```
package transactions;

import database.DbItem;

import java.time.LocalDateTime;

public class NoDeadlockTransaction extends Transaction {
    private final long transactionId;
    public NoDeadlockTransaction(DbItem mainDbItem, DbItem secondayDbItem){
        this.mainDbItem = mainDbItem;
        this.secondaryDbItem = secondayDbItem;
        transactionId = ++Transaction.id;
    }
    public void run() {
        try {
            System.out.println("\t -> Transaction " + this.transactionId+ " initialized (" + LocalDateTime.now() +
")");
            System.out.println("\t\t > [T" + this.transactionId + "] checking items availability...");

            if (!mainDbItem.isLocked() && !secondaryDbItem.isLocked()) {
                System.out.println("\t\t > [T" + this.transactionId + "] " + this.mainDbItem.getPosition() + " is
available.");
                System.out.println("\t\t > [T" + this.transactionId + "] " + this.secondaryDbItem.getPosition() + "
is available.");


                System.out.println("\t\t > [T" + this.transactionId + "] locking: " + this.mainDbItem.getPosition());
                this.mainDbItem.write_lock(this);

                System.out.println("\t\t > [T" + this.transactionId + "] locking: " +
this.secondaryDbItem.getPosition());
                this.secondaryDbItem.write_lock(this);

                System.out.println("\t\t > [T" + this.transactionId + "] trying acess item: " +
this.mainDbItem.getPosition());
                synchronized (mainDbItem) {
                    System.out.println("\t\t > [T" + this.transactionId+ "] reading content on: " +
this.mainDbItem.getPosition());
                    System.out.println("\t\t > [T" + this.transactionId + "] current content: " +
this.mainDbItem.read(this) );
                    System.out.println("\t\t > [T" +this.transactionId + "] processing");

                    Thread.sleep(500);
                    System.out.println("\t\t > [T" + this.transactionId + "] writting on: " +
this.mainDbItem.getPosition());
                    this.mainDbItem.write(this, "outroAluno A");
                    System.out.println("\t\t > [T" + this.transactionId + "] new content: " +
this.mainDbItem.read(this) );

                    System.out.println("\t\t > [T" + this.transactionId + "] trying acess item: " +
this.secondaryDbItem.getPosition());

                    synchronized (secondaryDbItem) {
                        System.out.println("\t\t > [T" + this.transactionId + "] reading content on: " +
this.secondaryDbItem.getPosition());
```

```
                    System.out.println("\t\t > [T" + this.transactionId + "] current content: " +
this.secondaryDbItem.read(this) );
                    System.out.println("\t\t > [T" + this.transactionId + "] processing");

                    Thread.sleep(500);
                    System.out.println("\t\t > [T" + this.transactionId + "] writting on: " +
this.secondaryDbItem.getPosition());
                    this.secondaryDbItem.write(this, "outroAluno B");
                    System.out.println("\t\t > [T" + this.transactionId + "] new content: " +
this.secondaryDbItem.read(this) );
                }

                System.out.println("\t\t > [T" + this.transactionId+ "] unlocking: " +
this.mainDbItem.getPosition());
                this.mainDbItem.unLock(this);

                System.out.println("\t\t > [T" + this.transactionId + "] unlocking: " +
this.secondaryDbItem.getPosition());
                this.secondaryDbItem.unLock(this);
            }
            System.out.println("\t <- Transaction " + this.transactionId + " finished (" + LocalDateTime.now()
+ ")");
        }else{
            System.out.println("\t <- [T" + this.transactionId + "] Ops! Locked item. Waiting for try execute
again(" + LocalDateTime.now() + ")");
            Thread.sleep(2000);
            run();
        }
    } catch (Exception ex) {
        ex.printStackTrace();
    }
  }

  @Override
  public long getId() {
    return this.transactionId;
  }
}
```