**Lab Assignment 1  
Candy Game**

|  |
| --- |
| **Due Date** |
| **Wednesday 9/18/19 @ 11:59pm** |

**Objectives**

* Review of CS1110
* Review of arrays (using an array to simulate a circle of objects)
* Review of methods and classes
* Using pseudocode in algorithm development

**Problem Specification**

Develop a Java application to simulate a game played in an elementary classroom. In this game, the teacher places herself in the center of a circle of students surrounding her. She then distributes an even number of pieces of candy to each student. Not all students will necessarily receive the same number of pieces; however, the number of pieces of candy for each student is even and positive. When the teacher blows a whistle, each student takes half of his or her candy and passes it to the student on the right. After they do this, the teacher checks the candy in each student’s hand, and if there is an odd number, she gives one more piece to the student, so that each student again has an even number of pieces of candy. She continues to blow the whistle in this manner, each time making sure the students all have an even number by adding one to those who don’t after the handoff. The game ends when all the students have the same amount of candy after a move.

**Design Requirements**

**Basic Structure**

Your program should have two classes as follows:

1. The main class which has the main method; this class should be called “**Controller**”. It should initiate an instance of the second class and use that instance to call the methods in that class to perform the required actions to play the game.
2. The second class should be called “**CandyGame**”. This class should contain the methods required to perform the necessary actions to play the game. The methods to be included in this class are described below.
   1. A method to get an integer from the user in the range from a lower limit to an upper limit inclusive. This method should receive the two limits as parameters. The integer returned from this method is the number of students playing the game.
   2. A method to get an even integer in the range from an even (not odd) lower limit to an even upper limit inclusive. Again, the upper and lower limits should be passed to this method. (The lower and upper limits passed to the method are the limits for the lower even number which in this case are 4 and 10.)
   3. A method that will distribute a number of pieces of candy to an array of integers (representing the students). The number given to each “student” must be random, even, and between two specified even limits inclusive. You will need to pass the lower and upper limits to this method.
   4. A method to print an array of integers on one line in a field width of size 4.
   5. A method that will pass the candy as described in the game. In essence, you are using an array to represent a circle. Portions (specifically half) of each element in the array are getting added to the element on the right, except for the last element, a portion of which gets added to the first.
   6. A method to test whether or not the game is over. It should return true if all values in the array are the same, otherwise, it should return false.

The execution of your program should follow the outline below.

1. The user should enter the number of students in the class, which is a number from 15 to 30 inclusive.
2. The program will then need an array of integers of that size. Each element of the array will hold information for one student.
3. The user should also enter the smallest number of pieces of candy that will be distributed, which is an even number in the range 4 to 10 inclusive.
4. Finally, the user should enter the largest number of pieces of candy that will be distributed, which is an even number greater than the smallest number (obtained in step 3 above) but less than or equal to the smallest plus 50.
   1. Note that the same method should be used for steps 3 and 4 but the correct lower and upper limits will need to be passed to the method for each method call.
5. After these numbers are entered, deal the candy and then indicate that the game is about to be played.
   1. Before playing, ask the user whether or not to print the array after each move. If yes, print it each time. If no, print only the final configuration.
6. When the game ends, print the final configuration, which should have the same number in all cells of the array.

The output and interface should be neat and well-described. See the two examples provided in the Testing Phase section. All user input must be done with methods and the methods should clearly specify the range for each integer. Description of what is being requested should be done in the main class (Controller).

**Pseudocode**

Write pseudocode for the required methods (as specified above, for the main method in class **Controller,** and for any other methods you may want to use in your code.

*Note:* ***Correct pseudocode development*** *will be worth* ***40%*** *of the total LA grade.*

**Implementation Phase**

Using the pseudocode developed, write the Java code for your assignment. This is a two-week assignment.

**Testing Phase**

* Your program should include code to validate user input and ensure that all input meets the specifications. If input is not valid, your program should keep looping to force the user to provide valid input. Each time your program loops, it should provide information to the user to indicate what is considered valid input.
* While testing, it would be a good idea to print out all configurations (even if the user says not to print) so that you can visualize the progression of your program’s execution and identify possible errors in implementation. Once the program is working correctly, you can then implement the part that prints the intermediate configurations if the user so desires.
* Build your program incrementally, carefully testing each method as you go.

**Example Output 1:**

Getting the number of students.

Enter an integer in [15, 30] inclusive:

14

Must be in [15, 30] inclusive! Re-enter:

31

Must be in [15, 30] inclusive! Re-enter:

16

Getting the lower number of starting candy pieces from 4 to 10.

