Answer y Question

Se observa `Codigo Duplicado` en varios metodos de las clases `Answer` y `Question`.
Para solucionarlo vamos a aplicar `Extract Superclass`. Creamos una nueva clase: `Publication` y de esta, heredaran las dos clases anteriores.

```
Object subclass: #Publication
    instanceVariableNames: ''
    classVariableNames: ''
    package: 'TP-Refactoring-Model'
```

```
Publication subclass: #Question
    instanceVariableNames: 'title answers topics timestamp user votes description'
    classVariableNames: ''
    package: 'TP-Refactoring-Model'
```

```
Publication subclass: #Answer
    instanceVariableNames: 'question timestamp user votes description'
    classVariableNames: ''
    package: 'TP-Refactoring-Model'
```

Identificamos las variables de instancias en comun: `timestamp`, `user`, `votes` y `description` y realizamos un `Pull Up Field` de estas.

```
Object subclass: #Publication
    instanceVariableNames: 'timestamp user votes description'
    classVariableNames: ''
    package: 'TP-Refactoring-Model'
```

```
Publication subclass: #Answer
    instanceVariableNames: 'question'
    classVariableNames: ''
    package: 'TP-Refactoring-Model'
```

```
Publication subclass: #Question
    instanceVariableNames: 'title answers topics'
    classVariableNames: ''
    package: 'TP-Refactoring-Model'
```

Identificamos los metodos en comun: `#addVote:`, `#description`, `#description:`,`#negativeVotes`, `#positiveVotes`, `#timestamp`, `#timestamp:`, `#user`, `#user:` y `#votes` y realizamos un `Pull Up Method` de estos.

Se puede observar aun, codigo duplicado en el metodo `#initialize`.

```
Answer>>initialize
    votes := OrderedCollection new.
    timestamp := DateAndTime now
```

```
Question>>initialize
    answers := OrderedCollection new.
    topics := OrderedCollection new.
    votes := OrderedCollection new.
    timestamp := DateAndTime now
```

Aplicamos un `Pull Up Method` en el metodo `Answer>>#intialize` y una pequena modificacion en `Question>>#initialize`

```
Publication>>initialize
    votes := OrderedCollection new.
    timestamp := DateAndTime now
```

```
Question>>initialize
    super initialize.
    answers := OrderedCollection new.
    topics := OrderedCollection new
```

Publication

negativeVotes / positiveVotes

Se observa que estos metodos presentan `Codigo duplicado` , y uso de variables temporales innecesarias.

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

```
Publication>>positiveVotes
    | r |
    r := OrderedCollection new.
    votes
        do: [ :vote |
            vote isLike
                ifTrue: [ r add: vote ] ].
    ^ r
```

Primero aplicamos `Substitute Algorithm`

```
Publication>>negativeVotes
    | r |
    r := reject: [ :vote | vote isLike ]
    ^ r
```

```
Publication>>positiveVotes
    | r |
    r := select: [ :vote | vote isLike ]
    ^ r
```

Seguidamente aplicamos `Replace Temp WithQuery`

```
Publication>>negativeVotes
    ^ votes reject: [ :vote | vote isLike ]
```

```
Publication>>positiveVotes
    ^ votes select: [ :vote | vote isLike ]
```

QuestionRetriever

retrieveQuestions:

Se Observa que el metodo `#retrieveQuestions:` es un `Long Method` que se vuelve complicado de seguir, por lo tanto empezamos a refactorizarlo.

```
QuestionRetriever>>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 ]
```

Vemos una `Cadena de Mensajes` y `Codigo Duplicado` al tratar de obtener la cantidad de votos positivos de una publicacion, por lo que aplicamos `Hide Delegate`

```
QuestionRetriever>>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 positiveVotesCount > b positiveVotesCount ].
            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 positiveVotesCount > b positiveVotesCount ].
            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 positiveVotesCount > b positiveVotesCount ].
            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 positiveVotesCount ]) / popularTCol size.
            temp := (popularTCol
                select: [ :q | q positiveVotesCount >= averageVotes ])
                asSortedCollection: [ :a :b | a positiveVotesCount > b positiveVotesCount ].
            qRet := temp last: (100 min: temp size) ].
    ^ qRet reject: [ :q | q user = aUser ]
```

```
Publication>>positiveVotesCount
    ^ self positiveVotes size
```

