For the refactoring of **RoutineMe**, our group wanted to implement several of the design patterns discussed in class into the RoutineMe app. The first, and most important refactoring step was to break up some of our enormous classes into smaller, more manageable classes. From the old class diagram pictured below, the Routine and UserProfile classes were very large and represented the Blob antipattern with several attributes, getter, and setter methods. To fix this, the attributes of these classes were grouped together and made into easier to manage sub classes.

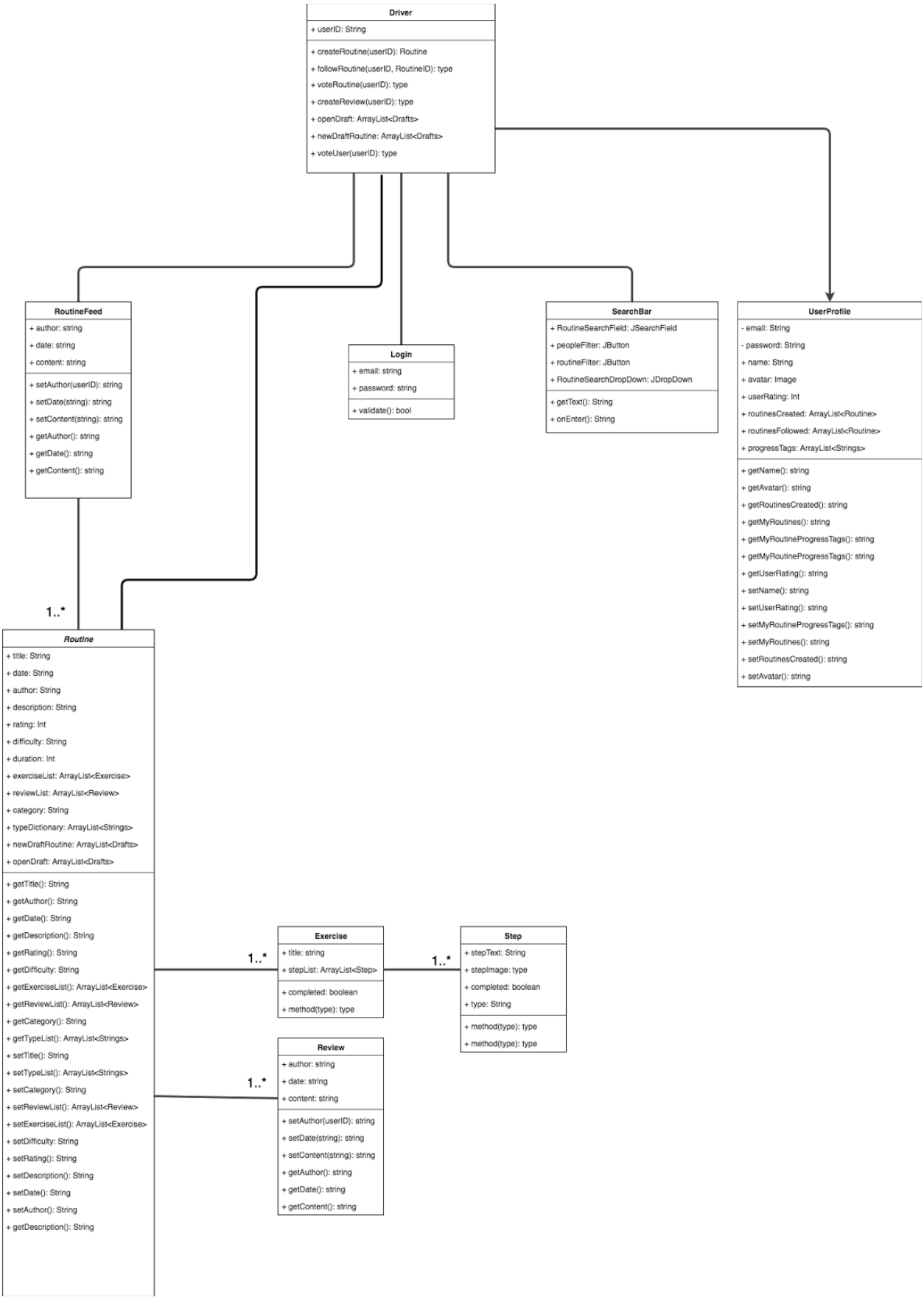
When implementing design patterns into our class layout, our group wanted to focus on four main areas of the app. These were routine creation, login and verification, routine drafts, and our connection to our database.

