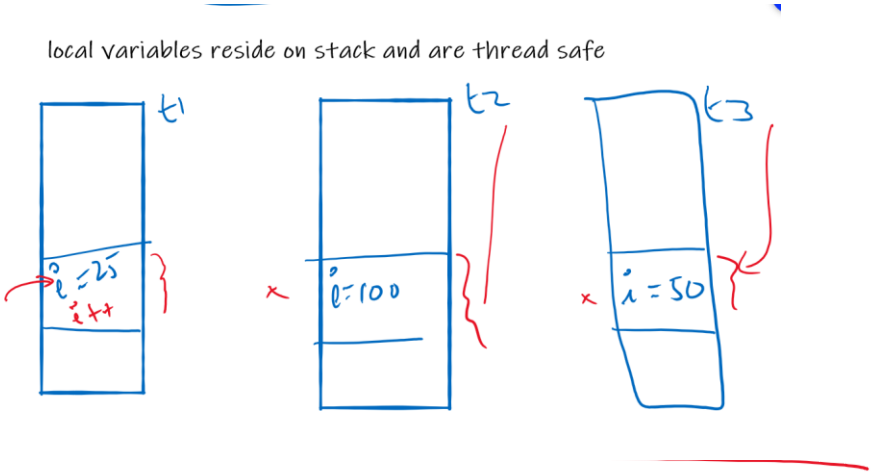
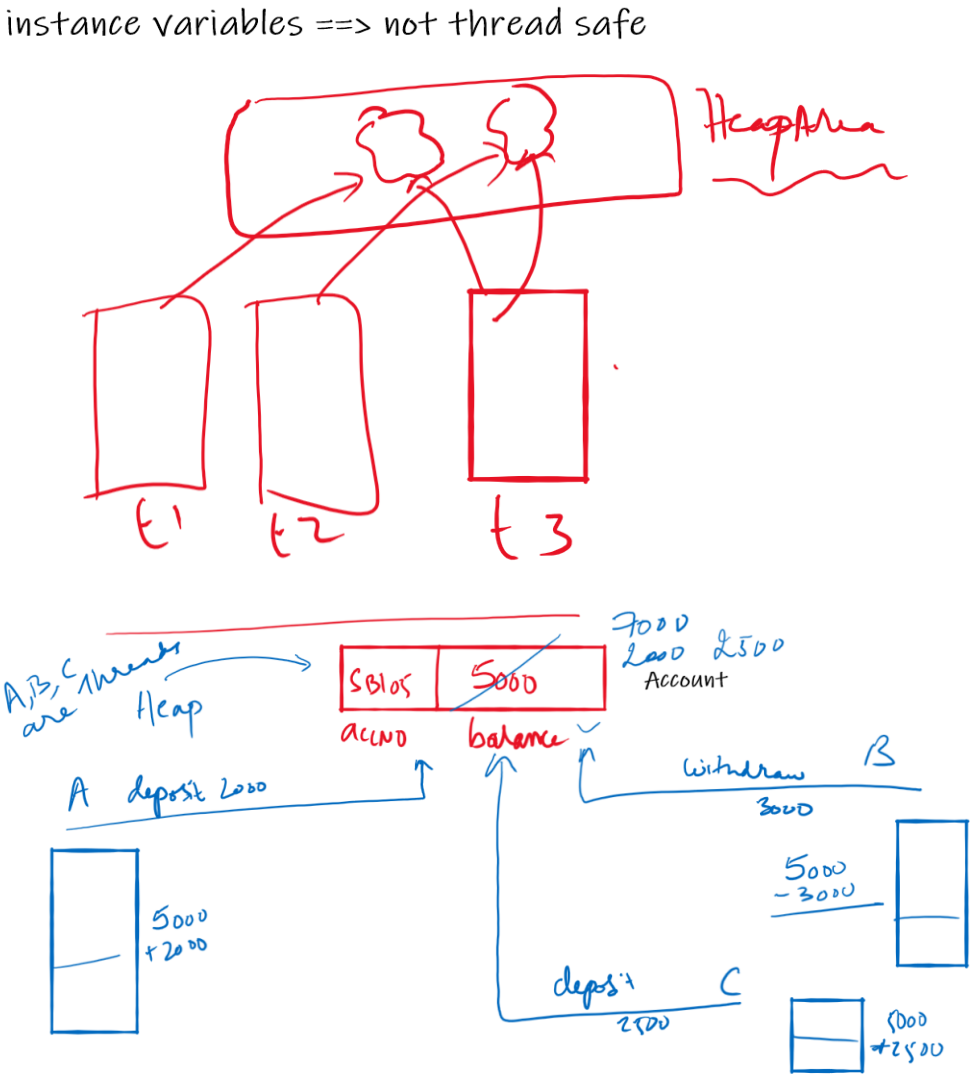
