```
package transactions;

import database.DbItem;

import java.time.LocalDateTime;
public class DeadlockTransaction extends Transaction {
    private final long transactionId;
    private Transaction transactionJoin;
    public DeadlockTransaction(DbItem mainDbItem, DbItem secondayDbItem){
        this.mainDbItem = mainDbItem;
        this.secondaryDbItem = secondayDbItem;
        this.transactionId = ++Transaction.id;
    }
    public DeadlockTransaction(DbItem mainDbItem, DbItem secondayDbItem, Transaction transactionJoin){
        this.mainDbItem = mainDbItem;
        this.secondaryDbItem = secondayDbItem;
        this.transactionId = ++Transaction.id;
        this.transactionJoin = transactionJoin;
    }
    public void run() {
        try {
            if( this.transactionJoin != null ) {
                this.transactionJoin.join();
            }

            System.out.println("\t -> Transaction " + this.transactionId + " initialized (" + LocalDateTime.now() + ")");
            System.out.println("\t\t > [T" + this.transactionId + "] trying acess item: " + this.mainDbItem.getPosition());

            while( !this.mainDbItem.read_lock(this) ){
                System.out.println("\t <- [T" + this.transactionId + "] Ops! already locked item. Waiting for trying again(" +
LocalDateTime.now() + ")");
                Thread.sleep(2000);
            }

            System.out.println("\t\t > [T" + this.transactionId + "] " + this.mainDbItem.getPosition() + " locked ");
            synchronized (this.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());

                while( !this.secondaryDbItem.read_lock(this) ){
                    System.out.println("\t <- [T" + this.transactionId + "] Ops! already locked item. Waiting for trying again(" +
LocalDateTime.now() + ")");
                    Thread.sleep(2000);
                }

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

            System.out.println("\t\t > [T" + this.transactionId + "] unlocking item: " + this.secondaryDbItem.getPosition() );
            this.secondaryDbItem.unLock(this);
        }
        System.out.println("\t <- Transaction " + this.transactionId + " finished (" + LocalDateTime.now() + ")");
    } catch (Exception ex) {
        ex.printStackTrace();
    }
  }

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