

ClientSide Code

Advertisement.java

```
package com.example;

import java.rmi.Remote;
import java.rmi.RemoteException;

// Class representing the advertisement data
public class Advertisement implements java.io.Serializable {
    private static final long serialVersionUID = 1L;
    private String content;
    private String advertiserName;
    private String publicationDate;
    // Additional advertisement attributes and methods can be added here

    public Advertisement(String content, String advertiserName, String
publicationDate) {
        this.content = content;
        this.advertiserName = advertiserName;
        this.publicationDate = publicationDate;
    }

    // Getter and setter methods for advertisement attributes
}
```

AdvertisementProducer.java

```
package com.example;

import org.apache.kafka.clients.producer.KafkaProducer;
import org.apache.kafka.clients.producer.ProducerRecord;
import java.util.Properties;

public class AdvertisementProducer {
    private static final String TOPIC_NAME = "advertisement-topic";

    public static void main(String[] args) {
        // Set Kafka producer properties
        Properties props = new Properties();
        props.put("bootstrap.servers", "localhost:9092");
        props.put("key.serializer", "org.apache.kafka.common.serialization.StringSerializer");
        props.put("value.serializer", "org.apache.kafka.common.serialization.StringSerializer");

        try (KafkaProducer<String, String> producer = new KafkaProducer<>(props)) {
            // Create and send advertisement messages to the Kafka topic
            for (int i = 0; i < 10; i++) {
                String advertisementContent = "Advertisement " + i + " Content";
                String advertiserName = "Advertiser X";
                String publicationDate = "2024-04-25";

                ProducerRecord<String, String> record = new ProducerRecord<>(TOPIC_NAME,
advertisementContent);
                producer.send(record);
                System.out.println("Sent advertisement: " + advertisementContent);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

AdvertisementsInterface.java

```
package com.example;

import java.rmi.Remote;
import java.rmi.RemoteException;

// Remote interface for handling advertisements
public interface AdvertisementsInterface extends Remote {
    void receiveAdvertisement(Advertisement advertisement) throws RemoteException;
}
```

MarketingDepartmentClient.java

```
package com.example;

import java.rmi.Naming;
import java.rmi.RemoteException;
import com.example.Advertisement;
import com.example.AdvertisementsInterface;

public class MarketingDepartmentClient {
    public static void main(String[] args) {
        try {
            // Lookup the remote object from the RMI registry
            AdvertisementsInterface server = (AdvertisementsInterface)
Naming.lookup("//localhost/PublishingCenter");
```

```

        // Create and send an advertisement to the Publishing Center
        Advertisement advertisement = new Advertisement("Advertisement Content", "Advertiser
X", "2024-04-25");
        server.receiveAdvertisement(advertisement);

        System.out.println("Advertisement sent successfully.");
    } catch (Exception e) {
        System.err.println("Client exception: " + e.toString());
        e.printStackTrace();
    }
}
}

```

SERVICESIDE CODE

Advertisement.java

```

package com.example;

import java.rmi.Remote;
import java.rmi.RemoteException;

// Class representing the advertisement data
public class Advertisement implements java.io.Serializable {
    private static final long serialVersionUID = 1L;
    private String content;
    private String advertiserName;
    private String publicationDate;
    // Additional advertisement attributes and methods can be added here

    public Advertisement(String content, String advertiserName, String
publicationDate) {
        this.content = content;
        this.advertiserName = advertiserName;
    }
}

```

```

        this.publicationDate = publicationDate;
    }

    // Getter and setter methods for advertisement attributes
    public String getContent() {
        return content;
    }

    public String getAdvertiserName() {
        return advertiserName;
    }

    public String getPublicationDate() {
        return publicationDate;
    }
}

```

AdvertisementsInterface.java

```

package com.example;

import java.rmi.Remote;
import java.rmi.RemoteException;

// Remote interface for handling advertisements
public interface AdvertisementsInterface extends Remote {
    void receiveAdvertisement(Advertisement advertisement) throws
    RemoteException;
}

```

PublishingCenterConsumer.java

```

package com.example;

import org.apache.kafka.clients.consumer.ConsumerRecords;
import org.apache.kafka.clients.consumer.KafkaConsumer;

```

```

import org.apache.kafka.common.serialization.StringDeserializer;
import java.time.Duration;
import java.util.Collections;
import java.util.Properties;

public class PublishingCenterConsumer {
    private static final String TOPIC_NAME = "advertisement-topic";

    public static void main(String[] args) {
        // Set Kafka consumer properties
        Properties props = new Properties();
        props.put("bootstrap.servers", "localhost:9092");
        props.put("group.id", "publishing-center-group");
        props.put("key.deserializer", StringDeserializer.class.getName());
        props.put("value.deserializer", StringDeserializer.class.getName());

        try (KafkaConsumer<String, String> consumer = new KafkaConsumer<>(props)) {
            // Subscribe to the Kafka topic
            consumer.subscribe(Collections.singletonList(TOPIC_NAME));

