Shuffleboard.java and ShufflePlayer.java

Shuffleboard is a classic game in which two players play against each other. One earns points by getting a biscuit (a sort of puck) in a region on the playing surface. For the digital version of the game, the surface will look like this, showing that a biscuit is waiting to be launched at the left:

x - - - - - - 10 8 7 -10
Your score is 0.
Press Enter to push your biscuit.

A launched biscuit on the surface will look something like this, which would be in response to generating the random number 4. This got the player no points:

Random number is 4.

- - - - x - - 10 8 7 -10
Your score is 0.

In our first version (Step 1), you'll push your biscuit once, then the game will stop.

Here's another version of what could happen after I pressed enter. The biscuit that got its location after I generated random number 8. This got the player 8 points:

Random number is 8.

- - - - - - - 10 x 7 -10
Your score is 8.

Since we have two players, player A and player B, we'll put names on their biscuits. When it's A's turn, at the beginning of the game, this will be shown. (Don't worry about showing later pucks in the leftmost positions to start, as in the basic version; we can only play one piece of a type at a time.):

A - - - - - - 10 8 7 -10
A's score is 0.
B's score is 0.
No one is winning.
A, press Enter to push your biscuit, -1 to quit.

This shows that the biscuit is on the playing surface, ready to be pushed towards the numbers. I hope to land on 10 so I can earn 10 points! If I land on 7, I'll earn only 7 points. If I land on -10, I will lose 10 points.

Random number is 9.

- - - - - - - 10 8 A -10
A's score is 7.
B's score is 0.
A is winning.
B, press Enter to push your biscuit, -1 to quit.

Random number is 10.

- - - - - - - 10 8 A B

A's score is 7.
B's score is -10.
A is winning.
A, press Enter to push your biscuit, -1 to quit.

Random number is 3.
- - - A - - - 10 8 7 B
A's score is 7.
B's score is -10.
A is winning.
B, press Enter to push your biscuit, -1 to quit.

Random number is 7.
- - - A - - - 10 8 7 B
A's score is 7.
B's score is 0.
A is winning.
A, press Enter to push your biscuit, -1 to quit.
-1

Let's see if we all follow how to play the basic shuffleboard game as described above. Answer the following questions with a neighbor:

a) What is everyone's score when we start?
b) How does one earn points?
c) If a single player were to play three times, what would be the highest possible score? The lowest possible score?
d) A biscuit shows up on the far left of the playing surface when?
e) What user input is used to push a biscuit?
f) How many players are there?
g) What is the sentinel value?
h) A player will be an object of its own class. What kind of variables will each player need?
i) How many biscuits can show up on the playing surface at one time?
j) A plays and gets random number 2. On its next turn, A plays and gets random number 3. Where does A end up? *Use ideas above.*
k) If B's biscuit is pushed farther than A's biscuit, does A's biscuit get pushed by B? *Use ideas above.*
l) You'll use loops to do several things when this gets programmed. See if you can reason which will happen iteratively (repetitively). Put a 1 next to each of the following if they happen 1 time per turn (but over and over, as several turns are taken), or 2 if they happen several times per turn (and this is further magnified when more turns are taken). If the even does not repeat, don't put a number next to it.

__Print the spots on the playing surface
__Change whose turn it is
__Calculate the score
__ Make a player A
__ Print a biscuit
__ Print a prompt to press Enter
__ Generate a random number
__ Print a score
__ Print a biscuit on the far left of the playing surface

We'll plan and program this in steps. Take time to review Step 1, below, then write some pseudocode. Use Pandas Don't Eat Oreos conventions! Methods you can reuse and very clear identifiers are a superb idea – you should be really thinking through the pseudocode to describe blocks (methods, control structures) so it's easy to describe your program, well before you type any code!

Step 1: Make a Shuffleboard game work for one person, one time:

1. Make a ShufflePlayer object (that means you'll need a ShufflePlayer class). Construct it using a name (e.g. "A") and give it a score of zero to start. It will also need a third variable for position. Be sure that this class has the constructor with no parameters as well as the constructor that uses the name. Be sure you use the constructor as taught in class to assign values to the object. Think about if your field variables should be private (convention) or if they will need to be public.
2. Draw the playing surface before play. Draw the biscuit on the far left as 'x'. You should be using a bunch of if/else-if conditions and statements in a single while or for loop. You will print something each cycle of the loop. What you print depends on what cycle of the loop it is; if it's the 7^{th} cycle of the loop, print 10. Note that there are a range of values where you will print '-', so write some efficient code to do this.
3. Print the score (it should be zero since you have not played yet) below the playing surface.
4. Upon user input of a "return," generate a random number using Math.random(). This will indicate where the biscuit will land. There are 10 possible places, so make 10 possible numbers. Print the random number once it is generated.
5. Draw the playing surface <u>with</u> the biscuit on it. Add to your loop so there is a check to see if the biscuit's associated random number is the same as the loop cycle you are on, and if so, print 'x' instead of what you would normally print on the playing surface. This should require only one new condition inside the body of the loop.
6. Outside the loop, calculate the score based on where the biscuit is on the playing surface. The player should have started with 0, and now you will add the score to the earlier score. Yes, players can have negative scores. Print the score below the playing surface.

