# Final project for the course

Aknur Shakhidani
Sima Sbouh

## Used libraries

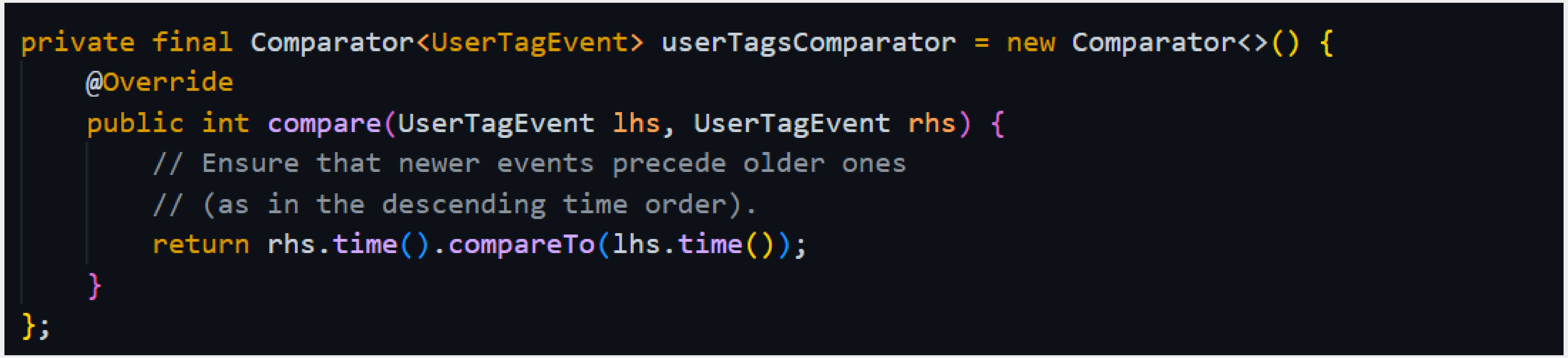
| |
|---|
| import java.time.Instant |
| import java.util.ArrayList |
| import java.util.Collections |
| import java.util.Comparator |
| import java.util.concurrent.ConcurrentHashMap |
| import java.util.concurrent.ConcurrentLinkedQueue |

## Handiling with user actions

```
private final ConcurrentHashMap<String, ConcurrentLinkedQueue<UserTagEvent>> viewEventsStore = new ConcurrentHashMap<>();
private final ConcurrentHashMap<String, ConcurrentLinkedQueue<UserTagEvent>> buyEventsStore = new ConcurrentHashMap<>();
```

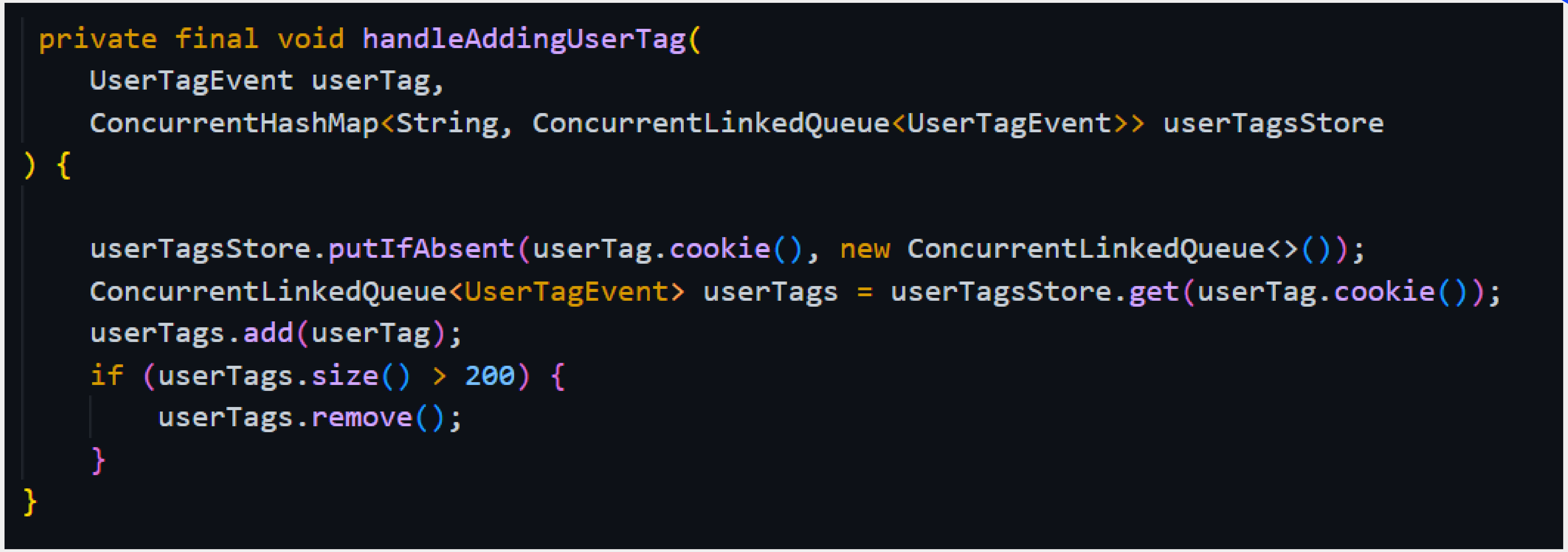
- 2 stores: one for each type of user tag events (VIEW/BUY)
- Both stores is indexed by a cookie (in ConcurrentHashMap) respectively
- ConcurrentLinkedQueue is used to store these events in the order of their arrival on the server

