Deliverable

***• GITHUB URL with the project checked into the GITHUB***

• Code

• Documentation

• Answers to the challenges

Part 1

Programming Challenge 1:

You are to create two small applications for this programming task; one is called Randomizer, the other Prime Randomizer‘s job is to generate a series of positive random integers and send those to Prime via a distributed queue of integers.

Primes job is to receive the integers and calculate whether the integer is a prime or not and return the answer to Randomizer via a distributed queue ( just a java Queue implementation , no need to implement JMS etc ) that contains the original number and a Boolean; which Randomizer will print to system out.

Points

1. Use only the standard java library

2. Both Applications will run on the same server

3. The system should be as fast as possible

4. The results do not have to be returned in the same order as received

5. You don’t have to go overboard tweaking the prime check

Solution:

**import** javax.naming.InitialContext;

**import** javax.jms.Queue;

**import** javax.jms.Session;

**import** javax.jms.TextMessage;

**import** javax.jms.QueueSender;

**import** java.util.Random;

**import** javax.jms.DeliveryMode;

**import** javax.jms.QueueSession;

**import** javax.jms.QueueConnection;

**import** javax.jms.QueueConnectionFactory;

**public class** Sender {

**public static void** main(String[] args) **throws** Exception {

// get the initial context

InitialContextctx = **new**InitialContext();

// lookup the queue object

Queue queue = (Queue) ctx.lookup("queue://" + args[0]);

// lookup the queue connection factory

QueueConnectionFactoryconnFactory = (QueueConnectionFactory) ctx.lookup("queue/connectionFactory");

// create a queue connection

QueueConnectionqueueConn = connFactory.createQueueConnection();

// create a queue session

QueueSessionqueueSession = queueConn.createQueueSession(**false**, Session.***DUPS\_OK\_ACKNOWLEDGE***);

// create a queue sender

QueueSenderqueueSender = queueSession.createSender(queue);

queueSender.setDeliveryMode (DeliveryMode.***NON\_PERSISTENT***);

// create a simple message to say "Hello"

Random random = **new**Random(1);

**int**num = Math.*abs*(random.nextInt());

TextMessagemessage = queueSession.createTextMessage();

message.setText(String.*valueOf*(num));

// send the message

queueSender.send(message);

// close the queue connection

queueConn.close();

}

}

**import** javax.jms.DeliveryMode;

**import** javax.jms.Queue;

**import** javax.jms.QueueConnection;

**import** javax.jms.QueueConnectionFactory;

**import** javax.jms.QueueReceiver;

**import** javax.jms.QueueSender;

**import** javax.jms.QueueSession;

**import** javax.jms.Session;

**import** javax.jms.TextMessage;

**import** javax.naming.InitialContext;

**public class** Receiver

{

**public static void** main(String[] args) **throws** Exception

{

// get the initial context

InitialContext ctx = **new**InitialContext();

// lookup the queue object

Queue queue = (Queue) ctx.lookup("queue://" + args[0]);

// lookup the queue connection factory

QueueConnectionFactoryconnFactory = (QueueConnectionFactory) ctx.

lookup("queue/connectionFactory");

// create a queue connection

QueueConnectionqueueConn = connFactory.createQueueConnection();

// create a queue session

QueueSessionqueueSession = queueConn.createQueueSession(**false**,

Session.***AUTO\_ACKNOWLEDGE***);

// create a queue receiver

QueueReceiverqueueReceiver = queueSession.createReceiver(queue);

// start the connection

queueConn.start();

// receive a message

TextMessagemessage = (TextMessage) queueReceiver.receive();

**int**inputnum = Integer.*parseInt*(message.getText());

// print the message

String isPrimeStr = inputnum+"-" + (*isPrime*(inputnum)? "true" :"false");

// create a queue sender

QueueSenderqueueSender = queueSession.createSender(queue);

queueSender.setDeliveryMode(DeliveryMode.***NON\_PERSISTENT***);

// create a simple message to say "Hello"

TextMessageresponseMessage = queueSession.createTextMessage();

responseMessage.setText(isPrimeStr);

// send the message

queueSender.send(message);

// close the queue connection

queueConn.close();

}

**public static boolean** isPrime(**int** num) {

**boolean**flag = **false**;

**for** (**int**i = 2; i<= num / 2; ++i) {

**if** (num % i == 0) {

flag = **true**;

**break**;

}

}

**return** flag;

}

Readme.txt

I created distributed Queue suing javax.jms.queue. Sender file generates random positive integer and sends it to receiver. Receiver will check if the number is prime or not and returns original number and result. It needs javax.jms-api-2.0.jar to be in lcasspath

Deliverable

• A project with code – eclipse or any IDE that you are using. However, eclipse is preferable

• A Readme.txt should contain the following

• What is your design and implementation?

• Sample Output

• Further work – if you have all the time in the world how would you implement

Programming Challenge 2:

• Reverse of a string without reverse method.

Solution:

//Reverse String using recursion

public class ReverseStringTest {

public static void main(String[] args) {

String str = "Test String";

String reversed = reverseString(str);

System.out.println("The reversed string is: " + reversed);

}

public static String reverseString(String str)

{

if (str.isEmpty())

return str;

return reverseString(str.substring(1)) + str.charAt(0);

}

}

• Find a palindrome

Ans :

public class PolyndromeTest {

public static void main(String[] args) {

PolyndromeTestpolyndromeTest = new PolyndromeTest();

System.out.println("Is Polyndrome ?:"+polyndromeTest.isPolyndrome("abcba"));

}

public booleanisPolyndrome(String str){

return str.equals(reverseString(str));

}

public static String reverseString(String str) {

if (str.isEmpty())

return str;

return reverseString(str.substring(1)) + str.charAt(0);

}

}

2)      Please create a class for the below and send it,

We have a table which has 4 columns as id, name, phone and address.

You need to have a method which will return me the data.

Create a method where you can hard code the data and print the details.

The main aim of above example is how you use data structure.

public class Employee {

private int id;

private String name;

private String phone;

private String address;

public Employee(int id, String name, String phone, String address) {

super();

this.id = id;

this.name = name;

this.phone = phone;

this.address = address;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getPhone() {

return phone;

}

public void setPhone(String phone) {

this.phone = phone;

}

public String getAddress() {

return address;

}

public void setAddress(String address) {

this.address = address;

}

}

public interface IEmployeeService {

public Employee getEmployee();

}

public class EmployeeService implements IEmployeeService {

@Override

public Employee getEmployee() {

return new Employee(1,"test","111-111-1111", "1 test str NY");

}

}

Part 2

Representative Questions – Please write your answers with an example for each questions.

• How do you design an application with JMS messaging?

• How do you handle exception in JMS consumers and how to you recover?

• How do you implement LRU or MRU cache?

• How would you implement Executor Service?

• Describe singleton design pattern – how would you implement?

• Describe properties of Java String.

Deliverable

• Simple document with Question and corresponding answers. If you have used data from anywhere please include reference.