Aplicamos `Consolidate Duplicate Conditional Fragments` para remover codigo repetido que se encuentra presente en todos las ramas del condicional.

```
QuestionRetriever>>retrieveQuestions: aUser
    | qRet temp followingCol topicsCol newsCol popularTCol averageVotes |
    temp := OrderedCollection new.
    option = #social
        ifTrue: [ followingCol := OrderedCollection new.
            aUser following
                do: [ :follow | followingCol addAll: follow questions ].
            temp := followingCol ].
    option = #topics
        ifTrue: [ topicsCol := OrderedCollection new.
            aUser topics do: [ :topic | topicsCol addAll: topic questions ].
            temp := topicsCol ].
    option = #news
        ifTrue: [ newsCol := OrderedCollection new.
            cuoora questions
                do: [ :q |
                    q timestamp asDate = Date today
                        ifTrue: [ newsCol add: q ] ].
            temp := newsCol ].
    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 positiveVotesCount ]) / popularTCol size.
            temp := popularTCol
                select: [ :q | q positiveVotesCount >= averageVotes ] ].
    qRet := (temp
        asSortedCollection: [ :a :b | a positiveVotesCount > b positiveVotesCount ])
        last: (100 min: temp size).
    ^ qRet reject: [ :q | q user = aUser ]
```

Aplicamos `Extract Method` y nos queda:

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

Aplicamos `Replace Temp With Query` para remover las variable temporale `qRet` y hacemos un `Rename Temp` de `temp` a `questions` para otorgarle un nombre mas descriptivo

```
QuestionRetriever>>retrieveQuestions: aUser
    | questions |
    questions := self getQuestionsFor: aUser.
    ^ ((questions
        asSortedCollection: [ :a :b | a positiveVotesCount > b positiveVotesCount ])
        last: (100 min: questions size)) reject: [ :q | q user = aUser ]
```

Aplicamos `Replace Magic Number with Symbolic Constant` creando el metodo `#questionsLimit` para reemplzar el `100`

```
QuestionRetriever>>questionsLimit
    ^ 100
```

```
QuestionRetriever>>retrieveQuestions: aUser
    | questions |
    questions := self getQuestionsFor: aUser.
    ^ ((questions
        asSortedCollection: [ :a :b | a positiveVotesCount > b positiveVotesCount ])
        last: (self questionsLimit min: questions size)) reject: [ :q | q user = aUser ]
```

Aplicamos `Extract Method` para extraer la parte del metodo que limita la cantidad de preguntas

```
QuestionRetriever>>retrieveQuestions: aUser
    | questions |
    questions := self getQuestionsFor: aUser.
    ^ (self
        limitQuestions:
            (questions
                asSortedCollection: [ :a :b | a positiveVotesCount > b positiveVotesCount ]))
        reject: [ :q | q user = aUser ]
```

```
QuestionRetriever>>limitQuestions: aQuestionCollection
    ^ aQuestionCollection
        last: (self questionsLimit min: aQuestionCollection size)
```

Aplicamos `Extract Method` para extraer la parte del metodo que ordena las preguntas

```
QuestionRetriever>>retrieveQuestions: aUser
    | questions |
    questions := self getQuestionsFor: aUser.
    ^ (self limitQuestions: (self sortQuestions: questions))
        reject: [ :q | q user = aUser ]
```

```
QuestionRetriever>>sortQuestions: aQuestionCollection
    ^ aQuestionCollection
        asSortedCollection: [ :a :b | a positiveVotesCount > b positiveVotesCount ]
```

Aplicamos `Replace Temp With Query` para remover la variable temporal `questions`

```
QuestionRetriever>>retrieveQuestions: aUser
    ^ (self
        limitQuestions: (self sortQuestions: (self getQuestionsFor: aUser)))
        reject: [ :q | q user = aUser ]
```

getQuestionsFor:

Notamos que este metodo se divide en 4 grandes ramas segun el valor de la variable de instancia `option` .

```
QuestionRetriever>>getQuestionsFor: aUser
    | popularTCol averageVotes temp topicsCol followingCol newsCol |
    temp := OrderedCollection new.
    option = #social
        ifTrue: [ followingCol := OrderedCollection new.
            aUser following
                do: [ :follow | followingCol addAll: follow questions ].
            temp := followingCol ].
    option = #topics
        ifTrue: [ topicsCol := OrderedCollection new.
            aUser topics do: [ :topic | topicsCol addAll: topic questions ].
            temp := topicsCol ].
    option = #news
        ifTrue: [ newsCol := OrderedCollection new.
            cuoora questions
                do: [ :q |
                    q timestamp asDate = Date today
                        ifTrue: [ newsCol add: q ] ].
            temp := newsCol ].
    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 positiveVotesCount ]) / popularTCol size.
            temp := popularTCol
                select: [ :q | q positiveVotesCount >= averageVotes ] ].
    ^ temp
```

Aplicamos `Replace Conditional with Polymorphism` creando 4 subclases de `QuestionRetriever`: `NewsQuestionRetriever`, `SocialQuestionRetriever`, `PopularTodayQuestionRetriever` y `TopicsQuestionRetriever`

```
QuestionRetriever>>getQuestionsFor: aUser
    self subclassResponsibility
```

```
NewsQuestionRetriever>>getQuestionsFor: aUser
    | newsCol |
    newsCol := OrderedCollection new.
    cuoora questions
        do: [ :q |
            q timestamp asDate = Date today
                ifTrue: [ newsCol add: q ] ].
    ^ newsCol
```

```
SocialQuestionRetriever>>getQuestionsFor: aUser
    | followingCol |
    followingCol := OrderedCollection new.
    aUser following
        do: [ :follow | followingCol addAll: follow questions ].
    ^ followingCol
```

```
PopularTodayQuestionRetriever>>getQuestionsFor: aUser
    | popularTCol averageVotes temp |
    popularTCol := OrderedCollection new.
    cuoora questions
        do: [ :q |
            q timestamp asDate = Date today
                ifTrue: [ popularTCol add: q ] ].
    averageVotes := (cuoora questions sum: [ :q | q positiveVotesCount ])
        / popularTCol size.
    temp := popularTCol
        select: [ :q | q positiveVotesCount >= averageVotes ].
    ^ temp
```

```
TopicsQuestionRetriever>>getQuestionsFor: aUser
    | topicsCol |
    topicsCol := OrderedCollection new.
    aUser topics do: [ :topic | topicsCol addAll: topic questions ].
    ^ topicsCol
```

Al hacer esto, vemos que los tests de la clase `QuestionRetrieverTest` ya no pasan, revisando nos damos cuenta que se debe a nuestro refactoring, en particular vemos que el metodo `QuestionRetrieverTest>>#setUp` esta instanciando instancias de `QuestionRetriever` que ahora es una clase abstracta

```
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.
```

Acomodamos instanciando la clase correspondiente en cada caso

```
QuestionRetrieverTest>>setUp
    "..."
    socialRetriever := SocialQuestionRetriever new: cuoora
    topicsRetriever := QuestionRetriever new: cuoora
    newsRetriever := QuestionRetriever new: cuoora
    popularTodayRetriever := QuestionRetriever new: cuoora
```

Hecho esto vemos que la clase prensenta `Dead Code` en particular, los metodos `#option:` , el metodo `#intialize` y el metodo de clase `#new: #and:` ya no son utilizados, asi que podemos removerlos

