43233

Assignment 6

- Title
  To develop any distributed application using Messaging System in Publisher-Subscriber paradigm
- Tools
  Java Programming Environment, JDK 8, Eclipse IDE, Apache ActiveMQ 4.1.1, JMS

- Theory
  Large distributed systems are often overwhelmed with complications caused by heterogeneity and interoperability. Heterogeneity issues may arise due to the use of different programming languages, hardware platforms, operating systems, and data representations. Interoperability denotes the ability of heterogeneous systems to communicate meaningfully and exchange data or services. With the introduction of middleware, heterogeneity can be alleviated and interoperability can be achieved.Middleware is a layer of software between the distributed application and the operating system and consists of a set of standard interfaces that help the application use network resources and services.
  - The publish/subscribe messaging paradigm
    The publish/subscribe messaging paradigm is built with the concept of a topic, which behaves like an announcement board. Consumers subscribe to receiving messages that belong to a topic, and publishers report messages to a topic. The JMS provider retains the

responsibility for distributing the messages that it receives from multiple publishers to many other subscribers based on the topic they subscribe to. A subscriber receives messages that it subscribes to based on the rules it defines and the messages that are published after the subscription is registered; they do not receive any messages that are already published

- JMS interfaces
    - JMS defines a set of high-level interfaces that encapsulate several messaging concepts. These high-level interfaces are further extended for the Point-To-Point and publish/subscribe messaging domains:
    - ConnectionFactory:This is an administered object with the ability to create a connection.
    - Connection:This is an active connection handle to the provider.
    - Destination:This is an administered object that encapsulates the identity of a message destination where messages are sent to/received from.
    - Session:This is a single-threaded context for sending/receiving messages. To Ensure a simple session-based transaction, concurrent access to a message by multiple threads is restricted. We can use multiple sessions for a multithreaded application
    - MessageProducer:This is used to send messages.
    - MessageConsumer:This is used to receive messages

- In ‘Publisher-Subscriber’ pattern, senders of messages, called publishers, do not program the messages to be sent directly to specific receivers, calledsubscribers.For example, consider there is a publisher publishes news (topics) related to politics and sports;they publish to the Messaging Broker, as shown in the following diagram. While Subscriber 1 receives news related to politics and Subscriber 3 receives news related to sports, Subscriber 2 will receive both political and sports news as it subscribed to the common topics.In designing our solution, we have created one publisher and subscriber wherein the publisher creates the topic.
- ThePublisher/Subscriberpattern is mostly implemented in anasynchronousway (using message queue).
- Publishers and subscribers have a timing dependency. A client that subscribes to a topic can consume only messages published after the client has created a subscription, and the subscriber must continue to be active in order for it to consume messages.JMS is a Java API that allows applications to create, send, receive, and read messages. The JMS API enables communication that is loosely coupled, asynchronous and reliable.To use JMS, we need to have a JMS provider that can manage the sessions, queues, and topics. Some examples of known JMS providers areApache ActiveMQ, WebSphere MQ from IBM or SonicMQ from Aurea Software. Starting from Java

EE version 1.4, a JMS provider has to be contained in all Java EE application servers

- Administered objects are preconfigured JMS objects, such as a connection factory (the object a client uses to create a connection to a provider) and a destination (the object a client uses to specify a target for its messages).JMS applications are usually developed in either the publish/subscribe or Point-To-Point paradigm.

Output

```
INFO | Successfully connected to tcp://localhost:61616
Received Message : 'Manvi 43233 R-10'

Message sent 'Manvi 43233 R-10'
```

Conclusion

In this assignment we studied the publisher subscriber model which was implemented using JMS.