3 Problems, 8 Pages	**Mid-Term Exam**	**Fall 2021**
0907522 Programming of Networks Protocols	**75 Minutes**	**December 8, 8:30AM** **30 points**
الرقم الجامعي:		**الاسـم :** key

Problem 1. For each of the following multiple-choice questions, choose the most accurate answer. (6 points)

1. **Which of the following methods <u>cannot</u> throw an *UnknownHostException*?**
   a. InetAddress.getByName("google.com");
   b. InetAddress.getAllByName("google.com");
   c. InetAddress.getLocalHost();
   d. InetAddress.getByAddress(byteArray); // byteArray is an array of bytes
   e. System.out.println(address.getHostName()); //address is an InetAddress objects
   f. c & e
   g. d & e

2. **When executed by an untrusted applet, under the control of the default security manager, which of the following statements throws a *SecurityException*? Assume that the applet is downloaded from a server with hostname *aol.com*.**
   a. InetAddress.getByName("aol.com");
   b. InetAddress.getByName("google.com");
   c. InetAddress.getLocalHost();
   d. InetAddress.getByName("60.1.12.30");
   e. All of the above
   f. b & c
   g. b & d

3. **Which of the following is a filter input stream?**
   a. ObjectInputStream
   b. SequenceInputStream
   c. InputStreamReader
   d. PrintStream
   e. System.in
   f. None of the above
   g. a & c
   h. b & e

4. **Which of the following is true regarding using streams in Java?**
   a. You can attach any filter stream to any low-level stream as long as they are both input or both output.
   b. You can attach any filter stream to any other filter stream as long as they are both input or both output.
   c. The first stream attached to a source (when reading) or a destination (when writing) can be either a low-level stream or a filter stream.
   d. Two-way communication cannot be achieved using streams in Java.
   e. The super-classes java.io.OutputStream and java.io.InputStream can be used to instantiate streams objects of a general type.
   f. a & b
   g. a & d

5. **Which of the following is true regarding the User Datagram Protocol (UDP)?**
   a. Corrupted packets are discarded.
   b. Unordered packets are discarded.
   c. Lost packets are retransmitted.
   d. Corrupted packets are retransmitted.
   e. Flow control is not provided.
   f. a & e
   g. a & b & e

6. **Which of the following is provided by the *DatagramSocket* class in Java?**
   a. A no-arg constructor that allows a socket to be created without binding to a port number.
   b. A method that allows communication to be restricted with one device by establishing a connection with it.
   c. A method to enable nonblocking reception of packets.
   d. Methods to get the IP address and port number of the application from which a *DatagramPacket* is received.
   e. Methods to write to the input stream and output stream of the socket.
   f. a & b
   g. a & c

Problem 2. Answer the following questions. **(8 points)**

a. Identify and explain how the loopback address 127.0.0.1 is useful in programming and debugging networking applications. **(2 points)**

① It's used to test the program regardless of whether a connection to the internet exists or not.

② To test the program without network problems.

b. Explain two reasons, with specific details, of why does the *InetAddress* class use static factory methods to create objects instead of providing public constructors. **(2 points)**

① They allow objects of the subclasses to be created (IPV4 or IPV6)

② They allow cached versions of the object to be returned

③ They allow for methods to be defined with names that define the way the method performs the initialization.

↑ any two ✓

c. **Explain how the effect of blocking I/O differs between reading operations and writing operations. Identify in which type of operations blocking is more significant and why?** **(2 points)**

reading is more significant.
⇒ reading: the other side is the producer program is the consumer. It depends on data being available from the other side
⇒ writing: program is the producer data normally available locally

d. **Given the following code segment, answer the questions below.** **(2 points)**

```
try {
    InputStream input = new FileInputStream("src.txt");
    OutputStream output = new FileOutputStream("dst.txt");
    byte[] in = new byte[16];
    int bytesRead = input.read(in);
    output.write(in);
    input.close();
    output.close();
}
catch (IOException ioe){
    System.err.println ("I/O error - " + ioe);
}
```

