CENTRE NATIONAL
DE LA RECHERCHE
SCIENTIFIQUE

CNT in JOTM

DUSART Doriane

June 10th 2002

Université de Valenciennes
LAMIH/ROI/SID
Le Mont Houy, 59313 Valenciennes Cedex 9
ddusart@meletu.univ-valenciennes.fr

LAMIH
LABORATOIRE
D'AUTOMATIQUE
ET DE MECANIQUE
INDUSTRIELLES
ET HUMAINES

Table of contents

- What exists
- CNT
- CNT in local and distributed

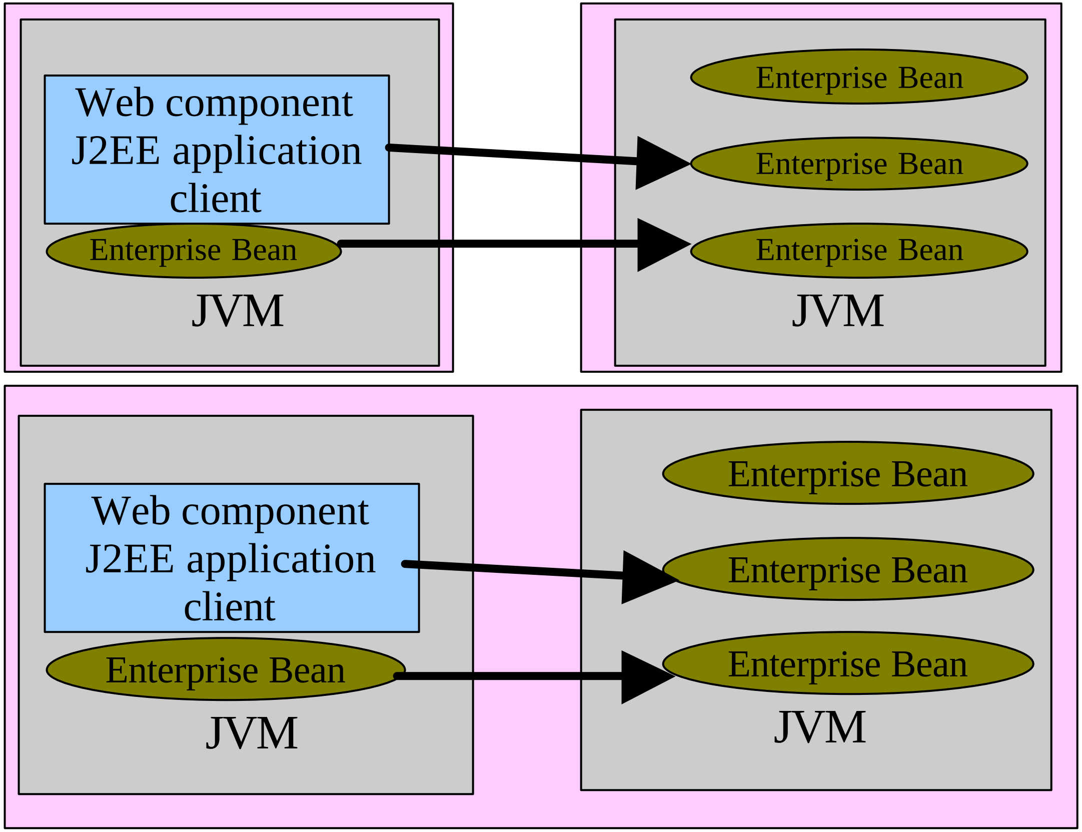
What exists

- – Local transaction / Distributed transaction
- – JOTM architecture
- – JTA
- – Local transaction
- – JTS
- – Distributed transaction

Distributed transaction in J2EE

Source : http://java.sun.com/j2ee/tutorial/1_3-cs/doc/EJBConcepts6.html#62995

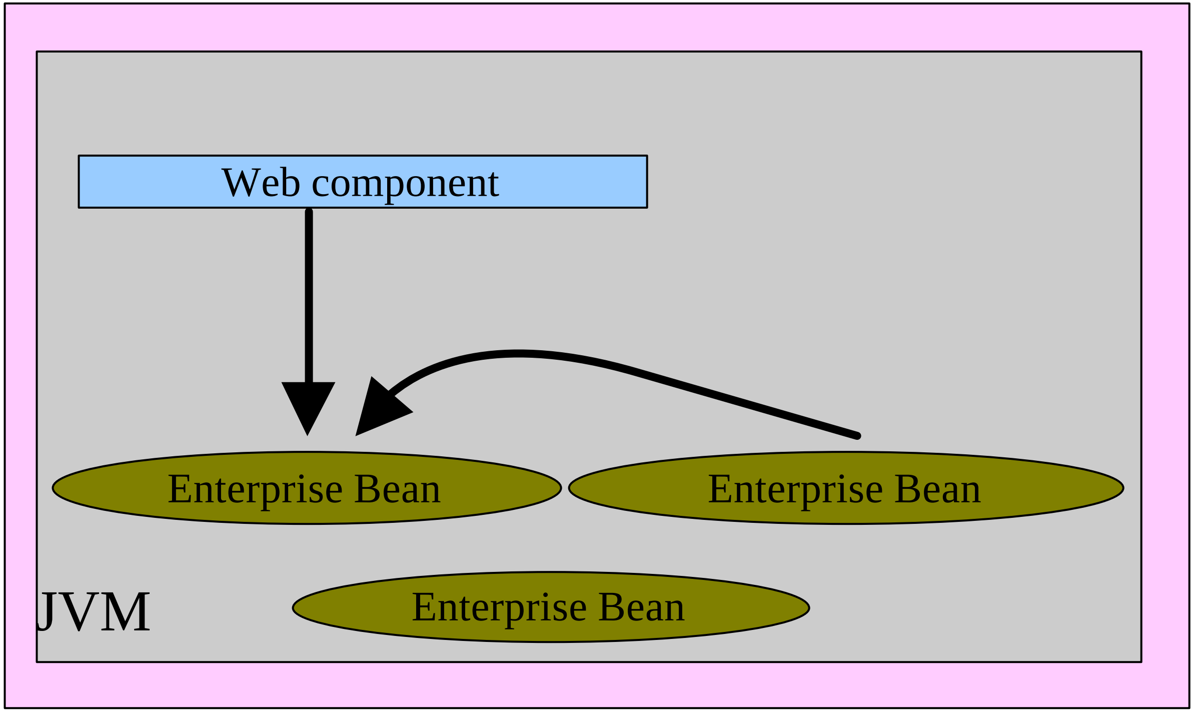
- ?A **remote clien**t of an enterprise bean has the following traits:



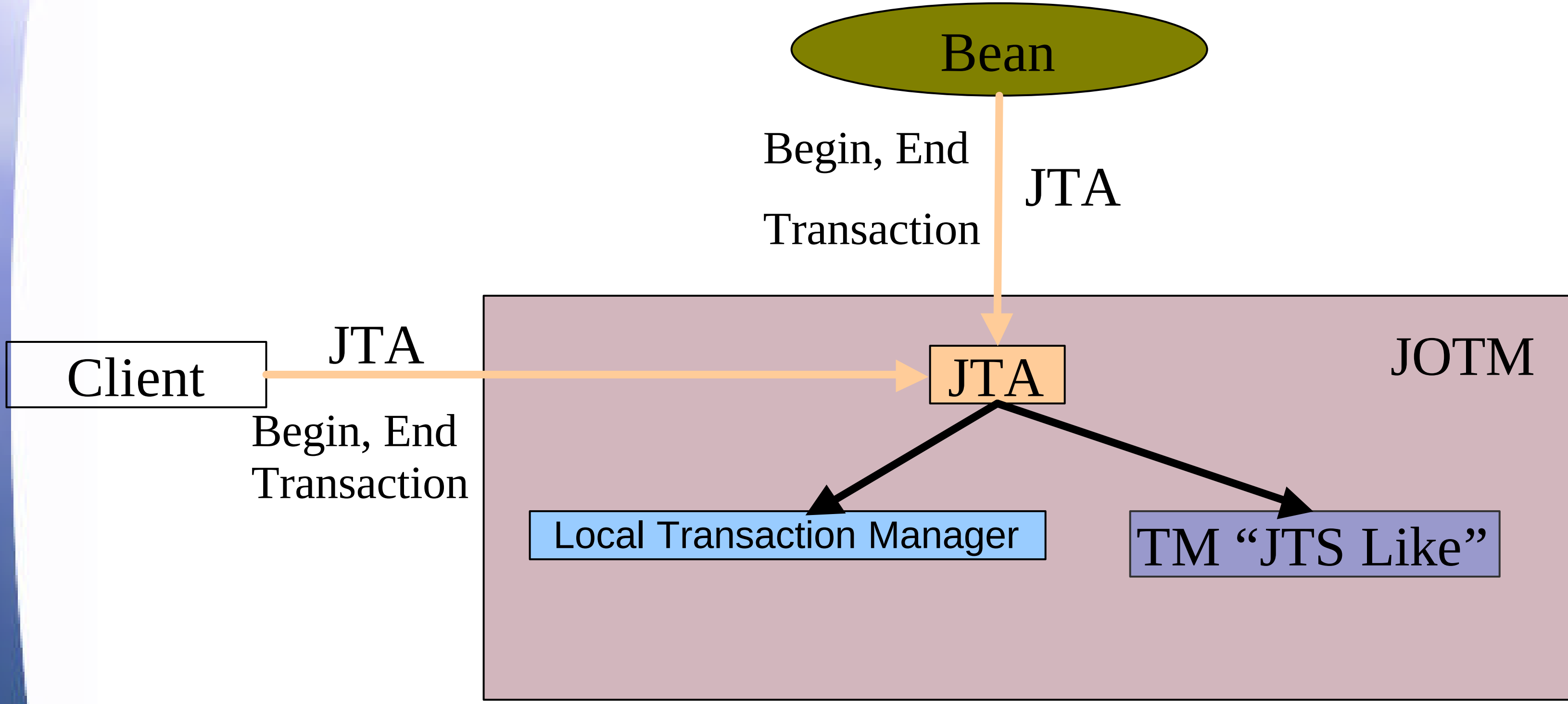
Local transaction in J2EE

Source : http://java.sun.com/j2ee/tutorial/1_3-cs/doc/EJBConcepts6.html#62995

?A **local client** has these characteristics:

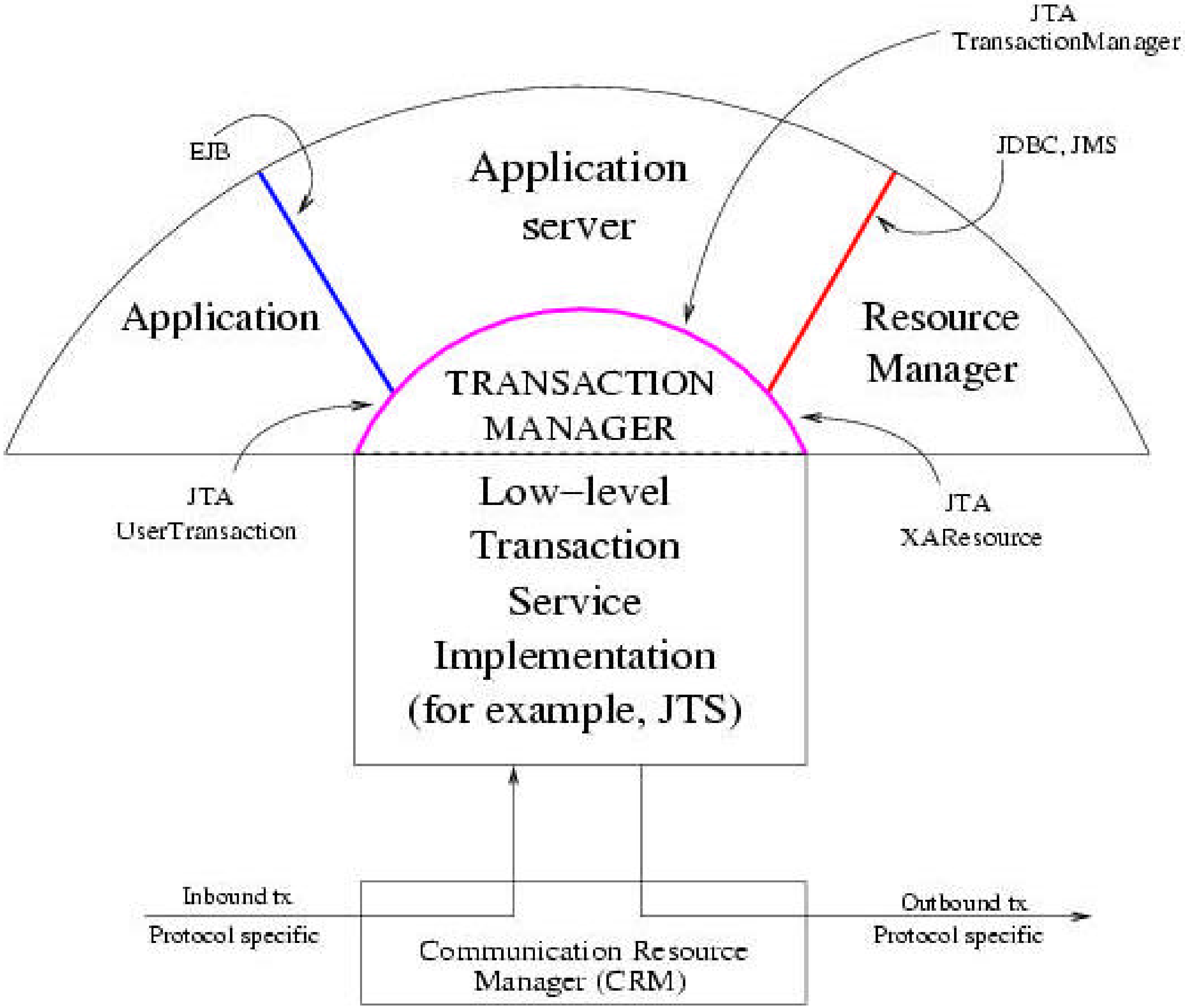


JOTM

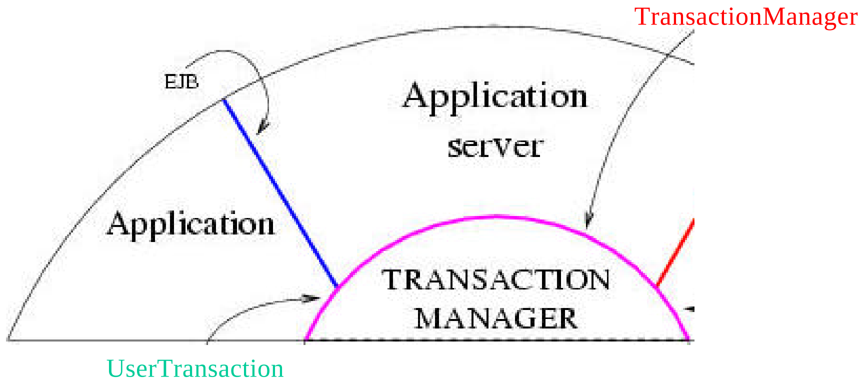

Bean
Begin, End
Transaction
JTA
Client
JTA
Begin, End
Transaction
JTA
JOTM
Local Transaction Manager
TM "JTS Like"

JTA Operation

JTA specification : ftp://ftp.java.sun.com/pub/jta/tui00/jta-spec1_0_1.pdf



JTA interfaces (1 /2)



- javax.transaction :

✓**Status** : status of the transaction.

✓**Synchronization** : synchronization allows transaction manager to notifie application server, before the begin of transaction, and after this end.

✓**Transaction** : This interface allows to associate a transaction with a transactionnal object when a transaction is created.

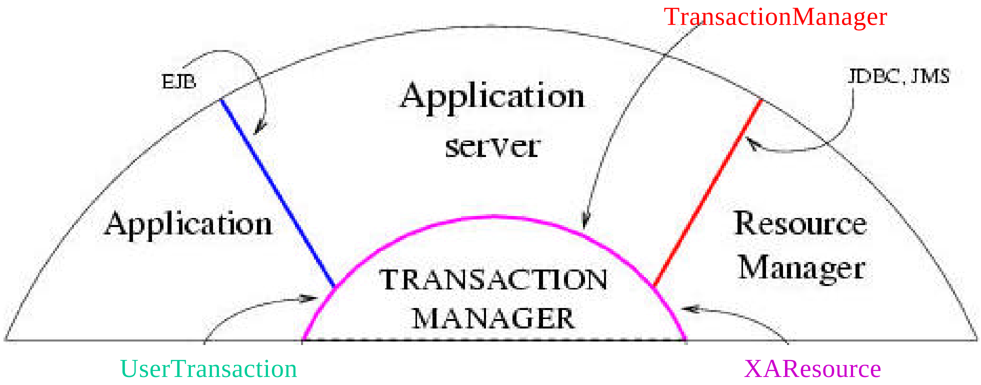
✓**TransactionManager** : This interface allows to manage states of a transaction.

✓**UserTransaction** : This interface can be used by a custom program or an EJB bean.

JTA interfaces (2 /2)

javax.transaction.xa :

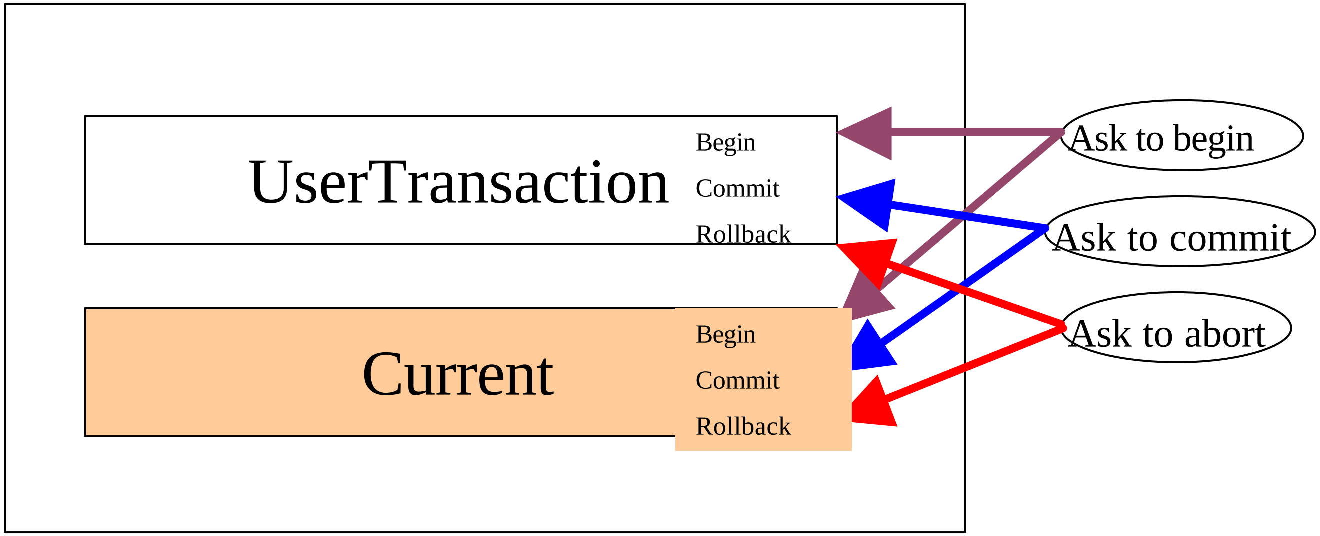
- ✓**XID** : It's a mapping of the XID identification structure od a transaction according to X/OPEN. It's uses by transaction and resourc managers.
- ✓**XAResource** : This interface give the contract between the transactio manager and resources manager in case of distributed transactions.



JTA / JOTM

Interfaces JTA	Implementation of JTA in JOTM
TransactionManager	Current
UserTransaction	Current
Transaction	TransactionImpl
Xid	XidImpl

Local transaction (1 /2)



UserTransaction

Begin

Commit

Rollback

Current

Begin

Commit

Rollback

Ask to begin

Ask to commit

Ask to abort

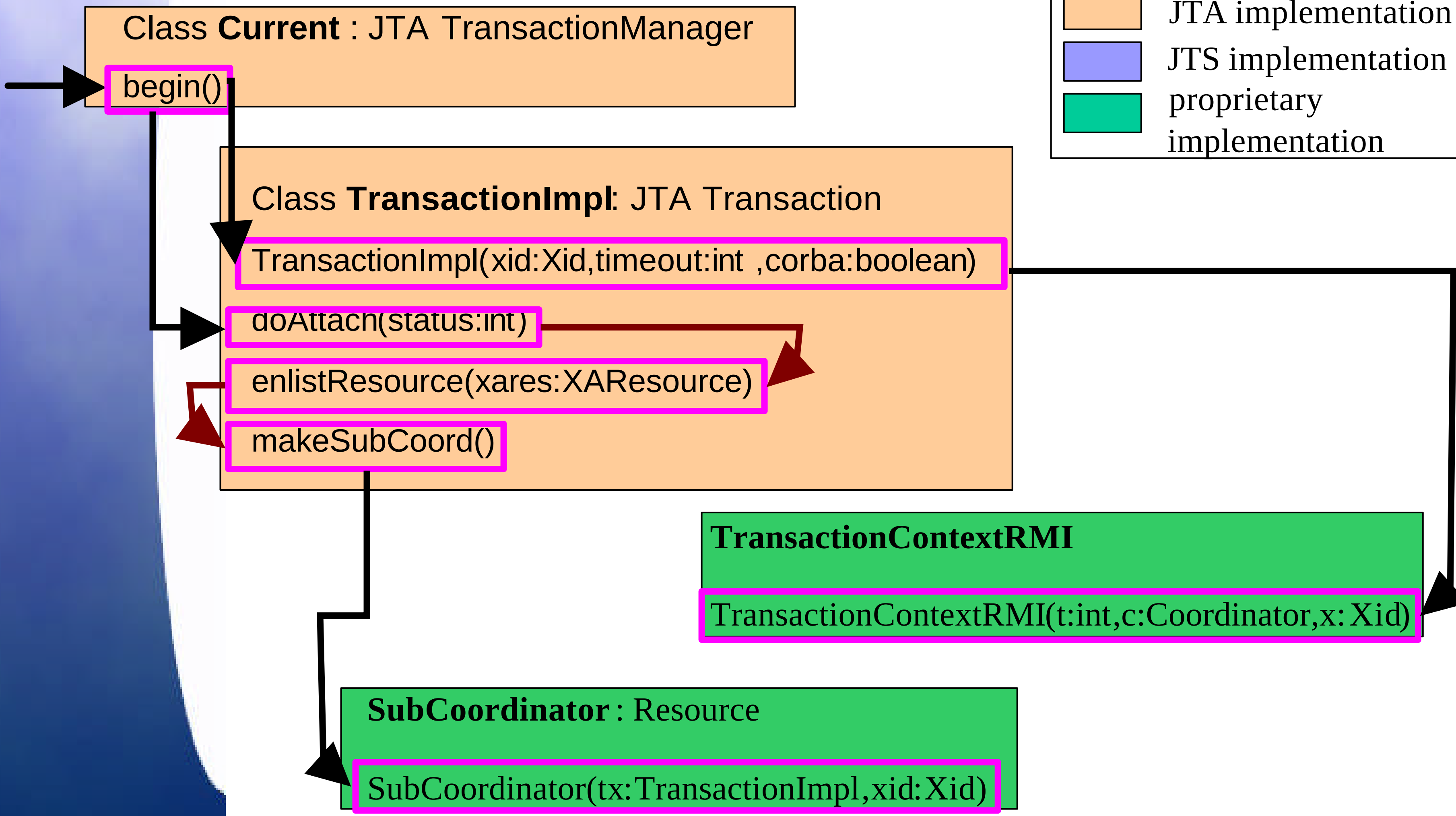
Local transaction (2 /2)

LEGENDE :

🟧 JTA implementation

🟦 JTS implementation

🟩 proprietary implementation



Class **Current** : JTA TransactionManager

begin()

Class **TransactionImpl**: JTA Transaction

TransactionImpl(xid:Xid,timeout:int ,corba:boolean)

doAttach(status:int)

enlistResource(xares:XAResource)

makeSubCoord()

TransactionContextRMI

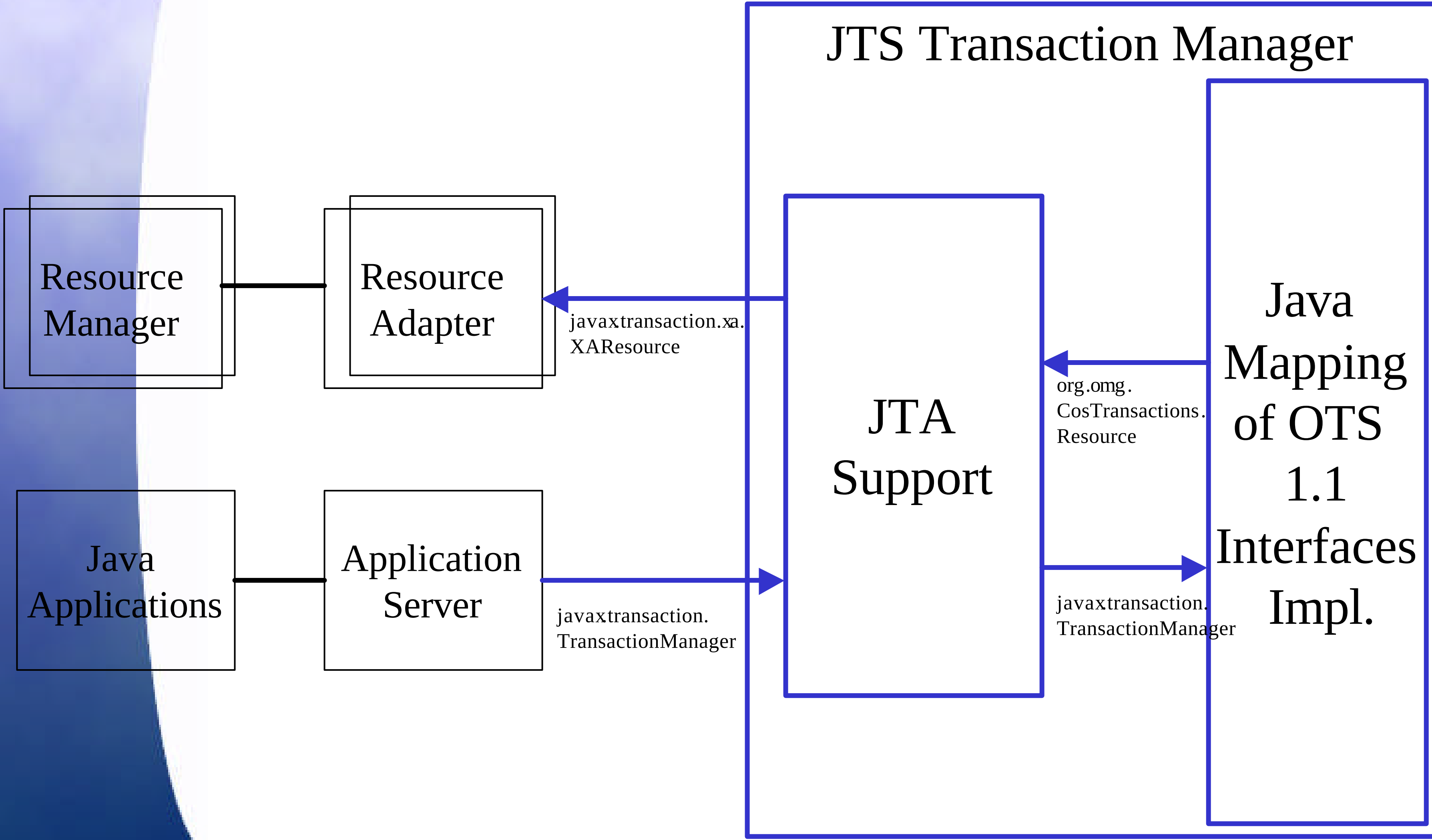
TransactionContextRMI(t:int,c:Coordinator,x:Xid)

SubCoordinator : Resource

SubCoordinator(tx:TransactionImpl,xid:Xid)

JTS API

JTS Specification : ftp://ftp.java.sun.com/pub/jts/2309856/jts1_0-spec.pdf



JTS Transaction Manager

Resource Manager

Resource Adapter

javaxtransaction.xa.
XAResource

Java Applications

Application Server

javaxtransaction.
TransactionManager

JTA Support

org.omg.
CosTransactions.
Resource

javaxtransaction.
TransactionManager

Java Mapping of OTS 1.1 Interfaces Impl.

JTS interface

- Mapping java of OTS Specification version 1.1
- Low level API
- A transaction manager (JTS compliant) gives the following services :

  - Possibility of controlling the range and the duration of a transaction.
  - Several objects allowing to do a work like part of a simple transaction.
  - Gives the possibility of associating a total transaction with a work carriec out transactionnal resources.
  - Coordinates the achievement of a total transaction through several resource manager.
  - Supports the synchronization of the transactions.
  - Gives the possibility to interoperate with other transaction managers using standard CORBA ORB/TS.

JTS / JOTM

Interface JTS	Implementation of JTSin JOTM
Control	ControlImpl
Coordinator	ControlImpl
RecoveryCoordinator	ControlImpl
Terminator	ControlImpl

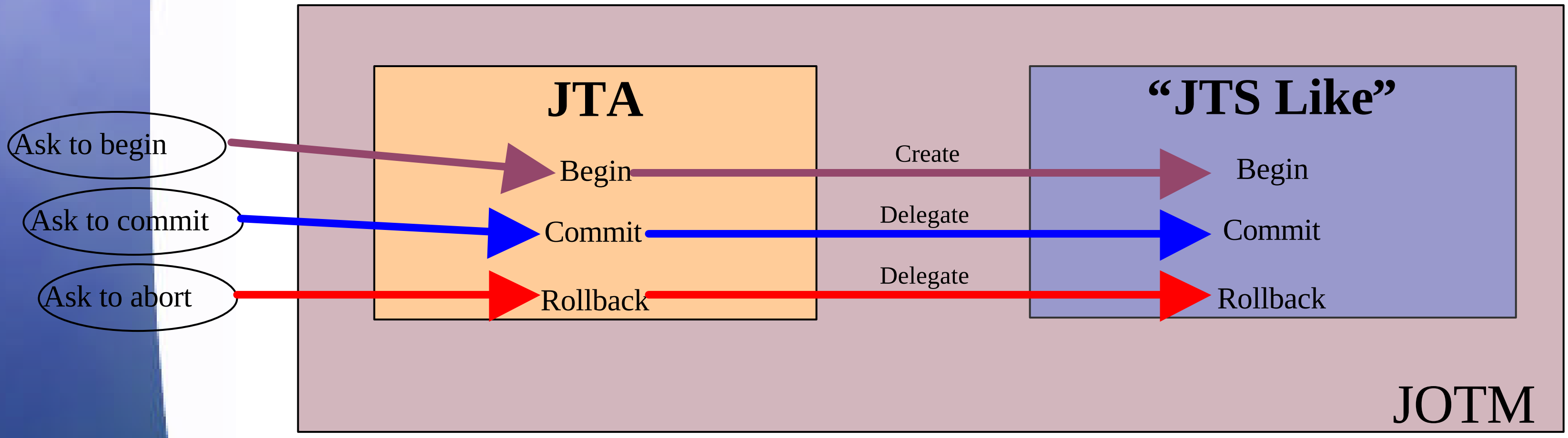
Distributed transaction (1 /2)

LEGENDE :

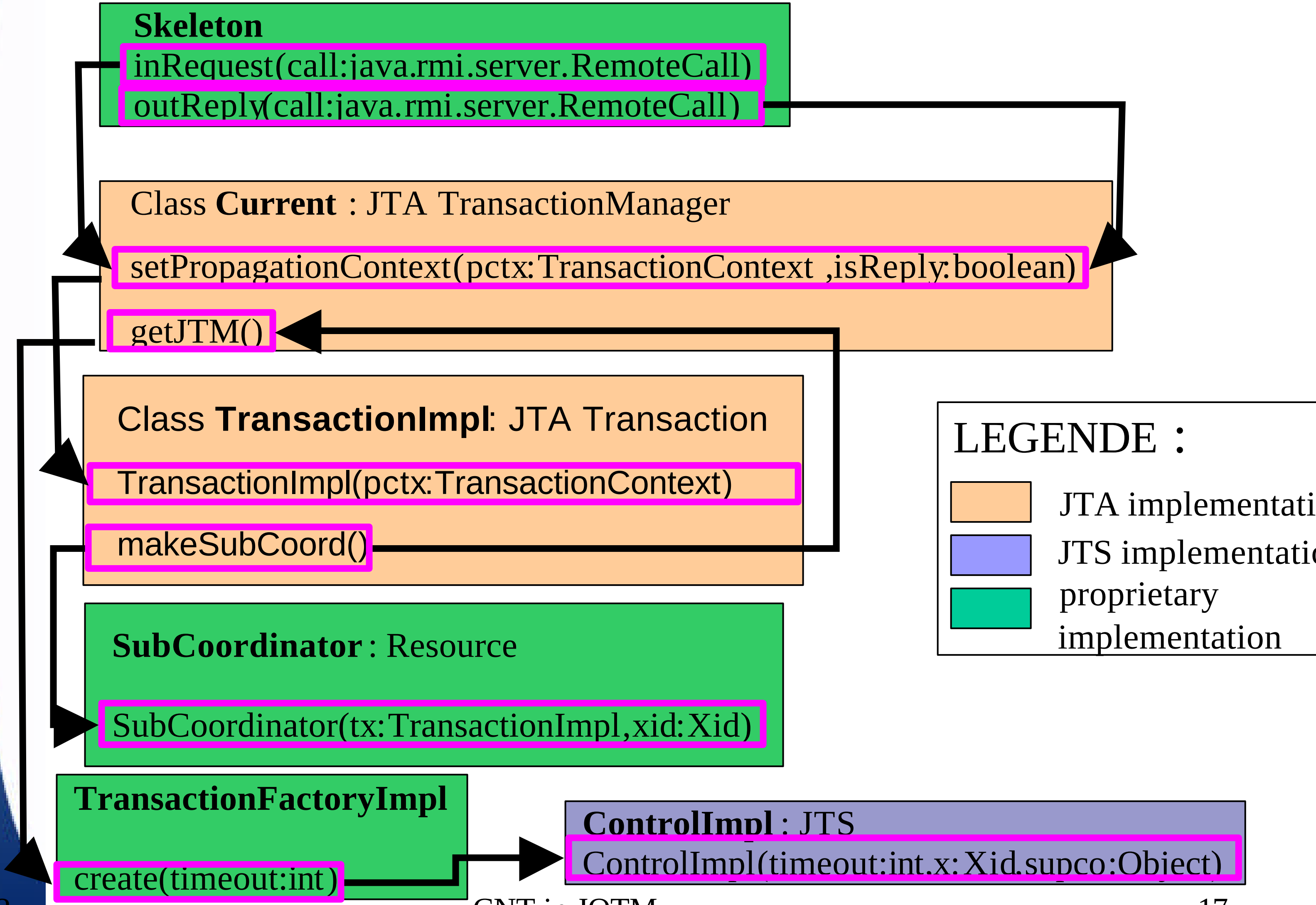
🟧 JTA implementation

🟦 JTS implementation

🟩 proprietary implementation



Distributed transaction (2 /2)



Skeleton

inRequest(call:java.rmi.server.RemoteCall)

outReply(call:java.rmi.server.RemoteCall)

Class **Current** : JTA TransactionManager

setPropagationContext(pctx:TransactionContext ,isReply:boolean)

getJTM()

Class **TransactionImpl**: JTA Transaction

TransactionImpl(pctx:TransactionContext)

makeSubCoord()

SubCoordinator : Resource

SubCoordinator(tx:TransactionImpl,xid:Xid)

TransactionFactoryImpl

create(timeout:int)

ControlImpl : JTS

ControlImpl(timeout:int,x:Xid,supco:Object)

LEGENDE :

🟧 JTA implementati

🟦 JTS implementati

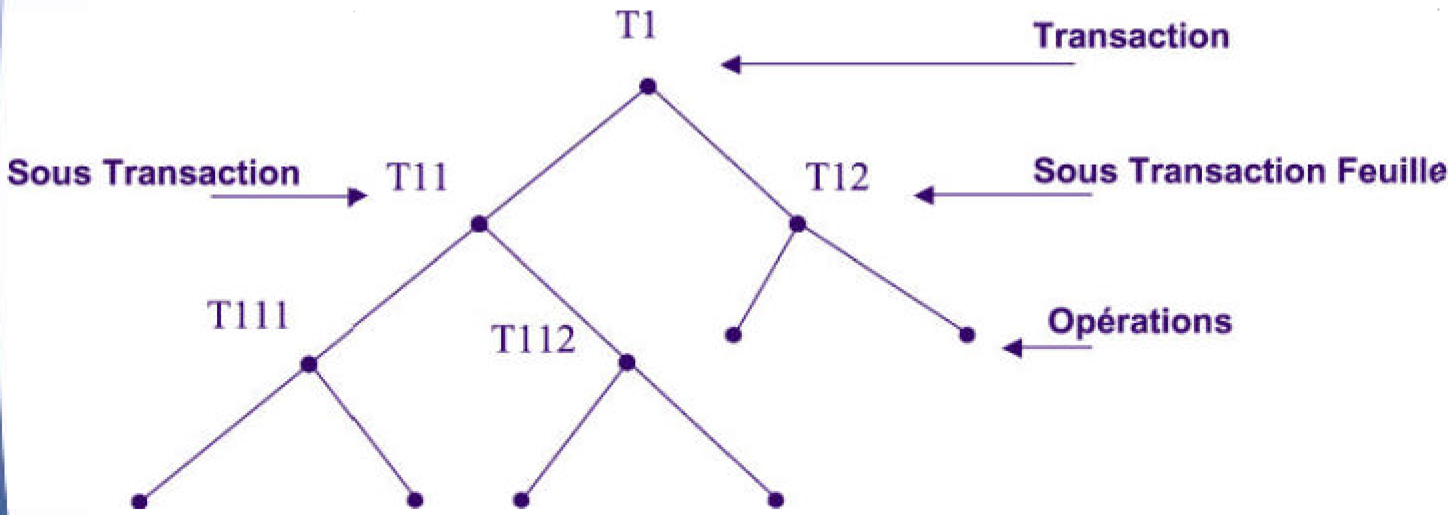
🟩 proprietary implementation

CNT

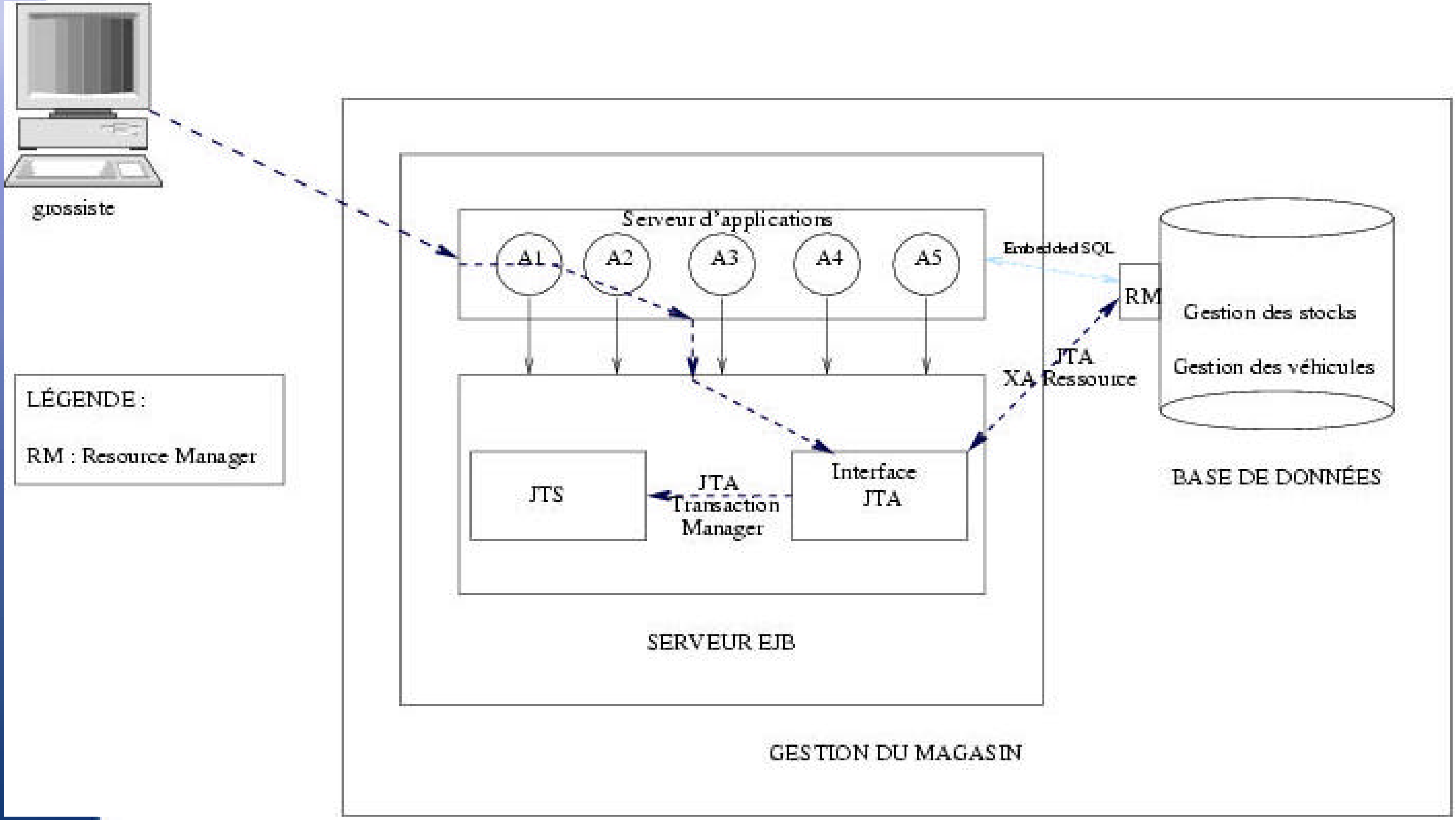
- – Description
- – CNT for distributed transactions
- – CNT for local transactions
- – Problem
- – Proposal

CNT Description

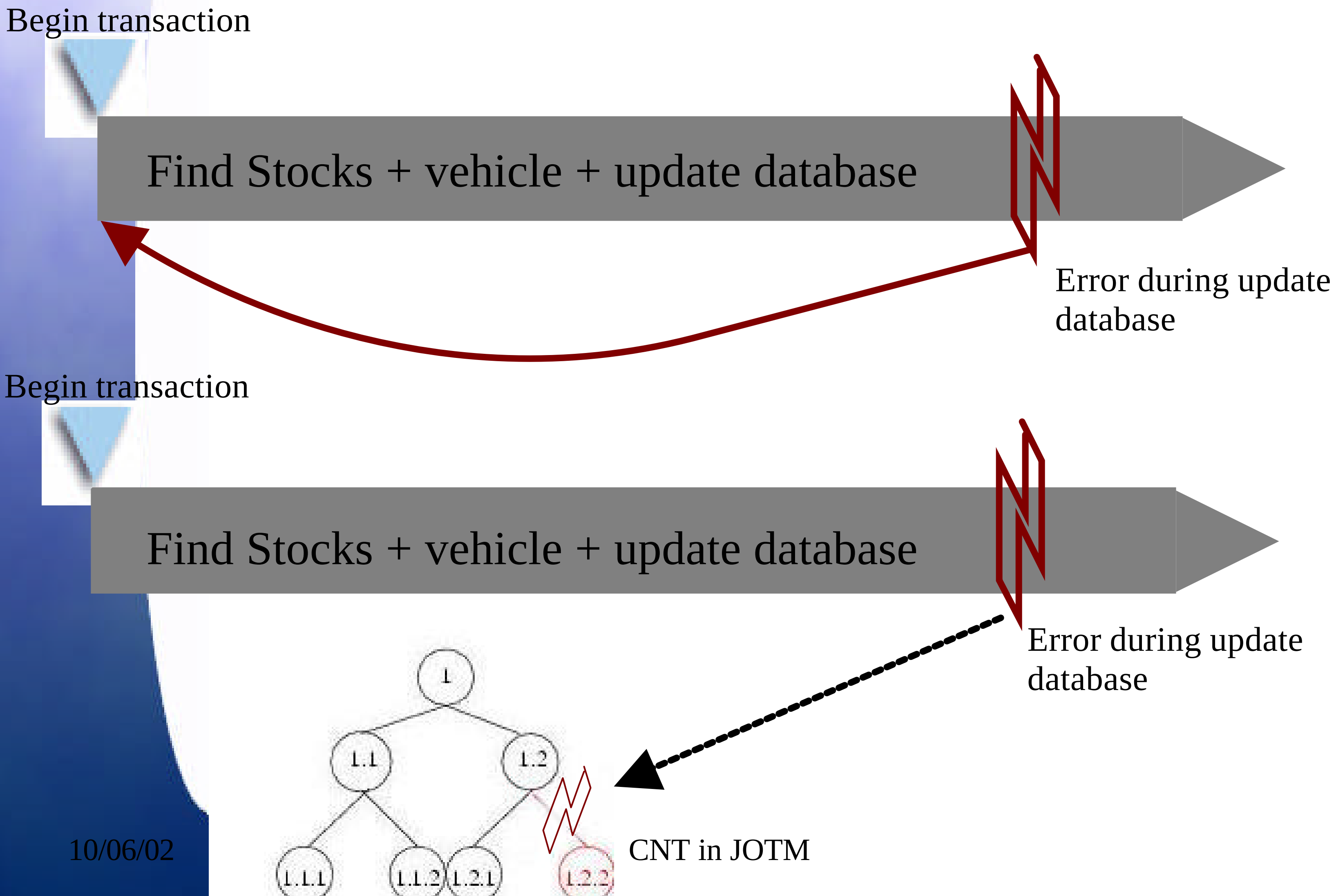
Cf cours de Didier Donsez



CNT
Advantages (1 /2)



CNT Advantages (2 /2)



Begin transaction

Find Stocks + vehicle + update database

Error during update database

Begin transaction

Find Stocks + vehicle + update database

Error during update database

CNT for distributed transaction (1 /4)

- Pepita project
- Allows to create CNT in a distributed environnement
- New interface and update of interfaces and classes

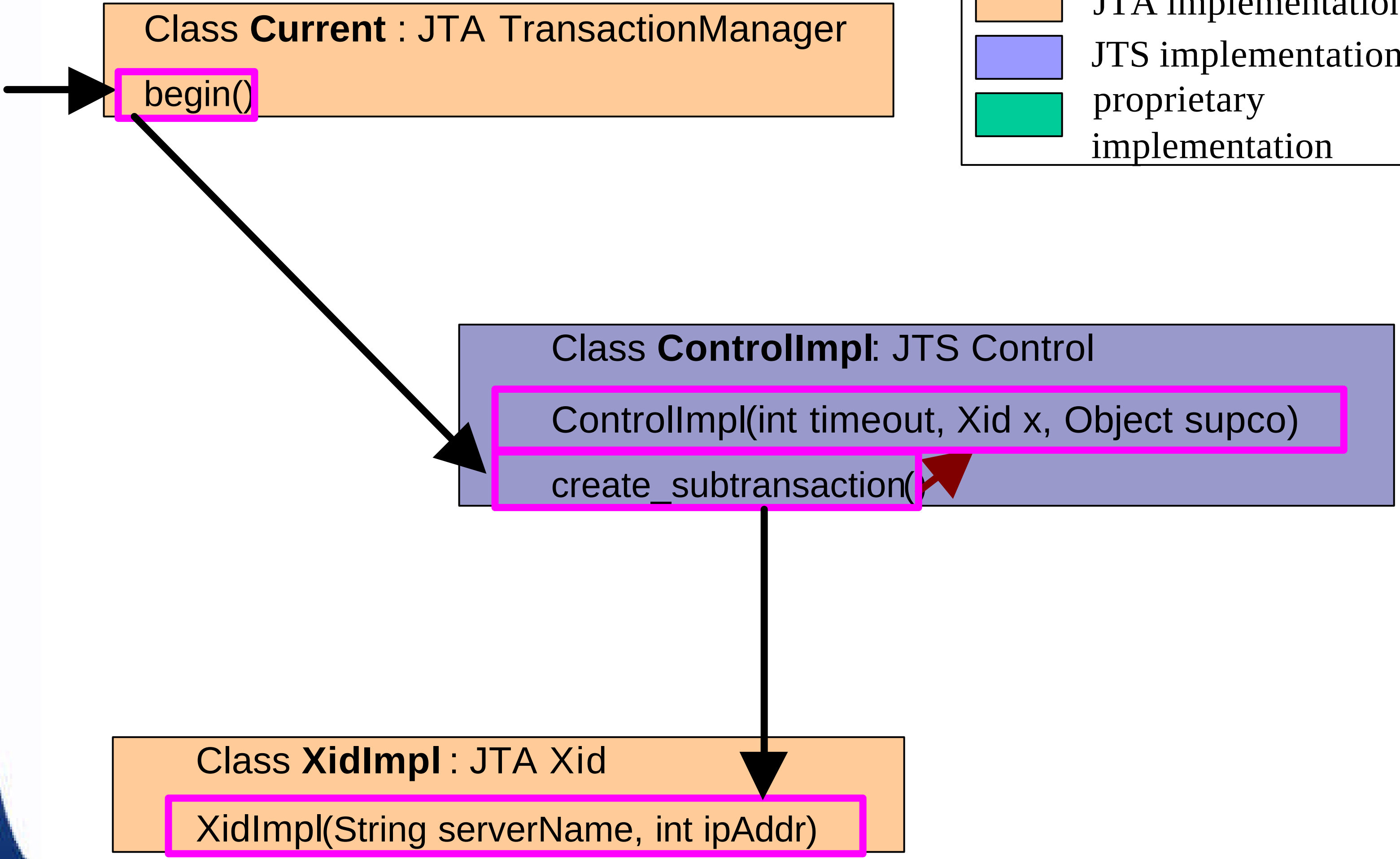
CNT for distributed transaction (2 /4)
Begin transaction

LEGENDE :

🟧 JTA implementation

🟦 JTS implementation

🟩 proprietary implementation



Class **Current** : JTA TransactionManager

begin()

Class **ControlImpl**: JTS Control

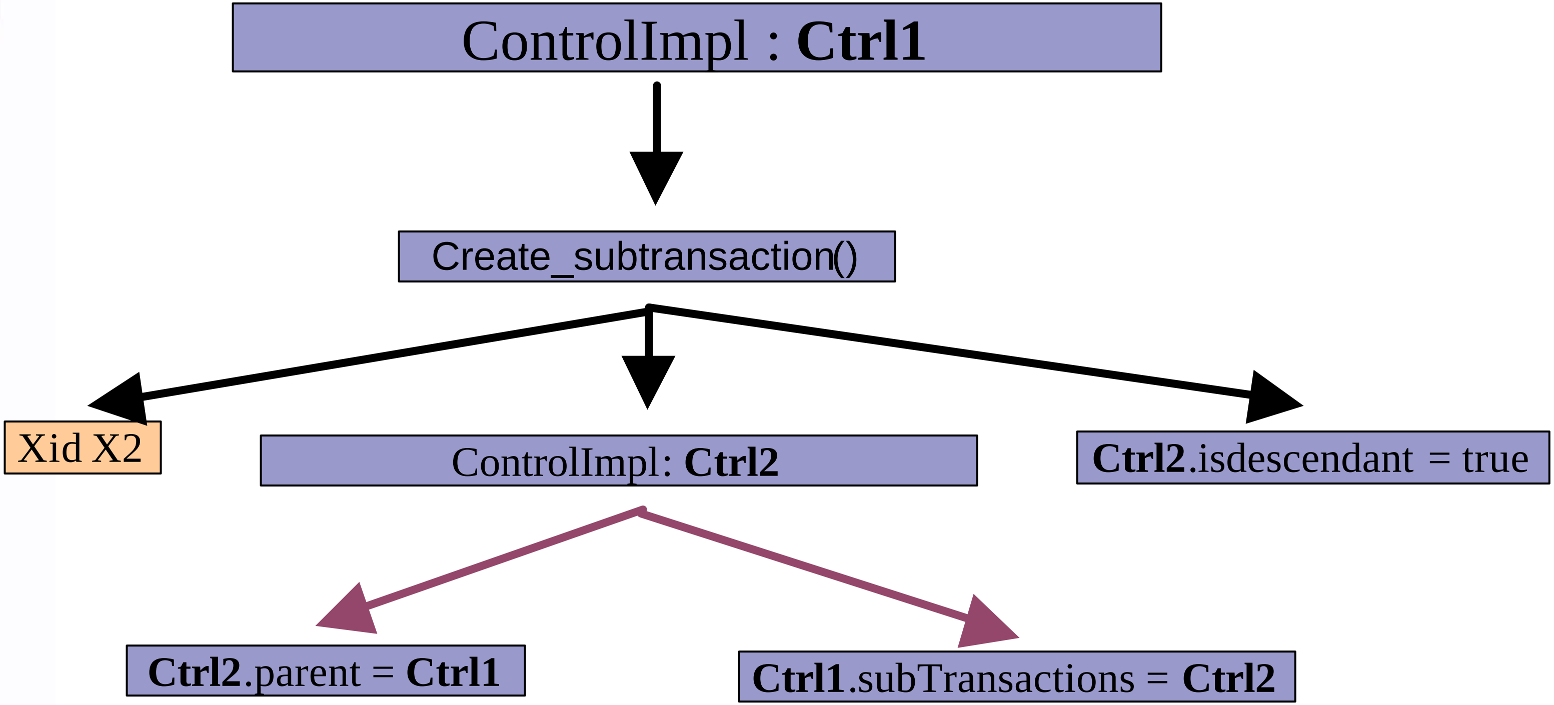
ControlImpl(int timeout, Xid x, Object supco)

create_subtransaction()

Class **XidImpl** : JTA Xid

XidImpl(String serverName, int ipAddr)

CNT for distributed transaction (3 /4)
Relation parent-chidren



ControlImpl : **Ctrl1**

Create_subtransaction()

Xid X2

ControlImpl: **Ctrl2**

Ctrl2.isdescendant = true

Ctrl2.parent = **Ctrl1**

Ctrl1.subTransactions = **Ctrl2**

Ctrl1.child = Ctrl2
Ctrl2.parent = Ctrl1

LEGENDE :

🟧 JTA implementati

🟦 JTS implementati

🟩 proprietary implementation

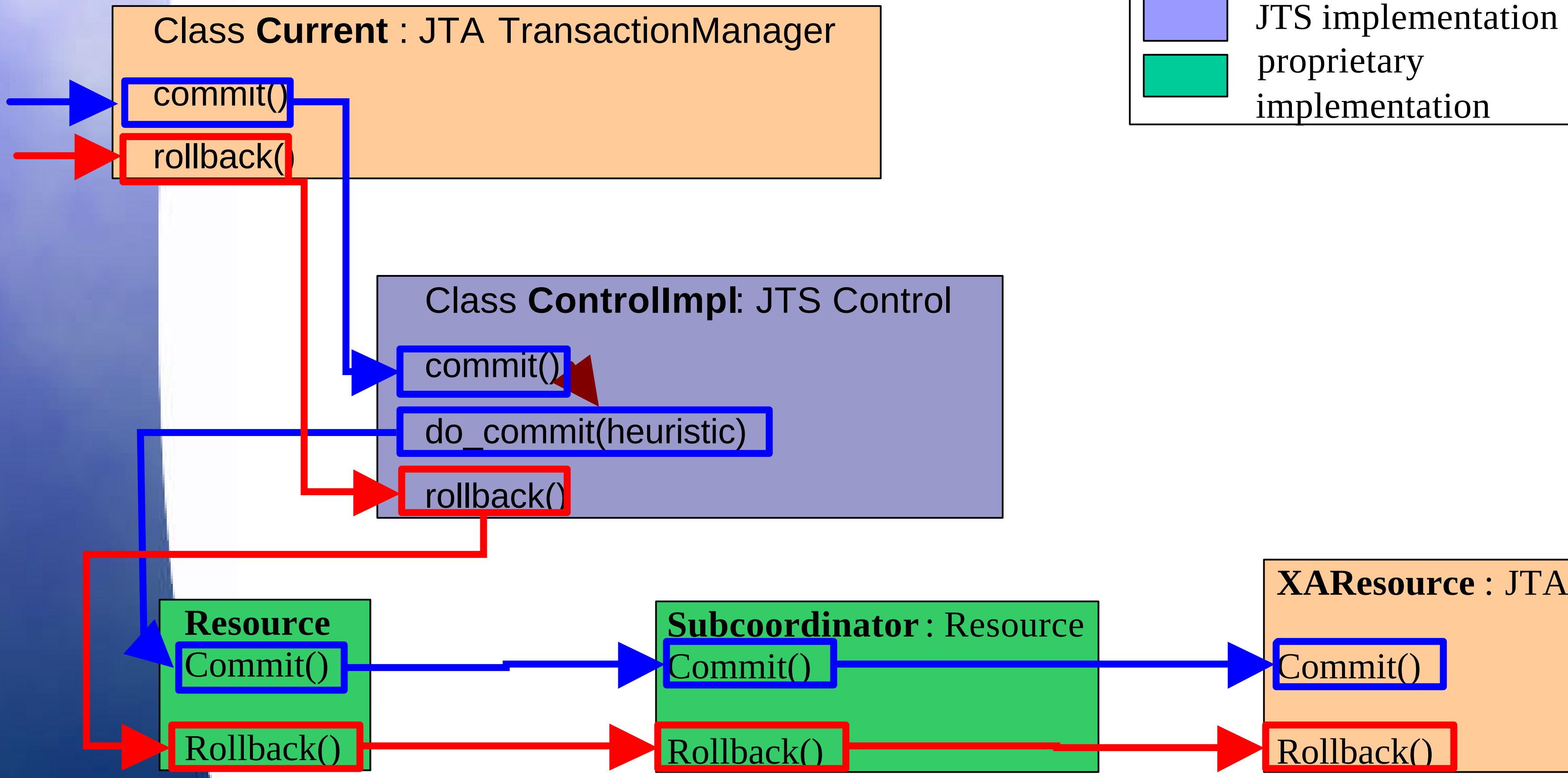
CNT for distributed transaction (4 /4)
End of transaction

LEGENDE :

🟧 JTA implementation

🟦 JTS implementation

🟩 proprietary implementation



CNT for local transaction

- JOTM uses a special transaction manager for local transactions
- **Keep coherence of JOTM's architecture**
- Needed to integrate CNT in local
  - ✓ For example, if an update of 1 000 000 000 of products in a database have a problem in the 999 999 product, we can keep the modifications of the 999 998 first products, and repeat only the subtransaction 999 999.

Problem

- Commit in local transaction :
  - ➢ Need of relation between parents and children.
  - ➢ Need to have a commit and rollback for local subtransaction.
  - ➢ Transaction manager for local transactions

- 2 solutions :

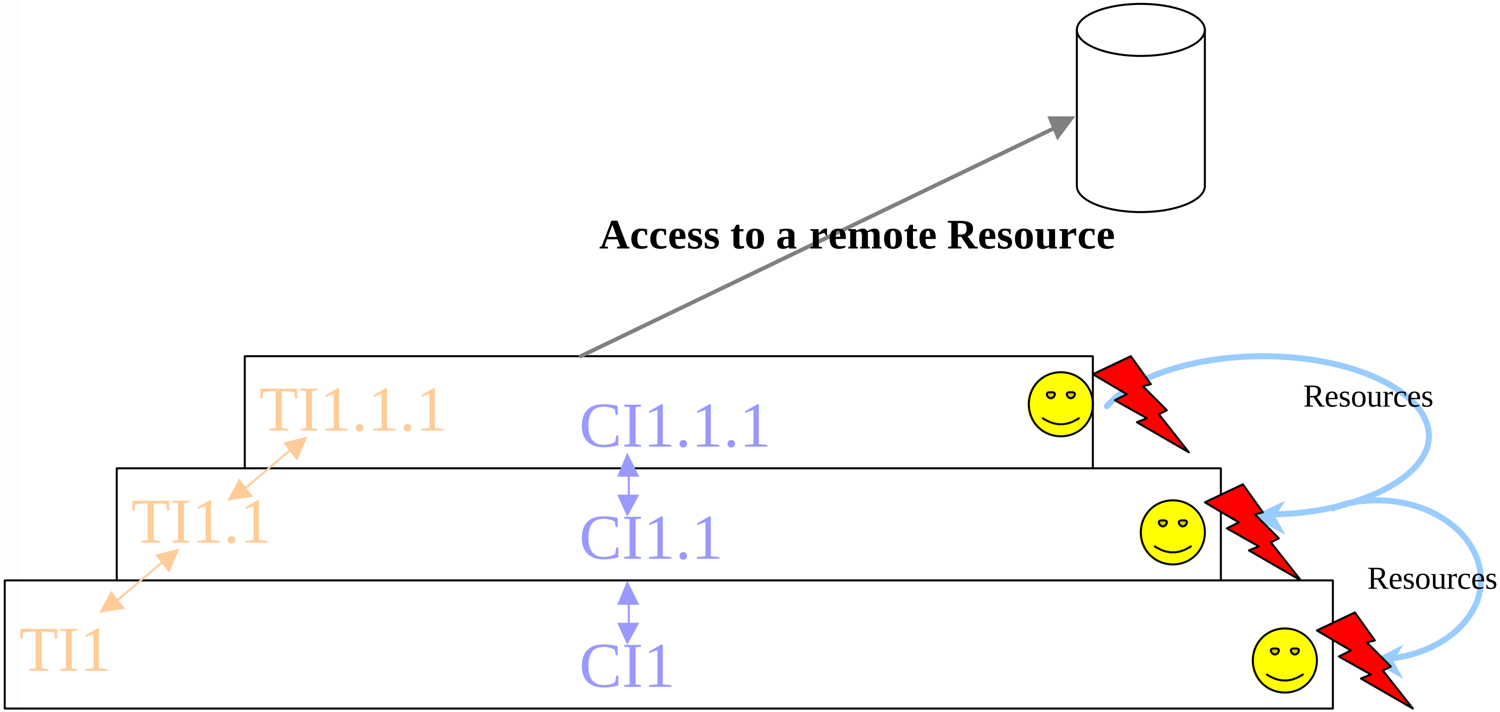
1. Use only one transaction manager for local and distributed transactions
   - ➢ JTS via JTA
2. Extends the transaction manager for local transaction to the CNT
   - ➢ A JTS light with relations parents-chidren and commit - rollback

Proposal (1 /2)

This proposal is an extension of the local transaction manager of JOTM => Extension of JTA implementation in JOTM

- **CoordinatorLocal** : Give identifier of top level transaction and the recovery coordinator
- **TerminatorLocal** : idem OTS
- **TransactionControlLocal** : idem OTS
- **TransactionLocal** : Relations between parents and chidren
- **SubtransactionAwareResourceLocal** : idem OTS

Proposal (2 /2)

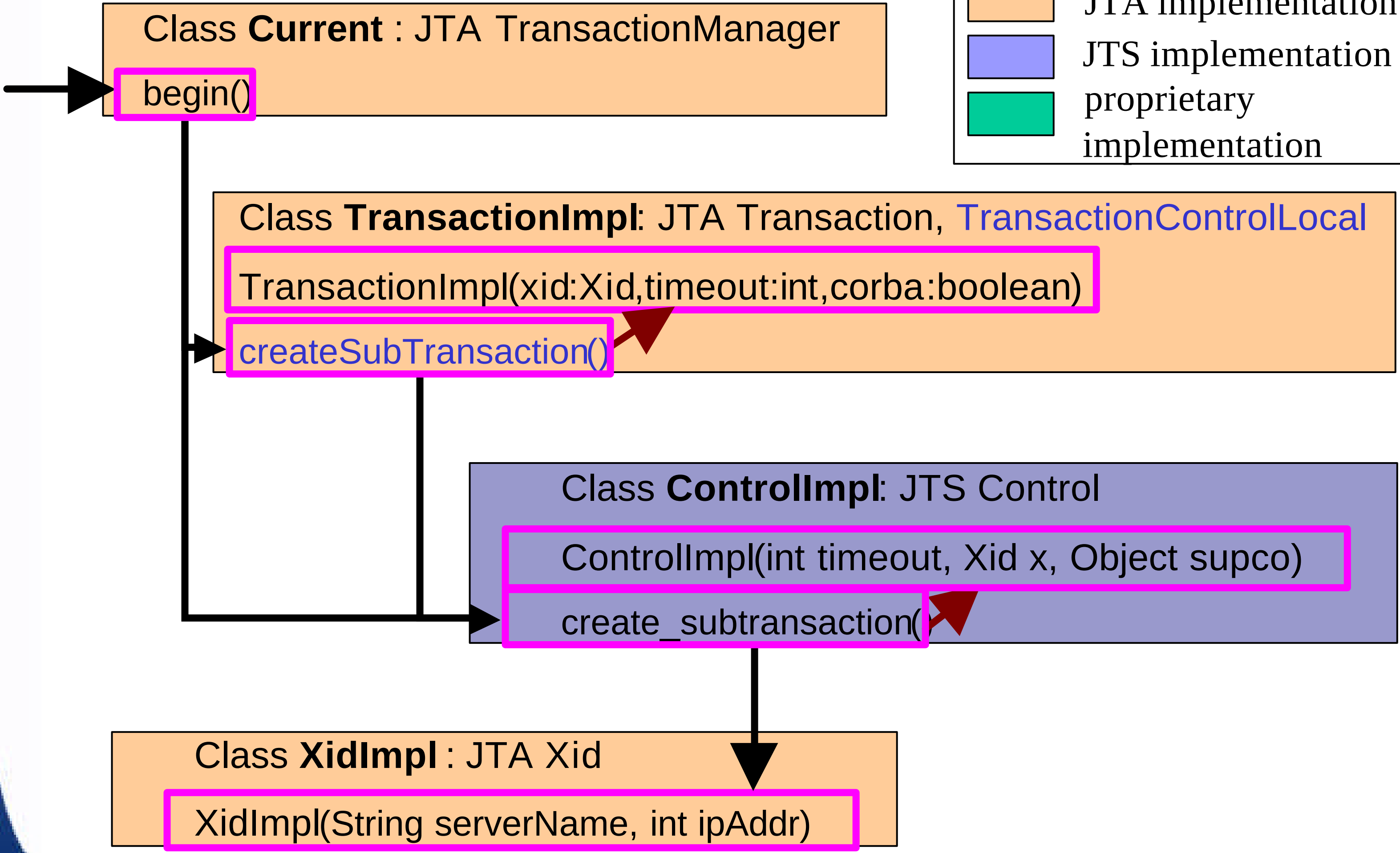


LEGENDE :
TI : TransactionImpl
CI : ControlImpl

CNT in distributed transaction now
Begin Transaction

LEGENDE :

🟧	JTA implementation
🟦	JTS implementation
🟩	proprietary implementation



Class **Current** : JTA TransactionManager

begin()

Class **TransactionImpl**: JTA Transaction, TransactionControlLocal

TransactionImpl(xid:Xid,timeout:int,corba:boolean)

createSubTransaction()

Class **ControlImpl**: JTS Control

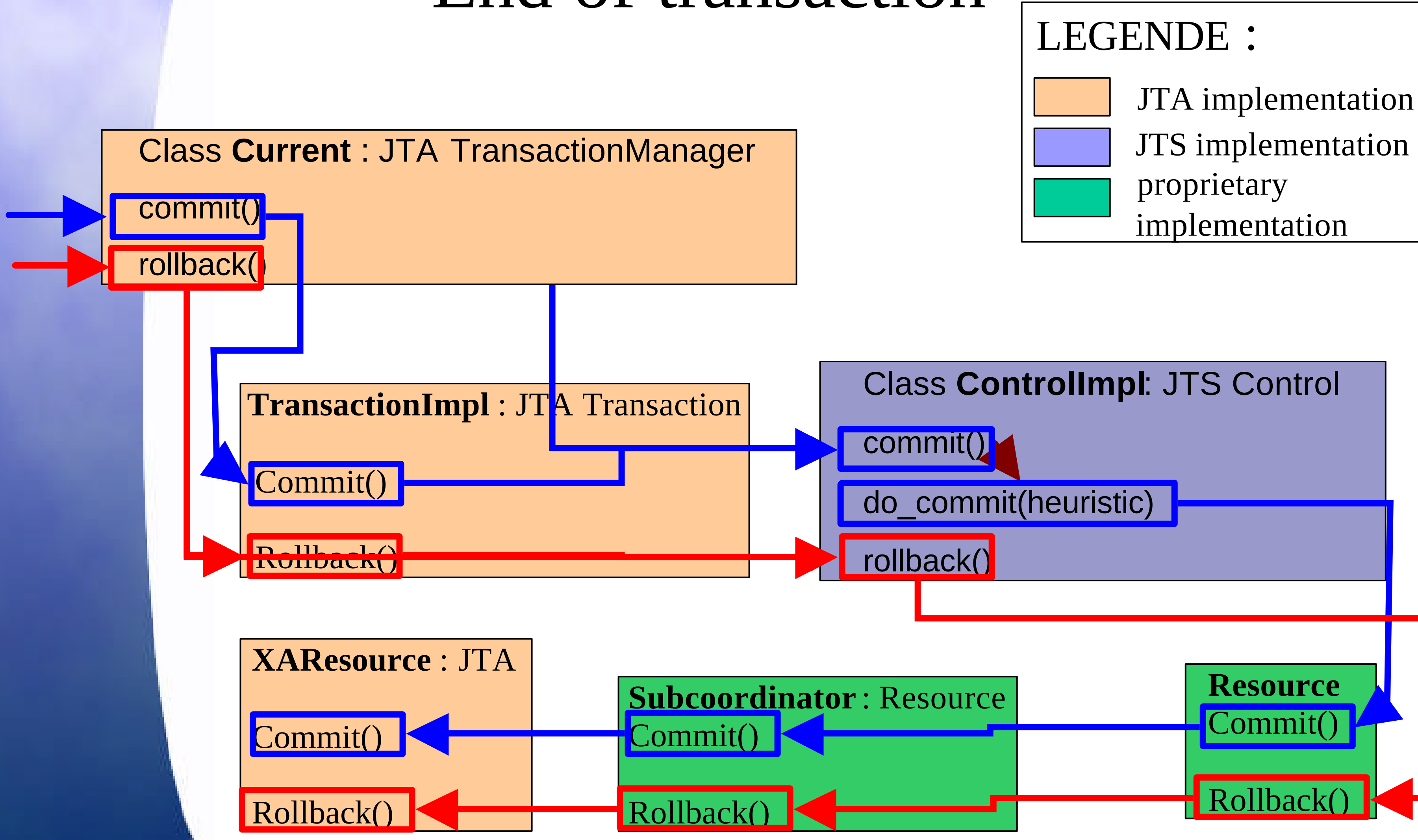
ControlImpl(int timeout, Xid x, Object supco)

create_subtransaction()

