**Controller Class**

1. Instantiate a new CandyGame object.
2. Call method on the CandyGame object to get the number of students.
3. Instantiate a new int array with a size of the number of students.
4. Call method on the CandyGame object to get the lower limit of starting pieces of candy using limits defined in the code (15, 30).
5. Instantiate an int with a value of the lower limit plus 50 to be used as the upper limit of the upper limit.
6. Instantiate an int with a value of the lower limit plus two because we want a variable for the next even number.
7. Call a method on the CandyGame object to get the upper limit of starting pieces of candy using the lower limit plus 50 variable and the next even number variable.
8. Call a method on the CandyGame object to distribute the starting candy to the array of students using the user defined lower and upper limits.
9. Call a method on the CandyGame object to pass the candy.

**CandyGame Class**

1. Define a Scanner attribute.
2. Declare a method that returns an int for getting the number of students and takes arguments for the ints lower limit and upper limit.
   1. If the lower limit is greater the upper limit or the lower limit is less than or equal to 0, throw an exception
   2. Instantiate an int for the number of students and initialize it as 0
   3. Set the scanner attribute to a new Scanner with the input stream as system input.
   4. Instantiate an int for the desired number of students and initialize it as 0 (I hate null)
   5. If the next token in the scanner's input can be interpreted as an int value then set the desired number of students to the next int value.
   6. If the desired number of students is greater than or equal to the lower limit and less than or equal to the upper limit, then set the number of students to the desired number of students. If not, then set the number of students to a recursive method call of the method to get the number of students.
   7. Return the number of students.
3. Declare a method that returns an int to get the limit of starting pieces of candy and that accepts arguments for the ints lower limit and upper limit.
   1. If the lower limit is greater than the upper limit, or the lower limit is less than or equal to 0, or both limits are not even, throw an exception.
   2. Instantiate an in for the limit of starting pieces of candy and initialize it as 0.
   3. Set the scanner attribute to a new Scanner with the input stream as system input.
   4. Instantiate an int for the desired limit of starting pieces of candy and initialize it as 0.
   5. If the next token in the scanner's input can be interpreted as an int value then set the desired limit of starting pieces of candy to the next int value.
   6. If the desired limit of starting pieces of candy is greater than or equal to the lower limit and is less than or equal to the upper limit, and is even, then the limit of starting pieces of candy is the desired limit. If not, then set the limit to a recursive method call of the method to get the limit.
   7. Return the limit of starting pieces of candy.
4. Declare a method that returns void to distribute the candy to the students and accepts arguments for the int array students, and the ints upper limit and lower limit.
   1. For all the students
      1. Instantiate a Random object.
      2. Set the students candy equal to (lower limit + ( ( (random int) / 2 ) \* 2 )
5. Declare a method that returns void to display the students and accepts arguments for the int array students
   1. For all the students, print the value of their candy with a field width of 4
   2. Print a new line
6. Declare a method that returns void to pass the candy and accepts the arguments for the int array students
   1. Set the scanner attribute to a new Scanner with the input stream as system input.
   2. Instantiate an int called each step desired as -1.
   3. If the next token in the scanner's input can be interpreted as an int value then set each step desired to the next int value.
   4. If each step desired is not 1 or 0, then recall the method.
   5. Instantiate a Boolean called game completed and initialize it as false.
   6. While the game is not completed
      1. Instantiate an int called passing candy with 0.
      2. For all the students
         1. Instantiate an int called candy to pass with the value of passing candy.
         2. Set the student’s candy to half their candy.
         3. Set the passing candy to the student’s candy.
         4. Add the candy to pass to the student’s candy.
      3. Add the passing candy to the first student.
      4. For all the students
         1. Add a candy if their candy is odd.
      5. If all the students candy values are the same
         1. Display the students.
         2. Declare the game as complete.
         3. Return.
      6. If each step is desired
         1. Display the student’s candy
7. Declare a method to check if the game is complete and accepts arguments for an int array called students
   1. Instantiate an int called first student that is equivalent to the first student’s amount of candy from the array.
   2. Instantiate a Boolean that stores whether the game is over and initialize it with true.
   3. For all the students
      1. If a student’s amount of candy is not equal to the first student’s, then the game is not over.
   4. Return whether the game is over or not.