## Sorting user tag events in descending time order



```java
private final Comparator<UserTagEvent> userTagsComparator = new Comparator<>() {
    @Override
    public int compare(UserTagEvent lhs, UserTagEvent rhs) {
        // Ensure that newer events precede older ones
        // (as in the descending time order).
        return rhs.time().compareTo(lhs.time());
    }
};
```

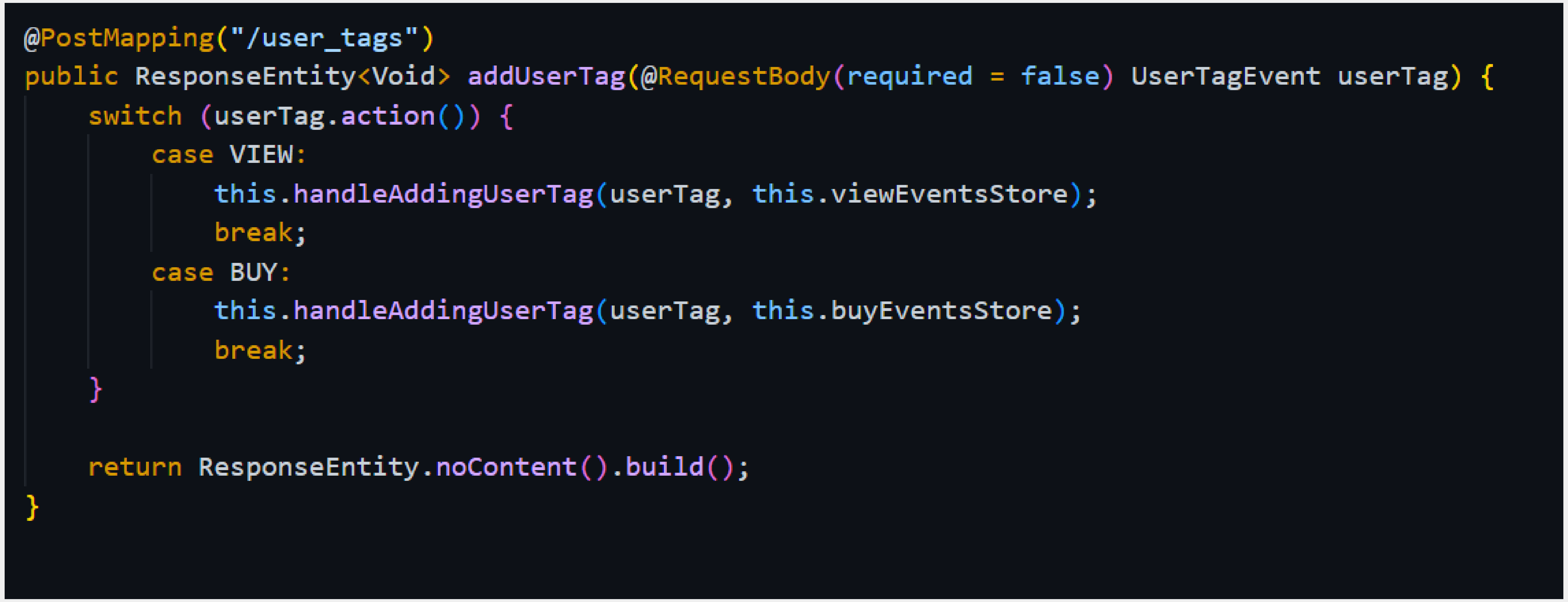
## Add a user tag event to the structure and remove the outdated ones



```java
private final void handleAddingUserTag(
    UserTagEvent userTag,
    ConcurrentHashMap<String, ConcurrentLinkedQueue<UserTagEvent>> userTagsStore
) {

    userTagsStore.putIfAbsent(userTag.cookie(), new ConcurrentLinkedQueue<>());
    ConcurrentLinkedQueue<UserTagEvent> userTags = userTagsStore.get(userTag.cookie());
    userTags.add(userTag);
    if (userTags.size() > 200) {
        userTags.remove();
    }
}
```

