Q0416

1. SHOW ALL YOUR WORK. REMEMBER THAT PROGRAM SEGMENTS ARE TO BE WRITTEN IN JAVA.

- Assume that the classes listed in the Java Quick Reference have been imported where appropriate.
- Unless otherwise noted in the question, assume that parameters in method calls are not `null` and that methods are called only when their preconditions are satisfied.
- In writing solutions for each question, you may use any of the accessible methods that are listed in classes defined in that question. Writing significant amounts of code that can be replaced by a call to one of these methods will not receive full credit.

This question involves the creation and use of a spinner to generate random numbers in a game. A `GameSpinner` object represents a spinner with a given number of sectors, all equal in size. The `GameSpinner` class supports the following behaviors.

- Creating a new spinner with a specified number of sectors
- Spinning a spinner and reporting the result
- Reporting the length of the current run, the number of consecutive spins that are the same as the most recent spin

The following table contains a sample code execution sequence and the corresponding results.

Statements	Value Returned (blank if no value returned)	Comment
`GameSpinner g = new GameSpinner(4);`		Creates a new spinner with four sectors
`g.currentRun();`	`0`	Returns the length of the current run. The length of the current run is initially `0` because no spins have occurred.
`g.spin();`	`3`	Returns a random integer between `1` and `4`, inclusive. In this case, `3` is returned.
`g.currentRun();`	`1`	The length of the current run is `1` because there has been one spin of `3` so far.
`g.spin();`	`3`	Returns a random integer between `1` and `4`, inclusive. In this case, `3` is returned.
`g.currentRun();`	`2`	The length of the current run is `2` because there have been two `3`s in a row.
`g.spin();`	`4`	Returns a random integer between `1` and `4`, inclusive. In this case, `4` is returned.
`g.currentRun();`	`1`	The length of the current run is `1` because the spin of `4` is different from the value of the spin in the previous run of two `3`s.
`g.spin();`	`3`	Returns a random integer between `1` and `4`, inclusive. In this case, `3` is returned.
`g.currentRun();`	`1`	The length of the current run is `1` because the spin of `3` is different from the value of the spin in the previous run of one `4`.
`g.spin();`	`1`	Returns a random integer between `1` and `4`, inclusive. In this case, `1` is returned.
`g.spin();`	`1`	Returns a random integer between `1` and `4`, inclusive. In this case, `1` is returned.
`g.spin();`	`1`	Returns a random integer between `1` and `4`, inclusive. In this case, `1` is returned.
`g.currentRun();`	`3`	The length of the current run is `3` because there have been three consecutive `1`s since the previous run of one `3`.

Write the complete `GameSpinner` class. Your implementation must meet all specifications and conform to the example.

Please respond on separate paper, following directions from your teacher.

Q0416

GameSpinner class

Select a point value to view scoring criteria, solutions, and/or examples and to score the response. +1 indicates a point earned and -1 indicates a general or question-specific penalty.

+1 [Skill 3.B] Declares all appropriate `private` instance variables

+1 [Skill 3.B] Declares method headers: `public int spin()` and `public int currentRun()`

+1 [Skill 3.B] Declares header: `GameSpinner (int __)` (*must not be* `private`)

+1 [Skill 3.B] Constructor initializes instance variable for number of sectors using parameter. Instance variables for previous spin and length of current run initialized correctly when declared or in constructor with default values.

Responses still earn the point even if they...

· declare instance variables incorrectly.

+1 [Skill 3.A] Computes random integer [1, number of sectors]

+1 [Skill 3.C] Compares new spin and last spin to determine required updates to state

Responses still earn the point even if they...

· use an incorrectly computed random integer for new spin; or

· incorrectly declare the instance variable intended to store last spin.

+1 [Skill 3.C] Updates instance variable that represents length of current run appropriately if new spin and previous spin are the same

Responses still earn the point even if they...

· incorrectly compare new spin and last spin.

+1 [Skill 3.C] Updates previous spin and length of current run appropriately when new spin differs from the previous spin

Responses still earn the point even if they...

· incorrectly compare new spin and last spin.

Q0416

+1 [Skill 3.B] `currentRun` returns updated instance variable value

Responses still earn the point even if they...

· incorrectly update instance variables in the `spin` method.

-1 (w) Extraneous code that causes side-effect (e.g., printing to output, incorrect precondition check)

-1 (x) Local variables used but none declared

-1 (z) Void method or constructor that returns a value

Canonical Solution:

```
public class GameSpinner
{
   private int sectors;
   private int previousSpin = 0;
   private int currentLength = 0;

   public GameSpinner(int s)
   {
      sectors = s;
   }

   public int spin()
   {
      int newSpin = (int)(Math.random() * sectors)
                    + 1;

      if (newSpin == previousSpin)
      {
         currentLength++;
      }
      else
      {
         previousSpin = newSpin;
         currentLength = 1;
      }
      return newSpin;
   }

   public int currentRun()
   {
      return currentLength;
   }
}
```

0	1	2	3	4	5	6	7	8	9 ✓

Total number of points earned (minus penalties) is equal to 9.

Q0416

- ☐ **+1** Declares all appropriate `private` instance variables **(points earned)**
- ☐ **+1** Declares method headers: `public int spin()` and `public int currentRun()` **(points earned)**
- ☐ **+1** Declares header: `GameSpinner (int __)` (*must not be* `private`) **(points earned)**
- ☐ **+1** Constructor initializes instance variable for number of sectors using parameter. Instance variables for previous spin and length of current run initialized correctly when declared or in constructor with default values. **(points earned)**
- ☐ **+1** Computes random integer [1, number of sectors] **(points earned)**
- ☐ **+1** Compares new spin and last spin to determine required updates to state **(points earned)**
- ☐ **+1** Updates instance variable that represents length of current run appropriately if new spin and previous spin are the same **(points earned)**
- ☐ **+1** Updates previous spin and length of current run appropriately when new spin differs from the previous spin **(points earned)**
- ☐ **+1** `currentRun` returns updated instance variable value **(points earned)**
- ☐ **-1** [penalty] (w) Extraneous code that causes side-effect (e.g., printing to output, incorrect precondition check) **(General penalties)**
- ☐ **-1** [penalty] (x) Local variables used but none declared **(General penalties)**
- ☐ **-1** [penalty] (z) Void method or constructor that returns a value **(General penalties)**

Canonical Solution:

Q0416

```
public class GameSpinner
{
   private int sectors;
   private int previousSpin = 0;
   private int currentLength = 0;

   public GameSpinner(int s)
   {
      sectors = s;
   }

   public int spin()
   {
      int newSpin = (int)(Math.random() * sectors)
                    + 1;

      if (newSpin == previousSpin)
      {
         currentLength++;
      }
      else
      {
         previousSpin = newSpin;
         currentLength = 1;
      }
      return newSpin;
   }

   public int currentRun()
   {
      return currentLength;
   }
}
```