := OrderedCollection new.
        option = #social
                ifTrue: [ qRet := self comprobarSocial: aUser ].
        option = #news
                ifTrue: [ qRet := self comprobarNews: aUser ].
        option = #topics
                ifTrue: [ qRet := self comprobarTopics: aUser ].
        option = #popularToday
```

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

Hacemos 4 nuevas clases, cada una representando a una opción. Todas tienen a QuestionRetriever como superclase.

```
Nuevas Clases creadas:
QuestionRetriever subclass: #NewsRetriever
QuestionRetriever subclass: #PopularTodayRetriever
QuestionRetriever subclass: #SocialRetriever
QuestionRetriever subclass: #TopicsRetriever
```

```
QuestionRetriever >> retrieveQuestions: aUser
        | qRet |
        qRet := OrderedCollection new.
        qRet := self retrieveFromUser: aUser.
        ^ qRet reject: [ :q | q user = aUser ]
```

```
QuestionRetriever >> retrieveFromUser: aUser
        self subclassResponsibility .
```

Ahora hacemos 4 **Move Method**. Al mismo tiempo hacemos un **Rename Method** de cada uno.
Desde QuestionRetriever a sus subclases.

```
QuestionRetriever >> comprobarNews: aUser
        | temp newsCol qRet |
        newsCol := OrderedCollection new.
        newsCol := cuoora questionsFromToday .
        temp := newsCol
                asSortedCollection: [ :a :b | a positiveVotes size > b positiveVotes size ].
        qRet := temp last: (100 min: temp size).
        ^ qRet

NewsRetriever >> retrieveFromUser: aUser
        | temp newsCol qRet |
        newsCol := OrderedCollection new.
        newsCol := cuoora questionsFromToday .
        temp := newsCol
                asSortedCollection: [ :a :b | a positiveVotes size > b positiveVotes size ].
        qRet := temp last: (100 min: temp size).
        ^ qRet
```

```
QuestionRetriever >> comprobarPopularToday: aUser
        | qRet popularTCol averageVotes temp |
        popularTCol := OrderedCollection new.
        popularTCol := cuoora questionsFromToday.
        averageVotes := cuoora sumAllPositiveVotes / popularTCol size.
        temp := (popularTCol
                select: [ :question | question positiveVotes size >= averageVotes ])
                asSortedCollection: [ :a :b | a positiveVotes size > b positiveVotes size ].
```

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

PopularTodayRetriever >> retrieveFromUser: aUser
        | qRet popularTCol averageVotes temp |
        popularTCol := OrderedCollection new.
        popularTCol := cuoora questionsFromToday.
        averageVotes := cuoora sumAllPositiveVotes / popularTCol size.
        temp := (popularTCol
                select: [ :question | question positiveVotes size >= averageVotes ])
                asSortedCollection: [ :a :b | a positiveVotes size > b positiveVotes size ].
        qRet := temp last: (100 min: temp size).
        ^ qRet
```

