ASSIGNMENT 23.1

Task 1:
Create a java program MyKafkaProducer.java that takes a file name and delimiter as input arguments.
It should read the content of file line by line.
Fields in the file are in following order
1. Kafka Topic Name
2. Key
3. value

For every line, insert the key and value to the repsective Kafka broker in a fire and forget mode.
After record is sent, it should print appropriate message on screen.
Pass **dataset_producer.txt** as the input file and -as delimiter.
LINK: https://drive.google.com/file/d/0B_Qjau8wv1KoSnR5eHpKOF9rTFU/view?usp=sharing

Task 2:
Modify the previous program MyKafkaProducer.java and create a new Java program KafkaProducerWithAck.java
This should perform the same task as of KafkaProducer.java with some modification.
When passing any data to a topic, it should wait for acknowledgement.
After acknowledgement is received from the broker, it should print the key and value which has been written to a specified topic.
The application should attempt for 3 retries before giving any exception.
Pass **dataset_producer.txt** as the input file and -as delimiter.

SOLUTION

The solution source code has been given below. Following are the main features of the program

1. We use the KafkaProducer class to create the Kafka producer
2. We provide properties of the Producer class by setting values in Properties() class
3. For task1 – we use send method of the producer class without any callback which returns immediately
4. For Task2 – We set the properties of the producer to wait for Ack and to retry 3 times. These are configured via props.put("acks", "all") and props.put("retries", 3);
5. Kafka automatically creates the topic if it does not exists when a producer sends a message with a given topic
6. For both the cases we kill the Kafka Broker to see the behavior of the producer. In first case nothing happens as it is fire and forget. For the second case we see that we encounter exceptions as the acknowledgement is not received

```java
package kafka;
import org.apache.kafka.clients.producer.*;
import java.util.Properties;
import java.io.*;


public class MyKafkaProducer {

	public static void main(String[] args)
	{

		String filename=null;
		String separator=null;
		if(args.length != 2)
		{
			System.out.println("usage: MyKafkaProducer <filename>
			<separator>");
			return;
		}
		// read file here based on the filename and delimer

		filename = args[0];
		separator = args[1];
		System.out.println("filename: " + filename);
		System.out.println("separator is: " + separator);


		// create instance for properties to configure producer
		Properties props = new Properties();

		// provide the connection port to the default kafka server
		props.put("bootstrap.servers", "localhost:9092");

		// since key and value are Strings, specify default serializer
		props.put("key.serializer",
			"org.apache.kafka.common.serialization.StringSerializer");

		props.put("value.serializer",
			"org.apache.kafka.common.serialization.StringSerializer");

		// create the producer
		Producer<String, String> producer = new KafkaProducer
			<String, String>(props);


		try
		{
			File file = new File(args[0]);

			BufferedReader br = new BufferedReader(new FileReader(file));

			String st;
			while ((st = br.readLine()) != null)
			{
				System.out.println(st);
```

```
                        // split the string in topic, key and value
                        String[] s = st.split(args[1]);
                        String topic = s[0];
                        String key = s[1];
                        String value = s[2];
                        // send these to broker via producer.
                        // Send is called immediately without waiting for the ack
                        producer.send(new ProducerRecord<String, String>(topic,
                                key,value));

                        System.out.println("Sent record with Topic: " + topic +
                                " Key: "+key + " value: "+value );
                }


        }
        catch(Exception e)
        {
                e.printStackTrace();
                return;
        }

        producer.close();


    }
}
```

