Master reactivity in Java 9 by implementing a real-time analytics platform, using PublisherSubscriber pattern and Flow API to process streaming data efficiently.

Scenario:

Imagine you are developing a real-time analytics platform that monitors social media feeds

for trending hashtags. The platform needs to react promptly to incoming data streams and identify

the top three trending hashtags at any given moment. To achieve this, you decide to implement

reactivity using Java 9 features.

Problem Statement: Reactivity in Java 9

- Implement the TrendingHashtagsProcessor class to process incoming hashtags. Use the

onNext method to update the trending hashtags list.

- Implement the TrendingHashtagsPublisher class to publish hashtag updates. Ensure proper

subscription handling in the subscribe method.

- Simulate the arrival of hashtags in the Main class and complete the publisher to trigger

completion in the processor.

- Print the top three trending hashtags using the getTopTrendingHashtags method in the

TrendingHashtagsProcessor class.

Codings:

```
import java.util.concurrent.Flow.*;

// Implement the TrendingHashtagsProcessor class to process incoming hashtags

class TrendingHashtagsProcessor implements Subscriber<String> {

// Add necessary fields and methods

@Override

public void onSubscribe(Subscription subscription) {
```

```
// Implement subscription handling
}
@Override
public void onNext(String hashtag) {
// Implement logic to process incoming hashtags
}
@Override
public void onError(Throwable throwable) {
// Implement error handling
}
@Override
public void onComplete() {
// Implement completion handling
}
// Add any additional methods if required
}
// Implement the TrendingHashtagsPublisher class to publish hashtag updates
class TrendingHashtagsPublisher implements Publisher<String> {
// Add necessary fields and methods
@Override
public void subscribe(Subscriber<? super String> subscriber) {
// Implement subscription logic
}
// Add any additional methods if required
```

```
}

public class Main {

public static void main(String[] args) {

// Create an instance of TrendingHashtagsPublisher

TrendingHashtagsPublisher hashtagsPublisher = new TrendingHashtagsPublisher();

// Create an instance of TrendingHashtagsProcessor

TrendingHashtagsProcessor hashtagsProcessor = new TrendingHashtagsProcessor();

// Subscribe the processor to the publisher

// Implement the necessary steps for subscription

// Simulate the arrival of hashtags

hashtagsPublisher.publishHashtag("#Java");

hashtagsPublisher.publishHashtag("#ReactiveProgramming");

hashtagsPublisher.publishHashtag("#Java9");

hashtagsPublisher.publishHashtag("#Concurrency");

// Complete the publisher to trigger completion in the processor

hashtagsPublisher.complete();

// Print the top three trending hashtags

System.out.println("Top Trending Hashtags: " +
hashtagsProcessor.getTopTrendingHashtags());

}

}
```

Expected Output:

Top Trending Hashtags: [#Java, #ReactiveProgramming, #Java9]

Learning Outcomes:

- Develop expertise in handling asynchronous data streams through the implementation of

reactive components in Java 9.

• Gain hands-on experience in utilizing the Publisher-Subscriber pattern and Flow API for efficient communication and data processing.

• Enhance problem-solving skills by addressing real-world scenarios, fostering proficiency in creating responsive applications.

```
import java.util.concurrent.Flow.*;
import java.util.*;
import java.util.stream.Collectors;

// Implement the TrendingHashtagsProcessor class to process incoming hashtags
class TrendingHashtagsProcessor implements Subscriber<String> {
    private Subscription subscription;
    private List<String> hashtags = new ArrayList<>();

    @Override
    public void onSubscribe(Subscription subscription) {
        this.subscription = subscription;
        subscription.request(1);  // Request one item at a time
    }

    @Override
    public void onNext(String hashtag) {
        // Add the incoming hashtag to the list
        hashtags.add(hashtag);
        // Request the next item
        subscription.request(1);
    }

    @Override
    public void onError(Throwable throwable) {
        // Handle errors (if any)
        System.err.println("Error: " + throwable.getMessage());
    }

    @Override
    public void onComplete() {
        // Once complete, we can print the top trending hashtags
        System.out.println("Processing complete. Top trending hashtags are:");
    }

    // Method to get top trending hashtags (top 3 most recent hashtags)
    public List<String> getTopTrendingHashtags() {
        // Get the top 3 trending hashtags
        return hashtags.stream()
                .limit(3)
                .collect(Collectors.toList());
    }
```

```
}

// Implement the TrendingHashtagsPublisher class to publish hashtag updates
class TrendingHashtagsPublisher implements Publisher<String> {
    private List<Subscriber<? super String>> subscribers = new ArrayList<>();

    @Override
    public void subscribe(Subscriber<? super String> subscriber) {
        // Add the subscriber to the list of subscribers
        subscribers.add(subscriber);
        // Notify the subscriber that the subscription has been accepted
        subscriber.onSubscribe(new Subscription() {
            @Override
            public void request(long n) {
                // For this simulation, we request one hashtag at a time
            }

            @Override
            public void cancel() {
                // Handle cancellation if needed
            }
        });
    }

    public void publishHashtag(String hashtag) {
        // Notify all subscribers with the new hashtag
        for (Subscriber<? super String> subscriber : subscribers) {
            subscriber.onNext(hashtag);
        }
    }

    public void complete() {
        // Notify all subscribers that the publishing is complete
        for (Subscriber<? super String> subscriber : subscribers) {
            subscriber.onComplete();
        }
    }
}

public class Main {
    public static void main(String[] args) {
        // Create an instance of TrendingHashtagsPublisher
        TrendingHashtagsPublisher hashtagsPublisher = new
TrendingHashtagsPublisher();

        // Create an instance of TrendingHashtagsProcessor
        TrendingHashtagsProcessor hashtagsProcessor = new
TrendingHashtagsProcessor();

        // Subscribe the processor to the publisher
        hashtagsPublisher.subscribe(hashtagsProcessor);

        // Simulate the arrival of hashtags
        hashtagsPublisher.publishHashtag("#Java");
        hashtagsPublisher.publishHashtag("#ReactiveProgramming");
        hashtagsPublisher.publishHashtag("#Java9");
        hashtagsPublisher.publishHashtag("#Concurrency");

        // Complete the publisher to trigger completion in the processor
```

```
        hashtagsPublisher.complete();

        // Print the top three trending hashtags
        System.out.println("Top Trending Hashtags: " +
hashtagsProcessor.getTopTrendingHashtags());
    }
}




//Output:
//Top Trending Hashtags: [#Java, #ReactiveProgramming, #Java9]
```