- @param userTag  - the user tag to add
- @param userTagsStore  - the structure where to add the user tag to
- The condition removes the events which came earlier than the 200 most recent ones (as described on the project requirements)

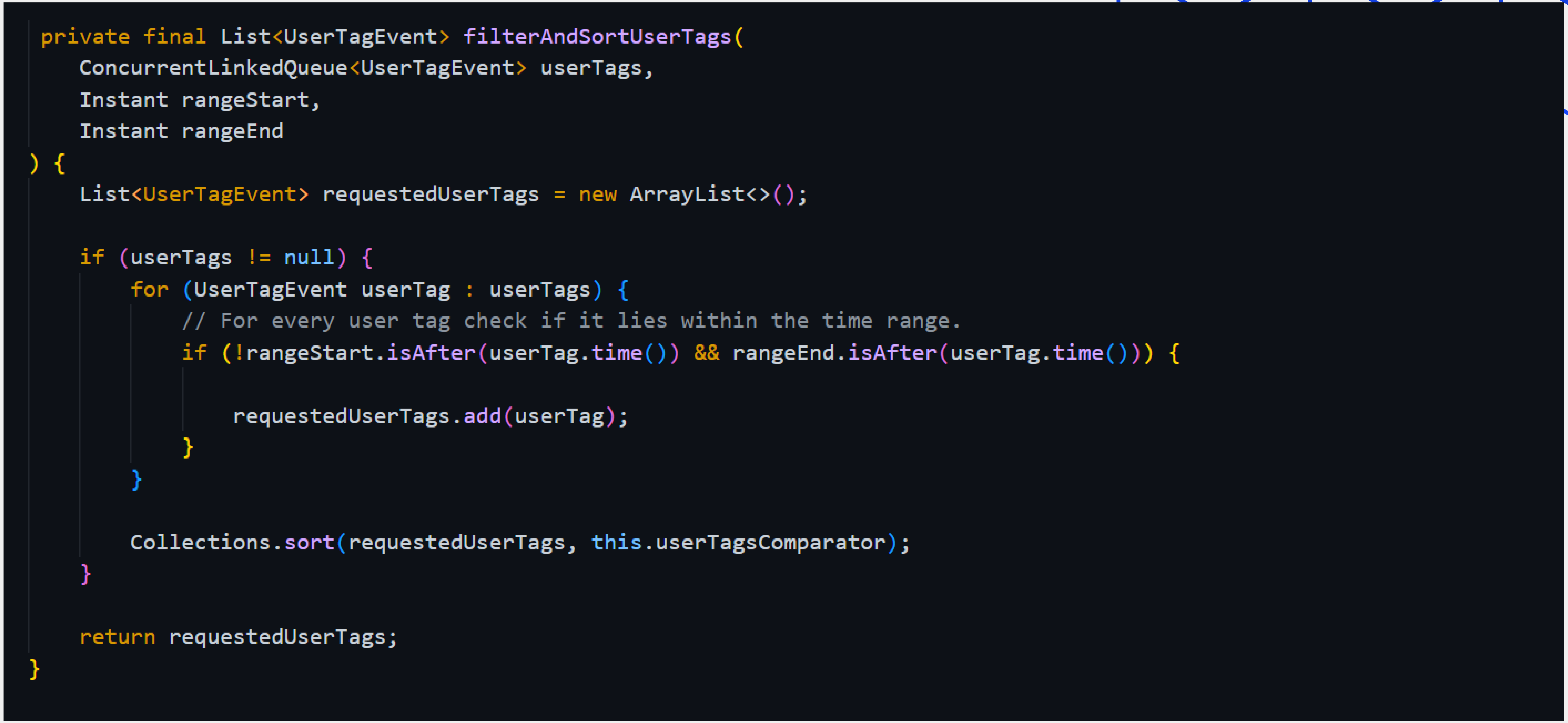
## Add the user tag event to the correct events store depending on its type