Program output

We first launch zookeeper and Kafka Server with default configuration
We run a console consumer with topic "UserTopic" to see if our producer is working
We run our program by providing the filename and separator as program argument

```
[acadgild.mmisra ~]$ java -jar MyKafkaProducer.jar dataset_producer.txt -
filename: dataset_producer.txt
separator is: -
ERROR StatusLogger No log4j2 configuration file found. Using default configuration: logging only errors to the console. Set system property 'org.apache.logging.log4j.simplelog.StatusLogger.level' to TRACE to show Log4j2 internal initialization logging.
ItemTopic-{"item_id":"101"}-{"user_id":"U101"}
Sent record with Topic: ItemTopic Key: {"item_id":"101"} value: {"user_id":"U101"}
UserTopic-{"name":"John"}-{"exp":16}
Sent record with Topic: UserTopic Key: {"name":"John"} value: {"exp":16}
ItemTopic-{"item_id":"101"}-{"user_id":"U106"}
Sent record with Topic: ItemTopic Key: {"item_id":"101"} value: {"user_id":"U106"}
UserTopic-{"name":"Mark"}-{"exp":18}
Sent record with Topic: UserTopic Key: {"name":"Mark"} value: {"exp":18}
ItemTopic-{"item_id":"102"}-{"user_id":"U110"}
Sent record with Topic: ItemTopic Key: {"item_id":"102"} value: {"user_id":"U110"}
UserTopic-{"name":"Cylin"}-{"exp":15}
Sent record with Topic: UserTopic Key: {"name":"Cylin"} value: {"exp":15}
```

```
ItemTopic-{"item_id":"102"}-{"user_id":"U101"}
Sent record with Topic: ItemTopic Key: {"item_id":"102"} value: {"user_id":"U101"}
UserTopic-{"name":"Prod"}-{"exp":14}
Sent record with Topic: UserTopic Key: {"name":"Prod"} value: {"exp":14}
ItemTopic-{"item_id":"104"}-{"user_id":"U102"}
Sent record with Topic: ItemTopic Key: {"item_id":"104"} value: {"user_id":"U102"}
UserTopic-{"name":"Abhay"}-{"exp":17}
Sent record with Topic: UserTopic Key: {"name":"Abhay"} value: {"exp":17}
ItemTopic-{"item_id":"107"}-{"user_id":"U104"}
Sent record with Topic: ItemTopic Key: {"item_id":"107"} value: {"user_id":"U104"}
UserTopic-{"name":"Misano"}-{"exp":19}
Sent record with Topic: UserTopic Key: {"name":"Misano"} value: {"exp":19}
[acadgild.mmisra ~]$

[acadgild.mmisra ~]$ $KAFKA_HOME/bin/kafka-console-consumer.sh --topic UserTopic --from-beginning \
> --zookeeper localhost:2181 --property print.key=true --property "parse.key=true" \
> --property "key.separator=-"
Using the ConsoleConsumer with old consumer is deprecated and will be removed in a future major release.
Consider using the new consumer by passing [bootstrap-server] instead of [zookeeper].
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [file:/home/acadgild/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/acadgild/install/kafka/kafka_2.12-0.10.1.1/libs/slf4j-log4j12-
1.7.21.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
ERROR StatusLogger No log4j2 configuration Set system property
'org.apache.logging.log4j.simplelog.StatusLogger.level' to TRACE to show Log4j2 internal initialization logging.
{"name":"John"}-{"exp":16}
{"name":"Mark"}-{"exp":18}
{"name":"Cylin"}-{"exp":15}
{"name":"Prod"}-{"exp":14}
{"name":"Abhay"}-{"exp":17}
{"name":"Misano"}-{"exp":19}
file found. Using default configuration: logging only errors to the console.
```

Kafka Producer with Ack

The source code is given below it is modified from the previous one in the following way

1. We set the ack-all properties and the retry count in the properties of the producer
2. During the send call, we provide the callback function which gets called when the producer receives the ack.

```
package kafka;

import org.apache.kafka.clients.producer.*;
import java.util.Properties;
import java.io.*;


public class MyKafkaProducerWithAck {

	public static void main(String[] args)
	{

		String filename=null;
		String separator=null;
		if(args.length != 2)
		{
			System.out.println("usage: MyKafkaProducerWithAck <filename>
<separator>");
			return;
		}
		// read file here based on the filename and delimer

		filename = args[0];
		separator = args[1];
		System.out.println("filename: " + filename);
		System.out.println("separator is: " + separator);



		// create instance for properties to configure producer
		Properties props = new Properties();

		// provide the connection port to the default kafka server
		props.put("bootstrap.servers", "localhost:9092");

		// since key and value are Strings, specify default serializer
		props.put("key.serializer",
				"org.apache.kafka.common.serialization.StringSerializer");

		props.put("value.serializer",
				"org.apache.kafka.common.serialization.StringSerializer");

		//Set acknowledgements for producer requests. This is needed for
		acknowledge
	       props.put("acks", "all");

		 //If the request fails, the producer can automatically retry, set to 3
	       props.put("retries", 3);

		// create the producer
		Producer<String, String> producer = new KafkaProducer
				<String, String>(props);
```

```
		try
		{
			File file = new File(args[0]);

			BufferedReader br = new BufferedReader(new FileReader(file));

			String st;
			while ((st = br.readLine()) != null)
			{
				System.out.println(st);

				// split the string in topic, key and value
				String[] s = st.split(args[1]);
				String topic = s[0];
				String key = s[1];
				String value = s[2];
			// provide a callback function as part of send which gets called
				// when the acknowledgement is received
				producer.send(new ProducerRecord<String, String>(topic,
							key,value),new Callback() {
                        public void onCompletion(RecordMetadata metadata, Exception
e) {
                            if(e != null) {
                               e.printStackTrace();
                            } else {

                                System.out.println("Ack received, Sent record
with Topic: " + topic +
                                                    " Key: "+key + " value: "+value
);

                            }
                        }
                    });

			}


		}
		catch(Exception e)
		{
			e.printStackTrace();
			return;
		}
		producer.close();

	}
}
```