The **DatabaseConnect** class is an observer class that connects straight to the Feed class. This is used to notify the Feed class when there is new data available in the database and it can automatically update the feed seen by the user using its own Feed.update function. The implementation of this observer pattern will keep the routine feed up to date with the latest routines/users available from the database.

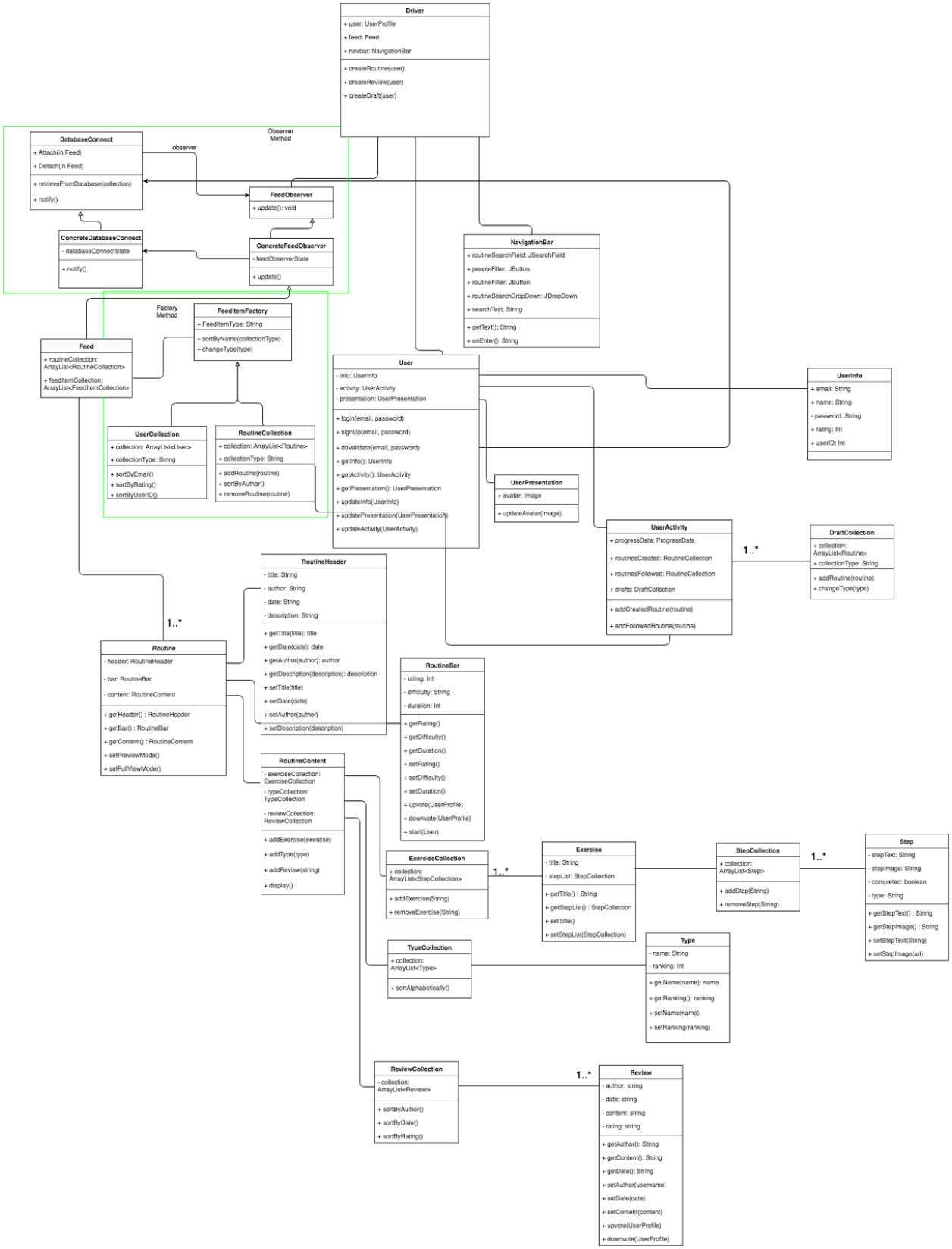
The **FeedItemFactory** is a class that allows us to sort the items in our Feed by the name of the item, whether it's the User or the Routine. If we decide in the future to include something other than Users or Routines (maybe articles), we can treat them as another FeedItem, and the sortByName method can sort by the name of the article as well.

Additionally, we created a lot of getters and setters so that everything keeps in sync with our database when other classes perform operations on other classes. Most attributes are private and most methods are public for this reason.