Step 2: Make the Shuffleboard game work for two people, starting a new biscuit each time:

1. **Add to your pseudocode before starting to code this, so you don't get confused about what is the same, and what needs to change.**
2. Building from what you created above, make sure the original biscuit is shown as 'A' instead of 'x.' Build two ShufflePlayer objects (not one, as in Step 1), each with their own zero score, position, and different name to start – you should not need to change the ShufflePlayer.java code to do this. Draw a single playing surface and print both A and B's scores below the playing surface.
3. Add a DO-WHILE loop on the outside, such that the loop you wrote before will be nested inside it. This loop allows us to know if it's A's turn or B's turn. When the loop count is odd (starting from 1), prompt for user A to press Enter, then print the appropriate A biscuit as moving in response to a random number being made. When the loop count is even, prompt for user B and show biscuit B as moving in response to a random number. Stop the DO-WHILE loop when the user types -1 (a sentinel value). Since you were reading in Enter before, you will want to use as a condition for a DO WHILE LOOP:

   ```
   ! (input.equals("-1")
   ```

   It's easiest to have a temporary variable for the biscuit name (either A or B) and a temporary variable for the score (either A's score or B's score) that you can toggle between depending on which cycle of the DO-WHILE loop you are running.
4. Tell who is winning: A or B.

Some notes about Step 2's version of play:

- Let's assume there is a playing surface and the biscuits cannot hit each other, for this version.
- A will show up on the left when you first begin, but after that, we won't show a new biscuit on the left, so it's easier to program as we progress; we don't intend to have more than two biscuits ever on the playing surface.
- Note that random numbers are not additive; biscuits are always thrown from the far left.
- If two biscuits land on the same spot, the last one to be thrown appears.
- Your programming teacher is not endorsing throwing food; if you simulate this game at home with one biscuit (Step 1), your mom might not say anything, but if you try to play with two (Step 2), you are bound to make her angry and made to think that you are wasting food. Biscuits here are not food, anyhow. ☺ Still, playing shuffleboard makes me hungry for biscuits!

Step 3: Make the Shuffleboard game work for two people, multiple times (**Extra Credit**):

1. Now let's throw a new biscuit on that can cause the old biscuit to move, if it is struck. This is for bonus credit. No arrays or lists allowed. Let's only use what we already know how to do in this class. This will require some additional conditions. **Save Step 2's version so you have it functioning and complete before you try Step 3. DO NOT OVERWRITE Step 2's version.** How it works:
    a. Pseudocode is required!
    b. After one biscuit has already been thrown by each player the usual way, the user can decide to throw hard, regular, light, or rethrow. Read the input with the same method and Scanner you were using before. Remember we use .equals() to check Strings.
    c. Throwing hard means that IF you equal or exceed your old biscuit's value, it will jump to the right 4 places. Print the calculated jump next to the random number generated, so you can check your work.
    d. Throwing regular means that IF you equal or exceed your old biscuit's value, it will jump to the right 2 places. Print the calculated jump next to the random number generated, so you can check your work.
    e. Throwing light means that IF you equal or exceed your old biscuit's value, it will jump to the right 1 place. Print the calculated jump next to the random number generated, so you can check your work.
    f. Throwing rethrow means that you throw again just like in Step 2 (above); there will be a complete replacement of your old throw with this new throw.
    g. All four types of throws generate the random number the same way as before.
    h. If you throw hard, regular, or light and do not exceed the old biscuit's value, your old biscuit will stay where it was. You will not collect points again (as you did not move).
    i. There are still only two biscuits (maximum) ever shown on the playing surface.
    j. If you throw (either of the four kinds) so the biscuit lands where the other player's biscuit was, the moving biscuit takes that position and the old biscuit disappears (just like in Step 2, above).
    k. When your biscuit moves, add points to your score.
    l. Assume player A can only move player A's biscuits, and player B can only move player B's biscuits, UNLESS a player lands on another player's position (as described above and in Step 2).
    m. If your placement on the playing surface exceeds 10, the biscuit should not be shown and you should print "____'s biscuit fell off the playing surface." and indicate who won (the blank is either A or B). The game should terminate (set your variable to the sentinel value or add to your DO-WHILE condition so it can use another sentinel value.

2. You can do part or all of the extra credit, to earn some extra points. You will have to print your code and turn it in with a written English statement about how you accomplished each of the requirements in Step 3, to earn your points.

Here's some sample output of Step 3, to help address any questions:

A - - - - - - 10 8 7 -10
A's score is 0.
B's score is 0.
No one is winning.
A, press Enter to push your biscuit.

Random number is 4.

- - - - A - - 10 8 7 -10
A's score is 0.
B's score is 0.
No one is winning.
B, press Enter to push your biscuit.

Random number is 6.

- - - - A - B 10 8 7 -10
A's score is 0.
B's score is 0.
No one is winning.
A, type hard, regular, or light for strength of throw to knock an old biscuit, or rethrow to throw brand new biscuit. Type -1 to quit.
hard

Random number is 5. Biscuit jumped 4 places to 7.
- - - - - - B A 8 7 -10
A's score is 10.
B's score is 0.
A is winning.
B, type hard, regular, or light for strength of throw to knock an old biscuit, or rethrow to throw brand new biscuit. Type -1 to quit.
light

Random number is 9. Biscuit jumped from 6 to 7.
- - - - - - - B 8 7 -10
A's score is 10.
B's score is 10.
No one is winning.
A, type hard, regular, or light for strength of throw to knock an old biscuit, or rethrow to throw brand new biscuit. Type -1 to quit.
rethrow

Random number is 2.

- - A - - - - B 8 7 -10

A's score is 10.

B's score is 10.

No one is winning.

B, type hard, regular, or light for strength of throw to knock an old biscuit, or rethrow to throw brand new biscuit. Type -1 to quit.

light

Random number is 1. Biscuit did not jump.

- - A - - - - B 8 7 -10

A's score is 10.

B's score is 10.

No one is winning.

A, type hard, regular, or light for strength of throw to knock an old biscuit, or rethrow to throw brand new biscuit. Type -1 to quit.

-1