Una vez hecho esto podemos remover la variable de instancia `option` que ya no es necesaria

```
Object subclass: #QuestionRetriever
    instanceVariableNames: 'cuoora'
    classVariableNames: ''
    package: 'TP-Refactoring-Model'
```

new:

Vemos que esta clase reimplementa de forma erronea el metodo `#new:`

```
QuestionRetriever>>new: cuoora
    ^ self new cuoora: cuoora; yourself
```

Aplicamos `Rename Method` para solucionar lo anterior y darle un nombre mas descriptivo

```
newForCuOOra: cuoora
    ^ self new cuoora: cuoora; yourself
```

NewsQuestionRetriever

getQuestionsFor:

Se observa una variable temporal innecesaria `newsCol`

```
NewsQuestionRetriever>>getQuestionsFor: aUser
    | newsCol |
    newsCol := OrderedCollection new.
    cuoora questions
        do: [ :q |
            q timestamp asDate = Date today
                ifTrue: [ newsCol add: q ] ].
    ^ newsCol
```

Aplicamos `Substitute Algorithm`

```
NewsQuestionRetriever>>getQuestionsFor: aUser
    | newsCol |
    newsCol := cuoora questions select: [ :q | q timestamp asDate = Date today ]
    ^ newsCol
```

Seguido de `Replace Temp With Query`

```
NewsQuestionRetriever>>getQuestionsFor: aUser
    ^ cuoora questions select: [ :q | q timestamp asDate = Date today ]
```

Vemos que `NewsQuestionRetriever` presenta `envidia de atributos` a `CuOOra` por lo tanto aplicamos `Extract Method` y seguidamente `Move Method`

```
NewsQuestionRetriever>>getQuestionsFor: aUser
    ^ cuoora getTodayQuestions
```

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

PopularTodayQuestionRetriever

getQuestionsFor:

En este metodo hay varias cosas a refactorizar

```
PopularTodayQuestionRetriever>>getQuestionsFor: aUser
    | popularTCol averageVotes temp |
    popularTCol := OrderedCollection new.
    cuoora questions
        do: [ :q |
            q timestamp asDate = Date today
                ifTrue: [ popularTCol add: q ] ].
    averageVotes := (cuoora questions sum: [ :q | q positiveVotesCount ])
        / popularTCol size.
    temp := popularTCol
        select: [ :q | q positiveVotesCount >= averageVotes ].
    ^ temp
```

empezamos aplicando `Extract Method`

```
PopularTodayQuestionRetriever>>getQuestionsFor: aUser
    | popularTCol averageVotes temp |
    popularTCol := OrderedCollection new.
    cuoora questions
        do: [ :q |
            q timestamp asDate = Date today
                ifTrue: [ popularTCol add: q ] ].
    averageVotes := self averageVotes: popularTCol size.
    temp := popularTCol
        select: [ :q | q positiveVotesCount >= averageVotes ].
    ^ temp
```

```
PopularTodayQuestionRetriever>>averageVotes: aNumber
    ^ (cuoora questions sum: [ :q | q positiveVotesCount ]) / aNumber
```

Podemos ver que el metodo `PopularTodayQuestionRetriever>>#averageVotes:` `Envidia Atributos` de `cuoora`, por lo tanto aplicamos `Move Method` y lo movemos hacia la clase `CuOOra`

```
PopularTodayQuestionRetriever>>getQuestionsFor: aUser
    | popularTCol averageVotes temp |
    popularTCol := OrderedCollection new.
    cuoora questions
        do: [ :q |
            q timestamp asDate = Date today
                ifTrue: [ popularTCol add: q ] ].
    averageVotes := cuoora averageVotes: popularTCol size.
    temp := popularTCol
        select: [ :q | q positiveVotesCount >= averageVotes ].
    ^ temp
```

```
CuOOra>>averageVotes: aNumber
    ^ (questions sum: [ :q | q positiveVotesCount ]) / aNumber
```