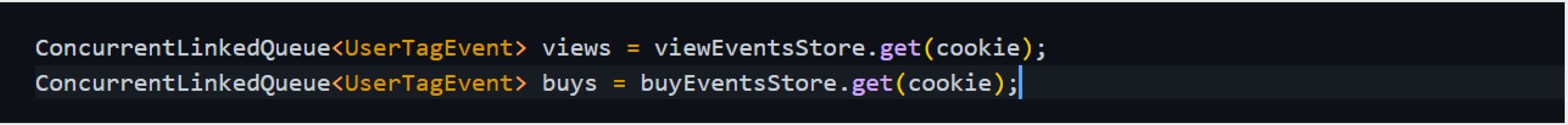
```java
@PostMapping("/user_tags")
public ResponseEntity<Void> addUserTag(@RequestBody(required = false) UserTagEvent userTag) {
    switch (userTag.action()) {
        case VIEW:
            this.handleAddingUserTag(userTag, this.viewEventsStore);
            break;
        case BUY:
            this.handleAddingUserTag(userTag, this.buyEventsStore);
            break;
    }

    return ResponseEntity.noContent().build();
}
```

## Filter the user tags according to the time range and sort them in descending time order

- @param rangeStart - the start of the time range
- @param rangeEnd - the end of the time range


```java
private final List<UserTagEvent> filterAndSortUserTags(
    ConcurrentLinkedQueue<UserTagEvent> userTags,
    Instant rangeStart,
    Instant rangeEnd
) {
    List<UserTagEvent> requestedUserTags = new ArrayList<>();

    if (userTags != null) {
        for (UserTagEvent userTag : userTags) {
            // For every user tag check if it lies within the time range.
            if (!rangeStart.isAfter(userTag.time()) && rangeEnd.isAfter(userTag.time())) {

                requestedUserTags.add(userTag);
            }
        }

        Collections.sort(requestedUserTags, this.userTagsComparator);
    }

    return requestedUserTags;
}
```

## Get user tag events of each type for the provided cookie

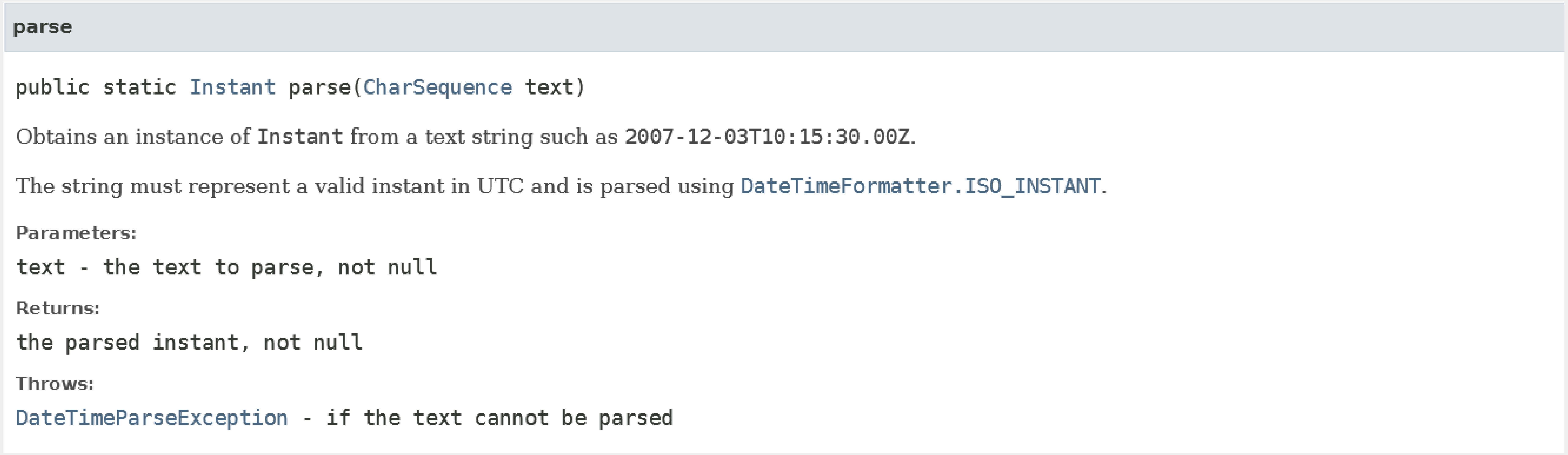


```java
ConcurrentLinkedQueue<UserTagEvent> views = viewEventsStore.get(cookie);
ConcurrentLinkedQueue<UserTagEvent> buys = buyEventsStore.get(cookie);
```

## Handling with time range

```java
String[] timeRange = timeRangeStr.split("_");

Instant rangeStart = Instant.parse(timeRange[0] + "Z");
Instant rangeEnd = Instant.parse(timeRange[1] + "Z");
```

- Split the time range to the start date and the end date
- Parse the time range dates with the ISO_INSTANT formatter. In order to do that, append "Z" to the end of the date string to convert it to the UTC format.



**parse**

```
public static Instant parse(CharSequence text)
```

Obtains an instance of `Instant` from a text string such as `2007-12-03T10:15:30.00Z`.

The string must represent a valid instant in UTC and is parsed using `DateTimeFormatter.ISO_INSTANT`.

**Parameters:**

`text - the text to parse, not null`

**Returns:**

`the parsed instant, not null`

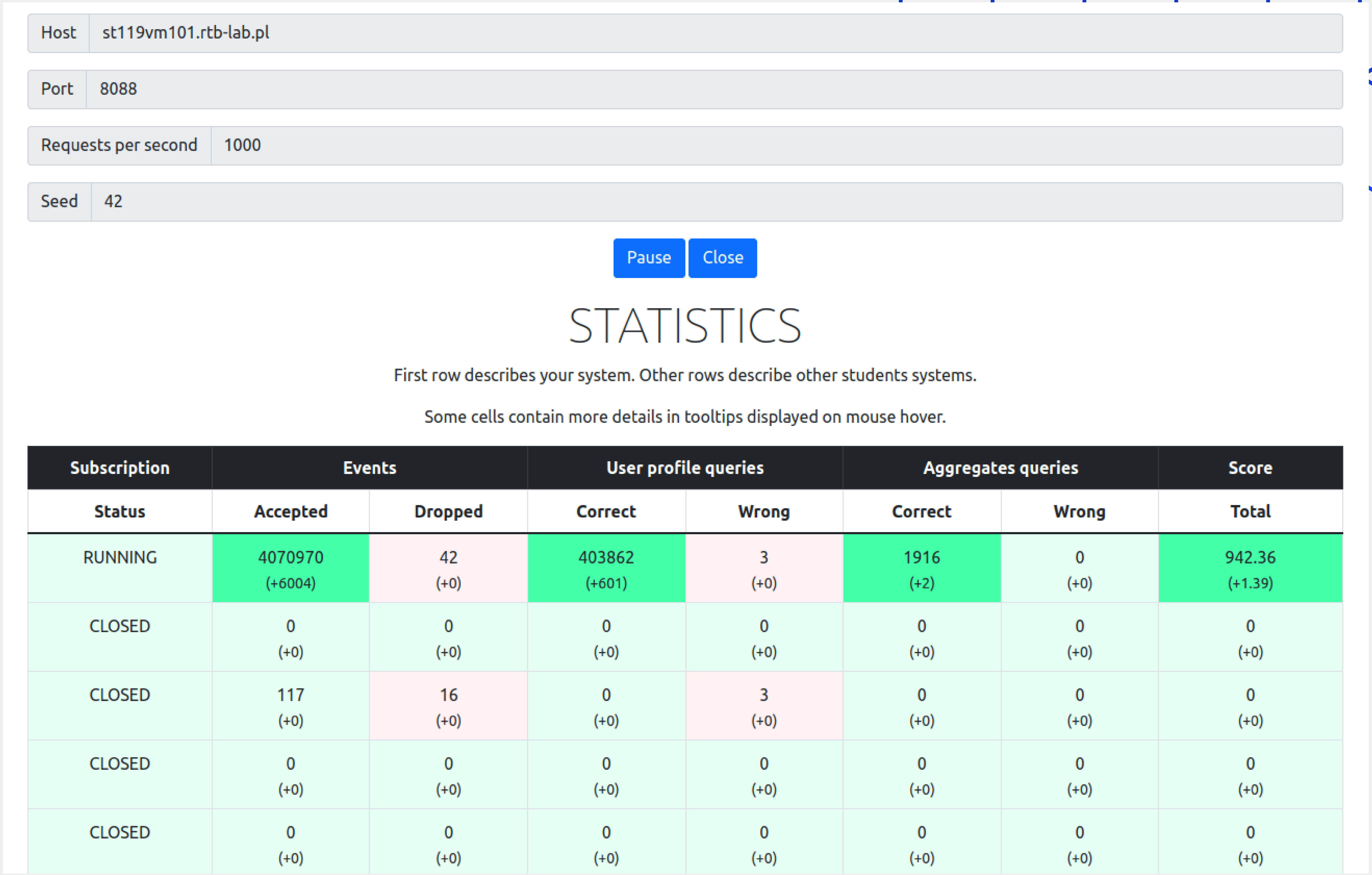
**Throws:**

`DateTimeParseException - if the text cannot be parsed`

## Get requested view and buy events

```java
List<UserTagEvent> requestedViews = this.filterAndSortUserTags(views, rangeStart, rangeEnd);
List<UserTagEvent> requestedBuys = this.filterAndSortUserTags(buys, rangeStart, rangeEnd);

UserProfileResult result = new UserProfileResult(cookie, requestedViews, requestedBuys);
```

```java
if (expectedResult != null && !expectedResult.equals(result)) {
    EchoClient.log.info("Expected: {}\nReceived: {}", expectedResult, result);
}

return ResponseEntity.ok(result);
```



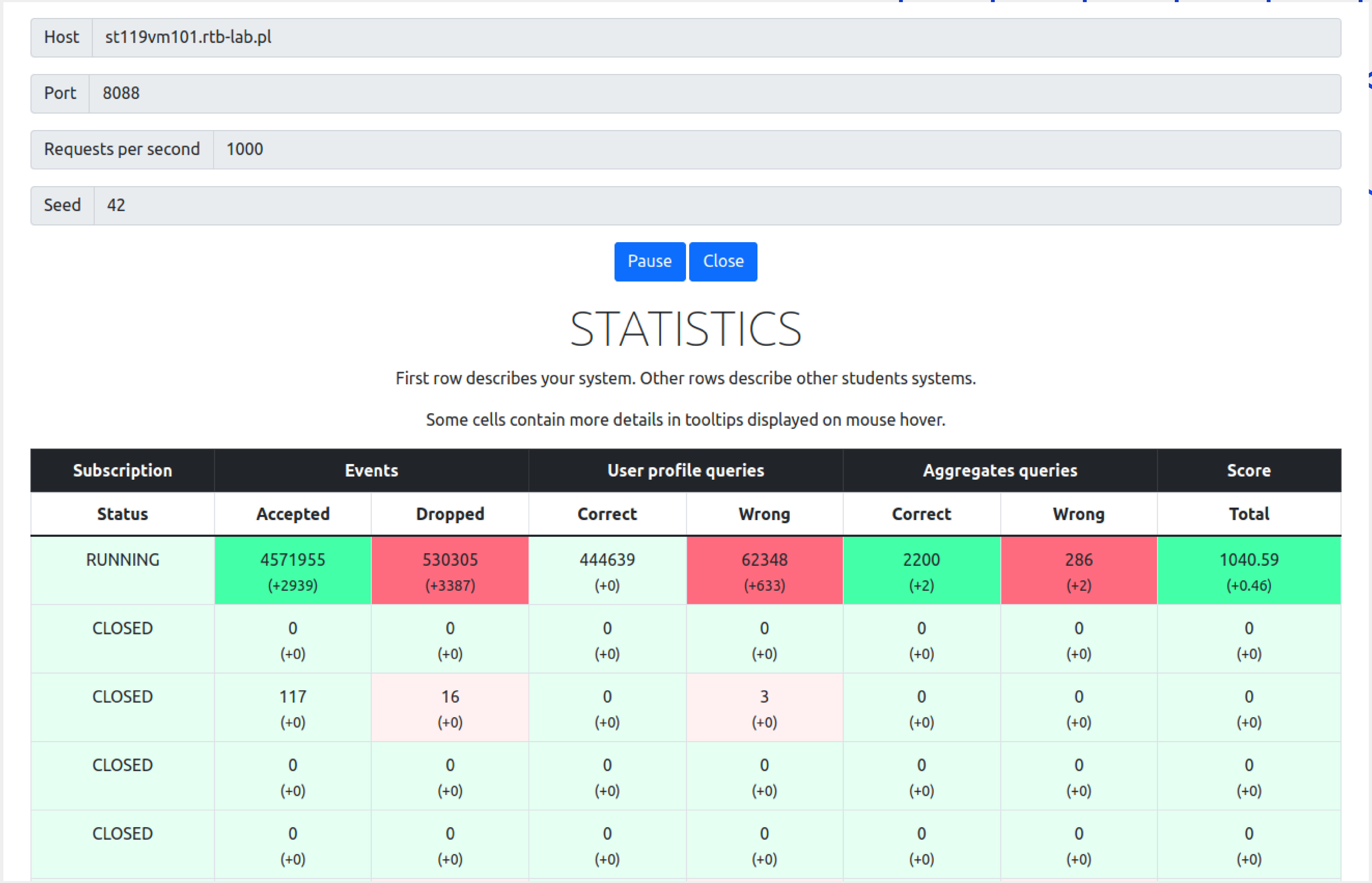
| | |
|---|---|
| Host | st119vm101.rtb-lab.pl |
| Port | 8088 |
| Requests per second | 1000 |
| Seed | 42 |

# STATISTICS

First row describes your system. Other rows describe other students systems.

Some cells contain more details in tooltips displayed on mouse hover.

| Subscription | Events | | User profile queries | | Aggregates queries | | Score |
|---|---|---|---|---|---|---|---|
| **Status** | **Accepted** | **Dropped** | **Correct** | **Wrong** | **Correct** | **Wrong** | **Total** |
| RUNNING | 4070970 (+6004) | 42 (+0) | 403862 (+601) | 3 (+0) | 1916 (+2) | 0 (+0) | 942.36 (+1.39) |
| CLOSED | 0 (+0) | 0 (+0) | 0 (+0) | 0 (+0) | 0 (+0) | 0 (+0) | 0 (+0) |
| CLOSED | 117 (+0) | 16 (+0) | 0 (+0) | 3 (+0) | 0 (+0) | 0 (+0) | 0 (+0) |
| CLOSED | 0 (+0) | 0 (+0) | 0 (+0) | 0 (+0) | 0 (+0) | 0 (+0) | 0 (+0) |
| CLOSED | 0 (+0) | 0 (+0) | 0 (+0) | 0 (+0) | 0 (+0) | 0 (+0) | 0 (+0) |



| Host | st119vm101.rtb-lab.pl |
|---|---|
| Port | 8088 |
| Requests per second | 1000 |
| Seed | 42 |

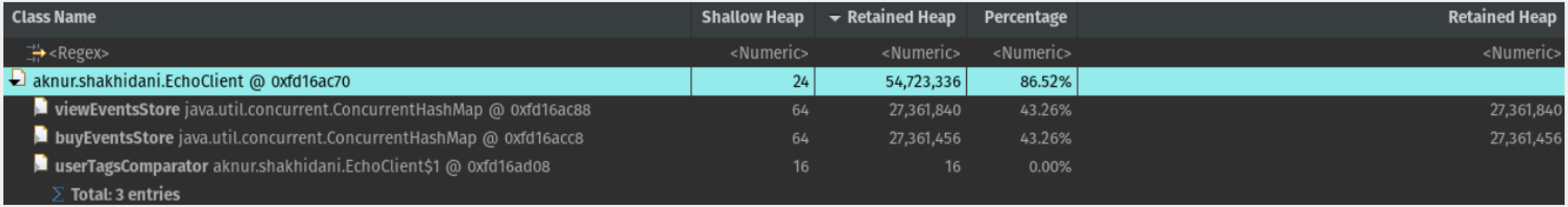
# STATISTICS

First row describes your system. Other rows describe other students systems.

Some cells contain more details in tooltips displayed on mouse hover.

| Subscription | Events | | User profile queries | | Aggregates queries | | Score |
|---|---|---|---|---|---|---|---|
| **Status** | **Accepted** | **Dropped** | **Correct** | **Wrong** | **Correct** | **Wrong** | **Total** |
| RUNNING | 4571955 (+2939) | 530305 (+3387) | 444639 (+0) | 62348 (+633) | 2200 (+2) | 286 (+2) | 1040.59 (+0.46) |
| CLOSED | 0 (+0) | 0 (+0) | 0 (+0) | 0 (+0) | 0 (+0) | 0 (+0) | 0 (+0) |
| CLOSED | 117 (+0) | 16 (+0) | 0 (+0) | 3 (+0) | 0 (+0) | 0 (+0) | 0 (+0) |
| CLOSED | 0 (+0) | 0 (+0) | 0 (+0) | 0 (+0) | 0 (+0) | 0 (+0) | 0 (+0) |
| CLOSED | 0 (+0) | 0 (+0) | 0 (+0) | 0 (+0) | 0 (+0) | 0 (+0) | 0 (+0) |

