CS2163 Java Homework 5 requirement

Homework 5 needs to be finished on the Greenfoot platform, and it does not involve Eclipse.

**Follow the steps below to finish homework 5:**

Based on the existing Greenfoot scenario WBC-4, you need to make the following changes:

1. in class Bloodstream, add the definition of four private instance variables, and their names are: numberOfBacteriaTouched, numberOfBacteriaMissed, numberOfVirusTouched , numberOfVirusMissed, respectively. Provide a public getter and setter for each variable.
2. here are the rules to increase these four instance variables:
   * when the white blood cell touches a bacteria, increase variable numberOfBacteriaTouched by one
   * when a bacteria reaches the left edge of the screen without touching the white blood cell, increase variable numberOfBacteriaMissed by one
   * when the white blood cell touches a virus, increase variable numberOfVirusTouched by one
   * when a virus reaches the left edge of the screen without touching the white blood cell, increase variable numberOfVirusMissed by one

**How to increase any of the above variables by one**?

**Hint**: first, you can use the public getter to obtain the current value of the variable, and then increase this value by one, and then use the public setter to set this increased value back to the variable. And that is why we provide each variable with a public setter and getter in step 1 above. In order to call the public getter and setter, you need to first obtain an instance (handler) of the class Bloodstream.

1. show the name and the values of the above four variables at the lower left corner of the screen using four lines of text. The sequence of the text follows the sequence shown in the requirement above.
2. change the way how score is calculated, and here is how to calculate the score under 4 conditions:
   * when the white blood cell touches one bacteria, increase the score by 1.
   * when a bacteria reaches the left edge of the screen without touching the white blood cell, decrease the score by 1.
   * when a white blood cell touches a virus, decrease the score by 2
   * when a virus reaches the left edge of the screen without touching the white blood cell, increase the score by 2

You should still invoke the addScore method to change score, and the actual parameter plugged in for the addScore method must follow the rules defined above on how to calculate the score. For example, you can plug in 1, or -1, or -2, or 2 as the actual parameters for the four conditions defined above, respectively. You cannot modify the signature of addScore method, which means that the addScore method can only take one formal parameter.

The places that you need to invoke the addScore method are:

* + in class WhiteCell, when the white blood cell touches a bacteria
  + in class Bacteria, when a bacteria reaches the left edge of the screen without touching the white blood cell
  + in class WhiteCell, when the white blood cell touches a virus
  + in class Virus, when a virus reaches the left edge of the screen without touching the white blood cell

1. in class Bloodstream, inside the implementation of the addScore method, the game stops when score is less than zero, and the code for this requirement is already in scenario WBC-4, thus no need for any code change. When the timer times out, you also need to show the final score, and the code for this requirement is already in scenario WBC-4, thus no need for any code change.
2. **Hint for the homework**: you need to read the source code for WBC-4 carefully, and understand how the score is calculated and displayed, before you start working on the new score system.
3. Your solution for step 3 must be efficient. When your program starts, you should display the value of the four counters, and they should all be zero. After that, you should update the display of the four counters **only when** there is a change in any of the counters. If you placed your code of showing the four counters in the act() method of a class, then that is not an efficient solution, because it will re-display the value of the four counters even though there is no change in any counter value.
4. Zip the entire scenario folder “WBC-4”, and rename the zip file as “***JohnDoeHw5.zip***”, where JohnDoe needs to be replaced by your first and last name
5. Submit the zip file ***JohnDoeHw5.zip*** to Moodle “homework 5 drop box”.

**After finishing this homework, how to verify the correctness of your submitted zip files:**

1. Download the zip files you have uploaded to Moodle homework drop box.
2. Unzip the zip file to a different local folder in your computer, other than the original local folder where the zip files are generated.
3. Run the Greenfoot project from the unzip folder, and make sure it compiles and runs correctly.
4. If your submitted zip file in the Moodle drop box
   1. **cannot be downloaded,** or
   2. **cannot be unzipped,** or
   3. **cannot compile,** or
   4. **cannot run,**
   5. then you need to figure out the reason and fix the error, and then submit the corrected zip file to the Moodle drop box. Then start this verification process again until you can download, unzip, compile and run successfully. To upload a corrected zip file to the Moodle drop box, you need to delete the previous submitted zip file from the Moodle drop box first.

In the first page of file “chap1-schedule.docx”, you can find the instructions on how to zip and unzip files.

**Grading components:**

* From step 1 ~ 4 in page 1, each step has 5 points, thus resulting in a total of 20 points.

**For any submitted zip file that still has syntax error and it cannot compile or run in Greenfoot, it will receive ZERO point**. No re-submission is allowed after the homework due day.

Please click the Moodle homework drop box to see the due day of this homework.

When coding in Eclipse or Greenfoot, please read document “RulesForIndentAndAlignCode.docx” in Moodle folder “chap 1”, and follow all the rules in code alignment and indentation.