Program output

We first launch zookeeper and Kafka Server with default configuration
We run a console consumer with topic “UserTopic” to see if our producer is working
We run our program by providing the filename and separator as program argument

```
[acadgild.mmisra ~]$ java -jar MyKafkaProducerWithAck.jar dataset_producer.txt -
filename: dataset_producer.txt
separator is: -
ERROR StatusLogger No log4j2 configuration file found. Using default configuration: logging only errors to the console. Set system property 'org.apache.logging.log4j.simplelog.StatusLogger.level' to TRACE to show Log4j2 internal initialization logging.
ItemTopic-{"item_id":"101"}-{"user_id":"U101"}
UserTopic-{"name":"John"}-{"exp":16}
Ack received, Sent record with Topic: ItemTopic Key: {"item_id":"101"} value: {"user_id":"U101"}
ItemTopic-{"item_id":"101"}-{"user_id":"U106"}
UserTopic-{"name":"Mark"}-{"exp":18}
ItemTopic-{"item_id":"102"}-{"user_id":"U110"}
UserTopic-{"name":"Cylin"}-{"exp":15}
ItemTopic-{"item_id":"102"}-{"user_id":"U101"}
UserTopic-{"name":"Prod"}-{"exp":14}
ItemTopic-{"item_id":"104"}-{"user_id":"U102"}
UserTopic-{"name":"Abhay"}-{"exp":17}
ItemTopic-{"item_id":"107"}-{"user_id":"U104"}
UserTopic-{"name":"Misano"}-{"exp":19}
Ack received, Sent record with Topic: UserTopic Key: {"name":"John"} value: {"exp":16}
Ack received, Sent record with Topic: UserTopic Key: {"name":"Mark"} value: {"exp":18}
Ack received, Sent record with Topic: UserTopic Key: {"name":"Cylin"} value: {"exp":15}
Ack received, Sent record with Topic: UserTopic Key: {"name":"Prod"} value: {"exp":14}
Ack received, Sent record with Topic: UserTopic Key: {"name":"Abhay"} value: {"exp":17}
Ack received, Sent record with Topic: UserTopic Key: {"name":"Misano"} value: {"exp":19}
Ack received, Sent record with Topic: ItemTopic Key: {"item_id":"101"} value: {"user_id":"U106"}
Ack received, Sent record with Topic: ItemTopic Key: {"item_id":"102"} value: {"user_id":"U110"}
Ack received, Sent record with Topic: ItemTopic Key: {"item_id":"102"} value: {"user_id":"U101"}
Ack received, Sent record with Topic: ItemTopic Key: {"item_id":"104"} value: {"user_id":"U102"}
Ack received, Sent record with Topic: ItemTopic Key: {"item_id":"107"} value: {"user_id":"U104"}
[acadgild.mmisra ~]$
```

Now we kill the broker and run the producer again. It should show exceptions as the acknowledgement of the messages is not received

```
[acadgild.mmisra ~]$ java -jar MyKafkaProducerWithAck.jar dataset_producer.txt -
filename: dataset_producer.txt
separator is: -
ERROR StatusLogger No log4j2 configuration file found. Using default configuration: logging only errors to the console. Set system property 'org.apache.logging.log4j.simplelog.StatusLogger.level' to TRACE to show Log4j2 internal initialization logging.
ItemTopic-{"item_id":"101"}-{"user_id":"U101"}
org.apache.kafka.common.errors.TimeoutException: Failed to update metadata after 60000 ms.
UserTopic-{"name":"John"}-{"exp":16}
```