            // Poll for new advertisement messages
            while (true) {
                ConsumerRecords<String, String> records = consumer.poll(Duration.ofMillis(100));
                records.forEach(record -> {
                    System.out.println("Received advertisement: " + record.value());
                    // Process the received advertisement message
                    // Further processing logic can be added here
                });
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

PublishingCenterServer.java

```

package com.example;

import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;

```

```

import com.example.Advertisement;
import com.example.AdvertisementsInterface;

public class PublishingCenterServer extends UnicastRemoteObject implements
AdvertisementsInterface {
    protected PublishingCenterServer() throws RemoteException {
        super();
    }

    @Override
    public void receiveAdvertisement(Advertisement advertisement) throws
RemoteException {
        // Logic to process received advertisement
        System.out.println("Received advertisement from " +
advertisement.getAdvertiserName());
        System.out.println("Content: " + advertisement.getContent());
        System.out.println("Publication Date: " +
advertisement.getPublicationDate());
        // Further processing of the advertisement can be added here
    }

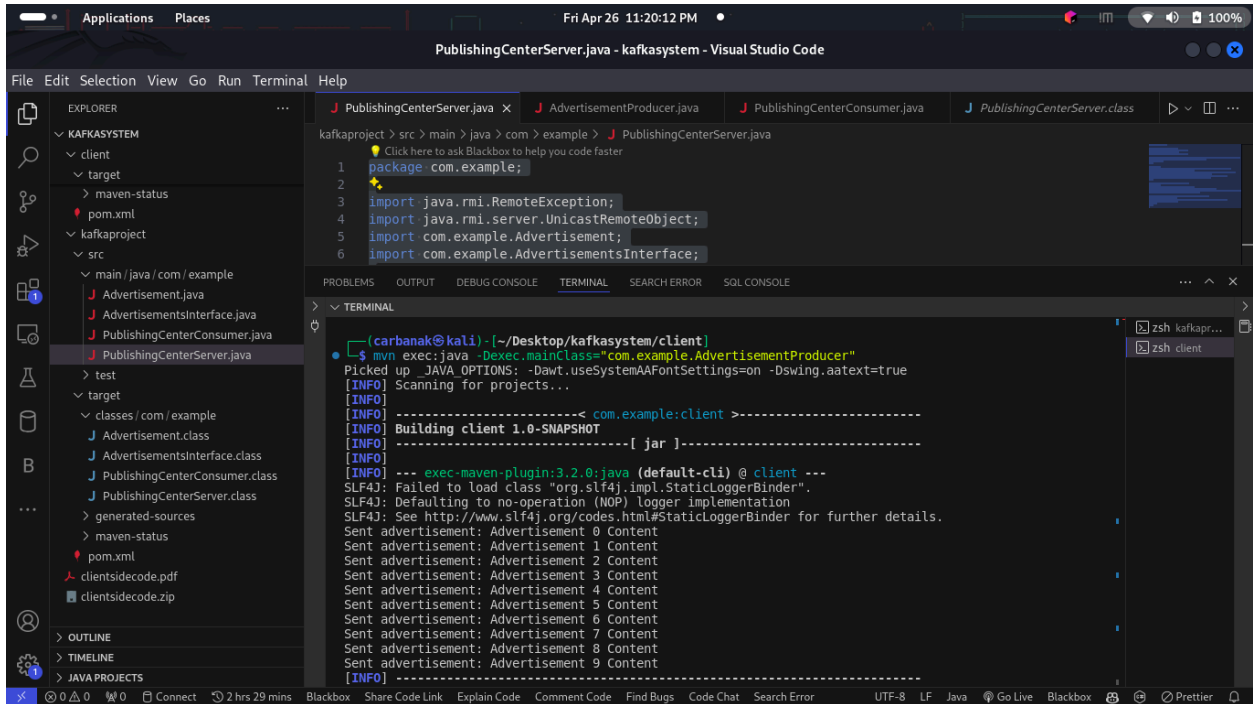
    public static void main(String[] args) {
        try {
            // Create and export the remote object
            AdvertisementsInterface server = new PublishingCenterServer();
            java.rmi.registry.Registry registry =
java.rmi.registry.LocateRegistry.createRegistry(1099);
            registry.rebind("PublishingCenter", server);

            System.out.println("Publishing Center Server ready.");
        } catch (Exception e) {
            System.err.println("Publishing Center Server exception: " +
e.toString());
            e.printStackTrace();
        }
    }
}