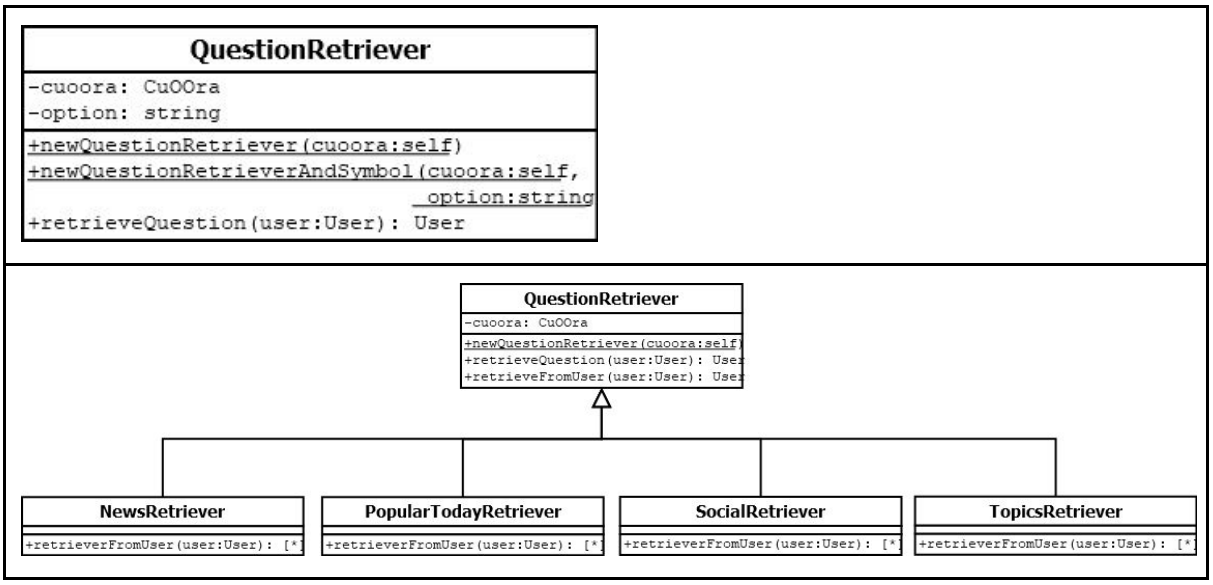
```
QuestionRetriever >> comprobarSocial: aUser
        | qRet temp followingCol |
        followingCol := OrderedCollection new.
        aUser following
                do: [ :follow | followingCol addAll: follow questions ].
        temp := followingCol
                asSortedCollection: [ :a :b | a positiveVotes size > b positiveVotes size ].
        qRet := temp last: (100 min: temp size).
        ^ qRet

SocialRetriever >> RetrieveFromUser: aUser
        | qRet temp followingCol |
        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).
        ^ qRet
```

```
QuestionRetriever >> comprobarTopics: aUser
        | qRet temp topicsCol |
        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).
        ^ qRet

TopicsRetriever >> retrieveFromUser: aUser
        | qRet temp topicsCol |
        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).
        ^ qRet
```

A modo de dejar una mejor visualización, mostramos el antes y el después a través de un diagrama UML:



Para testear estos cambios y asegurarnos de que no alteran el comportamiento, tuvimos que hacer cambios en los Tests de QuestionRetriever, en el método setUp

```
QuestionRetrieverTest >> setUp
        [...]
        socialRetriever := QuestionRetriever new: cuoora and: #social.
        topicsRetriever := QuestionRetriever new: cuoora and: #topics.
        newsRetriever := QuestionRetriever new: cuoora and: #news.
        popularTodayRetriever := QuestionRetriever new: cuoora and: #popularToday.
```

```
QuestionRetrieverTest >> setUp
        [...]
        socialRetriever := SocialRetriever new: cuoora.
        topicsRetriever := TopicsRetriever new: cuoora.
        newsRetriever := NewsRetriever new: cuoora.
        popularTodayRetriever := PopularTodayRetriever new: cuoora.
```

La variable de instancia “option” (de QuestionRetriever) ya no tiene sentido, la podemos eliminar (**Dead Code**).

- En las subclases de QuestionRetriever, en sus metodos retrieveFromUser:

Encontramos código duplicado (**Duplicated Code**).
Por ejemplo, la siguiente instrucción:

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

Se repite en cada uno de estos metodos
NewsRetriever >> retrieveFromUser: aUser
PopularTodayRetriever >> retrieveFromUser: aUser
SocialRetriever >> retrieveFromUser: aUser

TopicsRetriever >> retrieveFromUser: aUser

Como todas esas clases son subclases de QuestionRetriever y este método es siempre llamado por QuestionRetriever >> retrieveQuestions: aUser,
Podemos llevar esta instrucción al método que está “arriba” (**Pull Up Method**).

```
QuestionRetriever >> retrieveQuestions: aUser
        | qRet temp |
        qRet := OrderedCollection new.
        temp := self retrieveFromUser: aUser.
        qRet := temp last: (100 min: temp size).
        ^ qRet reject: [ :q | q user = aUser ]
```

Entonces, esta instrucción la podemos borrar de los 4 métodos que la repetían

```
NewsRetriever >> retrieveFromUser: aUser
        | temp newsCol qRet |
        newsCol := OrderedCollection new.
        newsCol := cuoora questionsFromToday .
        temp := newsCol
                asSortedCollection: [ :a :b | a positiveVotes size > b positiveVotes size ].
        qRet := temp last: (100 min: temp size).
        ^ qRet
```

```
PopularTodayRetriever >> retrieveFromUser: aUser
        | qRet popularTCol averageVotes temp |
        popularTCol := OrderedCollection new.
        popularTCol := cuoora questionsFromToday.
        averageVotes := cuoora sumAllPositiveVotes / popularTCol size.
        temp := (popularTCol
                select: [ :question | question positiveVotes size >= averageVotes ])
                asSortedCollection: [ :a :b | a positiveVotes size > b positiveVotes size ].
        qRet := temp last: (100 min: temp size).
        ^ qRet
```

```
SocialRetriever >> retrieveFromUser: aUser
        | qRet temp followingCol |
        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).
        ^ qRet
```

```
TopicsRetriever >> retrieveFromUser: aUser
        | qRet temp topicsCol |
        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).
        ^ qRet
```

