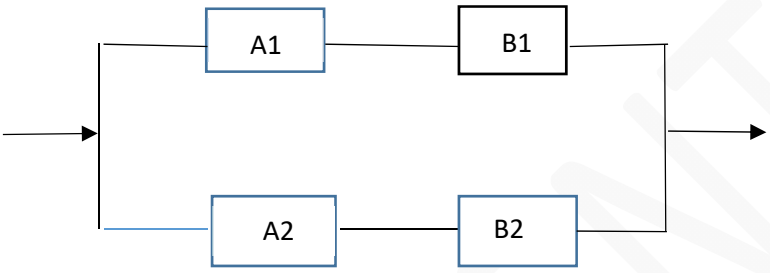
In Class Assignment: Please handwrite the answers on loose sheets and hand in the assignment.

Q1 (a) [0.5 mark] Propose a load test for the photogallery app or the app that you are developing for the course.
Ans: Incrementally increasing the size of the repository of the photos and evaluating its impact on the performance of the search.

(b) [0.5 mark] Assuming that each component, in the architecture depicted below, on the average can process 10 requests per second and the average arrival rate of the requests to the system is 3 requests per second, estimate the average latency or time it takes for a request to be handled by the system assuming that the system could be modeled as a network of queues.




Ans: 0.5 * [1/(10 – 1.5) + 1/(10 – 1.5)] + 0.5*[1/(10 – 1.5) + 1/(10 – 1.5)]
Where 0.5 is the probability that a request takes a particular path.

Q2. Illustrate the static composition, in the form of a UML class diagram [2 marks], and the dynamic behavior of the system, in the form of both the UML sequence diagram [2 marks] as well as the UML state machine/chart diagram [2 marks], for a simple Android application whose code is listed below. Marks will be based on how accurately and completely UML was utilized in capturing the correct composition and dynamic behavior of this software.