```

IMPLEMENTATION SCREENSHOTS

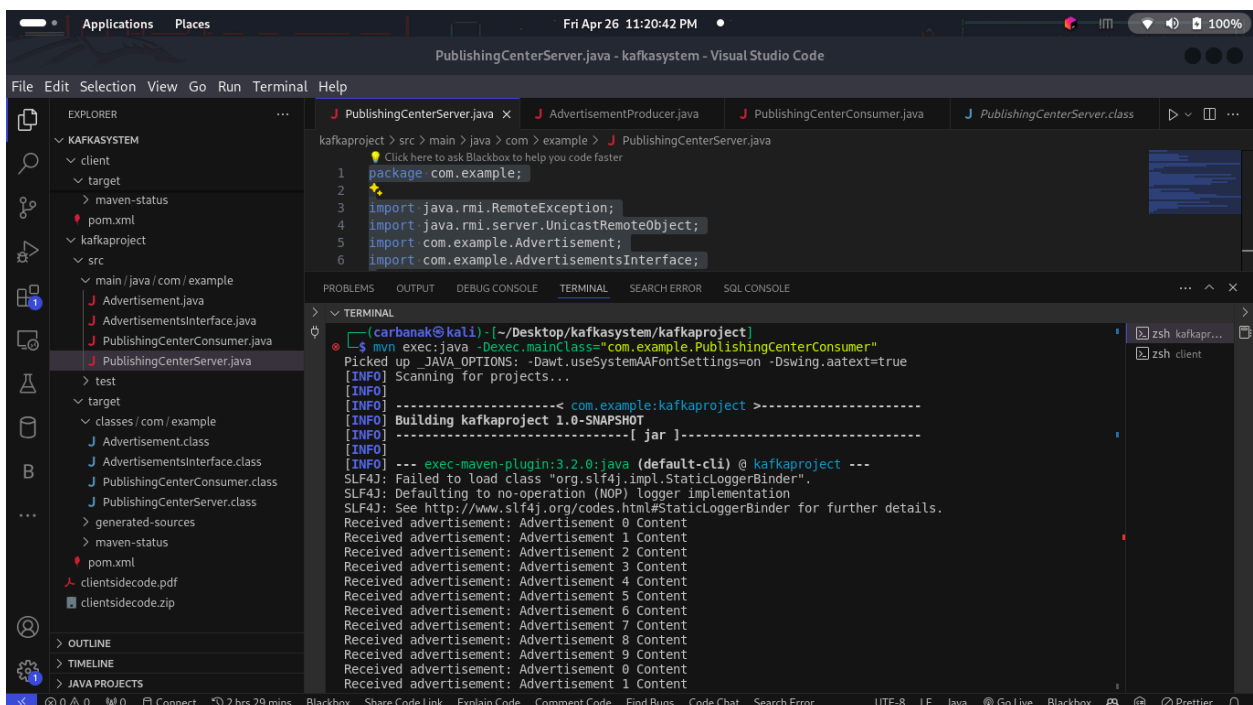
SENT ADVERTISEMENTS



```
1 package com.example;
2
3 import java.rmi.RemoteException;
4 import java.rmi.server.UnicastRemoteObject;
5 import com.example.Advertisement;
6 import com.example.AdvertisementsInterface;
```

```
carbanak@kali: ~/Desktop/kafkasystem/client
$ mvn exec:java -Dexec.mainClass=com.example.AdvertisementProducer
Picked up JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
[INFO] Scanning for projects...
[INFO]
[INFO] -----< com.example:client >-----
[INFO] Building client 1.0-SNAPSHOT
[INFO]
[INFO] -----[ jar ]-----
[INFO]
[INFO] --- exec-maven-plugin:3.2.0:java (default-cli) @ client ---
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
Sent advertisement: Advertisement 0 Content
Sent advertisement: Advertisement 1 Content
Sent advertisement: Advertisement 2 Content
Sent advertisement: Advertisement 3 Content
Sent advertisement: Advertisement 4 Content
Sent advertisement: Advertisement 5 Content
Sent advertisement: Advertisement 6 Content
Sent advertisement: Advertisement 7 Content
Sent advertisement: Advertisement 8 Content
Sent advertisement: Advertisement 9 Content
[INFO]
```

RECEIVED ADVERTISEMENTS



```
1 package com.example;
2
3 import java.rmi.RemoteException;
4 import java.rmi.server.UnicastRemoteObject;
5 import com.example.Advertisement;
6 import com.example.AdvertisementsInterface;
```

```
carbanak@kali: ~/Desktop/kafkasystem/kafkaproject
$ mvn exec:java -Dexec.mainClass=com.example.PublishingCenterConsumer
Picked up JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
[INFO] Scanning for projects...
[INFO]
[INFO] -----< com.example:kafkaproject >-----
[INFO] Building kafkaproject 1.0-SNAPSHOT
[INFO]
[INFO] -----[ jar ]-----
[INFO]
[INFO] --- exec-maven-plugin:3.2.0:java (default-cli) @ kafkaproject ---
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
Received advertisement: Advertisement 0 Content
Received advertisement: Advertisement 1 Content
Received advertisement: Advertisement 2 Content
Received advertisement: Advertisement 3 Content
Received advertisement: Advertisement 4 Content
Received advertisement: Advertisement 5 Content
Received advertisement: Advertisement 6 Content
Received advertisement: Advertisement 7 Content
Received advertisement: Advertisement 8 Content
Received advertisement: Advertisement 9 Content
Received advertisement: Advertisement 0 Content
Received advertisement: Advertisement 1 Content
```