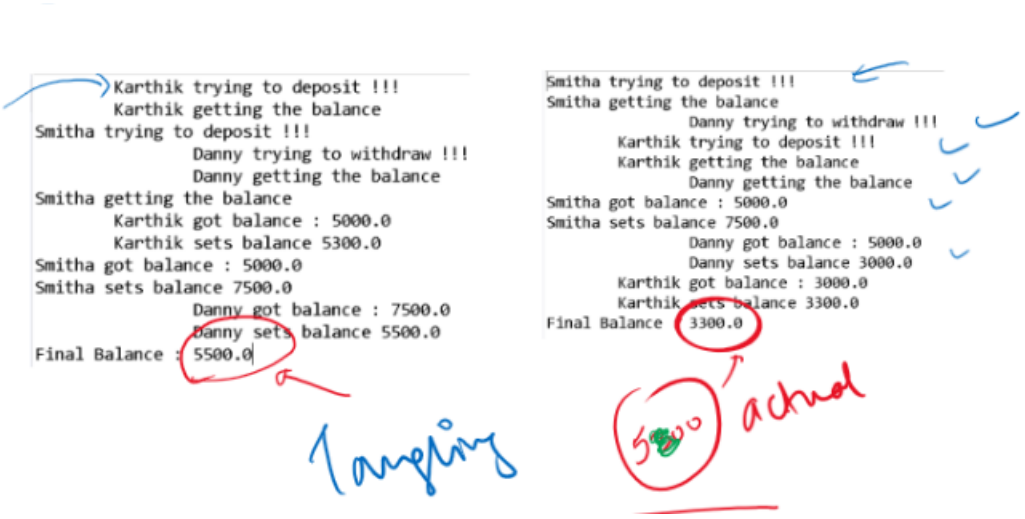
Thread Safe members:

Local variables are thread-safe, instance variables are not thread-safe:

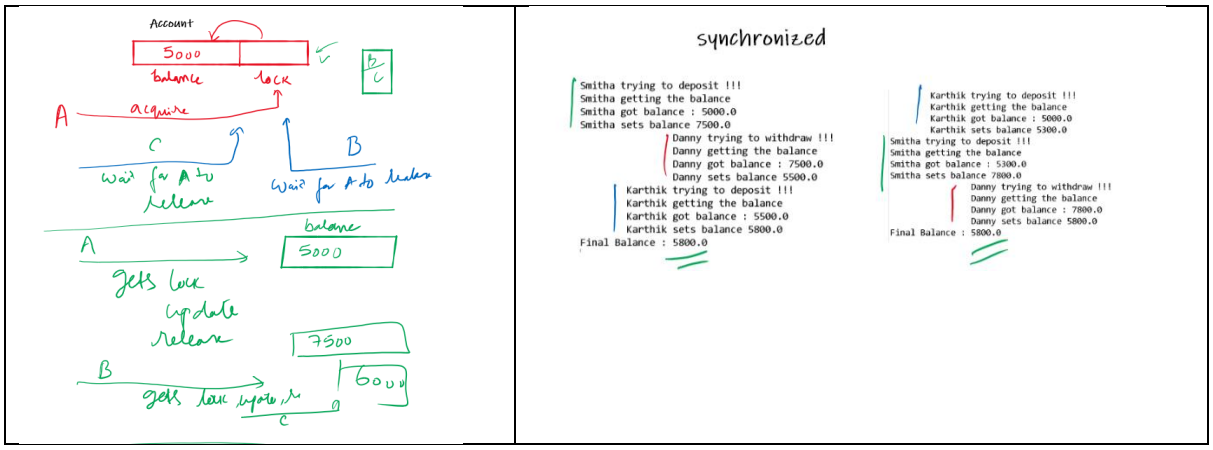




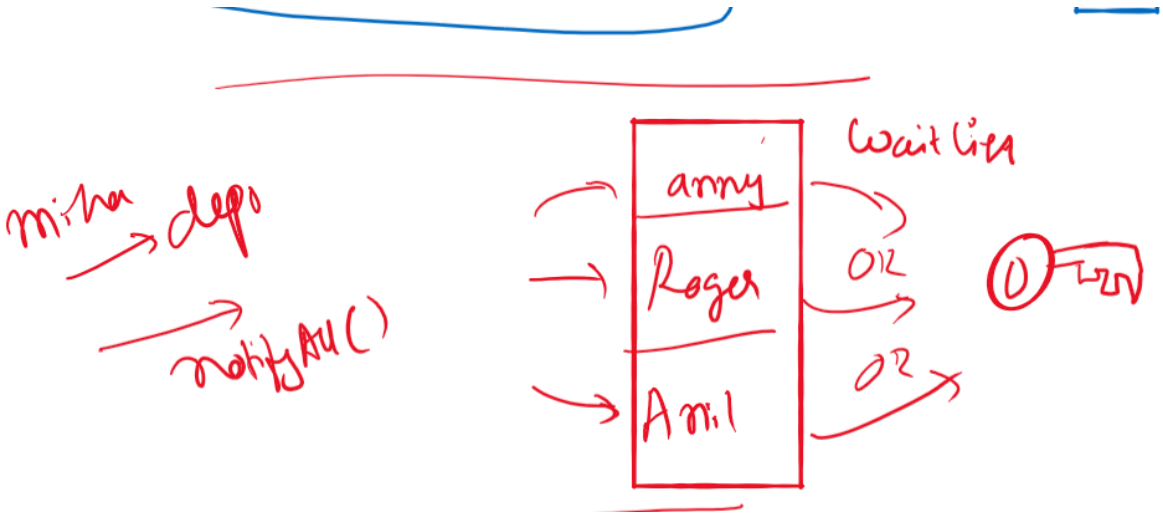
Data corruption in multi-threaded environment:



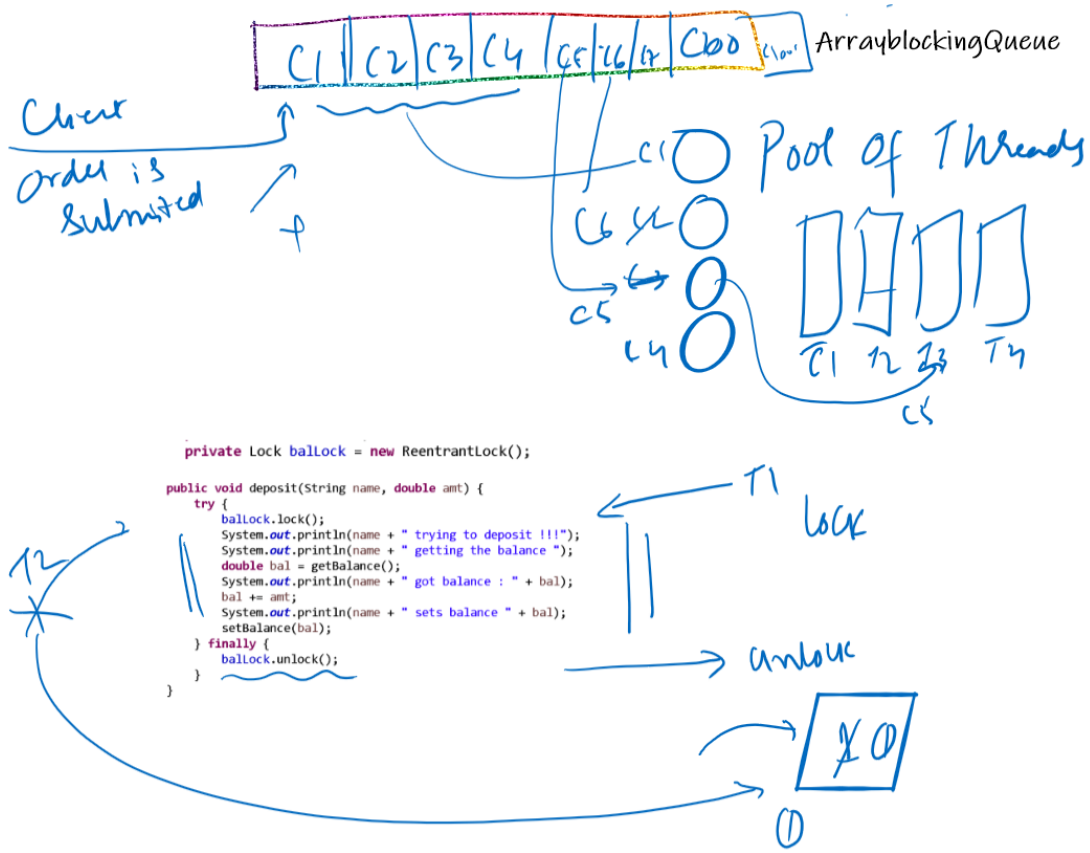
Synchronized and Monitor/Mutex/Lock:



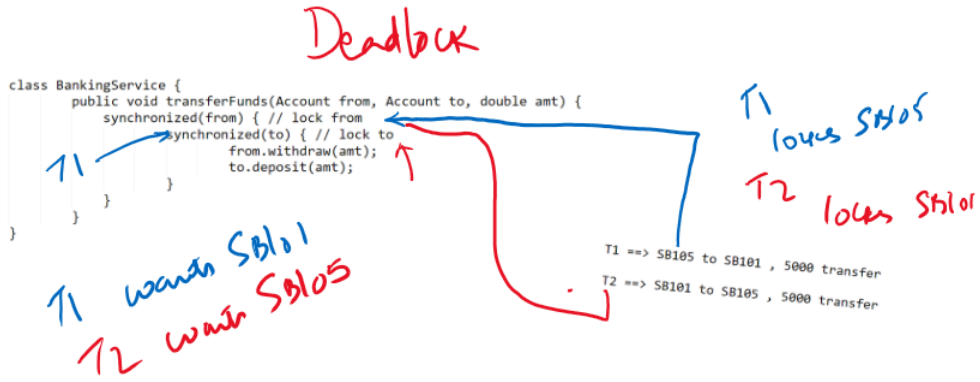
Wait and notify:



Lock API and ThreadPool:



DeadlocK



Lock API-> avoid deadlock

```
class BankingService {
    public void transferFunds(Account from, Account to, double amt) {
        if(from.balLock.tryLock(100)) {
            try {
                if(to.balLock.tryLock(100)) {
                    try {
                            from.withdraw(amt);
                            to.deposit(amt);
                        }
                    } finally {
                            to.balLock.unlock();
                    }
                }
            } finally {
                from.balLock.unlock();
            }
        }
    }
```

