3_F1

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 a game that is played with multiple spinners. You will write two methods in the `SpinnerGame` class below.

```
public class SpinnerGame
{
     /** Precondition: min < max
       * Simulates a spin of a spinner by returning a random integer
       * between min and max, inclusive.
       */
     public int spin(int min, int max)
     { /* to be implemented in part (a) */ }

     /** Simulates one round of the game as described in part (b).
       */
     public void playRound()
     { /* to be implemented in part (b) */ }
}
```

(a) The `spin` method simulates a spin of a fair spinner. The method returns a random integer between `min` and `max`, inclusive. Complete the `spin` method below by assigning this random integer to `result`.

```
/** Precondition: min < max
 * Simulates a spin of a spinner by returning a random integer
 * between min and max, inclusive.
 */
public int spin(int min, int max)
{
     int result;
  return result;
}
```

In each round of the game, the player and the computer each spin a spinner. The player spins a spinner numbered 1 to 10 , inclusive, whereas the computer spins a spinner numbered 2 to 8, inclusive.

Based on the results of the spins, a message is printed in the formats shown in the examples below.

3_F1

If the player obtains a higher result than the computer, the player gains a number of points equal to the positive difference between the spins. If the computer obtains a higher result than the player, the player loses a number of points equal to the positive difference between the spins.

In the event of a tie, the player and the computer each spin the spinner a second time. If the sum of the player's two spins are greater than the sum of the computer's two spins, the player gains one point. If the sum of the computer's two spins are greater than the sum of the player's two spins, the player loses one point. In the event of a tie after two spins, the round is reported as a tie and the player's score does not change.

Examples of the `playRound` method's intended behavior are shown in the following table.

Player Spin #1	**Computer Spin #1**	**Player Spin #2**	**Computer Spin #2**	**Printed String**
9	6			`You win! 3 points`
3	7			`You lose. -4 points`
4	4	6	2	`You win! 1 points`
6	6	1	2	`You lose. -1 points`
1	1	8	8	`Tie. 0 points`

(b) Complete the `playRound` method below. You must use the `spin` method appropriately in order to earn full credit.

```
/** Simulates one round of the game as described in part (b).
 */
public void playRound()
```

Part A - spin method (2 points)

Points earned:

+1 [Skill 3.C] Determine the correct number of possible random numbers in the range.

Response earns this point, but incurs general penalty x below if it...

· creates a local variable for the number of random choices, but does not declare it.

+1 [Skill 3.A] Generates a random integer in the determined range, inclusively, and is stored in result to be returned.

Response earn this point if it…

· redeclares result .

3_F1

Response do not earn this point if it…

· does not appropriately cast the random number to an int .

General Penalties:

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

-1 (x) Local variables used but none declared

Canonical Solution:

```
public int spin(int min, int max)
{
   int result;
   result = (int) (Math.random() * (max - min + 1))
            + min;
   return result;
}
```

0	1	2 ✓

Total number of points earned (minus penalties) is equal to 2.

- ☐ **+1** Determine the correct number of possible random numbers in the range. **(Points earned)**
- ☐ **+1** Generated random integer is in the determined, inclusive, and is stored in result to be returned **(Points earned)**
- ☐ **-1** (w) Extraneous code that causes side-effect (e.g., printing to output, incorrect precondition check) **(General Penalties)**
- ☐ **-1** (x) Local variables used but none declared **(General Penalties)**

Canonical Solution:

```
public int spin(int min, int max)
{
   int result;
   result = (int) (Math.random() * (max - min + 1))
            + min;
   return result;
}
```

Part B - playRound method (7 points)

Points earned:

+1 [Skill 3.A] Calls the spin method appropriately to generate first spins for the player and the computer

3_F1

Response earns this point, but incurs general penalty x below if it…

· Uses local variables for the spin of player and computer without declaring them,
+1 [Skill 3.C] Checks for a tie on the first pair of spins

+1 [Skill 3.C] Reports a player win or a computer win, if appropriate, based on only the first pair of spins

+1 [Skill 3.C] Stores second spins for the player and computer so that sums are available to compare (i.e., does not overwrite previously generated values)

Response earns the point if it…

· doesn't call the `spin` method correctly.

Response does not earn the point if it..

· does not create new `int` variables for the second computer and player spins. (Note that general penalty x is not incurred in this case.)

+1 [Skill 3.C] Compares correct sums in the case of a tie based on the first pair of spins

Response earns the point if it…

uses appropriate relational operators and arithmetic expression to compare the sums even if the `if` statement is incorrect.

+1 [Skill 3.C] Reports a player win, computer win, or tie, as appropriate, after the second pair of spins
+1 [Skill 3.A] All messages are in the specified format.

General Penalties:

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

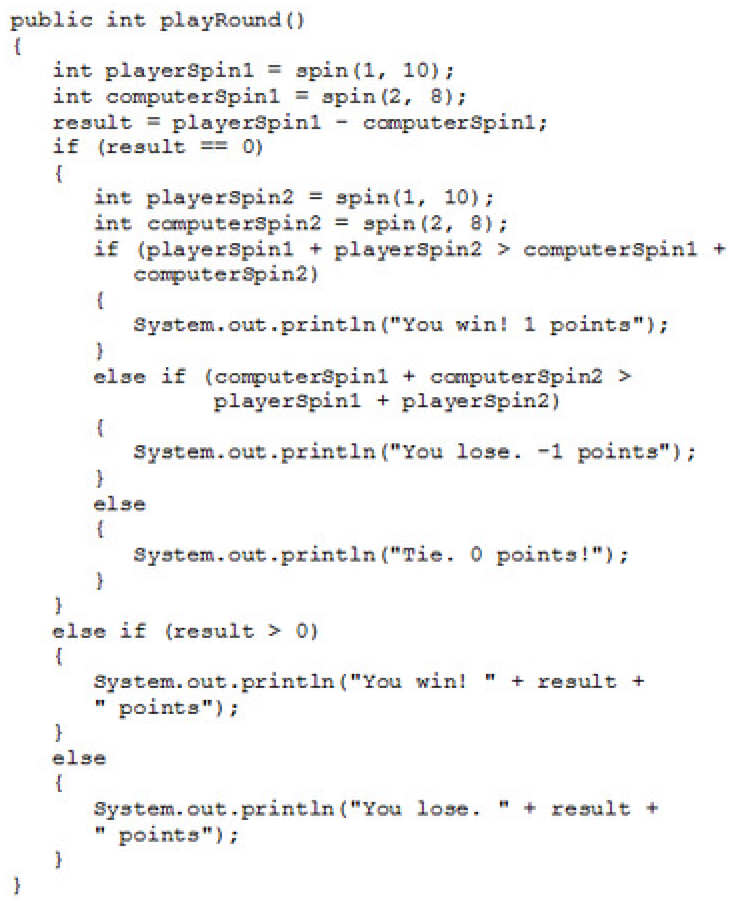
-1 (x) Local variables used but none declared

-1 (y) Destruction of persistent data (e.g., changing value referenced by parameter)

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

Canonical Solution:

3_F1



```
public int playRound()
{
   int playerSpin1 = spin(1, 10);
   int computerSpin1 = spin(2, 8);
   result = playerSpin1 - computerSpin1;
   if (result == 0)
   {
      int playerSpin2 = spin(1, 10);
      int computerSpin2 = spin(2, 8);
      if (playerSpin1 + playerSpin2 > computerSpin1 +
         computerSpin2)
      {
         System.out.println("You win! 1 points");
      }
      else if (computerSpin1 + computerSpin2 >
               playerSpin1 + playerSpin2)
      {
         System.out.println("You lose. -1 points");
      }
      else
      {
         System.out.println("Tie. 0 points!");
      }
   }
   else if (result > 0)
   {
      System.out.println("You win!  " + result +
      " points");
   }
   else
   {
      System.out.println("You lose.  " + result +
      " points");
   }
}
```

computerSpin1 + Line 11: computerSpin2) Line 12: { Line 13: System.out.println("You win! 1 points"); Line 14: } Line 15: else if (computerSpin1 + computerSpin2 > Line 16: playerSpin1 + playerSpin2) Line 17: { Line 18: System.out.println("You lose. -1 points"); Line 19: } Line 20: else Line 21: { Line 22: System.out.println("Tie. 0 points!"); Line 23: } Line 24: } Line 25: else if (result > 0) Line 26: { Line 27: System.out.println("You win! " + result + Line 28: " points"); Line 29: } Line 30: else Line 31: { Line 32: System.out.println("You lose. " + result + Line 33: " points"); Line 34: } Line 35: }">

0	1	2	3	4	5	6	[✓ 7]

Total number of points earned (minus penalties) is equal to 7.

- ☐ **+1** Calls the `spin` method appropriately to generate first spins for the player and the computer **(Points earned)**
- ☐ **+1** Checks for a tie on the first pair of spins **(Points earned)**
- ☐ **+1** Reports a player win or a computer win, if appropriate, based on only the first pair of spins **(Points earned)**
- ☐ **+1** Stores second spins for the player and computer so that sums are available to compare (i.e., does not overwrite previously generated values) **(Points earned)**
- ☐ **+1** Compares correct sums in the case of a tie based on the first pair of spins **(Points earned)**
- ☐ **+1** Reports a player win, computer win, or tie, as appropriate, after the second pair of spins **(Points**

3_F1

earned)

☐ **+1** All messages are in the specified format. **(Points earned)**

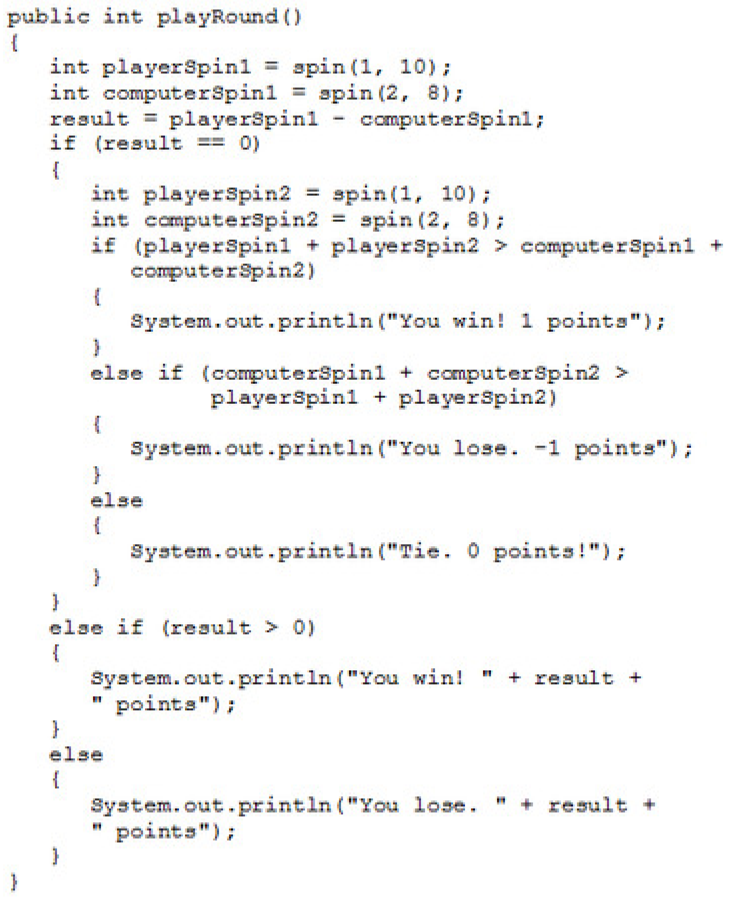
☐ **-1** (w) Extraneous code that causes side-effect (e.g., printing to output, incorrect precondition check) **(General Penalties)**

☐ **-1** (x) Local variables used but none declared **(General Penalties)**

☐ **-1** (y) Destruction of persistent data (e.g., changing value referenced by parameter) **(General Penalties)**

☐ **-1** (z) Void method or constructor that returns a value **(General Penalties)**

Canonical Solution:



```
public int playRound()
{
   int playerSpin1 = spin(1, 10);
   int computerSpin1 = spin(2, 8);
   result = playerSpin1 - computerSpin1;
   if (result == 0)
   {
      int playerSpin2 = spin(1, 10);
      int computerSpin2 = spin(2, 8);
      if (playerSpin1 + playerSpin2 > computerSpin1 +
         computerSpin2)
      {
         System.out.println("You win! 1 points");
      }
      else if (computerSpin1 + computerSpin2 >
               playerSpin1 + playerSpin2)
      {
         System.out.println("You lose. -1 points");
      }
      else
      {
         System.out.println("Tie. 0 points!");
      }
   }
   else if (result > 0)
   {
      System.out.println("You win! " + result +
      " points");
   }
   else
   {
      System.out.println("You lose. " + result +
      " points");
   }
}
```

computerSpin1 + Line 11: computerSpin2) Line 12: { Line 13: System.out.println("You win! 1 points"); Line 14: } Line 15: else if (computerSpin1 + computerSpin2 > Line 16: playerSpin1 + playerSpin2) Line 17: { Line 18: System.out.println("You lose. -1 points"); Line 19: } Line 20: else Line 21: { Line 22: System.out.println("Tie. 0 points!"); Line 23: } Line 24: } Line 25: else if (result > 0) Line 26: { Line 27: System.out.println("You win! " + result + Line 28: " points"); Line 29: } Line 30: else Line 31: { Line 32: System.out.println("You lose. " + result + Line 33: " points"); Line 34: } Line 35: }">