Domain Layer & EJBs: Developing **Session Beans**

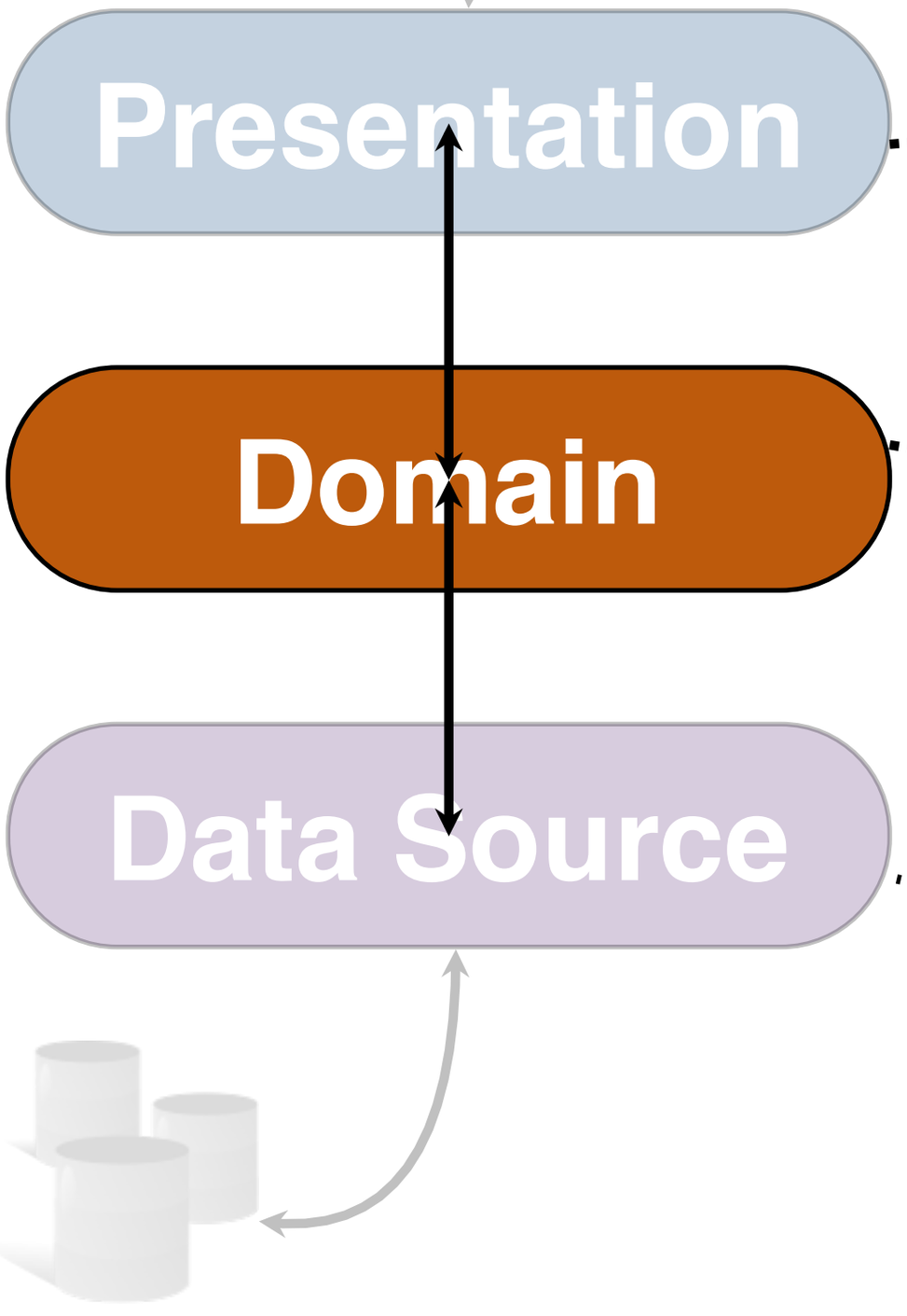
Sébastien Mosser

Lecture #2.2, 01.03.2018

Revu par AM Dery en avril 2020

pictures: sxc.hu

3-tiers architecture



Handle users

Business functionality

Handle data storage

1 Principles

2 State(lesslful) beans

3 Example

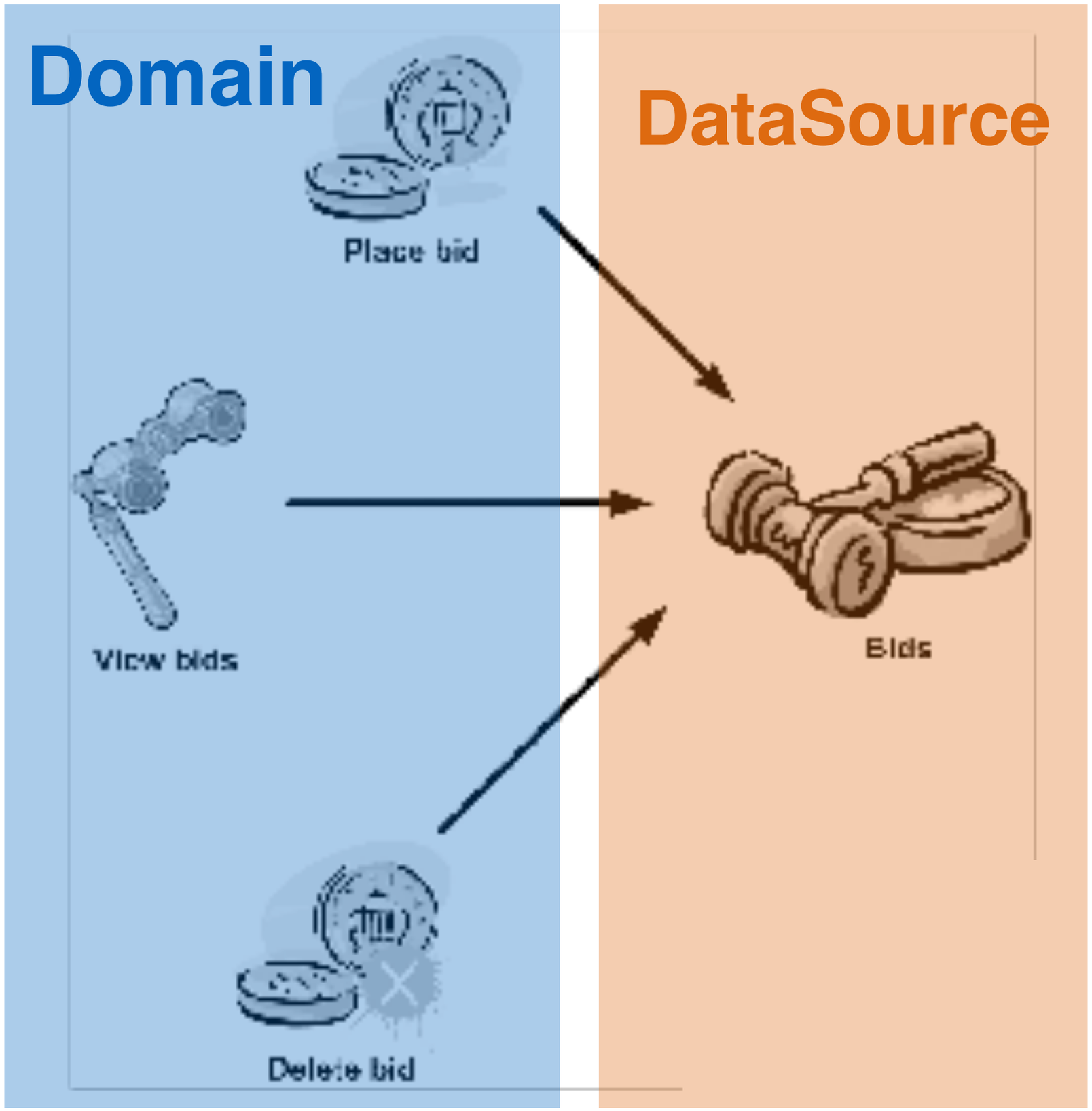
4

Principles

Rule of Thumb

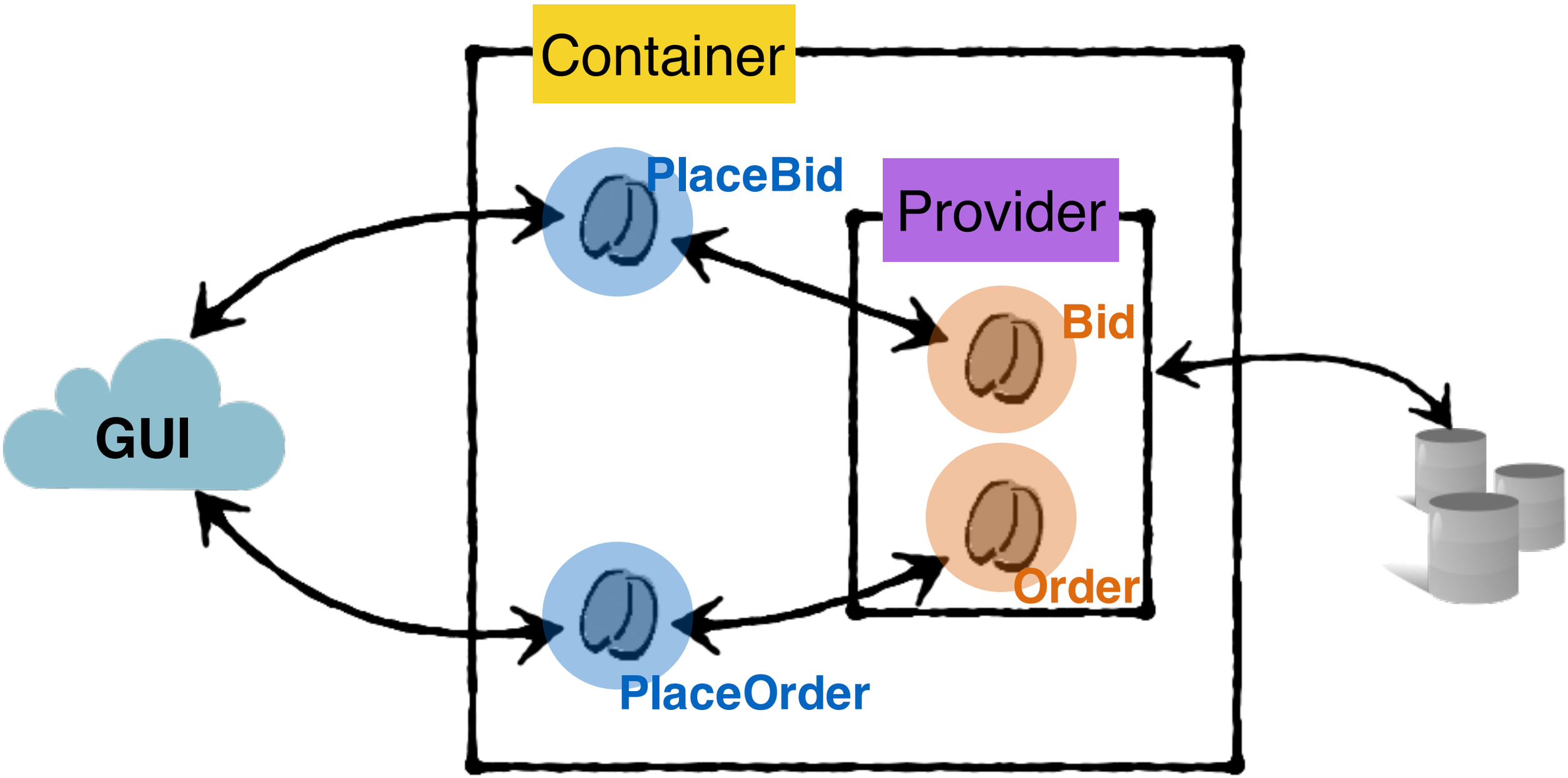
Domain Bean interfaces as **Verbs**

DataSource Beans as **Nouns**



[EiA]

Client **never calls** a datasource directly



[EiA]

You'll **never** instantiate a domain bean.

3 types de Beans

Session Beans

Traitements : services fournis

Entity Beans

Objets métiers qui existent dans le système de stockage permanent

Message Driven Beans

Traitent les messages asynchrones

State(less/ful) beans

Session Beans

Modélisent un processus métier (traitement ou session)

Durée de vie : la session (temps qu'un client reste connecté sur le bean)

Cycle de vie

•Création d'une instance par le conteneur lorsque le client se connecte sur le session bean

•Destruction par le conteneur possible lorsque le cieInt se déconnecte

non persistants

2 types de Session Beans

2 types de conversations différentes

Stateful Session Beans

Nécessité de conserver un état (plusieurs requêtes successives

Maintient l'état pendant la durée de vie du client : l'EJB est propre à un client pendant toute la durée de la session.

Une instance par client

Stateless Session Beans

Sans état ne conserve pas d'information entre 2 appels successifs

Importance des données transmises échangées entre cclients et composants

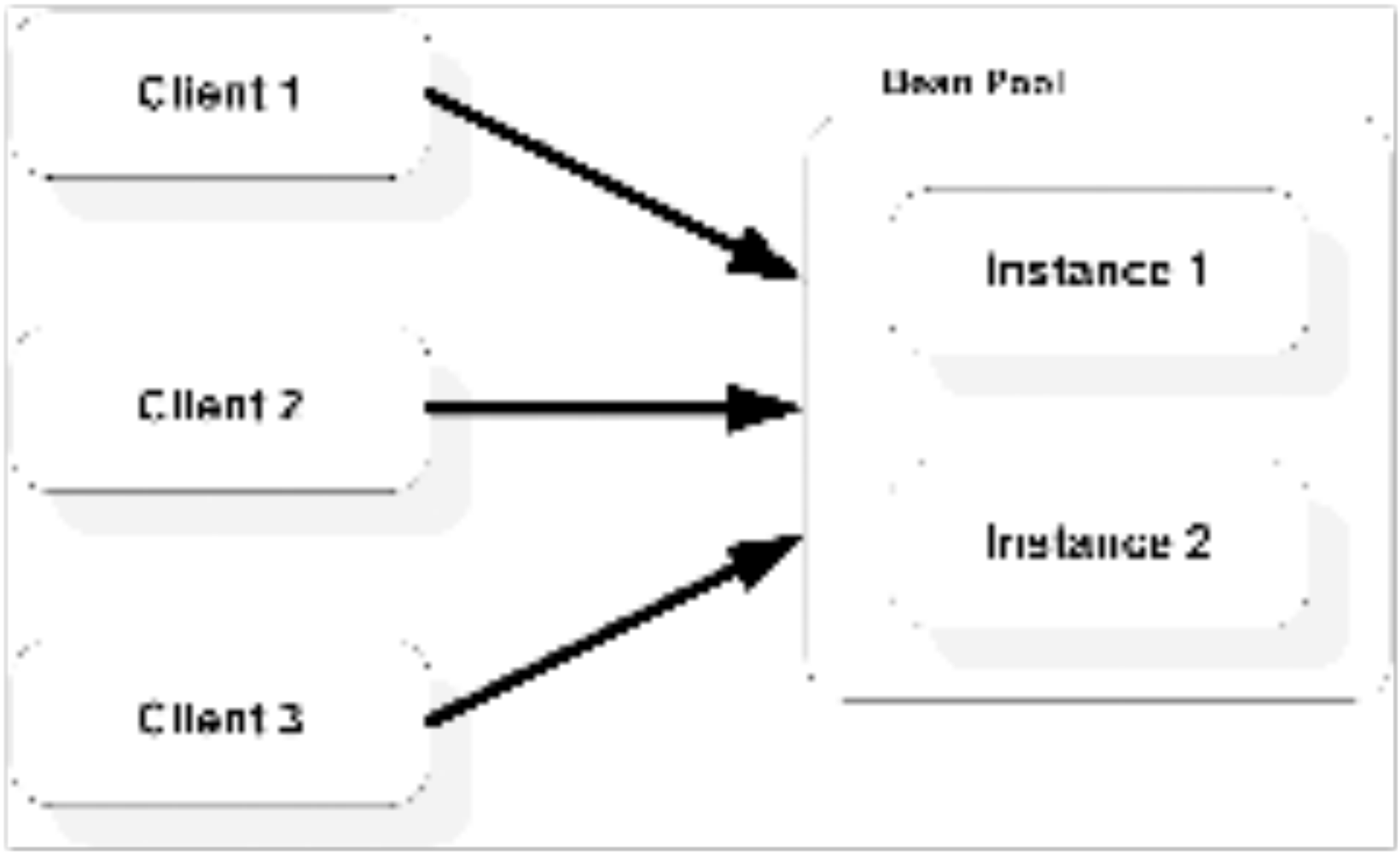
EJB partageable consécutivement par de nombreux clients

EJB's Lifecycle: **Inversion of Control**

"Don't call us, we'll call you"

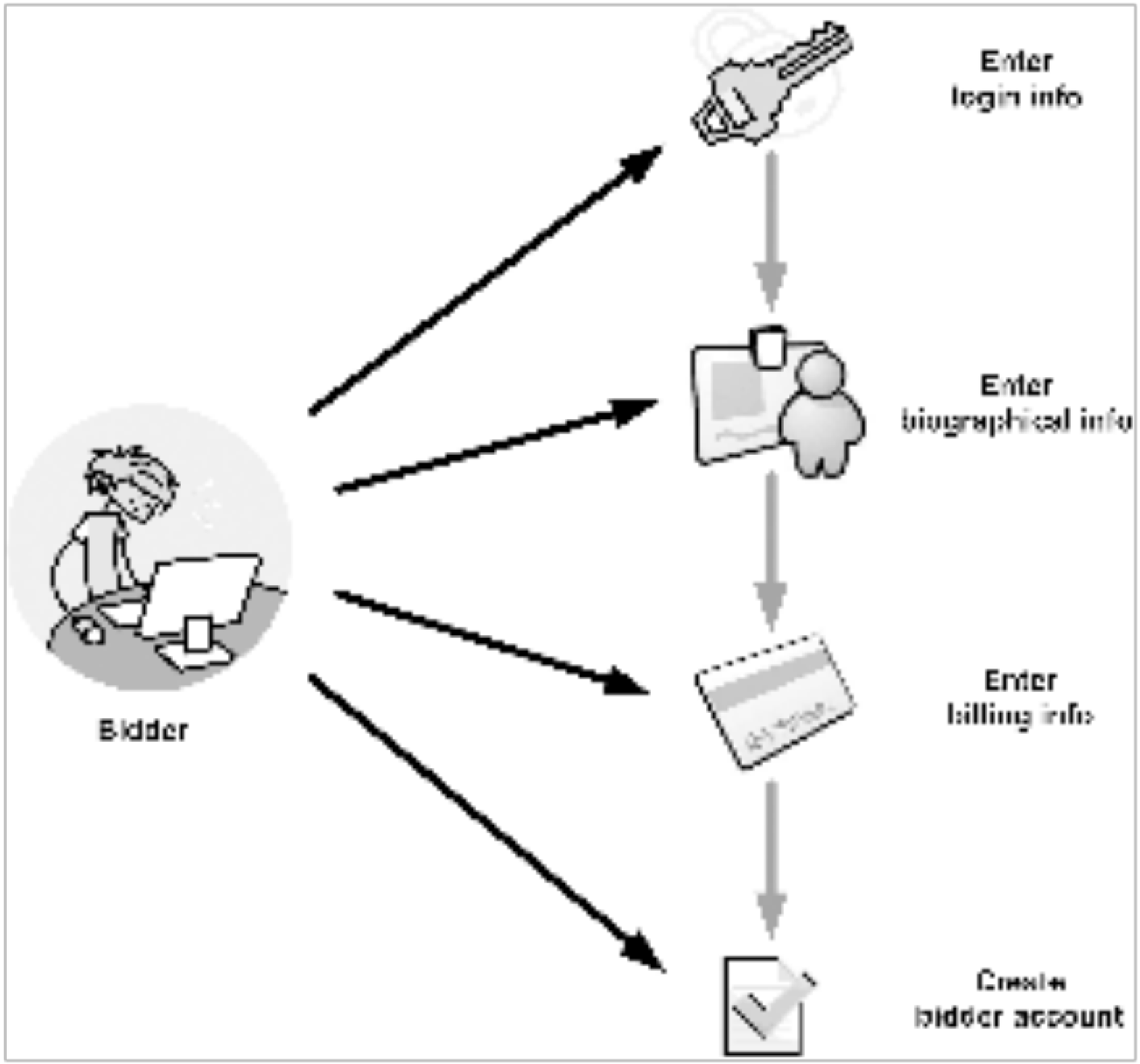
[EiA]

EJBs live in a "**pool**"



[EiA]

Domain beans live during a **Session**



[EiA]

Stateless beans

Stateless Cycle de vie

Le conteneur

1. gère un pool de beans

(création, destruction, mise à disposition)

2. Injecte les dépendances

3. Appelle la méthode @PostConstruct

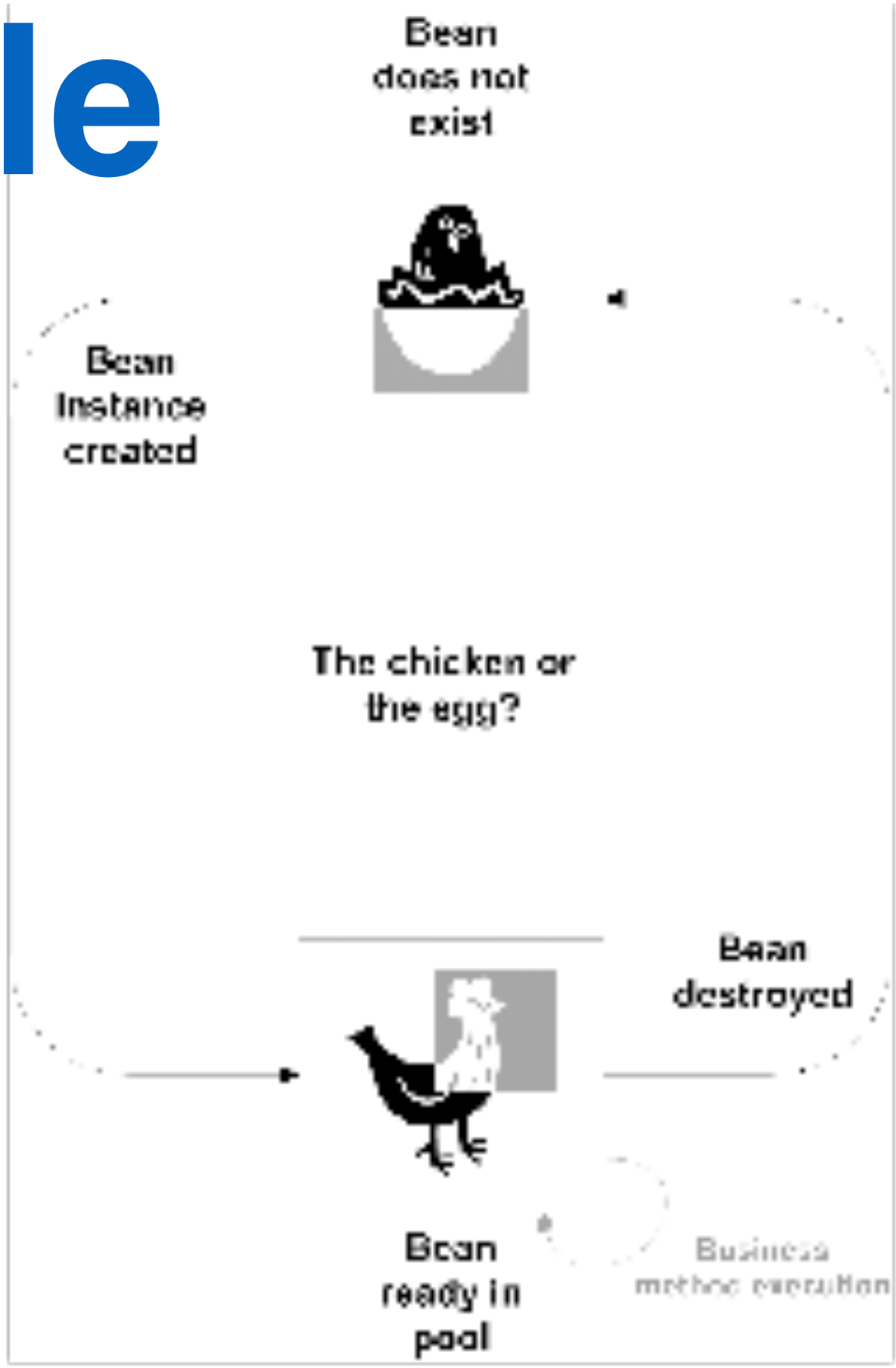
Et à la fin du cycle

4. Appelle la méthode @PreDestroy

https://fr.slideshare.net/HeithemAbbes1/entreprise-java-beans-ejb

Lifecycle

Handled by the container



[EiA]

Lifecycle **Hooks**: Construct, Destroy

```
@PostConstruct
public void initialize() {
   System.out.println("Initializing PetManager");
}

@PreDestroy
public void cleanup() {
   System.out.println("Destroying PetManager");
}
```

Stateless beans : POJO + Annotations

Interface

```
public interface PetManager {
    public Pet create(String name);
}
```

Bean

```
@Stateless
public class PetManagerBean implements PetManager {

    @PersistenceContext
    EntityManager entityManager;

    @Override
    public Pet create(String name){
        Pet p = new Pet(name);
        entityManager.persist(p);
        return p;
    }

}
```

Consuming a Bean: Inversion of Control

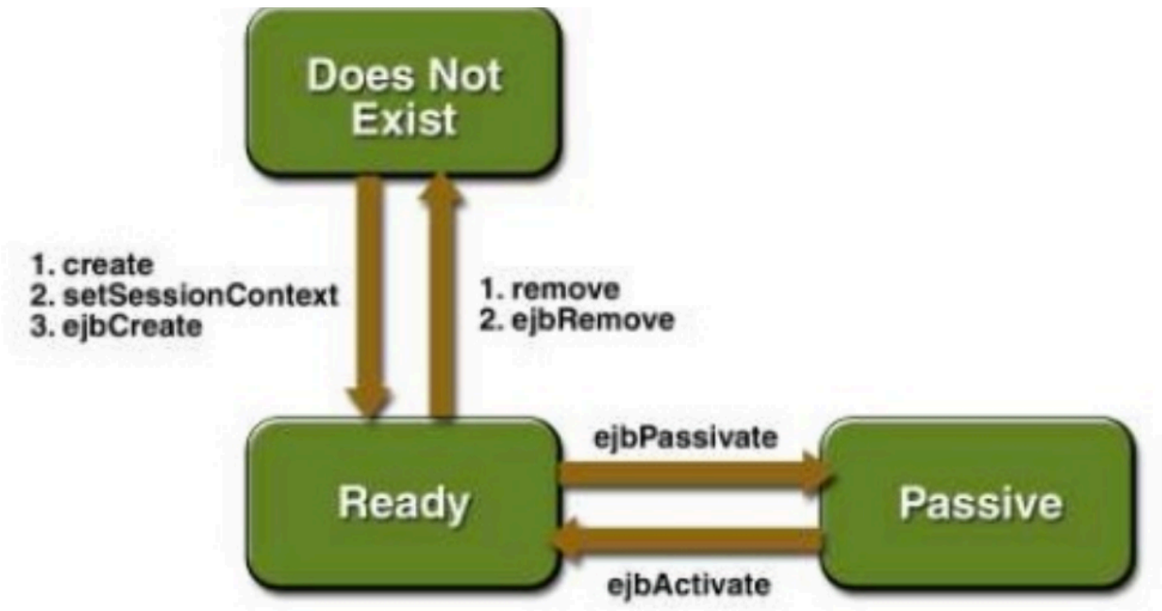
```
@EJB
private PetManager manager;
```

Interface

```
@Test
public void testCreation() throws Exception {
   Pet jinx = manager.create("Jinx");
   assertEquals(jinx.name, "Jinx");
}
```

Stateful beans

Stateful : Cycle de vie



Le conteneur fonctionne comme pour stateless mais en plus

Peut désactiver le bean en le sauvegardant et donc

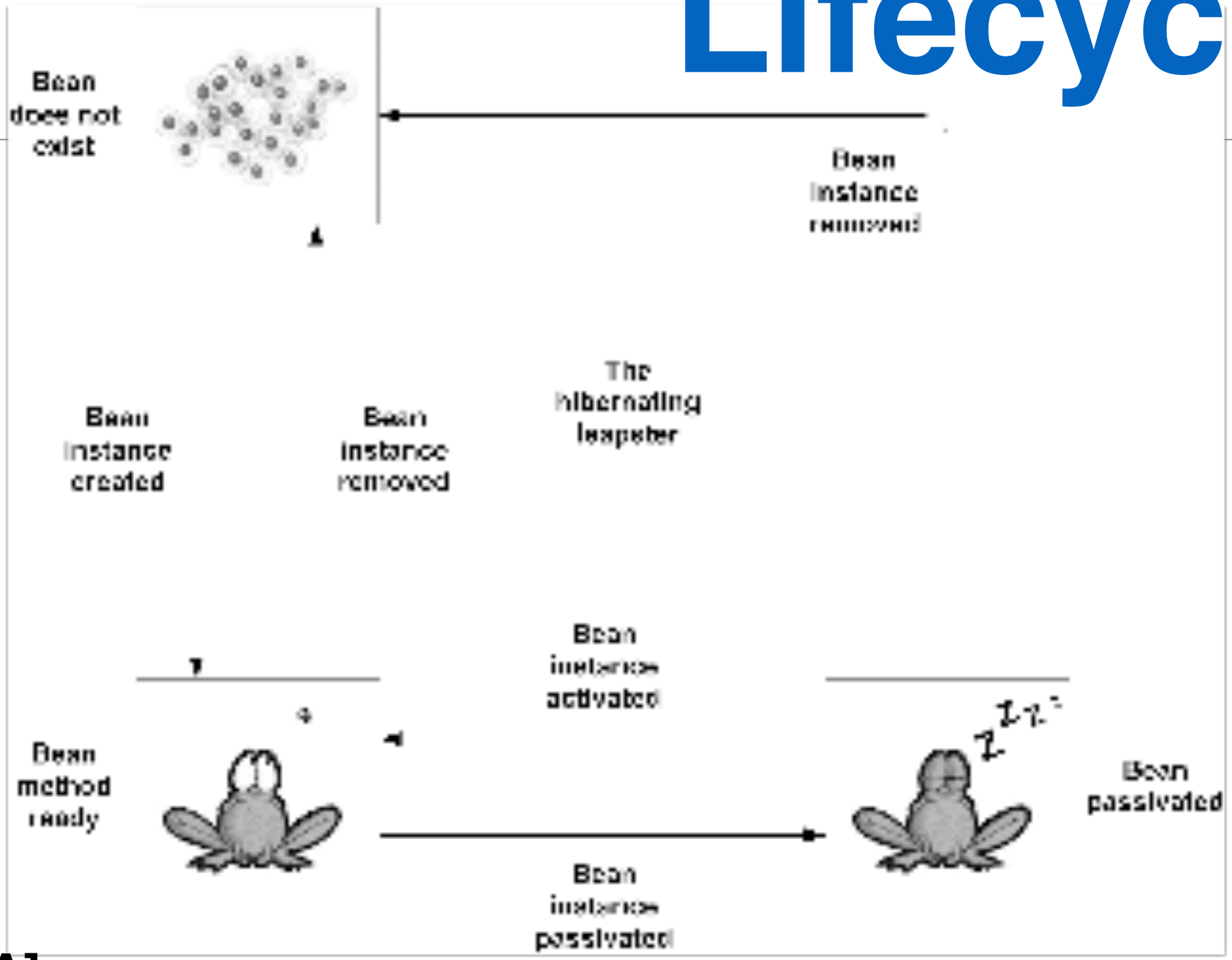
Appelle avant la passivation la méthode @PrePassivate

Et après l'activation

Appelle la méthode @PostActivate

https://fr.slideshare.net/HeithemAbbes1/entreprise-java-beans-ejb

Lifecycle



[EiA]

Lifecycle **Hooks**: Stateless + Passivate

@PostConstruct @PreDestroy

@PrePassivate @PostActivate

Maintaining **States** during **Sessions**

[EiA]

Stateful Bean : Classical Interface

```
public interface PetCart {

    public void addPet(Pet p);

    public List<Pet> getContents();

}
```

```
@Stateful
public class PetCartBean implements PetCart {

   private ArrayList<Pet> _contents =
      new ArrayList<Pet>();

   @Override
   public void addPet(Pet p) {
      _contents.add(p);
   }

   @Override
   public List<Pet> getContents() {
      return _contents;
   }
}
```

Stateless

versus

Stateful

Comment procéder ?

---

https://github.com/polytechnice-si/4A_ISA_TheCookieFactory/blob/develop/chapters/BusinessComponents.md

Implémentation de Card par un composant Stateful

Puis

Par un composant Stateless

Implémenttaions appuyée d’une argumentation privilégiant la solution stateless

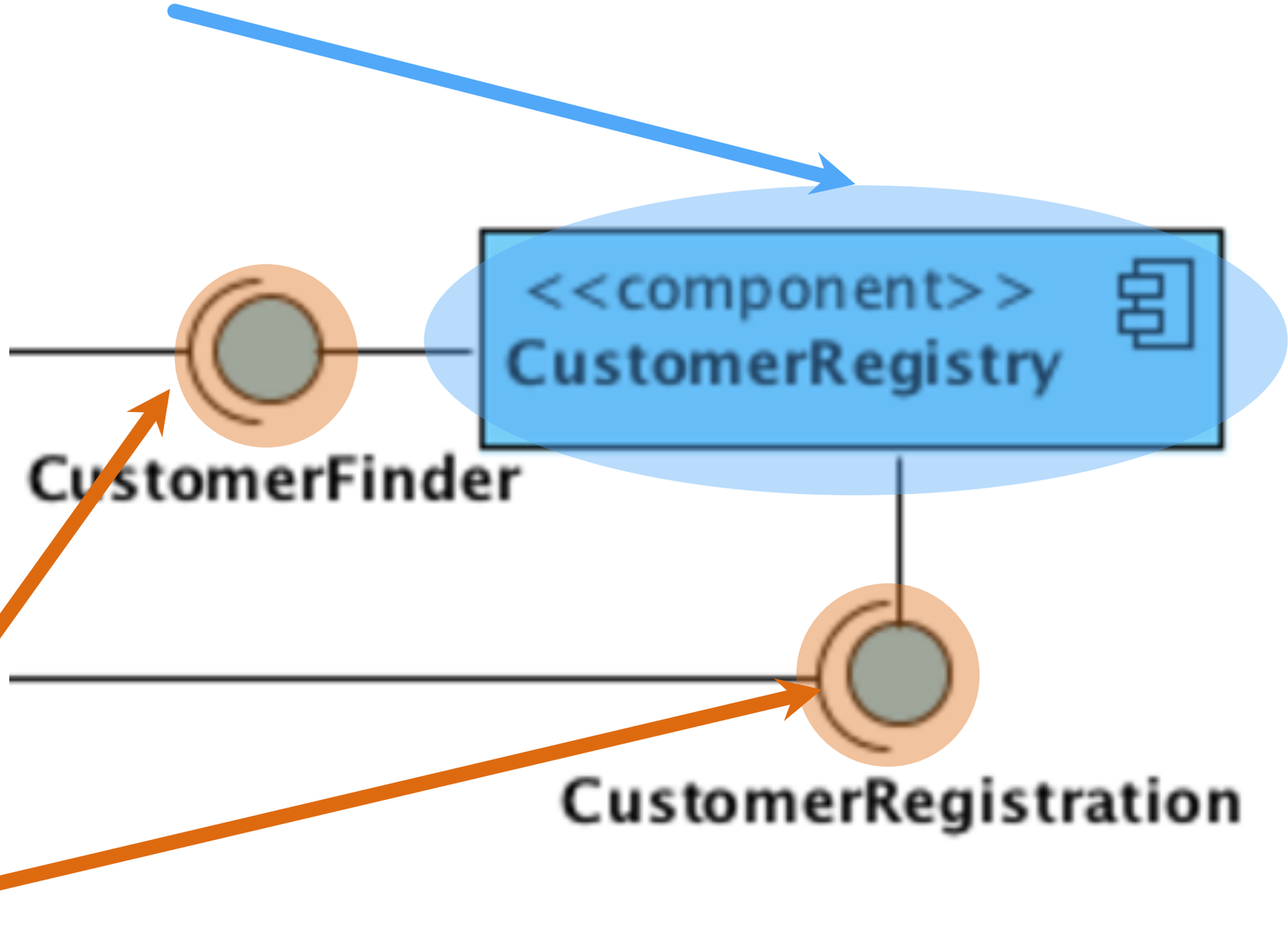
Stateless versus Statefull beans

Features	Stateless	Stateful
Conversational state	No	Yes
Pooling	Yes	No
Performance problems	Unlikely	Possible
Lifecycle events	`PostConstruct`, `PreDestroy`	`PostConstruct`, `PreDestroy`, `PrePassivate`, `PostActivate`
Timer (discussed in chapter 5)	Yes	No
`SessionSynchronization` for transactions (discussed in chapter 6)	No	Yes
Web services	Yes	No
Extended `PersistenceContext` (discussed in chapter 9)	No	Yes

[EiA]

Example 3

Stateless Bean

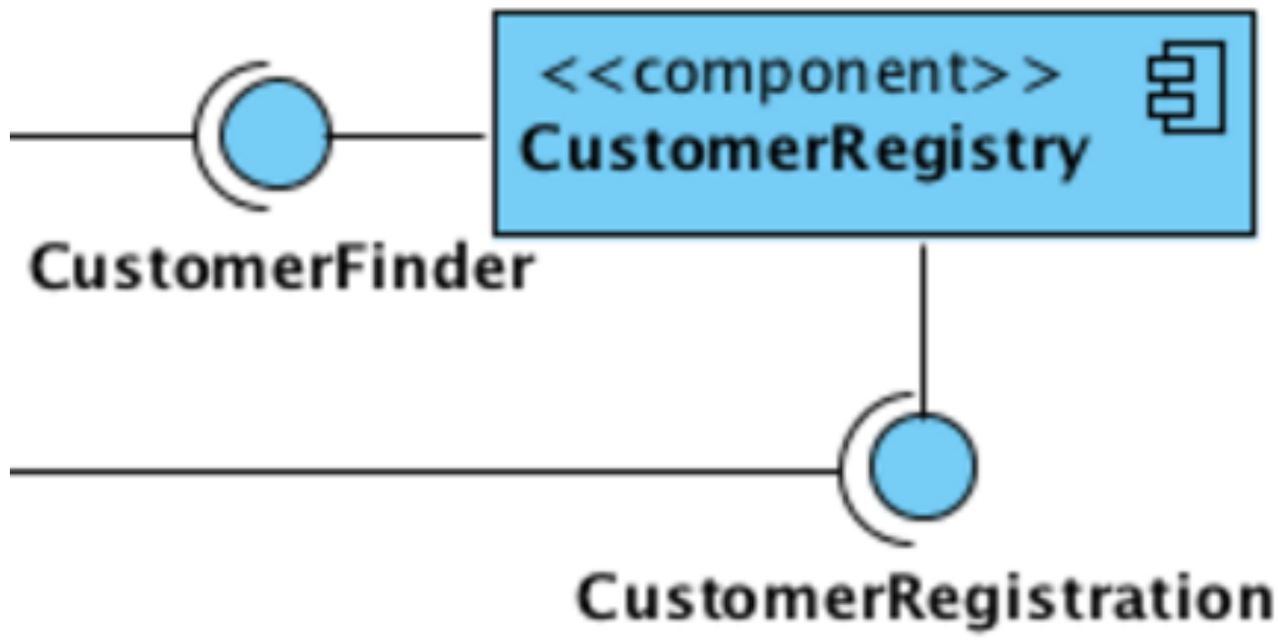


Interface

```java
@Local
public interface CustomerFinder {

    Optional<Customer> findByName(String name);

}
```




```java
@Local
public interface CustomerRegistration {

    void register(String name, String creditCard)
            throws AlreadyExistingCustomerException;

}
```

Inversion of control

```java
@Stateless
public class CustomerRegistryBean
        implements CustomerRegistration, CustomerFinder {

    @EJB
    private Database memory;

    /*****************************************
     ** Customer Registration implementation **
     *****************************************/

    @Override
    public void register(String name, String creditCard)
            throws AlreadyExistingCustomerException {
        if(findByName(name).isPresent())
            throw new AlreadyExistingCustomerException(name);
        memory.getCustomers().put(name, new Customer(name, creditCard));
    }


    /*****************************************
     ** Customer Finder implementation **
     *****************************************/

    @Override
    public Optional<Customer> findByName(String name) {
        if (memory.getCustomers().containsKey(name))
            return Optional.of(memory.getCustomers().get(name));
        else
            return Optional.empty();
    }

}
```

Persistence mock

Questions pour vérifier votre compréhension

Différences entre un composant stateful et un composant stateless ?

Quel est le rôle du conteneur d'EJBs ?

Quelles sont les méthodes que vous pouvez implémenter pour réagir au cycle de vie d'un composant stateful ?

Quelles sont les méthodes que vous pouvez implémenter pour réagir au cycle de vie d'un composant stateless ?

Comment expliqueriez vous l'action de ejbPassivate ?

Webservice et composants stateful sont ils compatibles ?

Persistance et composants stateless sont ils compatibles ?