JCore - How to implement a Service Task

Task:

The following document shall explain how an additional Service Task might be implemented.

Realization:

We want to implement an Email Task that automatically sends emails. Therefore we need a class *EmailTaskExecutionBehavior* that extends *TaskExecutionBehavior*.

```
1  package de.uni_potsdam.hpi.bpt.bp2014.jcore;
2
3
4  public class EmailTaskExecutionBehavior2 extends TaskExecutionBehavior {
5
6      public EmailTaskExecutionBehavior2(int activityInstance_id, ScenarioInstance scenarioInstance, ControlNodeInstance controlNodeInstance)
7          super(activityInstance_id, scenarioInstance, controlNodeInstance);
8      }
9
10     @Override
11     public void execute() {
12     }
13 }
14
```

Now we can implement the method execute. This method is going to get triggered every time the Service Task gets activated automatically. In this method we send our email. We can use the attributes from our class to get additional information for the scenario. In our meta model we have defined that the type of our new activity is *EmailTask*. One thing we have to do is to add our new type of Service Task to the following classes:

FragmentInstance.
Here you have to add a new case to the method *initializeNewNodeInstanceForFragment* with your Service Task.

```
private void initializeNewNodeInstanceForFragment() {
    //gets the Start Event and then the following Control Node to initialize it
    int startEvent = dbControlNode.getStartEventID(fragment_id);
    int controlNode = dbControlFlow.getNextControlNodeAfterStartEvent(startEvent);
    String controlNodeType = dbControlNode.getType(controlNode);
    ControlNodeInstance controlNodeInstance = null;
    switch (controlNodeType) {
        case "Activity":
        case "EmailTask":
        case "WebServiceTask":
            controlNodeInstance = new ActivityInstance(controlNode, fragmentInstance_id, scenarioInstance);
            break;
        case "AND":
        case "XOR":
            controlNodeInstance = new GatewayInstance(controlNode, fragmentInstance_id, scenarioInstance);
            break;
    }
    (controlNodeInstance.getIncomingBehavior()).enableControlFlow();
}
```

OutgoingBehavior.
Here it's nearly the same like in *FragmentInstance*. Add the new case for the Service Task to create an ActivityInstance in the method *createControlNode*.

```
protected ControlNodeInstance createControlNode(String type, int id) {
    ControlNodeInstance controlNodeInstance = null;
    //TODO: type
    switch (type) {
        case "Activity":
        case "EmailTask":
        case "WebServiceTask":
            controlNodeInstance = new ActivityInstance(id, fragmentInstance_id, scenarioInstance);
            break;
        case "Endevent":
            controlNodeInstance = new EventInstance(fragmentInstance_id, scenarioInstance, "Endevent");
            break;
        case "XOR":
        case "AND":
            controlNodeInstance = new GatewayInstance(id, fragmentInstance_id, scenarioInstance);
            break;
    }
    return controlNodeInstance;
}
```

ActivityInstance.
Here you have to add the Email Task to the constructor and the method *initActivityInstance*. Take care that you add the correct ExecutionBehavior here and set the automatic execution to true, so that the Service Task runs automatically.

```
public ActivityInstance(int controlNode_id, int fragmentInstance_id, ScenarioInstance scenarioInstance) {
    this.scenarioInstance = scenarioInstance;
    this.controlNode_id = controlNode_id;
    this.fragmentInstance_id = fragmentInstance_id;
    this.label = dbControlNode.getLabel(controlNode_id);
    this.references = dbReference.getReferenceActivitiesForActivity(controlNode_id);
    scenarioInstance.getControlNodeInstances().add(this);
    //creates a new Activity Instance also in database
    this.controlNodeInstance_id = dbControlNodeInstance.createNewControlNodeInstance(controlNode_id, "Activity", fragmentInstance_id);
    switch (dbControlNode.getType(controlNode_id)) {
        case "EmailTask":
            dbActivityInstance.createNewActivityInstance(controlNodeInstance_id, "EmailTask", "init");
            dbActivityInstance.setAutomaticExecution(controlNodeInstance_id, true);
            break;
        case "WebServiceTask":
            dbActivityInstance.createNewActivityInstance(controlNodeInstance_id, "WebServiceTask", "init");
            dbActivityInstance.setAutomaticExecution(controlNodeInstance_id, true);
            break;
        default:
            dbActivityInstance.createNewActivityInstance(controlNodeInstance_id, "HumanTask", "init");
    }
    this.stateMachine = new ActivityStateMachine(controlNodeInstance_id, scenarioInstance, this);
    ((ActivityStateMachine) stateMachine).enableControlFlow();
    this.initActivityInstance();
}
```

```
private void initActivityInstance(){
    this.canTerminate = dbActivityInstance.getCanTerminate(controlNodeInstance_id);
    this.automaticExecution = dbActivityInstance.getAutomaticExecution(controlNodeInstance_id);
    this.incomingBehavior = new TaskIncomingControlFlowBehavior(this, scenarioInstance, stateMachine);
    this.outgoingBehavior = new TaskOutgoingControlFlowBehavior(controlNode_id, scenarioInstance, fragmentInstance_id, this);
    switch (dbControlNode.getType(controlNode_id)) {
        case "EmailTask":
            this.taskExecutionBehavior = new EmailTaskExecutionBehavior(controlNodeInstance_id, scenarioInstance, this);
            this.isAutomaticTask = true;
            break;
        case "WebServiceTask":
            this.taskExecutionBehavior = new WebServiceTaskExecutionBehavior(controlNodeInstance_id, scenarioInstance, this);
            this.isAutomaticTask = true;
            break;
        default:
            this.taskExecutionBehavior = new HumanTaskExecutionBehavior(controlNodeInstance_id, scenarioInstance, this);
            this.isAutomaticTask = false;
    }
}
```

ParallelGatewaySplitBehavior and *ExclusiveGatewaySplitBehavior.*
In these both classes you have to add the case for the new Service Task to the method *setAutomaticExecutionToFalse*, so that the Service Task don't run automatically after an Exclusive Gateway.

```
private void setAutomaticExecutionToFalse(String type, ControlNodeInstance controlNodeInstance) {
    switch (type) {
        case "Activity":
        case "EmailTask":
        case "WebServiceTask":
            ((ActivityInstance) controlNodeInstance).setAutomaticExecution(false);
            break;
        case "AND":
            ((GatewayInstance) controlNodeInstance).setAutomaticExecution(false);
            break;
    }
}
```