- NewsRetriever >> retrieveFromUser:

Tenemos variables locales innecesarias (**Long Method**).
Tanto “newsCol” como “qRet” son variables con un mal nombre y que pueden ser eliminadas completamente ya que no son necesarias.

```
NewsRetriever >> retrieveFromUser: aUser
        | newsCol qRet |
        newsCol := OrderedCollection new.
        newsCol := cuoora questionsFromToday .
        qRet := newsCol
                asSortedCollection: [ :a :b | a positiveVotes size > b positiveVotes size ].
        ^ qRet
```

```
NewsRetriever >> retrieveFromUser: aUser
        ^ cuoora questionsFromToday
                asSortedCollection: [ :a :b | a positiveVotes size > b positiveVotes size ]
```

- PopularTodayRetriever >> retrieveFromUser:

Tenemos variables locales innecesarias (**Long Method**).
Tanto “popularTCol” como “qRet” son variables con un mal nombre y que pueden ser eliminadas completamente ya que no son necesarias.
La variable “averageVotes” nos parece importante ya que aporta legibilidad.

```
PopularTodayRetriever >> retrieveFromUser: aUser
        | qRet popularTCol averageVotes |
        popularTCol := OrderedCollection new.
        popularTCol := cuoora questionsFromToday.
        averageVotes := cuoora sumAllPositiveVotes / popularTCol size.
        qRet := (popularTCol
                select: [ :question | question positiveVotes size >= averageVotes ])
                asSortedCollection: [ :a :b | a positiveVotes size > b positiveVotes size ].
        ^ qRet
```

```
PopularTodayRetriever >> retrieveFromUser: aUser
        | averageVotes |
        averageVotes := cuoora sumAllPositiveVotes / cuoora questionsFromToday size .
        ^(cuoora questionsFromToday
                select: [ :question | question positiveVotes size >= averageVotes ])
                asSortedCollection: [ :a :b | a positiveVotes size > b positiveVotes size ].
```

- SocialRetriever >> retrieveFromUser:

Tenemos variables locales innecesarias (**Long Method**).
Tanto “followingCol” como “qRet” pueden ser confusas y son totalmente innecesarias.

La variable qRet puede ser eliminada totalmente.

```
SocialRetriever >> retrieveFromUser: aUser
	| qRet followingCol |
	followingCol := OrderedCollection new.
	aUser following
		do: [ :follow | followingCol addAll: follow questions ].
	qRet := followingCol
		asSortedCollection: [ :a :b | a positiveVotes size > b positiveVotes size ].
	^ qRet
```

```
SocialRetriever >> retrieveFromUser: aUser
	| followingCol |
	followingCol := OrderedCollection new.
	aUser following
		do: [ :follow | followingCol addAll: follow questions ].
	^followingCol
		asSortedCollection: [ :a :b | a positiveVotes size > b positiveVotes size ].
```

Sin embargo, al querer borrar la variable “followingCol” nos damos cuenta que hay otros errores más graves en la siguiente instrucción:

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

- Código Procedural: Usa el mensaje do: en vez de usar un mensaje apropiado para el caso.
- Envidia de atributos: Itera sobre un atributo de la clase Usuario (following)
- Responsabilidad mal asignada

Para resolver la envidia de atributos y la responsabilidad mal asignada, deberíamos extraer el método (**Extract Method**) y llevarlo a la clase User.
Para resolver el Código Procedural, usamos un flatCollect: en lugar del do:

Extraemos el método hacia la clase User, agregando el siguiente método

```
User >> questionsFromFollowedUsers
	^self following flatCollect: [ :user | user questions ].
```

Y reemplazamos en el origen

```
SocialRetriever >> retrieveFromUser: aUser
	| followingCol |
	followingCol := OrderedCollection new.
	followingCol := aUser questionsFromFollowedUsers .
	^followingCol
		asSortedCollection: [ :a :b | a positiveVotes size > b positiveVotes size ].
```