Note: For simplicity, assume that there is no other activity in the system and when this activity comes to the foreground it stays there forever. Also due to irrelevance, some of the xml layout attributes have been removed.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_main"
    tools:context="com.example.server.sampleapp.MainActivity">
    <TextView
        android:id="@+id/textView"
        android:text="Hello World!" />
</RelativeLayout>
```

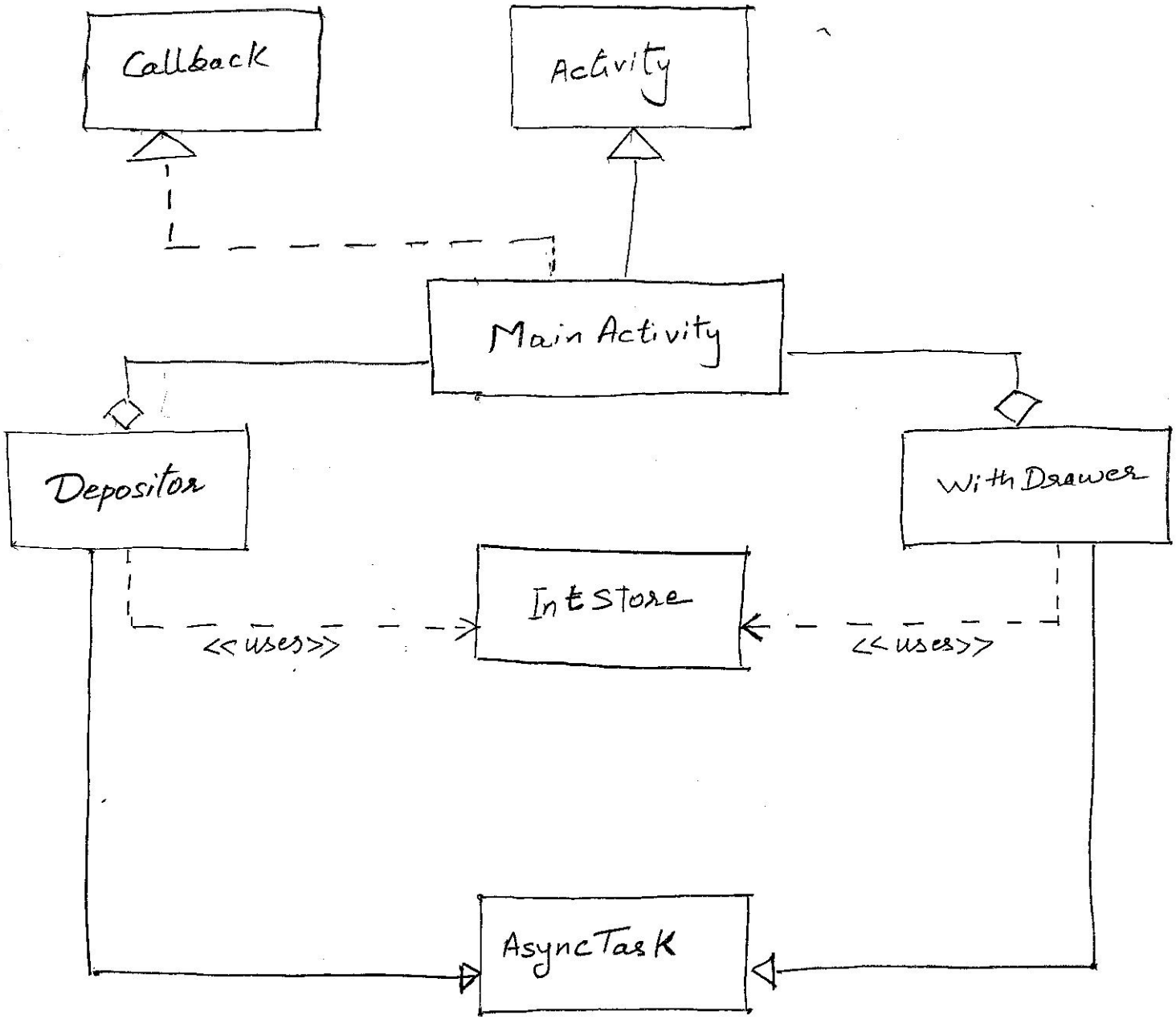
```
package com.example.server.sampleapp;
interface Callback {
```

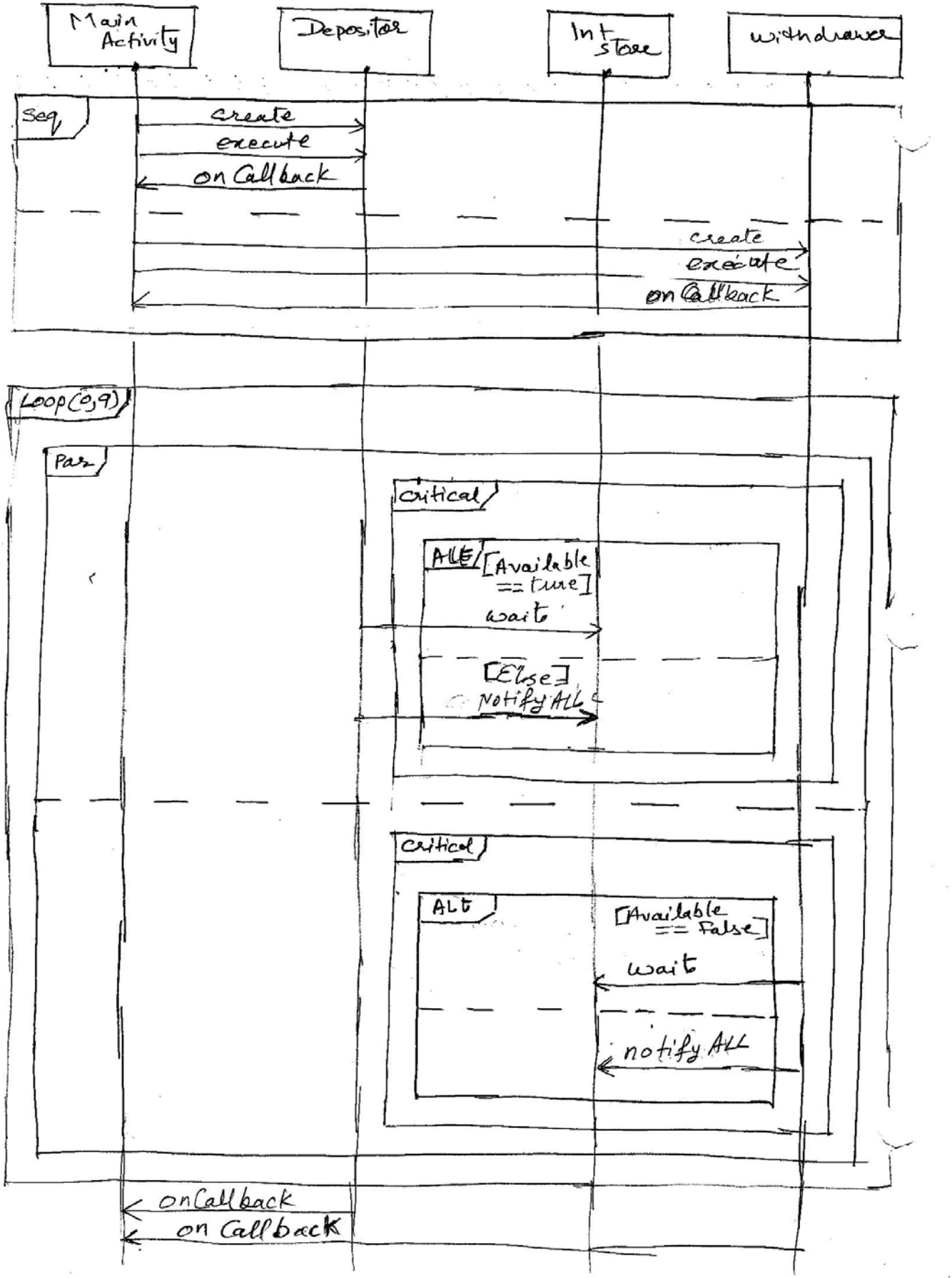
```
    void onCallback(String result);
}
```

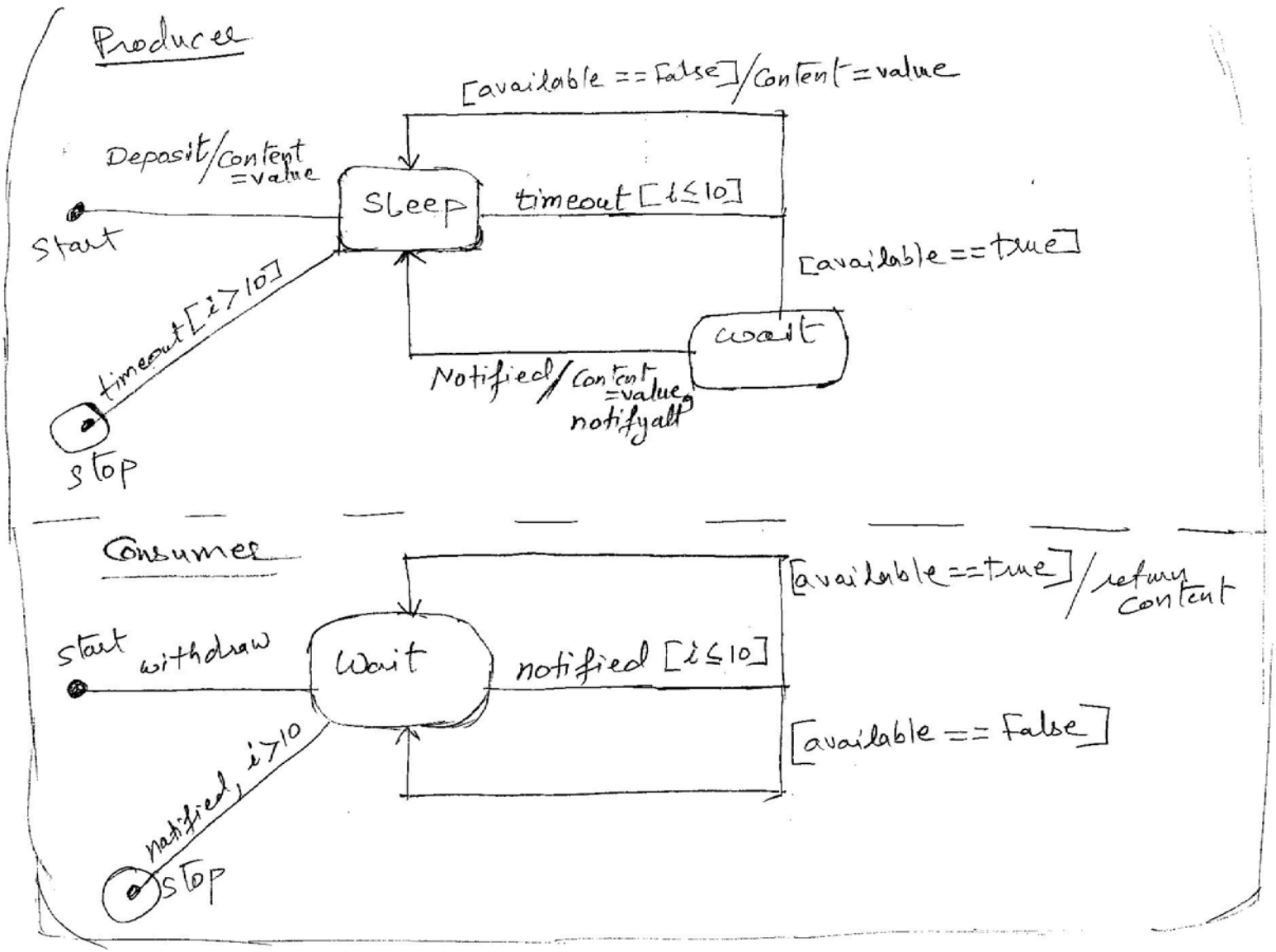
```
package com.example.server.sampleapp;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.TextView;
import android.os.AsyncTask;
import java.lang.Thread;
public class MainActivity extends AppCompatActivity implements Callback{
    TextView textView;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        textView = (TextView) findViewById(R.id.textView);
        IntStore intStore = new IntStore();
        Depositor depositor = new Depositor(this);
        depositor.executeOnExecutor(AsyncTask.THREAD_POOL_EXECUTOR, intStore);
        Withdrawer withdrawer = new Withdrawer(this);
        withdrawer.executeOnExecutor(AsyncTask.THREAD_POOL_EXECUTOR, intStore);
    }
    @Override
    public void onCallback(String result) {
        textView.setText(result);
    }
}