Enter an even integer in [4, 10] inclusive:

6

Getting the upper number of starting candy pieces.

Must be even and greater than 6 (the lower number) but less than or equal to 56 (the lower number plus 50).

Enter an even integer in [8, 56] inclusive:

25

Must be EVEN and in [8, 56] inclusive! Re-enter:

6

Must be EVEN and in [8, 56] inclusive! Re-enter:

24

The original deal is:

8 20 16 16 18 14 10 22 22 14 10 22 18 18 18 10

We are ready to play the game.

Do you want to print the status after each move? (1 for yes, 0 for no)

Enter an integer in [0, 1] inclusive:

0

20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20

**Example Output 2:**

Getting the number of students.

Enter an integer in [15, 30] inclusive:

20

Getting the lower number of starting candy pieces from 4 to 10.

Enter an even integer in [4, 10] inclusive:

8

Getting the upper number of starting candy pieces.

Must be even and greater than 8 (the lower number) but less than or equal to 58 (the lower number plus 50).

Enter an even integer in [10, 58] inclusive:

40

The original deal is:

30 24 26 10 36 34 26 38 22 20 22 38 18 14 10 22 14 22 36 32

We are ready to play the game.

Do you want to print the status after each move? (1 for yes, 0 for no)

Enter an integer in [0, 1] inclusive:

1

32 28 26 18 24 36 30 32 30 22 22 30 28 16 12 16 18 18 30 34

34 30 28 22 22 30 34 32 32 26 22 26 30 22 14 14 18 18 24 32

34 32 30 26 22 26 32 34 32 30 24 24 28 26 18 14 16 18 22 28

32 34 32 28 24 24 30 34 34 32 28 24 26 28 22 16 16 18 20 26

30 34 34 30 26 24 28 32 34 34 30 26 26 28 26 20 16 18 20 24

28 32 34 32 28 26 26 30 34 34 32 28 26 28 28 24 18 18 20 22

26 30 34 34 30 28 26 28 32 34 34 30 28 28 28 26 22 18 20 22

24 28 32 34 32 30 28 28 30 34 34 32 30 28 28 28 24 20 20 22

24 26 30 34 34 32 30 28 30 32 34 34 32 30 28 28 26 22 20 22

24 26 28 32 34 34 32 30 30 32 34 34 34 32 30 28 28 24 22 22

24 26 28 30 34 34 34 32 30 32 34 34 34 34 32 30 28 26 24 22

24 26 28 30 32 34 34 34 32 32 34 34 34 34 34 32 30 28 26 24

24 26 28 30 32 34 34 34 34 32 34 34 34 34 34 34 32 30 28 26

26 26 28 30 32 34 34 34 34 34 34 34 34 34 34 34 34 32 30 28

28 26 28 30 32 34 34 34 34 34 34 34 34 34 34 34 34 34 32 30

30 28 28 30 32 34 34 34 34 34 34 34 34 34 34 34 34 34 34 32

32 30 28 30 32 34 34 34 34 34 34 34 34 34 34 34 34 34 34 34

34 32 30 30 32 34 34 34 34 34 34 34 34 34 34 34 34 34 34 34

34 34 32 30 32 34 34 34 34 34 34 34 34 34 34 34 34 34 34 34

34 34 34 32 32 34 34 34 34 34 34 34 34 34 34 34 34 34 34 34

34 34 34 34 32 34 34 34 34 34 34 34 34 34 34 34 34 34 34 34

34 34 34 34 34 34 34 34 34 34 34 34 34 34 34 34 34 34 34 34

**Additional Requirements**

**Coding Standards**

You must adhere to all conventions in the CS 1120 Java coding standard (available on Elearning for your Lab). This includes the use of white spaces and indentations for readability, and the use of comments to explain the meaning of various methods and attributes. Be sure to follow the conventions also for naming classes, variables, method parameters and methods.

**Assignment Submission**

* Generate a .zip file that contains all your files including:
  + Program Files
  + Any input or output files
  + Detailed pseudocode for your program

You can refer to the link provided for instructions on how to export your project to a .zip file (“[How to submit a programming assignment](http://www.cs.wmich.edu/~wwshen/cs1120_Fall2012/documents/HowtoSubmitPAinEclipse.pdf)” ).

* Submit the .zip file to the appropriate folder on ELearning.

**NOTE**: The eLearning folder for LA submission will remain open beyond the due date but will indicate how many days late an assignment was submitted where applicable. The dropbox will be inaccessible seven days after the due date by which time no more credit can be received for the assignment.

The penalty for late submissions as stated in the course syllabus will be applied in grading any assignment submitted late.