SB105

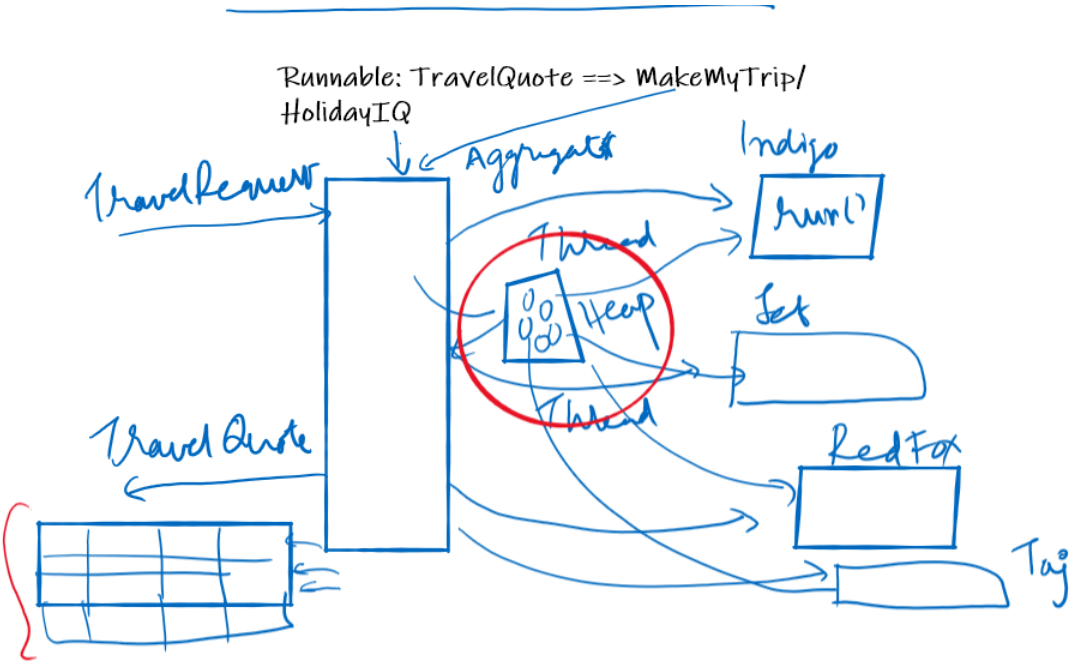
SB101

T1 ==> SB105 to SB101 , 5000 transfer

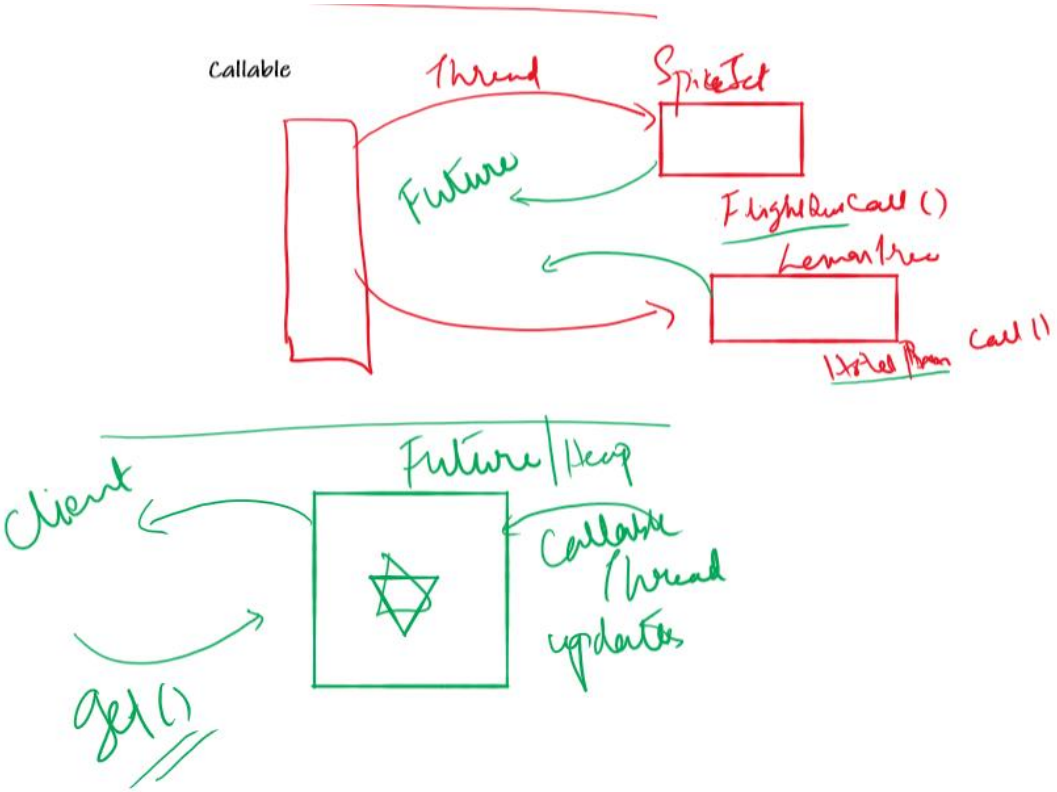
T2 ==> SB101 to SB105 , 2000 transfer

T1:
if t1 gets SB105 lock
try to get SB101

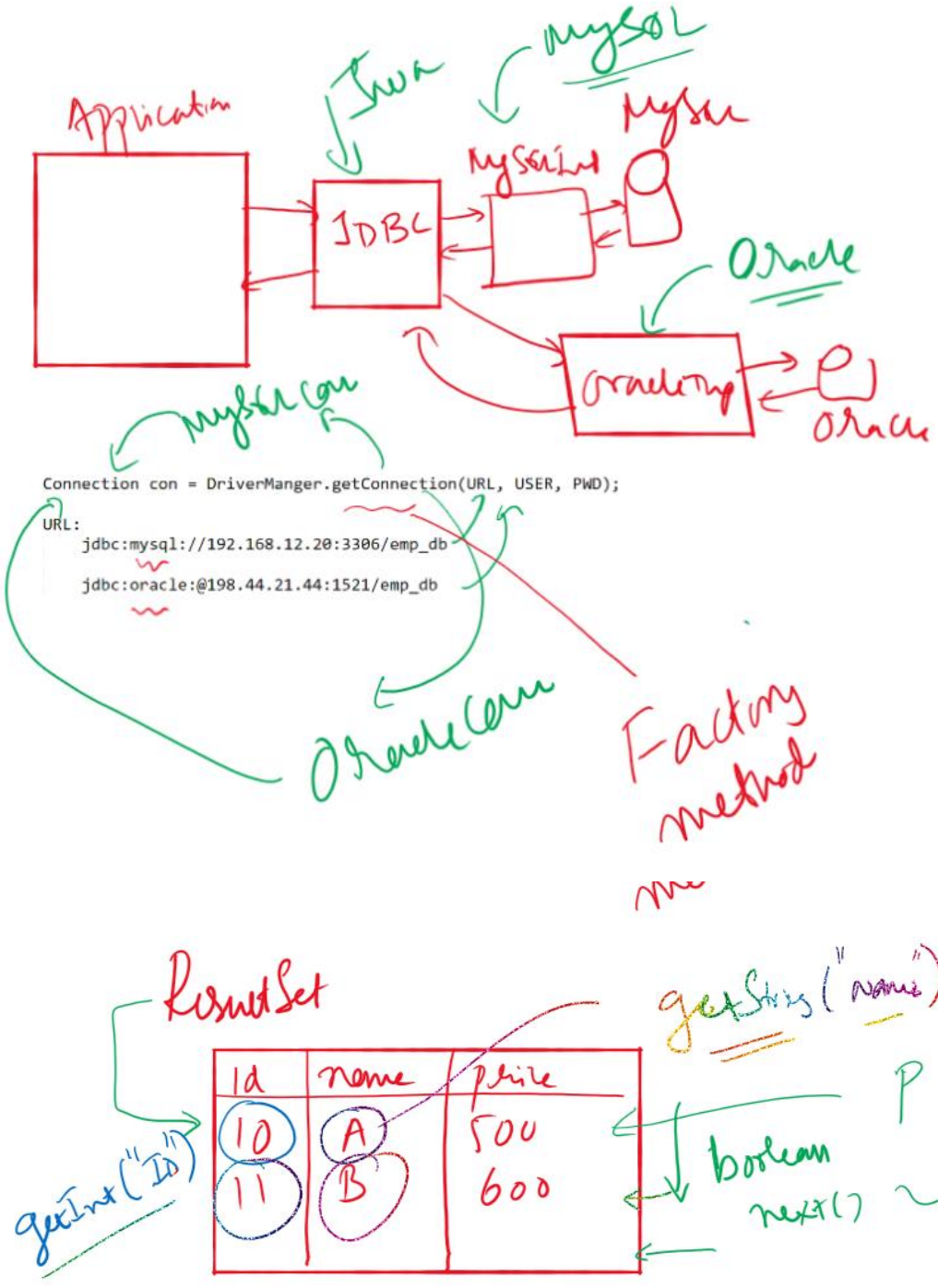
Without Callable and Future:



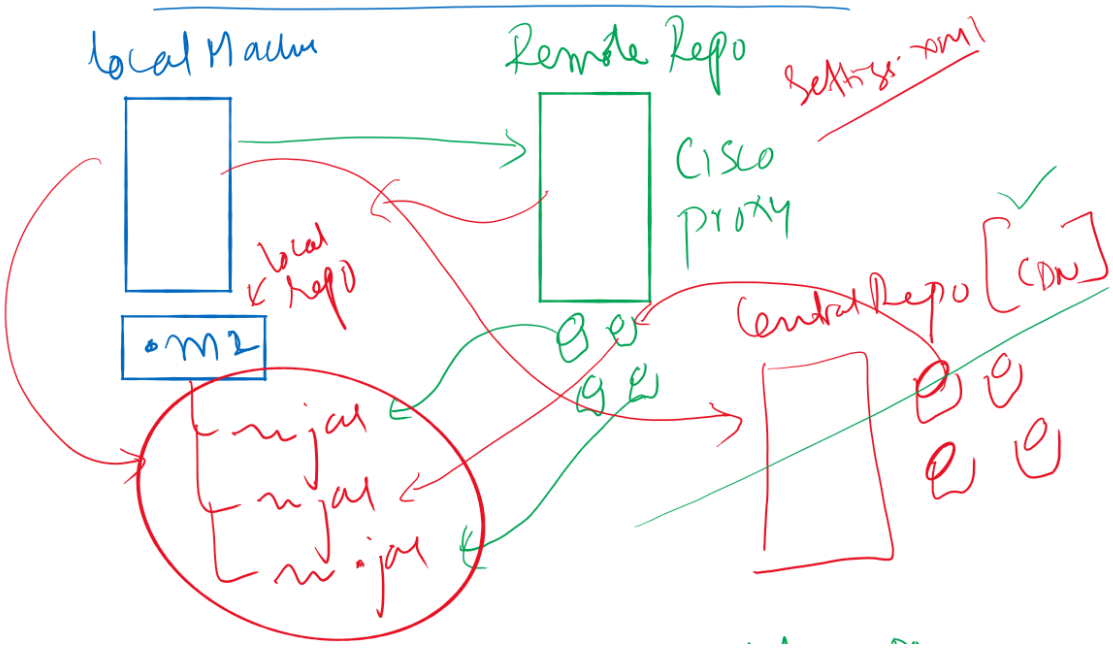
Issues with Runnable:
-> Shared heap area has to have locking mechanism
-> flags has to be used to signal Aggregator that job is done
-> maybe Jet server is down


Callable
Thread
SpiceJet
Future
FlightQuickCall ()
LemonTree
Hotel Room
Call ()
Future | Heap
client
Callable
Thread
updates
get ()

JDBC:


Application
Java
mysql
JDBC
Oracle
Connection con = DriverManger.getConnection(URL, USER, PWD);
URL:
jdbc:mysql://192.168.12.20:3306/emp_db
jdbc:oracle:@198.44.21.44:1521/emp_db
Factory method
ResultSet
id
name
price
10
11
A
B
500
600
getInt("ID")
getString("NAME")
boolean
next()

Maven:


local Machine
Remote Repo
Settings.xml
Cisco
proxy
local
Repo
.m2
Central Repo
CDN
.jar
.jar
.jar