package com.example.server.sampleapp;
public class IntStore {
    private int contents;
    private boolean available = false;
    public synchronized int withdraw() {
        while (available == false) {
            try {
                wait();
            } catch (InterruptedException e) { }
        }
        available = false;
        notifyAll();
        return contents;
    }
    public synchronized void deposit(int value) {
        while (available == true) {
            try {
```

```
                wait();
            } catch (InterruptedException e) { }
        }
        contents = value;
        available = true;
        notifyAll();
    }
}

package com.example.server.sampleapp;
import android.os.AsyncTask;
import android.util.Log;
public class Withdrawer extends AsyncTask<IntStore, Integer, String> {
    Callback callback;
    public Withdrawer(Callback callback){
        this.callback = callback;
    }
    @Override
    protected void onPreExecute() {
        callback.onCallback("Withdrawer Start");
    }
    @Override
    protected String doInBackground(IntStore... intStores) {
        int value = 0;
        for (int i = 0; i < 10; i++) {
            value = intStores[0].withdraw();
            publishProgress(value);
        }
        return "End";
    }
    @Override
    protected void onPostExecute(String message) {
        callback.onCallback("Withdrawer " + message);
    }
    @Override
    protected void onProgressUpdate(Integer... progress) {
        Log.i("Withdrawn: ", String.valueOf(progress[0]));
    }
}
```

```
package com.example.server.sampleapp;
import android.os.AsyncTask;
import android.util.Log;
class Depositor extends AsyncTask<IntStore, Integer, String> {
    Callback callback;
    public Depositor(Callback callback){
        this.callback = callback;
    }
    @Override
    protected void onPreExecute() {
        callback.onCallback("Depositor Start");
    }
    @Override
    protected String doInBackground(IntStore... intStores) {
        for (int i = 0; i < 10; i++) {
            intStores[0].deposit(i);
            publishProgress(i);
            try {
                Thread.sleep((int) (Math.random() * 100));
            } catch (InterruptedException e) {
            }
        }
        return "End";
    }
    @Override
    protected void onPostExecute(String message) {
        callback.onCallback("Depositor " + message);
    }
    @Override
    protected void onProgressUpdate(Integer... progress) {
        Log.i("Depositer", String.valueOf(progress[0]));
    }
}
```




Main Activity
Depositor
Int Store
withdrawer
Seq
create
execute
on Callback
create
execute
on Callback
Loop(0,9)
Par
critical
ALT
[Available == true]
wait
[Else]
notify ALL
critical
ALT
[Available == false]
wait
notify ALL
onCallback
on Callback


Producer
[available == False]/Content=value
Deposit/content =value
Sleep
timeout [i≤10]
start
[available == true]
timeout [i>10]
wait
Notified/Content =value, notifyall
stop
Consumer
[available == true]/return content
start
withdraw
Wait
notified [i≤10]
[available == False]
notified, i>10
stop