Class **XidImpl** : JTA Xid

XidImpl(String serverName, int ipAddr)

CNT for distributed transaction now
End of transaction

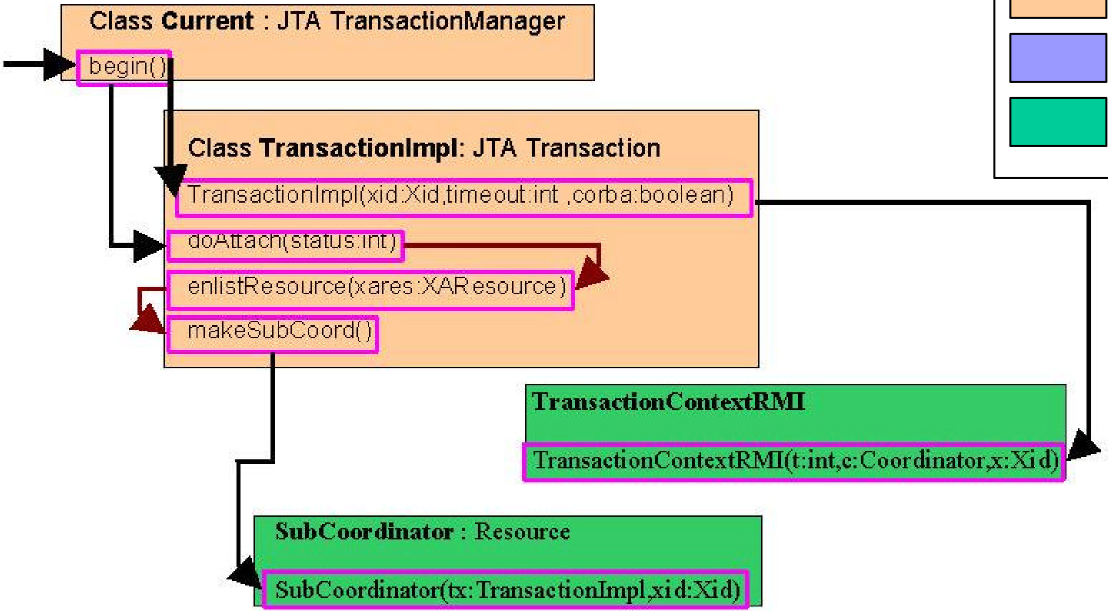


CNT in local transaction

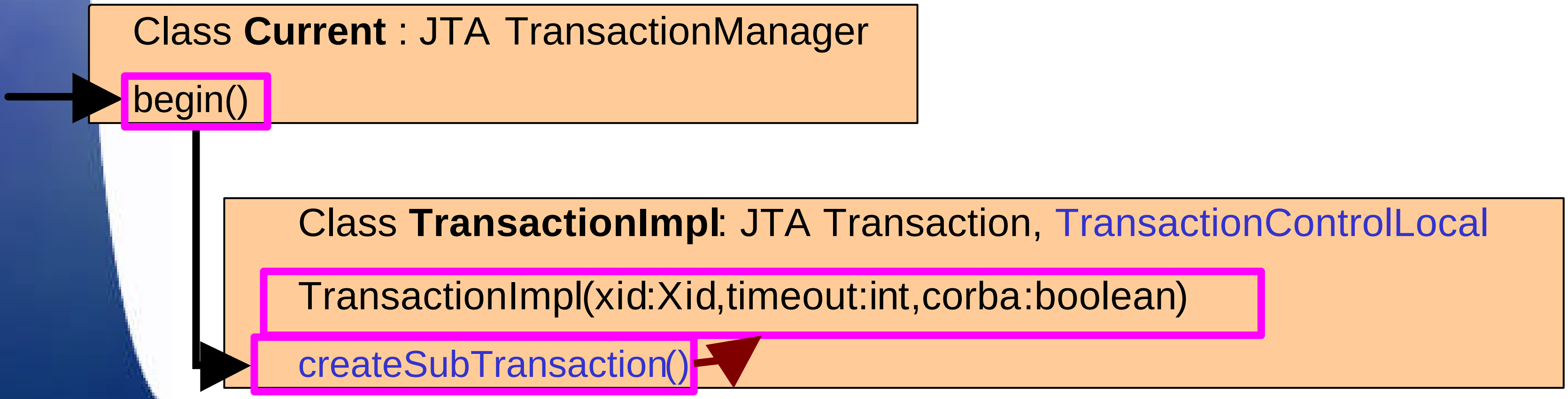
Begin Transaction

LEGENDE :

🟧	JTA implementatio
🟦	JTS implementatio
🟩	proprietary implementation

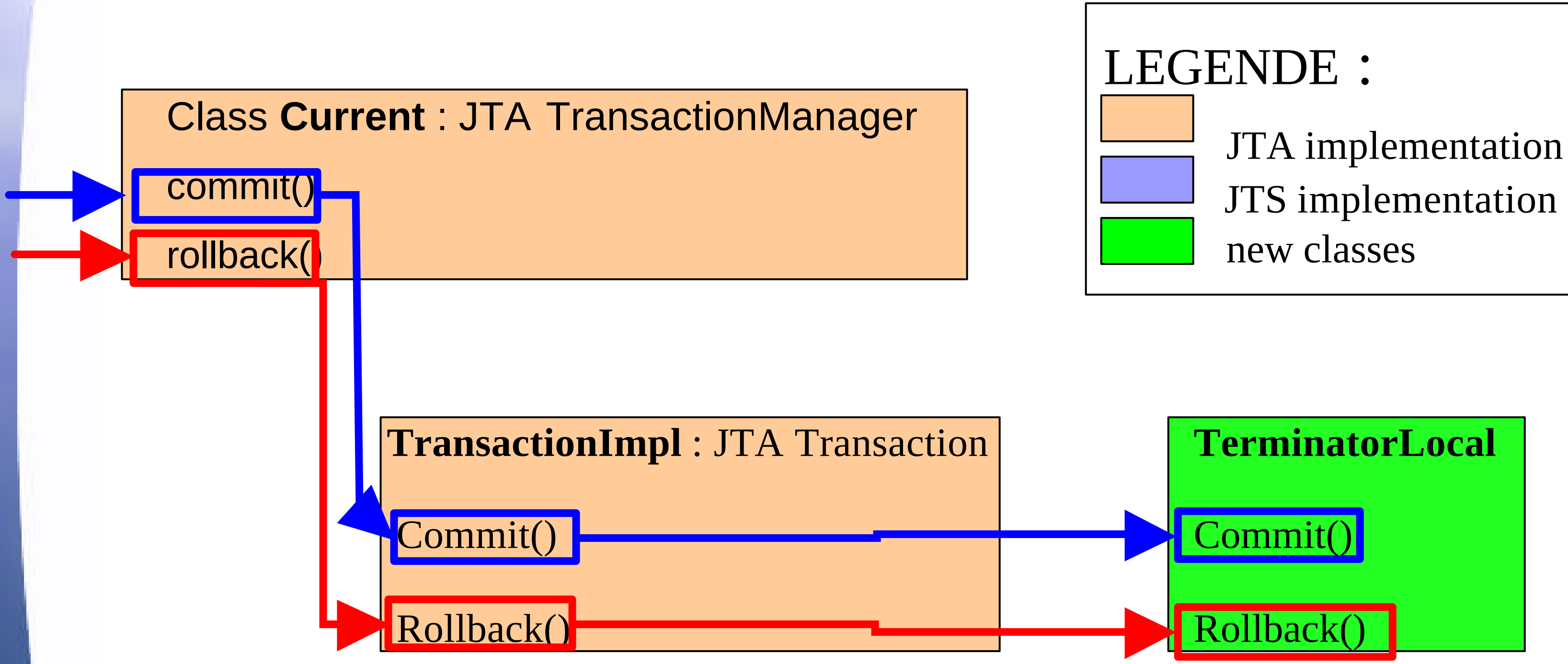


Class **Current** : JTA TransactionManager
begin()
Class **TransactionImpl**: JTA Transaction
TransactionImpl(xid:Xid,timeout:int ,corba:boolean)
doAttach(status:int)
enlistResource(xares:XAResource)
makeSubCoord()
TransactionContextRMI
TransactionContextRMI(t:int,c:Coordinator,x:Xid)
SubCoordinator : Resource
SubCoordinator(tx:TransactionImpl,xid:Xid)



Class **Current** : JTA TransactionManager
begin()
Class **TransactionImpl**: JTA Transaction, TransactionControlLocal
TransactionImpl(xid:Xid,timeout:int,corba:boolean)
createSubTransaction()

CNT for local transaction End of transaction



LEGENDE :

🟧 JTA implementation

🟪 JTS implementation

🟩 new classes

Class **Current** : JTA TransactionManager
commit()
rollback()
TransactionImpl : JTA Transaction
Commit()
Rollback()
TerminatorLocal
Commit()
Rollback()

Conclusion

- What exists in JOTM
- The 2 transactions managers
- CNT for distributed transaction
- Problem to keep the coherence of architecture
- Need to choose an architecture to include CNT
- Include CNT for local and distributed transactions
- Include the ONT in JOTM