Ruby - Feature #19905

Introduce `Queue#peek`

09/28/2023 11:10 PM - hi@joaofernandes.me (Joao Fernandes)

Status:	Open	
Priority:	Normal	
Assignee:		
Target version:		

Description

This ticket proposes the introduction of the Queue#peek method, similar to what we can find in other object oriented languages such as Java and C#. This method is similar to Queue#pop, but does not change the data, nor does it require a lock.

```
q = Queue.new([1,2,3])
=> #<Thread::Queue:0x00000001065d7148>
q.peek
=> 1
q.peek
=> 1
```

I have felt the need of this for debugging, but I think that it can also be of practical use for presentation. I believe that the only drawback could be that newcomers could misuse it in multi-threaded work without taking into account that this method is not thread safe.

I also volunteer myself to implement this method.

History

#1 - 09/29/2023 12:43 AM - ko1 (Koichi Sasada)

Could you describe more about use cases?

#2 - 09/29/2023 11:16 AM - hi@joaofernandes.me (Joao Fernandes)

ko1 (Koichi Sasada) wrote in #note-1:

> Could you describe more about use cases?

My main use case for this feature is observability and debugging. For example, I am experimenting with a car pooling simulation where multiple queues of passengers exist. I want to track the length of each queue, and how long the first passenger of each queue has been waiting. Thus, I would like to look at the first element of each queue without removing it, so that the workers can pick it up.

I was also considering using this method to decide which queue to serve first. While I would still need to be careful with concurrency issues for this feature, #peek would be extremely helpful.

Does this clarify potential use cases? Does it seem like a potential good addition to the language?

#3 - 10/06/2023 05:28 PM - shan (Shannon Skipper)

I'm just curious if it might be better for #peek to lock to avoid showing something that has already popped or other parallel access issues? Is there significant advantage having it not lock?

#4 - 10/06/2023 10:20 PM - Eregon (Benoit Daloze)

peek is inherently racy in that the value observed might be already popped by the time the value is returned.
If it's for observability and debugging I think to_a or each makes more sense, it's for sure useful to see other values than just the first one.

#5 - 02/13/2024 02:59 PM - sidonath (Damir Zekic)

I would like to provide another use case for Queue#peek. I'm using Queue in a project that's running commands prone to failures. I have a queue of commands, a worker and a monitor threads. The worker pops the commands, runs them in sequence and the monitor thread reports on the activity and restarts the worker if it observes a failure. Having ability to "peek" into the command list would allow informative output about what is the next command that will be executed. Obviously, having #to_a or #each would be as helpful here. The queue is not long for me to worry about looking up all elements versus the first one.

If there's a race condition and the proposed #peek doesn't lock, then the display of the "next command" might be briefly inaccurate at times, but that's not something I would consider an issue in this project. If it were locking, then I would need to use the retrieved instance outside of the locked block. This would beat the purpose of the lock, but that would be the conscious choice of me as the author of the code. I guess whether #peek locks or not would not be important in my situation.

#6 - 02/13/2024 05:22 PM - byroot (Jean Boussier)

Seems like all the use cases so far would be as well if not better handled by to_a. Which I admit I hoped existed a few times when debugging code that uses Queue. Should also be relatively straightforward to implement.

#7 - 02/13/2024 07:05 PM - ko1 (Koichi Sasada)

I feel it is natural that to_a returns [] immediately if there is no enqueued item.
(#peek is not clear on it)