Ahora podemos borrar la variable “followingCol”

```
SocialRetriever >> retrieveFromUser: aUser
	| followingCol |
	followingCol := OrderedCollection new.
	followingCol := aUser questionsFromFollowedUsers .
```

```
        ^followingCol
                asSortedCollection: [ :a :b | a positiveVotes size > b positiveVotes size ].
```

```
retrieveFromUser: aUser
        ^aUser questionsFromFollowedUsers
                asSortedCollection: [ :a :b | a positiveVotes size > b positiveVotes size ].
```

- TopicsRetriever >> retrieveFromUser:

Tenemos variables locales innecesarias (**Long Method**).
Tanto “topicsCol” como “qRet” pueden ser confusas y son totalmente innecesarias.

La variable qRet puede ser eliminada totalmente.

```
retrieveFromUser: aUser
        | qRet topicsCol |
        topicsCol := OrderedCollection new.
        aUser topics do: [ :topic | topicsCol addAll: topic questions ].
        qRet := topicsCol
                asSortedCollection: [ :a :b | a positiveVotes size > b positiveVotes size ].
        ^ qRet
```

```
retrieveFromUser: aUser
        | topicsCol |
        topicsCol := OrderedCollection new.
        aUser topics do: [ :topic | topicsCol addAll: topic questions ].
        ^topicsCol
                asSortedCollection: [ :a :b | a positiveVotes size > b positiveVotes size ].
```

Sin embargo, al querer eliminar la variable “topicsCol”, nos damos cuenta que hay otros errores más graves en la siguiente instrucción

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

- Código Procedural: Usa el mensaje do: en vez de usar un mensaje apropiado para el caso.
- Envidia de atributos: Itera sobre un atributo de la clase Usuario (topics)
- Responsabilidad mal asignada

Para resolver la envidia de atributos y la responsabilidad mal asignada, deberíamos extraer el método (**Extract Method**) y llevarlo a la clase User.
Para resolver el Código Procedural, usamos un flatCollect: en lugar del do:

Extraemos el método hacia la clase User, agregando el siguiente método

```
User >> questionsFromFollowedTopics
        ^self topics flatCollect: [ :topic | topic questions ].
```

Ahora reemplazamos en el origen

```
retrieveFromUser: aUser
        | topicsCol |
```

```
        topicsCol := OrderedCollection new.
        topicsCol := aUser questionsFromFollowedTopics .
        ^topicsCol
                asSortedCollection: [ :a :b | a positiveVotes size > b positiveVotes size ].
```

Ahora podemos eliminar la variable “topicsCol”

```
retrieveFromUser: aUser
        | qRet topicsCol |
        topicsCol := OrderedCollection new.
        topicsCol := aUser questionsFromFollowedTopics .
        qRet := topicsCol
                asSortedCollection: [ :a :b | a positiveVotes size > b positiveVotes size ].
        ^ qRet
```

```
retrieveFromUser: aUser
        ^aUser questionsFromFollowedTopics
                asSortedCollection: [ :a :b | a positiveVotes size > b positiveVotes size ].
```

- Subclases de QuestionRetriever

Notamos una sentencia repetida (**Duplicate Code**):

```
retrieveFromUser: aUser
        ^ (...)
                asSortedCollection: [ :a :b | a positiveVotes size > b positiveVotes size ].
```

En los métodos:
NewsRetriever >> retrieveFromUser:
PopularTodayRetriever >> retrieveFromUser:
SocialRetriever >> retrieveFromUser:
TopicsRetriever >> retrieveFromUser:

Podemos llevar dicha instrucción un escalón más arriba (**Pull Up Method**).
A la clase QuestionRetriever

```
QuestionRetriever >> retrieveQuestions: aUser
        | qRet temp |
        qRet := OrderedCollection new.
        temp := self retrieveFromUser: aUser.
        qRet := temp last: (100 min: temp size).
        ^ qRet reject: [ :q | q user = aUser ]
```

```
retrieveQuestions: aUser
        | qRet temp |
        qRet := OrderedCollection new.
        temp := (self retrieveFromUser: aUser)
                asSortedCollection: [ :a :b | a positiveVotes size > b positiveVotes size ].
        qRet := temp last: (100 min: temp size).
        ^ qRet reject: [ :q | q user = aUser ]
```

