Meme Magic
Sprint 3 - Build 2

You are proud of your team, they are making great progress. We will continue the building of the system in sprint 2. This sprint will include the `User` and `Feed` classes, along with some integration tasks.

First, set up your project:
- Create a new Java project on Eclipse, and call it `MemeMagic3`.
- Copy all of your `MemeMagic2` files into the `MemeMagic3` project.
- Maintain a copy of `MemeMagic2` files (i.e., do not overwrite them.)

Note: you may alternatively copy your project in Eclipse by selecting the `MemeMagic2` project in the Package Explorer, copy with Ctrl+c (or Command+c on mac), and paste with Ctrl+p (or Command+p on mac). On pasting, Eclipse will ask you to name the newly-pasted project.

Learning Goals

In this assignment, we will practice:
1. Utilizing `ArrayList` to store and organize objects
2. Integrating newly implemented classes into an existing codebase
3. Incorporating main-method testing to check correctness (i.e., behavior matches specifications)

Implementation

Implement methods for the classes as described below. When creating a constructor, if a class has more fields than specified in the constructor, initialize these fields with default values.

User

- create a constructor that accepts a `String` for userName
- `toString()` - returns "*username* has rated (*number of memes viewed*) memes, (*reputation*)"
    - **Note**: the reputation should be rounded to 1 decimal place.
    - ex:
      derrickstone has rated (10) memes, (9.0)
- `equals(Object)` - returns true if the parameter is a `User` object and this `User`'s `userName` is equal to that of the parameter.

- `rateMeme(Meme, int)` - this method accepts a `Meme` argument and an `int` for rating score. It will record that `Meme` as having been seen by this user (`memesViewed`) and give it a `Rating` of this `score`.
- `rateNextMemeFromFeed(Feed, int)` - this method accepts a `feed` argument and an `int` for rating score and returns a `boolean`. The method will get a `Meme` from the `Feed` (supplied as an argument) using the `getNewMeme(User)` method of the `Feed` class. It will record that `Meme` as having been viewed by the user, give it the rating score, and return true. If there are no `Memes` left to view, the method should return false (and should not throw an error).
  Note: the return type has been updated from Sprint 1 to return a `boolean`.
- `createMeme(BackgroundImage, String)` - creates a new `Meme` object using the supplied arguments (`String` is the `caption`) and with the current user set as the creator. This method will add the resulting `Meme` to the list of `createdMemes` for the current user.
- `deleteMeme(Meme)` - deletes this `Meme` if found in the `memesCreated` field for the current user, only if the shared field is false. *(Because anything shared on the Internet lives forever.)* If the deletion was successful, return true. Otherwise, return false.
- `shareMeme(Meme, Feed)` - marks that `Meme` as shared (sets the shared field to true) and copies it to the `ArrayList<Meme>` data structure on the supplied `Feed`.
- `calculateReputation()` - returns a value calculated as the average of all overall ratings (`caclulateOverallRating()`) for Memes created by this User. If the user has not created any `Memes` or had any `Memes` rated, `0.0` should be returned.

Feed

- `getNewMeme(User)` - return a `Meme` from the current `Feed` that the `User` has not seen (does not exist in that `User`'s `memesViewed` list) and that the `User` did not create themselves. If there is no `Meme` to return, return `null`.
- `toString()` - returns all the memes in the feed, each `Meme` on a new line. Note that the `toString()` methods for both `Meme` and `Rating` need to be updated to integrate with the `User` and `Feed` implementation.
    - Ex:
      How bots laugh <Image of Joquain Phoenix in his role as Joker, laughing maniacally> 'When your professor calls an in person meeting at 9 AM EST and you live in Cali' 5.0 [+1: 6, -1: 1] - created by derrickstone
      Too sad <Image of tearful kitten> 'When you laugh at insistence on comments in school and then get a job programming where nobody comments' -2.0 [+1: 4, -1: 6] - created by derrickstone
      Robots Eating <Robots seated at table with server> 'When you go out for a byte' 1.0 [+1: 3, -1: 2] - created by user01001011

Integration

Now that you have implemented the User class, we must integrate it into the previously-built portions of the system. Using the following specifications, update these methods of the `Meme` and `Rating` classes.