## Memory Analysis

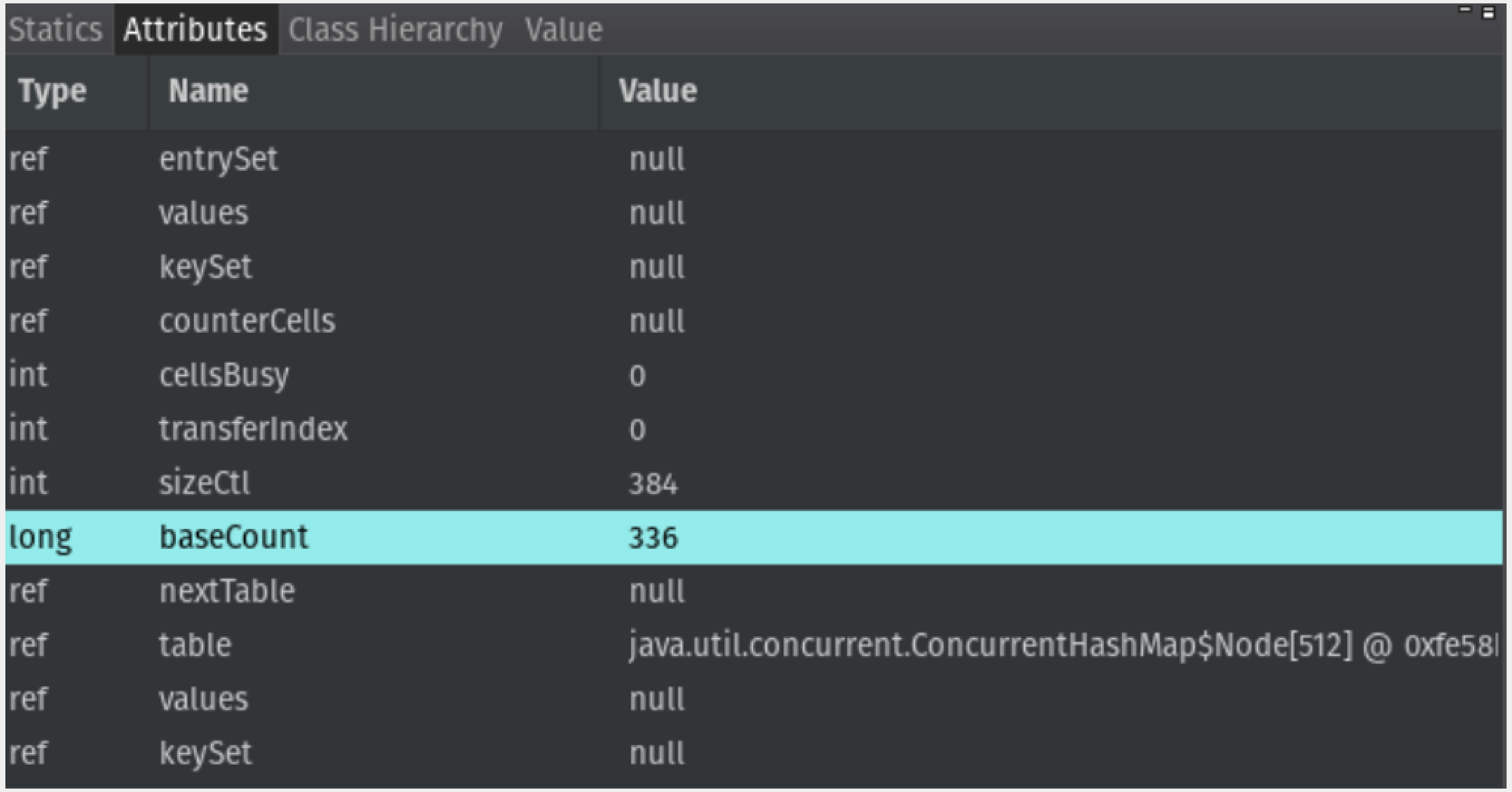


| Class Name | Shallow Heap | Retained Heap | Percentage | Retained Heap |
|---|---|---|---|---|
| <Regex> | <Numeric> | <Numeric> | <Numeric> | <Numeric> |
| aknur.shakhidani.EchoClient @ 0xfd16ac70 | 24 | 54,723,336 | 86.52% | |
| **viewEventsStore** java.util.concurrent.ConcurrentHashMap @ 0xfd16ac88 | 64 | 27,361,840 | 43.26% | 27,361,840 |
| **buyEventsStore** java.util.concurrent.ConcurrentHashMap @ 0xfd16acc8 | 64 | 27,361,456 | 43.26% | 27,361,456 |
| **userTagsComparator** aknur.shakhidani.EchoClient$1 @ 0xfd16ad08 | 16 | 16 | 0.00% | |
| Σ **Total: 3 entries** | | | | |

- Heap size limit was set to only 64 MB

## Memory Analysis

- We can focus only on our two structures (viewEventsStore and buyEventsStore) because they are responsible for almost all memory usage.



Statics | Attributes | Class Hierarchy | Value

| Type | Name | Value |
|---|---|---|
| ref | entrySet | null |
| ref | values | null |
| ref | keySet | null |
| ref | counterCells | null |
| int | cellsBusy | 0 |
| int | transferIndex | 0 |
| int | sizeCtl | 384 |
| long | baseCount | 336 |
| ref | nextTable | null |
| ref | table | java.util.concurrent.ConcurrentHashMap$Node[512] @ 0xfe58 |
| ref | values | null |
| ref | keySet | null |

- 80 KB per cookie for one action (in one structure / for 200 user tag events of one type);
- 160 KB per cookie overall;
- 13000 cookies is expected to take 2 GB;
- java.lang.OutOfMemoryError exceptions

## Thank you for your attention!