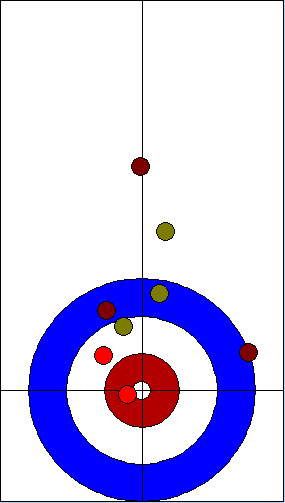
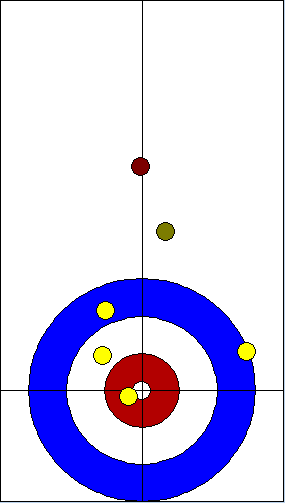
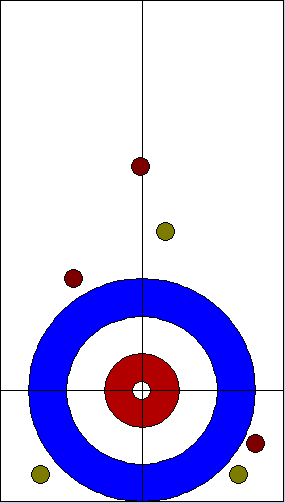
Curling Lab – More Practice with Arrays

In this lab, you will be creating a program which displays the results of a curling match. Curling is a sport played on ice with granite stones and brooms. In a game of curling, two teams of 4 compete against each other to try to slide their stones (44 pounds, 1 foot in diameter, red for one team, yellow for the other) closer to the center of a target on the ice. A game is divided into “ends” like the frames in a game of bowling or innings in baseball. A typical game will have 8 or 10 ends. In each end, the two teams alternate delivering (sliding) stones down the ice. Each end, each team delivers 8 stones (for a total of 16). Some of the stones will land in play, others will land out of play. Sometimes, a stone in play will be hit by a later stone and removed. After all 16 stones have been thrown, one team will score points based on how many of their stones are in the “house” (target).

Scoring an End

The diagrams below show the final positions of several stones after different ends (the diagrams are as if you are looking down on the ice from above):

In the first diagram, there are 5 red stones and 3 yellow stones. Six of these stones (4 red, 2 yellow) are “In the house” meaning they are touching part of the target area. Of these stones, the two closest to the “tee” (center of the target) are red (note the lighter color in the diagram), the third closest is yellow, then another red, then another yellow, then the last red. In this end, the red team will score 2 points because they had 2 stones in the house which were closer to the tee than the yellow team’s closest stone. The yellow team scores nothing. Note that:

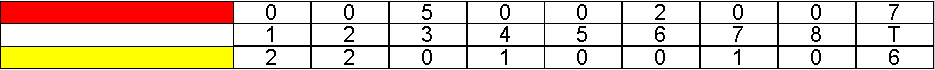
1. Not all the stones in the house count… only the ones that are closer than the other team’s closest stone
2. Only one team scores. The other team gets nothing.

In the second diagram, the yellow team scores 4 points because they have 4 stones in the house closer than the red team’s closest stone. In this end, since the red team didn’t have any stones in the house, all of the yellow team’s stones that are touching the house score. Also note, that a stone only has to touch the house to be eligible to score (notice the yellow stone on the far right).

In the third diagram, there are no stones in the house, so neither team scores any points. This is called a “blank end”.

The Scoreboard

A game in curling consists of several (usually 8 or 10) ends. After each end (16 total stones are thrown), one team scores points, and then the playing area is cleared and the teams play the next end. The score is usually recorded on a scoreboard. There are two styles of scoreboard. The first is similar to what you would see in a baseball game:



In this style of scoreboard, the ends (numbered 1-8) are labeled in the center of the board with each team’s score for that end recorded in the top (red) or bottom (yellow) line. Each team’s total is shown on the far right. In this game, the yellow team scored 2 points in the first end, then two more points in the second end. The red team answered with 5 points in the third end. Then yellow scored 1 in the fourth end. The fifth end was blank (neither team scored). Red scored 2 in the sixth, yellow scored 1 in the seventh, and the eighth end was blank. Red scored a total of 7 (5+2) points, yellow scored a total of 6 (2+2+1+1) points. Red won. This style of board is typically seen at major curling events (like national or world championships, the Olympics etc) and is usually displayed electronically. The reason for this is that a mechanical baseball-style board in which you hang number-plaques for scores above and below the end number would require a large number of numbers to have ready to display on the board. When implemented electronically, however, the numbers can just be printed or drawn to the screen as needed. You don’t need a supply of numbers, many of which will be used very rarely if ever. While the baseball-style boards are easy and intuitive to read, the major drawback is that they are either very expensive (electronic) or very cumbersome (mechanical).

The second style of boards, used mainly at the club level, solves these problems with the tradeoff of being slightly less intuitive to read. Here’s a club-style scoreboard for the same game:



In this scoreboard, the middle line indicates the SCORE rather than the END number. The numbers in the first and third line indicate the END in which a particular total score was attained. For example, in this game, yellow scored 2 points in the first end. After the first end, yellow had scored a total of 2 points, so a 1 is placed in the second box from the left on the third line to indicate this. There is a 2 in the 4th box on the third line. This indicates that yellow scored in the second end and that their total score after the second end was 4. In other words, they scored 2 in the second end, so the 2 (for the end number) is placed 2 spaces (for the score) to the right of their previous scoring marker. Red scored 5 in the third end for a total score of 5 so far, so the 3 (end) is placed in the 5th (score) box on the top row. Yellow scores 1 in the fourth end for a total of 5, so the 4 (end) is placed under the 5 (total score). You can tell that yellow scored 1 in the fourth end because the 4 is 1 box to the right of the 2 (the previous time that yellow scored). Neither team scored in the 5th end, so there is no 5 in the board. Red scored 2 in the sixth end for a total of 7, so the 6 is in the 7th box on the top row. Yellow scored 1 more for a total of 6 in the seventh end, so the 7 is in the sixth box on the bottom row. Finally, the eighth end was blanked, so there is no 8 in either row. The total score can be read quickly by noting where the right most number in each row is: Red has 7 points, Yellow has 6. As mentioned earlier, the club-style board is less intuitive. However, it has the chief advantage of only needing 8 total end numbers (1 – 8) to put on the board, most of which will be used every game. This means that there are much less numbers to keep track of.

The Stone Class:

Class Stone is already written for you. It contains methods to get its color (Color.RED or Color.YELLOW) and the x and y coordinates of the center of the stone. The x and y coordinates give the position, in feet, relative to the tee (center of the target). That is, a stone that is exactly on the tee would have position (0.0, 0.0). A stone 4 feet in front of the tee would have position (0.0, 4.0).

public class Stone {

. . .

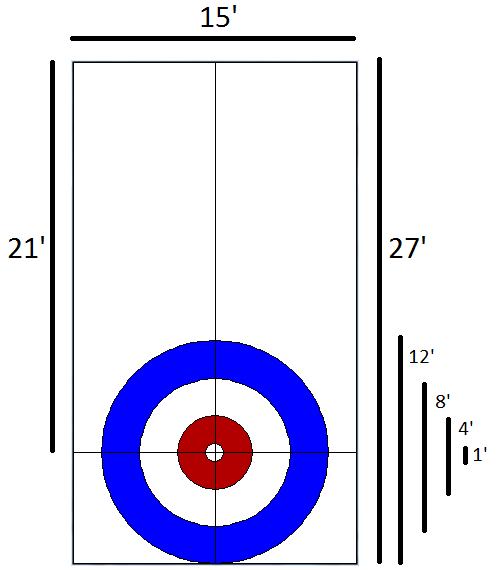
public Color getColor() { . . . }

public double getX() { . . . }

public double getY() { . . . }

. . .

}

Exercise 1: CurlingSheetView Part 1

Class CurlingSheetView has a paintSheet method which consumes a Graphics object and two ints which are the screen width and height of the sheet (the playing surface in curling is called a “sheet”). This method should draw the house (target) and the center and tee lines (the black lines which run vertically and horizontally through the tee (the center of the house). Also, draw in the bounding box for the whole (visible) sheet. The lines should all be black, and the circles should be outlined in black. The blue circle is Color.BLUE. The red circle is one shade darker than Color.RED (the Color class has a method called darker which returns a new color which is one shade darker).

It’s important to get the scale correct here. In real life, the sheet is 15 feet across and the portion of the sheet in which we will be interested (from the hog line (top of the diagram) to the base line (bottom of the diagram)) is 27 feet long. The tee line is 21 feet from the hog line. The large blue circle is 12 feet in diameter. The white circle is 8 feet in diameter. The red circle is 4 feet in diameter and the white circle around the tee is 1 foot across. Your picture should be to scale using the provided height and width parameters.

To test, use the runEndCreator method in class Curling (there should be a line for it in the main function). You should see a window with the picture above.

Exercise 2: CurlingSheetView Part 2

Class CurlingSheetView has a second method called paintStone which consumes a Graphics object, g, a width, a height, a Stone to paint, and a boolean called highlight. Use the x and y coordinates of the stone with the provided width and the height of the sheet to draw a red or yellow circle (depending on the color of the stone) in the appropriate location. If highlight is true, use either Color.RED or Color.YELLOW depending on the stone’s color. If highlight is false, use colors which are two shades darker.

The paint method consumes an EndResult. Class EndResult provides several methods:

public class EndResult {

. . .

public Stone [] getStones() { . . . }

public boolean isScoringStone( Stone s ) { . . . }

. . .

}

The getStones method returns an array of the Stones in play. The isScoringStone consumes a Stone and returns true if that stone is a scoring stone (ie is in the house and closer than all the stones for the other team). You will implement isScoringStone later, but for right now it returns true.

In the paint method for CurlingSheetView, you will need to get the array of Stones from the EndResult. For each stone in the array, call paintStone. If the stone is a scoring stone, use true for the highlight parameter. Otherwise, use false.

To test this out, you should run the runEndCreator function again. This time, click in the window. A red stone should be added, centered at your cursor. It should be dark red. Press the ‘y’ key on your keyboard (this will switch colors to add yellow stones instead of red stones), and click in a different spot. There should be a dark yellow stone centered at your cursor. You can move already created stones around the window by clicking on them to pick them up, and clicking again to set them down. To delete a stone, press ‘d’ then click the stone. To return to “pick-up mode”, press ‘p’.

Exercise 3: CurlingEndResult Part 1

In this part, you will implement several of the methods of CurlingEndResult. Class CurlingEndResult contains an array of Stones called stones. All of the functions (except distanceToCenter which is static, and anyway doesn’t need access to the array of stones) have direct access to this variable.

public class CurlingEndResult {

. . .

private Stone [] stones;

. . .

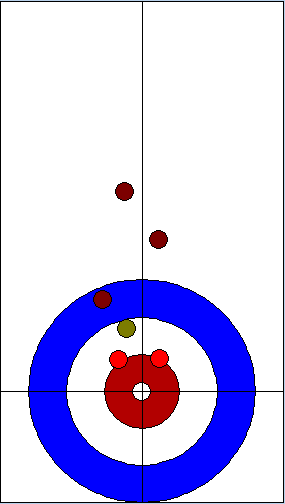
}

The first method to implement is distanceToCenter. This method consumes a Stone and returns the distance from the center of the stone to the center of the house. Use the distance formula:



And don’t forget that the tee is (0, 0) and that the Math class has some helper functions that will be useful here.

Next, implement method isScoringStone. This method consumes a Stone, s, and returns true if s is scoring. That is, s is in the house (what is the maximum distance it’s center can be from the tee?) and there are no stones of the other color which are closer. There are some hints in the comments for the function.

To test these two functions, run the runEndCreator method again. Create this picture:

Make sure that the two red stones near the center are brighter than the other red stones. Remove these two stones (use the delete function by pressing ‘d’ and clicking on the stones) to make sure that the yellow stone is now bright. Delete this stone to make the third red stone in the house bright. Move this red stone to the edge of the house to make sure it stays bright as long as it’s touching the house but goes dark when it’s outside the house.

Exercise 4: CurlingEndResult Part 2

In this part you will implement several more functions in CurlingEndResult to help calculate the score for the end.

getDistanceToClosestStone - consumes a Color and returns the distance to the closest stone for that color or Double.MAX\_VALUE if there are no stones of that color in play.

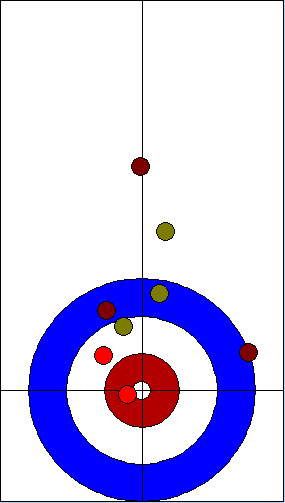
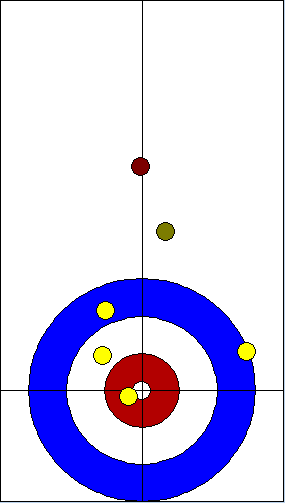
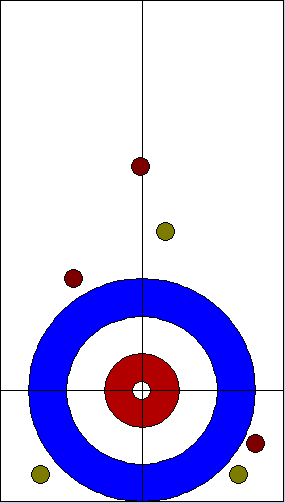
getNumScoringStones - consumes a Color and returns the number of stones of that color that are scoring. That is, the number of stones of this color that are in the house and closer to the tee than all of the other team’s stones. Hint: you can use isScoringStone and/or getDistanceToClosestStone to make this function very simple. Also, note that if the yellow team has scoring stones, then the red team has 0 scoring stones and vice versa.

getEndScore – this method returns the score for the end. That is, the number of rocks that are scoring plus a sign indicating which team is scoring: If the red team is scoring, the score will be positive. If the yellow team is scoring, the score will be negative. If neither team is scoring (there are no stones in the house), the score will be 0. For example, the scores for the diagrams below are:

2 - red is scoring 2 stones, so the score is positive 2

-4 - yellow is scoring 4 stones so the result is -4

0 - blank end

Hint: use the other functions that you have implemented to make this very easy.

To test these three functions, run the runEndCreator function again. Pressing the space bar will pop up a box with the current score.

Exercise 5: CurlingUtilityFunctions Part 1

In this exercise, you will write a function which returns an array of EndResults based on data from a file.

Open the file CurlingGame.txt in a text editor (Notepad will work). You should see this:

8

8

red 0.4225352112676056 10.458167330677291

red 1.426056338028169 8.629482071713149

red 1.8485915492957745 5.241035856573706

red 3.063380281690141 3.358565737051794

red -2.9577464788732395 1.3147410358565745

yellow 0.7394366197183099 4.380478087649403

yellow 0.7922535211267605 1.2609561752988054

yellow -0.4753521126760563 -0.9442231075697205

7

yellow -1.3204225352112675 10.243027888446216

yellow 1.6373239436619718 7.661354581673308

yellow 0.0528169014084507 0.9920318725099608

yellow 1.267605633802817 2.6593625498007976

red -2.112676056338028 7.715139442231076

red -2.9577464788732395 -1.96613545816733

red 2.112676056338028 -4.278884462151394

7

red -1.3204225352112675 11.910358565737052

red -0.8450704225352113 2.498007968127491

red 0.7394366197183099 2.1752988047808772

red 0.7394366197183099 0.8306772908366541

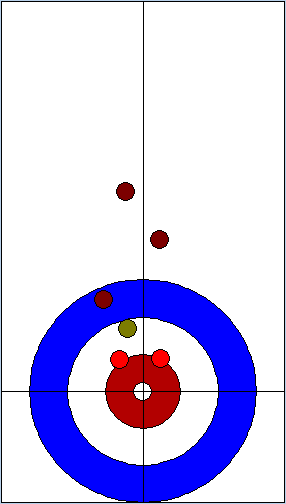
red -0.7394366197183099 -0.8904382470119515

yellow 1.2147887323943662 8.790836653386455

red 2.165492957746479 5.617529880478089

. . .

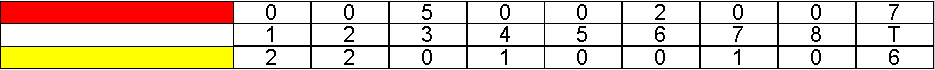
This file describes a complete curling game with several ends. At the beginning of the file is a number which indicates the number of ends in the game. In this case, the game had 8 ends. After that follow several (in this case, 8) sections each of which describes an end. Each section starts with a number indicating the number of stones for that end. For example, the first end had 8 stones in play. After that, there are several (in this case, 8) lines each of which describes a stone. Each line contains a color (either “red” or “yellow”) followed by two doubles indicating the x and y coordinate of the center of the stone.

readCurlingGameFromFile – this method consumes a String which represents a file name and returns an array of CurlingEndResults. Fairly detailed instructions are in the comments for the function.

To test, run the runEndCreator function again. Press ‘l’ to load ends from the file “CurlingGame.txt”. You should see the diagram to the right.

Exercise 6: CurlingUtilityFunctions Part 2

In this exercise, you will implement function getBaseballStyleLine which consumes a curling game (an array of CurlingEndResults) and a team color (RED or YELLOW) and produces an array of ints which represents the number of points that team scored in each end of the game plus the team’s total for the whole game. That is, if the game has N ends, then the result of this function is an array with N+1 entries. The first N entries will be the team’s score in each of the N ends and the N+1st entry will be the total score for the team for the whole game. For example, in the game of 8 ends depicted by this scoreboard:

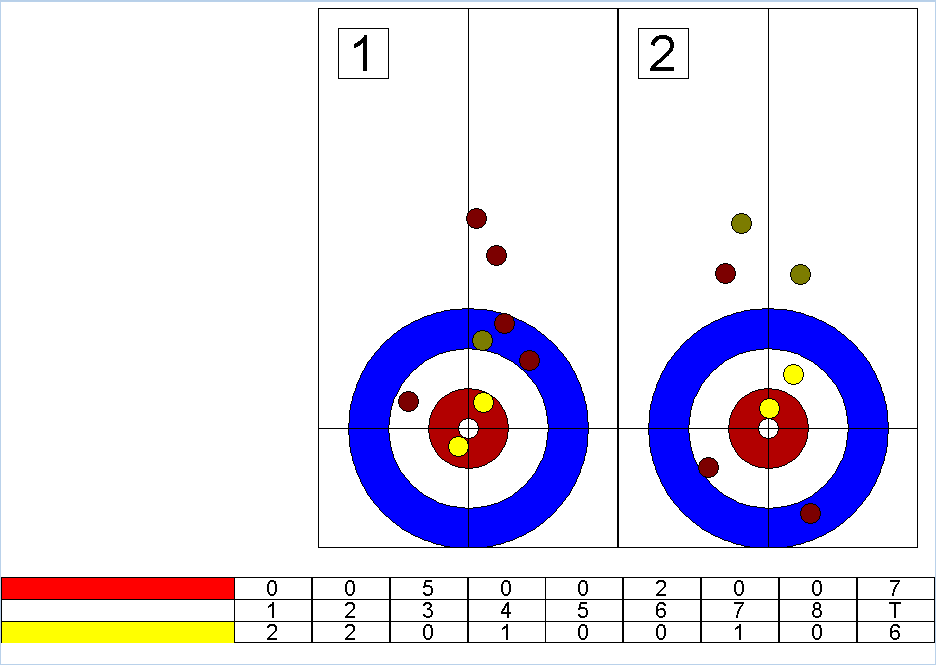


getBaseballStyleLine( game, Color.RED ) will produce an array of 9 entries: { 0, 0, 5, 0, 0, 2, 0, 0, 7 }

and

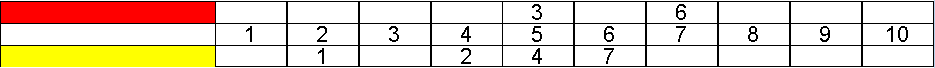
getBaseballStyleLine( game, Color.YELLOW) will produce the array { 2, 2, 0, 1, 0, 0, 1, 0, 6 }

To test your function, run the runGameViewer function in main. You should see this:



Exercise 7: CurlingUtilityFunctions Part 3

In this exercise, you will implement getClubStyleLine which consumes a curling game (array of CurlingEndResults), a team color (RED or YELLOW) and a maxScore, and produces an array of ints which has maxScore entries representing the score line on a club-style scoreboard. For example, in the game above, the club-style board would look like this:



getClubStyleLine( game, Color.RED, 10 ) would produce the array { 0, 0, 0, 0, 3, 0, 6, 0, 0, 0 }

and

getClubStyleLine( game, Color.YELLO, 10 ) would produce the array { 0, 1, 0, 2, 4, 7, 0, 0, 0, 0 }

Hint: As you traverse the array of end results, the *cumulative* score for the team gives the index in the array while the end number gives the entry at that index.

To test, run runGameViewer again and press the space bar. This will change the scoreboard from baseball style to club style. You should see this:

