This exam comprises 4 exercises. Make sure you read each question carefully before attempting an answer. Be sure to clearly indicate your final answer for each question. Also, be sure to state any assumptions that you are making in your answers. Each question answer should be on a separate sheet.

Exercise 1: Given the following UML Class Diagram

- The Class **ShPhone** is a Shareeha that allows two operations: call or send message.
- The Class **ShNet** is a Shareeha that allows downloading information measured in Megabytes from the Internet (each Megabyte costs 2 SR).
Class Shareeha:

Attributes:
- owner: name of the Shareeha owner.
- number: Shareeha number.
- credit: Account balance of Shareeha in riyals (SR).
- consume: Amount consumed in riyals (SR).

Methods:

remaining (): int

- In the Class ShPhone: returns the remaining duration in minutes. Note: one minute costs 0.50 SR.
- In the class ShNet: returns the remaining number of Megabytes. Note: one Megabyte costs 2 SR.

makeConnection(int x)

- In the class ShPhone: Accepts call duration x and updates credit and consume attributes.
- In the class ShNet: Accepts number of Megabytes x that must be downloaded and updates all attributes of the class ShNet (Note: one Megabyte costs 2 SR).

Class ShPhone: is a Shareeha that allows two operations: call or send message.

Attributes:
- nbofMessage: number of sent messages.

Methods:

call(in x: int, in t: int): Boolean

The parameter t represents the type of the operation:
- If t equals 0, the operation is a call and x represents the duration in minutes of the call.
- If t equals 1, the operation is a message and x represents the number of lines of this message. Each 5 lines or less is considered as 1 minute.

In the two cases, if the balance is enough then the makeConnection can be done.

Class ShNet: is a Shareeha that allows downloading information measured in Megabytes from the Internet (each Megabyte costs 2 SR).

Attributes:
- nbAccess: number of access to the Internet.
- overAccess: this class allows the owner to exceed his credit. Then the exceeded number of Megabytes is stored in the attribute overAccess.

Methods:

computeOA(): int

Computes and returns the amount in Riyals (SR) of the overAccess.

Question: Implement the class Sareeha, ShPone and ShNet.
Solution Exercise 1:

```java
public class Shareeha implements Serializable, implements Interf {
    protected String owner;
    protected int id;
    protected int credit;
    protected int consume;

    public abstract Shareeha(String w, int n, int c) {
        owner = w;
        id = n;
        credit = c;
        consume = 0;
    }

    public abstract int remaining();

    public abstract void makeConnection(int x);
}

public class ShNet extends Shareeha {
    private int nbAccess;
    private int overAccess;

    public ShNet(String w, int n, int c) {
        super(w, n, c);
        nbAccess = 0;
        overAccess = 0;
    }

    public int computeOAccess() {
        return (overAccess * 2);
    }

    public int remaining() {
        return (credit / 2);
    }

    public void makeConnection(int x) {
        nbAccess++;
        if (credit >= x * 2)
            credit -= x * 2;
        else {
            if (credit > 0)
                overAccess += ((x * 2) - credit);
            credit = 0;
        }
        else overAccess += x * 2;
        consume += x * 2;
    }
}
```
public Class ShPhone extends Shareeha
{
    private int nbofMessage;

    public ShPhone(String w, int n, int c)
    {
        super(w,n,c);
        nbofMessage = 0;
    }

    public int remaining()
    {
        return (credit/ 2);
    }

    public boolean call (int x, int t)
    {
        if (t = = 0)
        {
            if (x<=remaining())
            {
                makeConnection(x);
                return true;
            }
            else return false;
        }
        if (t = = 1)
        {
            if ((x/5)<=remaining())
            {
                nbofMessage++;
                makeConnection(x/5);
                return true;
            }
            else return false;
        }
        return false;
    }

    public void makeConnection(int x)
    {
        credit -= x /2; //x*0.5;
        consume += x/2;
    }
}
### Exercise 2: Given the following UML Class Diagram

#### Class List<\(T\)>:

- `getData(in p:int):T` returns the data at the position \(p\).
- `search(in s: String): int` returns the number of data that satisfies the condition using the string \(s\).
- `displayAll():` displays all the data in the list.
- `size():` returns the size of the list.

**Question:** Implement the class List and what you need for their methods.
Solution Exercise 2:

public class List<T extends Interf>
{
    private String name;
    private Node<T> first;

    public List<T> (s: String)
    {
        name = s;
        first = null;
    }

    public int size()
    {
        int x = 0;
        Node<T> N = first;
        while (N != null)
        {
            x++;
            N = N.getNext();
        }
        return x;
    }

    public T getData(int p)
    {
        int x = size();
        Node<T> N = null;
        if ((p <= x) && (p >= 1))
        {
            N = first;
            for (int i = 1; i < p; i++)
            {
                N = N.getNext();
            }
        }
        return N;
    }

    public int search(String s)
    {
        int x = 0;
        Node<T> N = first;
        while (N != null)
        {
            x++;
            if (N.getData().compare(s))
            {
                return x;
            }
            N = N.getNext();
        }
        return 0;
    }

    public void displayAll()
    {
        Node<T> N = first;
        while (N != null)
        {
            N.getData().display();
            N = N.getNext();
        }
    }

    public Interface Interf
    {
        public boolean compare(String s);
        public void display();
    }
}
Exercise 3: Given the following UML Class Diagram

- **List<T >**
  - name: String
  + List<T> (s: String)
  + size(): int
  + getData(in p:int):T
  + search(in s: String): int
  + displayAll()

- **Shareeha**

- **Telecom**
  - name: String
  + Telecom (in n1: String, in n2: String)
  + fatoura()
  + nbShOveraccess():int
  + nbShOwner(in n:String):int
  + saveShNetOA(in f:String):int

- **TL**

- **List<T>** (s: String)

- **Shareeha**

- **Telecom** (in n1: String, in n2: String)

- **fatoura()**: displays all Shareeha Objects in the list TL.
- **nbShOveraccess()**: returns the number of ShNet having an over access.
- **nbShOwner(in n: String)**: returns the number of Shareeha for a given owner n.
- **saveShNetOA(in f: String)**: saves all ShNet having an over access on the file f and returns the total amount of over access.

**Question**: Implement the class Telecom.
Solution Exercise 3:

```java
public class Telecom {
    private String name;
    private List<Shareeha> TL;

    public Telecom (String n1, String n2) {
        name = n1;
        TL = new List(n2);
    }

    public void fatoura() {
        TL.displayAll();
    }

    public int nbShOveraccess() {
        int x = size();
        int y=0;
        for (int i = 1 ; i<=x; i++) {
            if(TL.getData(i) instanceof ShNet) {
                ShNet S = (ShNet)TL.getData(i);
                if (S.getOverAccees()!=0) y++;
            }
        }
        return y;
    }

    public int nbShOwner(String n) {
        return TL.search(n);
    }

    public int saveShNetOA(String f) throws IOException {
        File outF = new File(f);
        FileOutputStream outFS = new FileOutputStream(outF);
        ObjectOutputStream outObj = new ObjectOutputStream(outFS);
        int x = size();
        int y=0; Shareeha H=null;
        for (int i = 1 ; i<=x; i++) {
            if(TL.getData(i) instanceof ShNet) {
                ShNet S = (ShNet)TL.getData(i);
                if (S.getOverAccees()!=0) {
                    outObj.writeObject(S);
                    y+=S.computeOAccess();
                }
            }
        }
        outObj.close();
        return y;
    }
}
```
**Exercise 4:** Given the following UML Class Diagram

<table>
<thead>
<tr>
<th>Stack&lt;T&gt;</th>
<th>Telecom</th>
</tr>
</thead>
<tbody>
<tr>
<td>- name: String</td>
<td></td>
</tr>
<tr>
<td>+ Stack&lt;T&gt; (s: String)</td>
<td></td>
</tr>
<tr>
<td>+ pop():T</td>
<td></td>
</tr>
<tr>
<td>+ push(in s: T)</td>
<td></td>
</tr>
<tr>
<td>- isEmpty():boolean</td>
<td></td>
</tr>
<tr>
<td>&lt;Shareeha&gt;</td>
<td></td>
</tr>
</tbody>
</table>

The following is the code of the class Stack.

```java
public class Stack<T> extends List<T>{
    public Stack(String s) {
        super(s);
    }
    public void push( T obj ) {
        insertAtFront(obj);
    }
    public T pop() throws Exception{
        if (isEmpty())
            throw new Exception("Stack is empty");
        return removeFirst();
    }
}
```

Write a method zeroShPhone() that returns a stack containing all ShPhone with credit zero and the original ST should contain all shareehaa with credit different to zero. (You should use the concept of exception)
Solution of Exercise 4:

Public Stack<Shareeha> zeroShPhone()
{
    Stack<Shareeha> S1 = new Stack(“Zero”);
    Stack<Shareeha> S2 = new Stack(“This”);
    Shareeha H = null;
    try
    {
        while (true)
        {
            H = ST.pop();
            if ((H instanceof ShPhone) && (H.getCredit() == 0))
                S1.push(H);
            else
                S2.push(H);
        }
    }
    catch(Exception e) {} 

    try
    {
        while (true)
        {
            ST.push(S2.pop());
        }
    }
    catch(Exception e) {} 

    return S1;
}