Entonces ahora podemos eliminar dicha sentencia de cada una de las subclases

```
NewsRetriever >> retrieveFromUser: aUser
	^ cuoora questionsFromToday
		asSortedCollection: [ :a :b | a positiveVotes size > b positiveVotes size ]
```

```
NewsRetriever >> retrieveFromUser: aUser
	^ cuoora questionsFromToday
```

```
PopularTodayRetriever >> retrieveFromUser: aUser
	| averageVotes |
	averageVotes := cuoora sumAllPositiveVotes / cuoora questionsFromToday size .
	^(cuoora questionsFromToday
		select: [ :question | question positiveVotes size >= averageVotes ])
		asSortedCollection: [ :a :b | a positiveVotes size > b positiveVotes size ].
```

```
PopularTodayRetriever >> retrieveFromUser: aUser
	| averageVotes |
	averageVotes := cuoora sumAllPositiveVotes / cuoora questionsFromToday size .
	^(cuoora questionsFromToday
		select: [ :question | question positiveVotes size >= averageVotes ])
```

```
SocialRetriever >> retrieveFromUser: aUser
	^aUser questionsFromFollowedUsers
		asSortedCollection: [ :a :b | a positiveVotes size > b positiveVotes size ].
```

```
SocialRetriever >> retrieveFromUser: aUser
	^aUser questionsFromFollowedUsers
```

```
TopicsRetriever >> retrieveFromUser: aUser
	^aUser questionsFromFollowedTopics
		asSortedCollection: [ :a :b | a positiveVotes size > b positiveVotes size ].
```

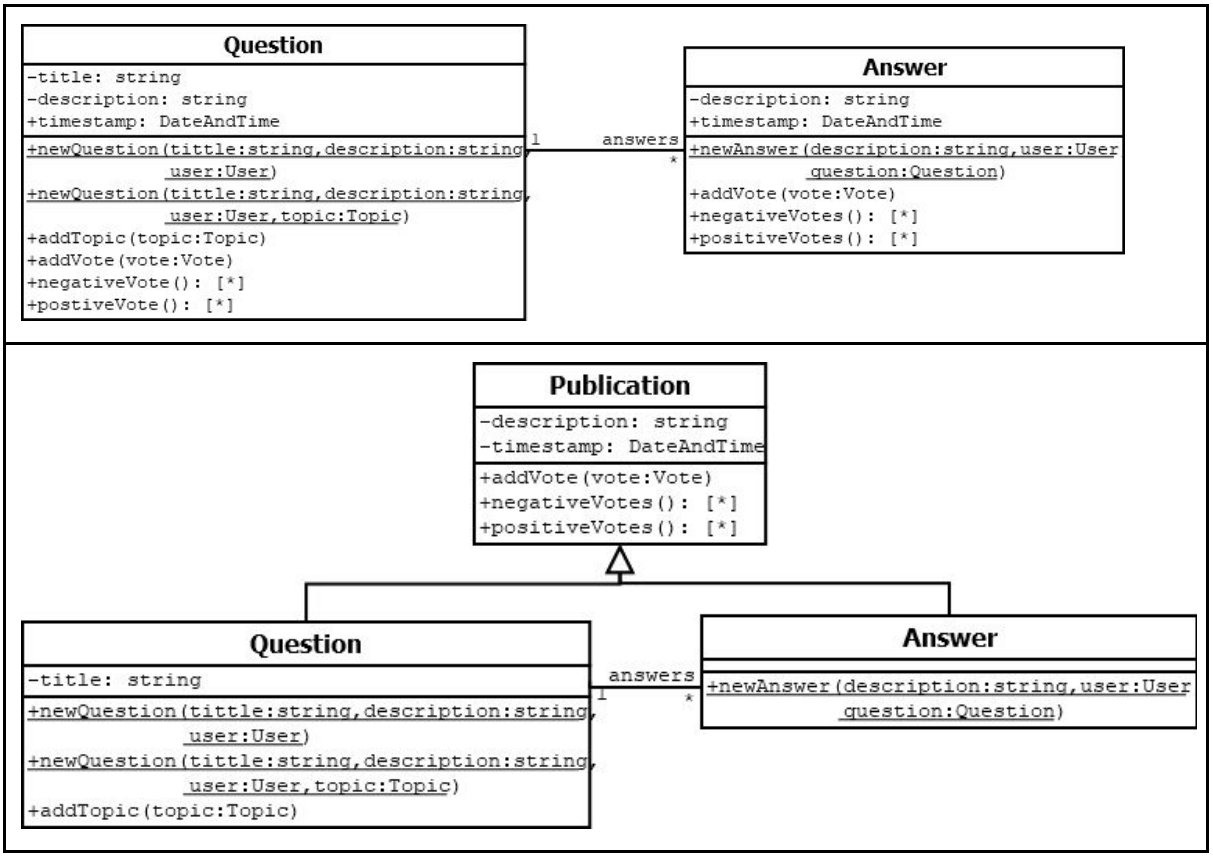
```
TopicsRetriever >> retrieveFromUser: aUser
	^aUser questionsFromFollowedTopics.
```

