



# FESA Class Relationships

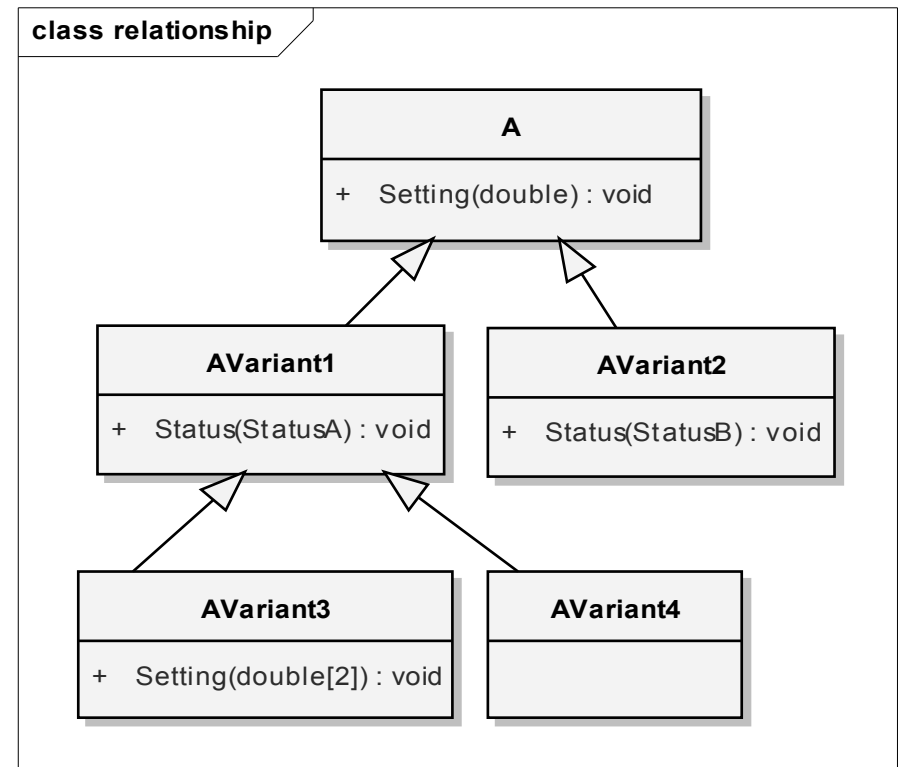
▼ <b>e</b> relationship	(association)
▼ <b>e</b> association	(class-name,
<b>e</b> class-name	class-name
<b>e</b> class-major-version	0
<b>e</b> class-minor-version	0
<b>e</b> class-tiny-version	0
▷ <b>e</b> composition	(class-name,
▷ <b>e</b> inheritance	(class-name,

# Topics

- **Inheritance**
- **Composition**
- **Association**

# Inheritance

- FESA Inheritance definition:
  - Properties/RTActions defined by a base-class are available for any sub-class
  - Properties/RTAction defined in a base-class can be overridden (explicitly)
  - Fields of the base-class can be used in any sub-class
  - Custom-types of the base can be used in any sub-class
- Example:
  - PowerSupplyBase
  - MyPowerSupply



# Inheritance

▼ <input type="checkbox"/> information	((class-name, cl
<input type="checkbox"/> class-name	MyVoltmeter
<input type="checkbox"/> class-major-version	0
<input type="checkbox"/> class-minor-version	1
<input type="checkbox"/> class-tiny-version	0
<input checked="" type="checkbox"/> type	Final
<input type="checkbox"/> state	Concrete
<input type="checkbox"/> description	Abstract
<input type="checkbox"/> fesa-version	Final
<input type="checkbox"/> repository-path	undefined

## Abstract

- cannot be instantiated (no devices)
- used for base-classes

## Concrete

- no restriction, can be instantiated and/or extended

## Final

- Can only be instantiated, cannot be extended

# Inheritance

```
using namespace fesa;
namespace InheritanceTestChild
{
class Device : public InheritanceTestBase::Device
{
    public:
        Device();











        SettingFieldScalar<int32_t> myChildField;

    private:
};
}
#endif
```

generated code

▼ [e] actions	(set-server-action*, get-server-action*)
▼ [e] rt-action	((description*), (triggered-event*))
@a name	MyRTAction
▼ [e] notified-property	
@a property-name-ref	InheritanceTestBase::Status
@a automatic	false

# Inheritance

Node	Content
?-? xml	version="1.0" encoding="UTF-8"
▼  deploy-unit	(include?, information, ownership, class+, sched
 xmlns:xsi	http://www.w3.org/2001/XMLSchema-instance
 xsi:noNamespaceSchemaLocation	file:/common/home/bel/schwinn/lnx/tmp/opt/fesa
▷  include	(class-scheduling-view+)
▷  information	(deploy-unit-name, deploy-unit-major-version, de
▷  ownership	(responsible, creator, editor*)
▷  class	((class-name, class-major-version, class-minor-v
▷  class	((class-name, class-major-version, class-minor-v
▷  scheduler	(concurrency-layer)+
▷  executable	(rt?, server?, mixed?)