Meme

- `toString()` - returns "backgroundImage ‘caption' *overallRating* [+1: the number of +1 ratings, -1: the number of -1 ratings] - created by *userName*"
    - ex:
      How bots laugh <Image of Joquain Phoenix in his role as Joker, laughing maniacally> 'When your professor calls an in person meeting at 9 AM EST and you live in Cali' 5.0 [+1: 6, -1: 1] - created by derrickstone
    - **Note**: overallRating is the value returned by `calculateOverallRating()`.
    - **Note**: userName is the `userName` of the `creator User`.
    - **Hint**: there is a lot going on in this string. Consider how additional *private* helper methods might make this easier to read.

Rating

- `toString()` - returns "*userName* rated as *type_of_rating*"
    - For example, if the user object has userName *derrickstone:*
        - if the score is +1, then it will return: *derrickstone rated as an upvote*
        - if the score is -1, then it will return: *derrickstone rated as a downvote*
        - if the score is 0, then it will return: *derrickstone rated as a pass*

Main Method Testing

In this sprint, we expect you to do your own main method testing. Although the amount of main method testing is not limited, please provide **at least two tests each** for the following:

- The new constructor we’ve asked for in this sprint (User)
- User, Feed, Meme, and Rating’s `toString()` methods
- User’s `equals()` method
- Feed’s `getNewMeme()`
- User’s `rateMeme()`
- User’s `rateNextMemeFromFeed()`
- User’s `createMeme()`
- User’s `deleteMeme()`
- User’s `shareMeme()`
- User’s `calculateReputation()`

Note: Your main method should have enough testing to provide sufficient evidence to determine that the behavior implemented matches the behavior described above. It is *highly recommended* to write the tests before implementation as it will help you to understand the exact behavior and what is the expected output of different inputs. That will be especially important for corner cases, such as deleting a shared meme in `deleteMeme()` or getting the next meme from an empty feed in `rateNextMemeFromFeed()`.

You will only be able to submit to Gradescope a total of 10 times for this assignment; please test your code's functionality with main method testing before submitting.

Additional Resources

The following resources may be helpful when completing and submitting this sprint.

- JavaDoc style documentation for each of the classes and methods described above
- Video on submitting to Gradescope
- The following code will print the string `5.0`
  `System.out.println(String.format("%.1f", 4.999999));`

Submission Information

Method and Class Naming: You must match method names, instance variable names, and data types exactly. You must use correctly formatted Java code. Declare fields in the class definition, and create a default constructor for each class that initializes every instance variable. For methods that you are overriding (i.e., `equals()` and `toString()`), use the `@Override` annotation before the method header.

Coding Style: In real-world software development, it is paramount to create readable and easily maintainable code. That is typically achieved through the use of style and commenting guidelines. Since you will be updating this code over the next few weeks, we have provided a style guide and formatting guide that we strongly encourage you to follow:

- Coding Style Guide (includes installation instructions for Eclipse)
- Eclipse Style File

Submitting: Upload your Eclipse project (the `.java` files) to the "Meme Magic 3" assignment on Gradescope. You should submit `User.java`, `Feed.java`, `Rating.java`, `BackgroundImage.java`, and `Meme.java`. This submission utilizes an autograder to check that your code follows these specifications. If it spots a disconnect or bug, it will alert you, but you should **NOT** use the submission system as your testing. Testing should be done during the implementation phase. Therefore, for this assignment, you may upload your code a **maximum of ten (10) times**.

Note: After the 10th submission, Gradescope will still allow submissions, but they will NOT be graded by the system.

Grading Rubric

The assignment will be worth a total of 100 points:

80 points	- Method and implementation correctness, auto-graded using Gradescope
15 points	- Main method testing
5 points	- Code readability (organized, well-indented, readable by others)

Academic Integrity and Moving into Sprint 4

After submitting this sprint (*and after the late submission deadline*), you are encouraged to collaborate with your group members on what you missed and you *are* allowed to share code for those parts. However, everyone who collaborates **must have already submitted their code and *may not resubmit* to this sprint**.

You *must* understand any code you incorporate, as you'll be using it to complete the next (and subsequent) sprint. We will *not* be directly testing the code from this sprint again. You **must** list your collaborators in the comment at the top of each file that has any amount of shared code.