- Question>>addVote: aVote , negativeVotes() , positiveVotes() , setters y getters.
- Answer>>addVote: aVote, negativeVotes() , positiveVotes() , setters y getters.

Encontramos el Bad Smells **Duplicate Code**, ya que estos métodos hacen exactamente lo mismo.

Utilizamos el Refactoring **Pull Up Method**. Se creó una jerarquía de clases, con una clase padre llamada Publication y se “subieron” los métodos duplicados a esta misma. Se realizó el mismo procedimiento con las variables duplicadas (description, user, timestamp y la colección votes).

A modo de verlo mejor mostramos el antes y el después con un diagrama UML.



- Publication>>negativeVotes
- Publication>>positiveVotes

Tenemos código procedural.

Debemos usar métodos de la clase Collection (select:, collect:, sum:, etc).

```
negativeVotes
    | r |
    r := OrderedCollection new.
    votes
        do: [ :vote |
            vote isLike
                ifFalse: [ r add: vote ] ].
    ^ r

negativeVotes
        | r |
        r := OrderedCollection new.
        r:= votes select: [ :vote | vote isLike = false ].
        ^ r
```

```
positiveVotes
    | r |
    r := OrderedCollection new.
    votes
        do: [ :vote |
```

```
                    vote isLike
                            ifTrue: [ r add: vote ] ].
    ^ r

positiveVotes
        | r |
        r := OrderedCollection new.
        r := votes select: [ :vote | vote isLike ].
        ^ r
```

A modo de simplificar el código, se eliminaron las líneas

```
negativeVotes
        | r |
        r := OrderedCollection new.
        r:= votes select: [ :vote | vote isLike = false ].
        ^ r
```

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

```
positiveVotes
        | r |
        r := OrderedCollection new.
        r := votes select: [ :vote | vote isLike ].
        ^ r
```

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

- De acuerdo al documento de Criterios y Heurísticas brindado por la Cátedra, encontramos un punto que señala que “**Los nombres de las variables deben indicar su rol**.”, tomando esto como referencia encontramos algunas variables poco expresivas y las modificamos, las mismas se detallan aca abajo.

```
CuOOra>>sumAllPositiveVotes
    ^ questions sum: [ :q | q positiveVotes size ]

CuOOra>>sumAllPositiveVotes
    ^ questions sum: [ :question | question positiveVotes size ]
```

```
CuOOra>>questionsFromToday
    ^ questions
            select: [ :q | q timestamp asDate = Date today ]
```

```
CuOOra>>questionsFromToday
    ^ questions
            select: [ :question | question timestamp asDate = Date today ]
```

- QuestionRetriever >> retrieveQuestions:

Tiene variables locales que parecen innecesarias y son poco descriptivas: “qRet” y “temp”.
“qRet” la eliminamos completamente.
“temp” elegimos renombrarla, porque si bien se puede eliminar y reducir aún más el código, quedaría un muy poco legible.

```
QuestionRetriever >> retrieveQuestions: aUser
        | qRet temp |
        temp := (self retrieveFromUser: aUser)
                asSortedCollection: [ :a :b | a positiveVotes size > b positiveVotes size ].
        qRet := temp last: (100 min: temp size).
        ^ qRet reject: [ :q | q user = aUser ]
```

```
retrieveQuestions: aUser
        | retrievedQuestions |
        retrievedQuestions := (self retrieveFromUser: aUser)
                asSortedCollection: [ :a :b | a positiveVotes size > b positiveVotes size ].
        ^ (retrievedQuestions last: (100 min: retrievedQuestions size))
                reject: [ :q | q user = aUser ]
```