As suggested by Josh, our group threw out our original copy of our program and started from scratch. After writing it the first time, we knew what changes would benefit our program and make it run more smoothly. These changes are as follows:

* To the User class we kept all of the previous methods and added new methods which include setFirstName(String), setLastName(String), setEmailAddress(String), getFirstName(), getLastName(), getEmailAddress(), hashCode(), equals(Object), and compareTo(User). In our previous UML diagram, we only had getter methods for last name and email. We realized this time around that some Users may have the same last name which would require us to be able to distinguish between the two. This realization led to the creation of the getFirstName method as well as the compareTo methods. the equals and hashCode methods were added to better the comparing between Users.
* To the Administrator class we changed the constructor so that it takes three String values for the user’s first name, last name, and email address. This way we can create a new Administrator user and enter in the information upon creation instead of setting the values later. Additionally, the getVolunteerList method from Deliverable two was removed from this class and the capabilities of this method were moved to a separate method in the Handler class in order to keep functions relatively unrelated to a particular class outside of that class. For that reason, it was necessary to move the method.
* To the Park Manager class we removed two methods, viewMyParkJobs and viewVolunteers, for the same reason that we removed getVolunteerList from the Administrator class, they were not related closely enough to the Park Manager class. Like the Administrator class, we changed the parameters to take three String values for first name, last name, and email address.
* To the Volunteer class we removed viewUpcomingJobs and addJobToMyList. ViewUpcomingJobs was removed because it wasn’t related to the Volunteer class closely enough to be accessed directly in that class. The addJobToMyList method was replaced by the addJob(Job) method. This new method adds the job to the Volunteer’s personal list of jobs and returns a boolean value to show whether the job was successfully added or not. Additional changes include changing the constructor in the same way that the Administrator and ParkManager constructors were changed as well as adding the canSignUpForJob(Job) which checks that the Job that the Volunteer wants to add is not on a date that conflicts with the date of a Job that Volunteer has already signed up for and that the date of the Job to be added is not in the past.
* To the Job class we removed getParkLocation because we no longer have locations for our Jobs, getParkName because its relevance moved to the new Park class, and displayJob because that was a console method. The rest of the methods from Deliverable two remained. Although this class was already fairly long in Deliverable two, after our first run of this project we realized we needed many more methods to achieve the results we wanted. To do so we added setMaxNumLightVolunteers(int), setMaxNumMediumVolunteers(int), setMaxNumHeavyVolunteers(int), getMaxNumLightVolunteers(), getMaxNumMediumVolunteers(), getMaxNumHeavyVolunteers(), isLightVolunteersFull(), is MediumVolunteersFull(), isHeavyVolunteersFull(), addLightVolunteer(Volunteer), addMediumVolunteer(Volunteer), addHeavyVolunteer(Volunteer), isJobScheduleValid(Date, Date), hashCode(), equals(Object), compareTo(Job), and getPark(). Like in the User class, the hashCode and equals methods were added for better comparisons between Jobs. The setters and getters added reflect the new information that was added since Deliverable two including the max number of each weight of volunteer. Methods were also added to check to see if those weights are full and check to see if the date of the job being added is valid based on the list of jobs already added. The getPark() method is used to access information about the Park that Job is at, such as the Park Name, Park Manager, and Park Location. We also changed the constructer so that the parameters it takes in include the jobID, startDate, endDate, and Park.
* The Park class was a completely new class for Deliverable three. In it we included a constructor that took a parkID, a parkName, and a ParkManager. The methods we added were setParkName(String), setParkManager(ParkManager), getParkName(), getParkManager(), getJobs(), addJob(Job), hashCode(), equals(Object), and compareTo(Park). Like the User and Job classes, the hashCode and equals methods were added for better comparisons between Parks. Unlike the addJob methods in the ParkManager and Volunteer classes, the addJob method in this class adds the Job passed in to a set of Jobs for that specific Park. The getJobs method returns all of the jobs that the specific Park has.   
  \*\*reason for adding this new class
* The Handler class is also a completely new class for Deliverable three. The methods added include addUser(User), getAllJobs(), getUpcomingJobs(), isPendingJobsFull(), isJobScheduleAvailable(Date, Date), getVolunteersByLastName(String), getUserByEmail(String), getAdministrators(), getParkManagers(), and getVolunteers(). This class holds the power to do the actions that were not related enough to put in certain other classes. For example, we moved the getVolunteerByLastName method from the Administrator class to the Handler class for Deliverable three. In this way, Administrator does not need to access the Volunteer class directly. This was the rationale for most of the methods included actually, that we did not want each class accessing many other classes. Instead, have one class that is able to access them all and have the methods that use multiple classes stored in there. That of course isn’t the case for all of the methods in our diagram as you can see that Volunteer accesses Job and ParkManager accesses Park. Those however are special cases since ParkManagers have Parks and need to access them often, just as Volunteers have Jobs and need to access those often.
* The JobComparatorByDate, ParkComparatorByName, and UserComparatorByName classes were all added in order to be able to organize the jobs by date and by park and to organize a list of users by name. This is used in the console when information is shown to the user.
* The RunAllTest class was added because Josh wanted us to be able to run all of our test classes at once with a suite class.
* The DateUtil class was added in order to fulfill certain business rules regarding Dates such as checking if two dates are equal and if they are within the three month bounds.
* Each view class is directly related to a specific menu that is viewed when a particular user uses the program. These classes don’t actually run the console, but they output the Strings that the user sees when running the program. Unlike in Deliverable two, we took Josh’s advice to separate the console into separate classes with a separate one for each menu. That way information stays with relevant information and there is not one extremely large class handling the entire console.
* The controller classes on the other hand are the ones running the console. When an option is selected from a menu, that is done in the individual controller classes. Like the view classes, we decided to separate the controller into separate classes for each user as well as one for the data and session which are used for loading and saving data and the current session of the program respectively.

After Deliverable two, each member of the group kept the same classes that they worked on the first time since they now knew the ins and outs of how their class(es) were supposed to work. The only change in group member responsibilities of the non-console classes came from the Handler and Park classes which took the place of Jason’s JobSchedule class from Deliverable two. Accordingly, Jason volunteered to write both the Handler class and the Park class.

As you can see, many changes were made to the classes, so accordingly many changes had to be made to the test classes. For the methods that are new in Deliverable three, we wrote test classes for each that have the potential to go wrong in some way. For the returning methods, we either rewrote our tests or adjusted them so that they reflected what we really understood should be tested. Our main changes to the tests from Deliverable two to Deliverable three was that we split up tests so that they only tested one element of a method at a time and made sure to name tests in a way that would make it easy for others looking at them to understand. As for the console, most of the work on it was done between Ankit and Jason with smaller collaborations from Leda and Katie.