Agile Developer Quiz

Thank you for your interest in this challenge. We strive to hire the best of the best. This quiz is your opportunity to put your best foot forward. We think this might take about an hour of your time, but don't be afraid to put more work into it. We want to see what you are capable of. The type of person we're looking for will find this an enjoyable challenge!

Instructions

Please answer each question according to the directory structure provided with this document.

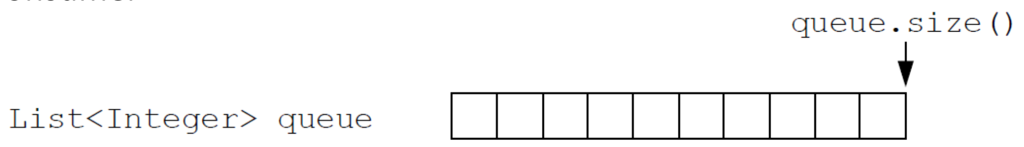
You can provide your answers in the Q#-answer.txt files in the respective sub folders per question, or add your own files if you think you need them.

Question 3 is a full programming exercise whose outcome should be working code that can be executed. Here is where you can show your best technical skills. We want to see how you would structure a project and a problem solution, given the space to do so. You can import external libraries to help with the implementation, if you so desire. The only rule is that they have to be publicly available under a free license (such as the LGPL, or a BSD style license). You may provide an IDE project (Eclipse, IntelliJ etc.) or, if preferred, a maven project to build the source code. Again, you may add additional information via a Q3-answer.txt file.

Question 1:

A producer thread periodically produces elements and puts them in a queue, and a consumer thread takes the elements from the queue and does some processing with them:

Consumer



Consumer

```
while(true)
{
        ___
        while(___)
        {
           ___
        }
        var el = queue.remove(0);
        ___
        processElement(el);
}
```

Producer

```
timer_tick()
{
         ___
         queue.Add(Math.random());
         ___
         ___
}
```

Question Outcomes

Please write the missing instructions for the producer and consumer threads into the blank spaces provided.

Question 2:

Given the following code:

```
public class Solution {
    public int sum(int[] input) {
        // write your solution here
        recursiveSum(...) { ... }
    }
}
```

Question Outcomes

- Please complete the recursiveSum() method. Use a recursive implementation.
- What can go wrong with this recursive function if the number of input numbers becomes really big (you can assume that there is enough RAM to fit all the numbers into memory).

Question 3

Please write a Java console application with the following behavior:

1. When the user enters the name of a shape followed by the corresponding number of numeric parameters, define that shape and keep it in memory. The numbers may be of type float. Examples:

   ```
   circle 1.7 -5.05 6.9
   square 3.55 4.1 2.77
   rectangle 3.5 2.0 5.6 7.2
   ```

- For the circle, the numbers are the x and y coordinates of the center followed by the radius.
- For the square they are x and y coordinates of one corner followed by the side length.
- For the rectangle they are the x and y coordinates of one corner followed by the two sides.

  In addition, every time such a line is entered, the application should give it a unique identifier and print it out in a standardized form. For example:

  ```
  => shape 1: circle with centre at (1.7, -5.05) radius 6.9
  ```

2. Provide a `contains` command. When the user enters a pair of numbers, the application should print out all the shapes that include that point in the (x, y) space, i.e. it should print out shape X if the given point is inside X.
3. It should accept the commands "help" for printing usage instructions and "exit" for terminating the execution
4. Please provide unit test cases that cover at least the implementation of the `contains` command.
5. *Bonus if you've got time*: Implement a donut shape as well. Hint: A point is inside a donut, if the point is inside of the outer circle but not inside the inner circle.

Remember that this is where you can show your best technical skills. Think about things like avoiding duplication of code, maintainability and conciseness and testing your code. Make this a shining example of your engineering skills.