Continuamos aplicando `Replace Temp With Query`

```
PopularTodayQuestionRetriever>>getQuestionsFor: aUser
    | popularTCol temp |
    popularTCol := OrderedCollection new.
    cuoora questions
        do: [ :q |
            q timestamp asDate = Date today
                ifTrue: [ popularTCol add: q ] ].
    temp := popularTCol
        select:
            [ :q | q positiveVotesCount >= (cuoora averageVotes: popularTCol size) ].
    ^ temp
```

Aplicamos `Substitute Algorithm`

```
PopularTodayQuestionRetriever>>getQuestionsFor: aUser
    | popularTCol |
    popularTCol := cuoora questions
        select: [ :q | q timestamp asDate = Date today ].
    ^ popularTCol
        select: [ :q | q positiveVotesCount >= (cuoora averageVotes: popularTCol size) ]
```

Se observa `codigo duplicado` y una variable temporal innecesaria `popularTCol` al obtener las preguntas del dia, por lo tanto aplicamos `Replace Tempo With Query`

```
PopularTodayQuestionRetriever>>getQuestionsFor: aUser
    ^ cuoora getTodayQuestions
        select: [ :q |
            q positiveVotesCount
                >= (cuoora averageVotes: cuoora getTodayQuestions size) ]
```

Vemos que el metodo presenta `envidia de atributos` a `CuOOra` por lo tanto aplicamos `Extract Method` seguido de `Move Method`

```
PopularTodayQuestionRetriever>>getQuestionsFor: aUser
    ^ cuoora getPopularTodayQuestions
```

```
CuOOra>>getPopularTodayQuestions
    ^ self getTodayQuestions
        select: [ :q |
            q positiveVotesCount
                >= (self averageVotes: self getTodayQuestions size) ]
```

SocialQuestionRetriever

getQuestionsFor:

Se observa una variable temporal innecesaria `followingCol`

```
SocialQuestionRetriever>>getQuestionsFor: aUser
    | followingCol |
    followingCol := OrderedCollection new.
    aUser following
        do: [ :follow | followingCol addAll: follow questions ].
    ^ followingCol
```

Aplicamos `Substitute Algorithm`

```
SocialQuestionRetriever>>getQuestionsFor: aUser
    | followingCol |
    followingCol := aUser following flatCollect: [ :follow | follow questions ]
    ^ followingCol
```

Seguido de `Replace Temp with Query`

```
SocialQuestionRetriever>>getQuestionsFor: aUser
    ^ aUser following flatCollect: [ :follow | follow questions ]
```

Notamos que el metodo presenta `envidia de atributos` a `User` por lo tanto, aplicamos `Extract Method` seguido de `Move Method`

```
SocialQuestionRetriever>>getQuestionsFor: aUser
    ^ aUser getFollowingUsersQuestions
```

```
User>>getFollowingUsersQuestions
    ^ following flatCollect: [ :follow | follow questions ]
```

TopicsQuestionRetriever

getQuestionsFor:

Se observa una variable temporal innecesaria `topicsCol`

```
TopicsQuestionRetriever>>getQuestionsFor: aUser
    | topicsCol |
    topicsCol := OrderedCollection new.
    aUser topics do: [ :topic | topicsCol addAll: topic questions ].
    ^ topicsCol
```

Aplicamos `Substitute Algorithm`

```
TopicsQuestionRetriever>>getQuestionsFor: aUser
    | topicsCol |
    topicsCol := aUser topics flatCollect: [ :topic | topic questions ]
    ^ topicsCol
```

Seguido de `Replace Temp with Query`

```
TopicsQuestionRetriever>>getQuestionsFor: aUser
    ^ aUser topics flatCollect: [ :topic | topic questions ]
```

Notamos que el metodo presenta `envidia de atributos` a `User` por lo tanto, aplicamos `Extract Method` seguido de `Move Method`

```
TopicsQuestionRetriever>>getQuestionsFor: aUser
    ^ aUser getFollowingTopicsQuestions
```

```
User>>getFollowingTopicsQuestions
    ^ topics flatCollect: [ :topic | topic questions ]
```