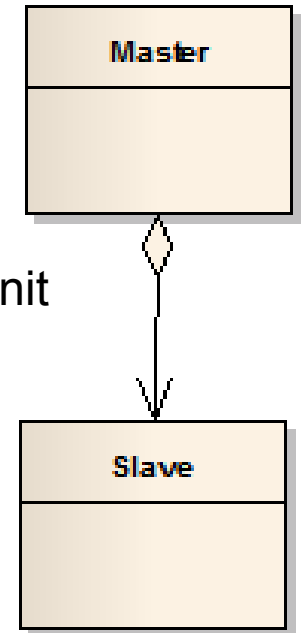
# Topics

- Inheritance
- **Composition**
- Association



# Composition

- Strong coupling ( “Master” can access fields of “Slave” )
- Deployed on a single computer, by the use of one deployment-unit
  - Priority management
  - Reusability of compound-classes
- Example:
  - InterfaceModuleMaster + ChannelCards













# Composition

▼ <b>e</b> relationship	(associ
▼ <b>e</b> composition	(class-
<b>e</b> class-name	Slave
<b>e</b> class-major-version	0
<b>e</b> class-minor-version	1
<b>e</b> class-tiny-version	0

```
namespace Master
{
    void MyAction::execute(fesa::RTEvent* pEvt)
    {
        Slave::Device* slave = this->SlaveServiceLocator_->getDevice("myDevice");
        bool myValue = slave->myField.get(pEvt->getMultiplexingContext());
        this->SlaveServiceLocator_->getDeviceCollection();
        this->SlaveServiceLocator_->getGlobalDevice();
    }
}
```

# Composition

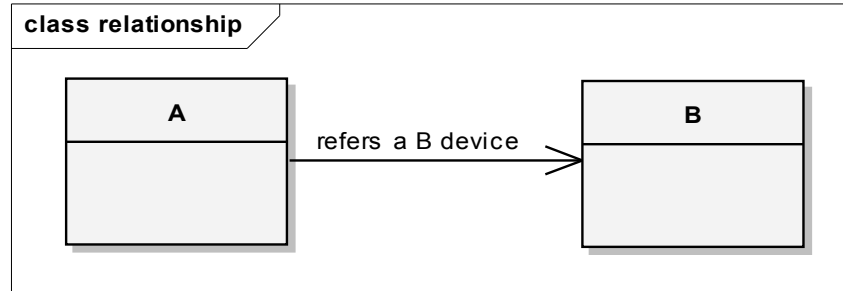
Node	Content
?? xml	version="1.0" encoding="UTF-8"
▼  deploy-unit	(include?, information, ownership, class+, sched
 xmlns:xsi	http://www.w3.org/2001/XMLSchema-instance
 xsi:noNamespaceSchemaLocation	file:/common/home/bel/schwinn/lnx/tmp/opt/fesa
▷  include	(class-scheduling-view+)
▷  information	(deploy-unit-name, deploy-unit-major-version, de
▷  ownership	(responsible, creator, editor*)
▷  class	((class-name, class-major-version, class-minor-v
▷  class	((class-name, class-major-version, class-minor-v
▷  scheduler	(concurrency-layer)+
▷  executable	(rt?, server?, mixed?)

# Topics

- Inheritance
- Composition
- **Association**

# Association

- Light coupling through the Middleware (properties).
- Stand-alone FESA classes running independently.
- Independent lifetime: “A” can shutdown while “B” is still running.
- The classes can be deployed on different computers.
- Example: MyPowerSupply + DataAggregator



# Association

▼ <b>e</b> events	(sources?, logical
▼ <b>e</b> sources	(timing-event-sou
▼ <b>e</b> on-subscription-event-source	(description?)
<b>a</b> name	OnSubscription
▼ <b>e</b> event-configuration	(timing   timer   ONDe
<b>a</b> name	OnSubscriptionConfig
▼ <b>e</b> OnSubscription	(on-subscription-event
▼ <b>e</b> on-subscription-event	
<b>a</b> context	NONE
<b>a</b> device	MyBDevice
<b>a</b> property	MyBProperty

```

void RTOnSubscription::execute(fesa::RTEvent* pEvt)
{
    PayloadOnSubscription_DataType payloadData;

    const OnSubscriptionRTEEventPayload* payload =
        dynamic_cast<const OnSubscriptionRTEEventPayload*> (pEvt->getPayload().get());

    rdaData data = payload->getRDADData();
    payloadData.setData(data, false, false);
}

```

# Mission - Inheritance

- PowerSupplyBase
  - Create an abstract base-class “PowerSupplyBase” by using the GSIClassTemplate
  - Set information/type to “abstract”
  - Generate the source-code + compile
- MyPowerSupply
  - Create a child-class “MyPowerSupply” which inherits from “PowerSupplyBase” ( add relationship/inheritance )
  - Define a RTAction which notifies the property “Acquisition” from the baseclass (automatic notification)
  - Inside the RTAction, set some value to the field “acquisitionContext” of the base-class and print something to the screen
  - Trigger the RT action periodically, once a second
- FESA-Explorer
  - Try to subscribe to the property

On any problem: [fesa-support@gsi.de](mailto:fesa-support@gsi.de)



# Mission

```
for (std::vector<Device*>::iterator device = deviceCol_.begin(); device != deviceCol_.end(); ++device)
{
    try
    {
        int64_t stamp = 12345678;
        (*device)->acquisitionContext.insert(pEvt->getMultiplexingContext(), stamp);
        std::cout << "Base class field 'acquisitionContext' set successfully !" << std::endl;
    }
    catch (...)
    {
        std::cout << "Some error happened in the user-code !!!" << std::endl;
        throw;
    }
}
```