Programming Challenge 1:

Design and Implementation:

I used a simple concurrent access provided by java util package. It provides ConcurrentLinkedQueue which can be accessed by various threads by time sharing and updated. The important thing about it is that it is synchronized and to access it java class only needs to implement runnable interface.

Files are uploaded on github under the folder name RandomizedPrimeChecker

The RandomizedPrime class calls the randomizer thread and prime thread by passing the input and output queue.

InputQueue: Integer concurrent linked queue

OutputQueue: concurrent linked queue holding two objects.

Randomizer thread generates the number and adds it to the input queue and checks the output queue for printing any results.

Prime thread checks input queue for input and calls the isPrime(int num) method and add the output to the output queue. The sample run works for 10 members and prime number within range of 0-100.

Sample output

```
Sample run for Prime Checker:
Prime checker is started
Randomizer started
Added to input queue from randomizer:10
Added to input queue from randomizer:38
 Integer: 10 isPrime: false
 Integer: 38 isPrime: false
Added to input queue from randomizer:68
Added to input queue from randomizer:1
 Integer: 68 isPrime: false
 Integer: 1 isPrime: false
Added to input queue from randomizer:35
 Integer: 35 isPrime: true
Added to input queue from randomizer:24
Added to input queue from randomizer:41
 Integer: 24 isPrime: false
 Integer: 41 isPrime: true
Added to input queue from randomizer:37
 Integer: 37 isPrime: true
Added to input queue from randomizer:60
```

Future Work:

JMS provides the same functionality to communicate between software components of application. JMS provides destination queue instance along with the connection between the sender/consumer and receiver server.

If I will be having much time to complete the challenge then I would have setup destination queue in xml. And after that I will be starting J Boss or any other middleware application server compatible with eclipse. The I will be defining JNDI context class as both client and server would need this which contains the port number on which the J boss is running to serve the request.

Then I will write a randomizer which will be generating random number and checking the output queue which is defined in the similar fashion as input queue using xml initially. First start by creating JNDI context instance, get connection factory, create connection and use the connection object create session and link up for the input queue and add elements to it and then look up output queue if it has any object to print.

After that I would have written prime number function providing the service to check if the input is prime or not which will start by creating JNDI context instance, get connection factory, create connection and session and lookup for the queue defined and eventually start connection. I will try to implement this way in an efficient manner.

Programming Challenge 2:

1. Reverse of a string without reverse method

Approach:

1. Start at the end index of string and add it to null string rev
2. Keep traversing and adding it to rev string.
3. At the end, append null and return.

Code:

```
package challenge2;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

public class StringReverse {

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

        BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));

        //option to end the program

        char opt = 'Y';

        while(opt!='N')
        {
         //ask for the input string to reverse from the user
             System.out.println("Enter the input string you want to
reverse: ");


             String input = br.readLine();
             String rev = "";
   //traverse from end index to the zero index and add it to the
rev   string
```

```
            for(int i=input.length()-1;i>=0;i--) {

                 rev+=input.charAt(i);
            }
            //display the result
            System.out.println("Reversed String is: " +rev);

            //ask for options

            System.out.println("Do you wish to continue ? press Y
for Yes and N for No");

            opt=(br.readLine().charAt(0));

            }
        }
}
```

Sample Output:

```
Enter the input string you want to reverse:
Anshul
Reversed String is: luhsnA
Do you wish to continue? press Y for Yes and N for No
Y
Enter the input string you want to reverse:
hey
Reversed String is: yeh
Do you wish to continue? press Y for Yes and N for No
N
```

Palindrome

Code:

```
package challenge2;

import java.util.Scanner;

public class CheckPalindrome {

   public static void main(String[] args) {

        String str, rev="";
        Scanner sc = new Scanner(System.in);

        System.out.println("Enter a string");
        str = sc.nextLine();

        int length = str.length();

        for (int i = length-1; i>=0; i--)
             rev= rev+str.charAt(i);

        if(str.equals(rev))
             System.out.println("String is a palindrome");
        else
             System.out.println("String is not a palindrome");


   }

}
```

Sample Output:

```
Enter a string
radar
String is a palindrome
```

2. **Approach**:

Create a class for holding all the properties of the Contact Object. I am using array list to store collection of this objects as its dynamic and we can use list iterator which can iterate in both the directions, providing easy iteration. So, constructor allows to set the values of the object and getData() method in class returns the data in String format. (As it is asked in the question that to return the data and not the object wrapper)

In the main method, we hard code the data and store it in the array list. We use the Iterator Interface to iterate through array list and call getData() method for each object of Contact type which return the data in String format and we print the data as required.

Code:

```
  package challenge2;
public class Contact {

   //Properties of contact class declared private for data
encapsulation


   private int id;
   private String name;
   private String phone;
   private String address;

   public Contact(int id, String name, String phone, String
address) {
         this.id = id;
         this.name = name;
         this.phone = phone;
         this.address = address;
```

```
    }

    //get data method as per the requirement which return the data


    public String getData() {

return String.format("ID: %d NAME: %s PHONE: %s ADDRESS: %s",
this.id,this.name,this.phone,this.address);

    }
}



package challenge2;

import java.util.ArrayList;
import java.util.Iterator;

public class ContactClass {

    public static void main(String[] args) {

        ArrayList<Contact> contact = new ArrayList<>();
        Contact c1 = new Contact(1, "Anshul", "6602382090", "410
Franklin Ave");
        Contact c2 = new Contact(2, "Dev", "660238441", "310 B
Anderson St");
        Contact c3 = new Contact(3, "Abhideep", "660237867", "554
Jefferson St");

        contact.add(c1);
        contact.add(c2);
        contact.add(c3);


        Iterator<Contact>  i  = contact.iterator();

        while(i.hasNext()) {
```

```
                System.out.println(i.next().getData());
        }

    }

}
```

Sample Output:

```
ID: 1 NAME: Anshul PHONE: 6602382090 ADDRESS: 410 Franklin Ave
ID: 2 NAME: Dev PHONE: 660238441 ADDRESS: 310 B Anderson St
ID: 3 NAME: Abhideep PHONE: 660237867 ADDRESS: 554 Jefferson St
```