I. **Assuming that the file *src.txt* contains the line "Programming of networks protocols.", and is encoded using UTF-8. What is the content of the file dst.txt after the code segment is executed?**

Programming of n

II. **Assuming that the file *src.txt* contains the line "Programming of networks protocols.", and is encoded using Unicode. What is the content of the file dst.txt after the code segment is executed?**

Program

Problem 3. The following program implements a UDP server that acts as a chatroom for clients. In this question, you are required to fill-in the missing implementations of the class methods to make the server fully operational. (16 points)

Note: The program requires that the server operates on port 2500, and that a client must operate on port 4000.

```
1.  import java.net.*;
2.  import java.io.*;
3.  import java.util.ArrayList;
4.
5.  public class UDPServer {
6.     private DatagramSocket socket;
7.     private ArrayList<InetAddress> clients;
8.     public UDPServer() {
9.        //missing code
10.    }
11.    public void loadClients() {
12.       //missing code
13.    }
14.    public void printSocketDetails() {
15.       //missing code
16.    }
17.    public void operateChatRoom() {
18.       printSocketDetails();
19.       for (;;) {
20.          try {
21.             DatagramPacket packet = new DatagramPacket(new byte[128], 128);
22.             try {
23.                socket.receive(packet);
24.             } catch (InterruptedIOException io) {
25.                awakenClients();
26.             }
27.             forwardPacket(packet);
28.             backupPacket(packet);
29.          } catch (IOException ioe) {
30.             System.err.println("Error : " + ioe);
31.          }
32.       }
33.    }
34.
35.  public void forwardPacket(DatagramPacket packet) {
36.       //missing code
37.    }
38.
39.    public void awakenClients() throws IOException {
40.       //missing code
41.    }
```

```
42.     public void backupPacket(DatagramPacket packet) {
43.       //missing code
44.     }
45.
46.     public static void main(String args[]) {
47.       UDPServer server = new UDPServer();
48.       server.loadClients();
49.       server.operateChatRoom();
50.     }
51. }
```

a. Write the code segment that must be filled in the constructor *UDPServer()* – line 8 such that it performs the following: 3 **(2 points)**

- **Create the socket object for the data field *socket* and bind to port number 2500.**
- **Set the timeout on receiving packets through the socket to 10 minutes.**
- **If any error occurs, it must print the statement: "Failed to start server: socket could not be created.".**

```
try {
    socket = new DatagramSocket(2500);
    socket.setSoTimeout(10*60*1000);
}
catch (Exception e) {
    System.out.println("Failed to ...");
}
```

b. What requirements are satisfied in the *InetAddress* class such that it can be serialized/deserialized successfully? (1 point)

1. It implements java.io.Serializable.
2. It contains a no-arg constructor

c. **Write the code segment that must be filled in the method *loadClients* – line 11 such that it performs the following:** **(3 points)**

- **Read (deserialize) an *ArrayList* of *InetAddresses* (which represents registered clients IPs) serialized previously on the file *clients.out* and assign it to the *clients* data field.**
- **If a problem occurs in finding the class definition, you should print the following statement to the console: *" Failed to start server: could not retrieve clients.".***
- **If a problem occurs in finding the file, you should print the following statement to the console: *"Failed to start server: no clients registered.".***
- **If a general I/O problem occurs, you should print the following statement to the console: *"Failed to start server: error occurred while loading clients.".***

```
try{
    FileInputStream fin = new FileInputStream(clients.out);
    ObjectInputStream oin = new ObjectInputStream(fin);
    try{
        clients = (ArrayList) oin.readObject();
    }
    catch(ClassNotFoundException cnf){System.out.println("Failed to start server: could not retrieve clients");}
}
catch(FileNotFoundException fnf)
{System.out.println("Failed to start server: no clients registered");}
catch(IOException io){}
```

d. **Write the code segment that must be filled in the method *printSocketDetails* – line 14 such that it performs the following:** **(2 points)**