## Original class diagram



**Refactored class Diagram (too large to fit, link provided)**
https://drive.google.com/file/d/0B4vwX5PFtpTEZjRETS15VnFsTUE/view?usp=sharing


Driver
+ user: UserProfile
+ feed: Feed
+ navbar: NavigationBar
+ createRoutine(user)
+ createReview(user)
+ createDraft(user)
Observer Method
DatabaseConnect
+ Attach(in Feed)
+ Detach(in Feed)
+ retrieveFromDatabase(collection)
+ notify()
observer
FeedObserver
+ update(): void
ConcreteDatabaseConnect
- databaseConnectState
+ notify()
ConcreteFeedObserver
- feedObserverState
+ update()
NavigationBar
+ routineSearchField: JSearchField
+ peopleFilter: JButton
+ routineFilter: JButton
+ routineSearchDropDown: JDropDown
+ searchText: String
+ getText(): String
+ onEnter(): String
Factory Method
FeedItemFactory
+ FeedItemType: String
+ sortByName(collectionType)
+ changeType(type)
Feed
+ routineCollection: ArrayList<RoutineCollection>
+ feedItemCollection: ArrayList<FeedItemCollection>
User
- info: UserInfo
- activity: UserActivity
- presentation: UserPresentation
+ login(email, password)
+ signUp(email, password)
+ dbValidate(email, password)
+ getInfo(): UserInfo
+ getActivity(): UserActivity
+ getPresentation(): UserPresentation
+ updateInfo(UserInfo)
+ updatePresentation(UserPresentation)
+ updateActivity(UserActivity)
UserInfo
+ email: String
+ name: String
- password: String
+ rating: Int
+ userID: Int
UserCollection
+ collection: ArrayList<User>
+ collectionType: String
+ sortByEmail()
+ sortByRating()
+ sortByUserID()
RoutineCollection
+ collection: ArrayList<Routine>
+ collectionType: String
+ addRoutine(routine)
+ sortByAuthor()
+ removeRoutine(routine)
UserPresentation
+ avatar: Image
+ updateAvatar(image)
UserActivity
+ progressData: ProgressData
+ routinesCreated: RoutineCollection
+ routinesFollowed: RoutineCollection
+ drafts: DraftCollection
+ addCreatedRoutine(routine)
+ addFollowedRoutine(routine)
1..*
DraftCollection
+ collection: ArrayList<Routine>
+ collectionType: String
+ addRoutine(routine)
+ changeType(type)
1..*
Routine
- header: RoutineHeader
- bar: RoutineBar
- content: RoutineContent
+ getHeader() : RoutineHeader
+ getBar() : RoutineBar
+ getContent() : RoutineContent
+ setPreviewMode()
+ setFullViewMode()
RoutineHeader
- title: String
- author: String
- date: String
- description: String
+ getTitle(title): title
+ getDate(date): date
+ getAuthor(author): author
+ getDescription(description): description
+ setTitle(title)
+ setDate(date)
+ setAuthor(author)
+ setDescription(description)
RoutineBar
- rating: Int
- difficulty: String
- duration: Int
+ getRating()
+ getDifficulty()
+ getDuration()
+ setRating()
+ setDifficulty()
+ setDuration()
+ upvote(UserProfile)
+ downvote(UserProfile)
+ start(User)
RoutineContent
- exerciseCollection: ExerciseCollection
- typeCollection: TypeCollection
- reviewCollection: ReviewCollection
+ addExercise(exercise)
+ addType(type)
+ addReview(string)
+ display()
ExerciseCollection
+ collection: ArrayList<StepCollection>
+ addExercise(String)
+ removeExercise(String)
1..*
Exercise
- title: String
- stepList: StepCollection
+ getTitle() : String
+ getStepList() : StepCollection
+ setTitle()
+ setStepList(StepCollection)
StepCollection
+ collection: ArrayList<Step>
+ addStep(String)
+ removeStep(String)
1..*
Step
- stepText: String
- stepImage: String
- completed: boolean
- type: String
+ getStepText() : String
+ getStepImage() : String
+ setStepText(String)
+ setStepImage(url)
TypeCollection
+ collection: ArrayList<Type>
+ sortAlphabetically()
Type
- name: String
- ranking: Int
+ getName(name): name
+ getRanking(): ranking
+ setName(name)
+ setRanking(ranking)
ReviewCollection
- collection: ArrayList<Review>
+ sortByAuthor()
+ sortByDate()
+ sortByRating()
1..*
Review
- author: string
- date: string
- content: string
- rating: string
+ getAuthor(): String
+ getContent(): String
+ getDate(): String
+ setAuthor(username)
+ setDate(date)
+ setContent(content)
+ upvote(UserProfile)
+ downvote(UserProfile)