Token Bucket

The TokenBucket algorithm is a rate-limiting algorithm that simulates a fixed-capacity bucket that is continuously refilled with tokens at a specified rate. The main idea behind this algorithm is to allow an operation (such as a request or an API call) to be executed only if there are enough tokens available in the bucket.

You have been given a `TokenBucket` class. The code has issues that cause the tokens to be refilled at a slower rate than intended. Your task is to identify and fix the bug(s) in the `TokenBucket` class.

Here is the given `TokenBucket` class:

```java
import java.util.concurrent.TimeUnit;

public class TokenBucket {
    private final int capacity;
    private final int refillRate;
    private int tokens;
    private long timestamp;

    public TokenBucket(int bucketSize, int refillRate) {
        this.capacity = bucketSize;
        this.refillRate = refillRate;
        this.tokens = bucketSize;
        this.timestamp = System.currentTimeMillis();
    }

    public boolean consume() {
        refresh();
        if (tokens > 0) {
            tokens -= 1;
            return true;
        }
        return false;
    }

    private void refresh() {
        long secondsPassed = (System.currentTimeMillis() - timestamp) / 1000;
        timestamp = System.currentTimeMillis();
        tokens = Math.min(
            capacity, tokens + (int) (secondsPassed * refillRate)
        );
    }
}
```

Click here to expand...

Bug 1: The use of integer division when calculating the `secondsPassed`.

Fix:

```java
long millisPassed = System.currentTimeMillis() - timestamp;
double secondsPassed = millisPassed / 1000.0;
```

Bug 2: The update of the `timestamp` in the `refresh()` method.

To fix the bug, we can make `tokens` a double instead of an integer.

If for some reason `timestamp` must be an integer, then we can change the timestamp update to consider only the elapsed time used for refilling tokens.

```java
long millisPassed = System.currentTimeMillis() - timestamp;
double secondsPassed = millisPassed / 1000.0;
int tokensToRefill = (int) (secondsPassed * refillRate);
tokens = Math.min(capacity, tokens + tokensToRefill);

// Update the timestamp considering only the elapsed time used for refilling tokens
long millisUsedForRefill = (long) (tokensToRefill / (double) refillRate * 1000);
timestamp += millisUsedForRefill;
```