- **Print the IP address (in dotted decimal notation) and port number to which the server socket is bound on separate lines in the following format:**
  Socket IP: …..
  Socket Port#: ……
- **Print the name (display name) of the interface to which the IP address of the server socket is bound on a new line as follows:**
  Socket NIC: …..
- **If an error occurs in retrieving info of the interface, you should print the following statement to the console: *" Could not retrieve interface info!".***

```
try{
    System.out.println("Socket IP" + socket.getLocalAddress().getHostAddress());
    System.out.println("Socket Port" + socket.getLocalPort());
    NetworkInterface nic = NetworkInterface.getByInetAddress(socket.getLocalAddress());
    System.out.println("Socket nic" + nic.getDisplayName());
}
catch(Exception e){ System.out.println( );}
```

e. Write the code segment that must be filled in the method *forwardPacket* – line 35 such that it takes a *DatagramPacket* object named *packet* and performs the following: (3 points)

- **Check if the packet is received from one of the registered clients (whose IPs are in the *ArrayList clients*). If not, it must return and discard the packet.**
- **If so, it must forward the packet to all clients whose IPs are registered in the *ArrayList clients*, except the sender. Remember that we assume that all clients operate on port number 4000.**
- **If an error occurs while forwarding the packet to one of the clients, you should print the following statement to the console: *"A problem occurred while trying to forward message to client: ….."*. In …., print the *toString* method of the *InetAddress* that represents the client. Note that you must continue forwarding packets to the remaining clients after that.**

```
InetAddress receivedIP = packet.getAddress();
boolean found = false;
for(int i=0; i<clients.size(); i++){
    if(receivedIP.equals(clients.get(i))){
        found = true;
        break;
    }
}
if(!found) return;
for(int i=0; i<clients.size(); i++){
    if(!receivedIP.equals(clients.get(i))){
        try{
            packet.setAddress(clients.get(i));
            socket.send(packet);
        }
        catch(IOException io){
            System.out.println("A problem occured
                                while trying...");
        }
    }
}
```

f. **Write the code segment that must be filled in the method *awakenClients* – line 39 such that it performs the following:** **(3 points)**

- **Broadcast the message "*Where are you guys?*" to all clients (whose IPs are registered in the *clients ArrayList*).**
- **<u>You must use a Writer to implement this method.</u>**
- **This method must not handle any thrown exception, it must be declared that it throws the exception in its header.**

```
ByteArrayOutputStream bout = new
                             ByteArrayOutputStream();
OutputStreamWriter writer = new
                            OutputStreamWriter(bout);
writer.write("Where are you guys?");
byte[] msg = bout.toByteArray();
DatagramPacket packetToSend = new
                              DatagramPacket(msg, msg.length);
packetToSend.setPort(4000);
for(int i=0; i< client.size(); i++) {
    packetToSend.setAddress(client.get(i));
    socket.send(packetToSend);
}
```

g. Write the code segment that must be filled in the method *backupPacket* – line 42 such that it takes a *DatagramPacket* named *Packet* and performs the following: (2 points)

- **Write the contents of the packet to a file name "*backup.out*" in the following format:**
  - ✓ **IP of the packet must be written as four integers representing the four octets of the IP.**
  - ✓ **Port number of the packet must also be written as an integer.**
  - ✓ **The array of bytes that represents the data must be written as an array of bytes to the file.**
- **<u>You must use Streams to implement this method.</u>**
- **If any error occurs, you should print the following statement to the console: *"Could not backup packet!"*.**

```
try {
    FileOutputStream fout = new
                        FileOutputStream("backup.out");
    DataOutputStream Dout = new
                        DataOutputStream(fout);
    byte[] address = packet.getAddress().getAddress();
    int [] intAddress = new int [address.length];
    for(int i=0; i < address.length; i++) {
        intAddress[i] = address[i];
        dout.writeInt(intAddress[i]);
    }
    dout.writeInt(packet.getPort());
    dout.write(packet.getData());
}
catch (Exception io) {
    System.out.println("Could not ...");}
```