GRASP

1. The Creator:
    a. HappinessGraph
        i. constructor creates Chart because Happiness Graph aggregates Chart objects
    b. PostToWall
        i. constructor creates Post object (by current user) because PostToWall has all the initializing data that will be passed to Post when it is created
    c. Feed
        i. updateFeed creates Post object because Feed contains three Post objects
    d. Dashboard
        i. constructor creates Feed because Dashboard contains a Feed object
    e. Dashboard
        i. constructor creates PostToWall because Dashboard contains a PostToWall object
    f. Post
        i. constructor creates StarRater because a Post object has a Star Rater object
    g. Dashboard
        i. constructor creates Search because it contains a Search object
    h. DiaryService
        i. create() is responsible for creating a DiaryEntry object because it has all of the necessary information to do so
    i. ReportMaker
        i. createReport() is responsible for creating a Report object because it has all of the necessary information to do so
    j. RegistrationService
        i. createUser() is responsible for creating a User object because it has all of the necessary information to do so
    k. MoodService
        i. createMood() is responsible for creating a Mood object because it has all of the necessary information to do so

2. Information Expert:
    a. Chart
        i. constructor is responsible for creating a basic graph that displays mood history for a 7 day interval because it has all of the necessary information to do so
    b. Feed

        i. updateFeed is responsible for updating the posts displayed on the wall after a new post is made because it has access to Post objects
    c. ReportMaker
        i. createReport creates a report based off of user data because it has all of the necessary information to do so
    d. FriendNetworkService
        i. addFriend() will be responsible for notifying DBSStore to add userA and userB to the same network because it will have the necessary information (email addresses) to do so
        ii. sendRequest() will be responsible for inviting userB to be friends with userA because it has access to userB's email
        iii. notify() is responsible for altering userB that userA wants to be friends because it has access to userB's email

3. Low Coupling and High Cohesion:
    a. Though DashBoard depends on many classes, it redirects to other JPanels or opens JDialogues. As a result, a change to one of its dependencies does not grossly affect its functionality.
    b. HappinessGraph is coupled to Chart because it contains a Chart object and displays it on its JPanel. Because it simply adds Chart to its JPanel, a change to Chart should not affect the functionality of HappinessGraph.
    c. Likewise, Feed contains Post objects. It must access a Post object's member variables, resulting in High Coupling. Because the purpose of Feed is to update Posts displayed on a Dashboard, the high coupling was justified as it allows for high cohesion.
4. PureFabrication
    a. DBSStore will have access to Database and program logic to maintain high cohesion and low coupling
5. Controller
    a. RegistrationController
        i. createLogin() is responsible for altering RegisterService after taking in user input to limit its responsibilities
    b. NetworkController
        i. calls searchFriend() which will alert FriendNetworkService to call DBSStore to check if the user exits. Because it only receives a username as a parameter, it has insufficient information to do anything else
    c. UserController will handle the adding of an event. We are assigning this responsibility as a use case scenario
    d. UserController will handle the posting to a wall. We are assigning this responsibility as a use case scenario

    e. DiaryController
        i. save() is responsible for altering DiaryService to save the diaryEntry
6. Law of Demeter: Our classes only know about each other if one class must somehow manipulate another.

Micah Dadson- 3 sequence diagrams, Chart.java, Feed.Java, MenuBar.Java, Post.java, PostToWall.java, Search.Java, StarRater.java, Dashboard.java, HappinessGraph.java, Report.java, CronScheduler.java, ReportMaker.java, applicationContext.xml, GRASP, UML class diagram
Points: 5/5
Time Spent: 25 hours

Jeremy Meadows- 3 sequence diagrams, Calendar.java, Database.java, Password.java, User.java, Registration.java, CustomUtilities.java, SpringUtilities.java, CustomUtilities.java, Mentality.Java, Runner.java, logo.png, logo.svg
Points:
Time Spent:

Shivani Bobbala- 3 sequence diagrams, Mood.java, DiaryFrame.java
Points:
Time Spent:

Joel Futagawa- 3 sequence diagrams
Points:
Time Spent:

Shunting Chen- 3 sequence diagrams
Points:
Time Spent:

Francis Ning- n/a
Points:
Time Spent: