1. In Ball.java the constructor wasn't calling its private variables. “Volume = volume;” and “color = color” wouldn’t change the variables and the code wouldn’t know you are calling the variables so instead I had to change it so that it used this.volume and this.color to make “this.volume = volume; this.color = color;” I then had to fix the other constructor that uses string as its volume parameter. First I had to delete the previous code to try to catch the exception for when a string is actually imputed into the code rather than a string number. To do this I used a try {} catch() {} to catch the “NumberFormatException e” and then I put “this.volume = Double.parseDouble(volume); this.color = color;” to keep the old code the same and added “this.volume = 0; this.color = color;” to the catch so that volume would actually exist when a string is entered. Then I had to fix the remaining two functions for getVolume() and getColor() since they weren’t utilizing the variables and since this time they didn’t have parameters of volume or color I used “return volume;” and “return color;”.
2. The second approach is better (Keep track of the total volume) since by keeping track of the volume consistently when calling the total volume it would only take O(1) time since you already know the volume. Adding up all the ball’s volume right before takes minimum O(n) time since it adds up all the n balls’ volume at that time which would make running this function slower.
3. The first approach I did to implement getBallsFromSmallest()is by using array list and its comparators. I first did this by importing using “import java.util.ArrayList; import java.util.Comparator; import java.util.List;”. Then I made an array list “List<Ball> sortBalls = new ArrayList<>();” and went through the ball container to add each ball into the sortBalls list “for (Ball ball : ballContainer) { sortBalls.add(ball);}” I then sorted it using the .sort “sortBalls.sort(Comparator.comparingDouble(Ball::getVolume));” and used .iterator() to return the list.
   The second approach I used is priority queue which required the imports of “import java.util.PriorityQueue;” and “import java.util.Comparator;”. It then requires a priority queue to be made and the comparing attributes which is going to be the volume. I used the code “PriorityQueue<Ball> sortBalls = new PriorityQueue<>(Comparator.comparingDouble(Ball::getVolume));” to do this. I then went over the balls in the container again and added it into the priority queue in “for (Ball ball : ballContainer) { sortBalls.add(ball); }”.I then returned the priority queue “return sortedBalls.iterator();”.
   I think the better strategy is the first strategy because it seems neater since the line isn’t as long to read and more people probably know about an array list rather than a priority queue.