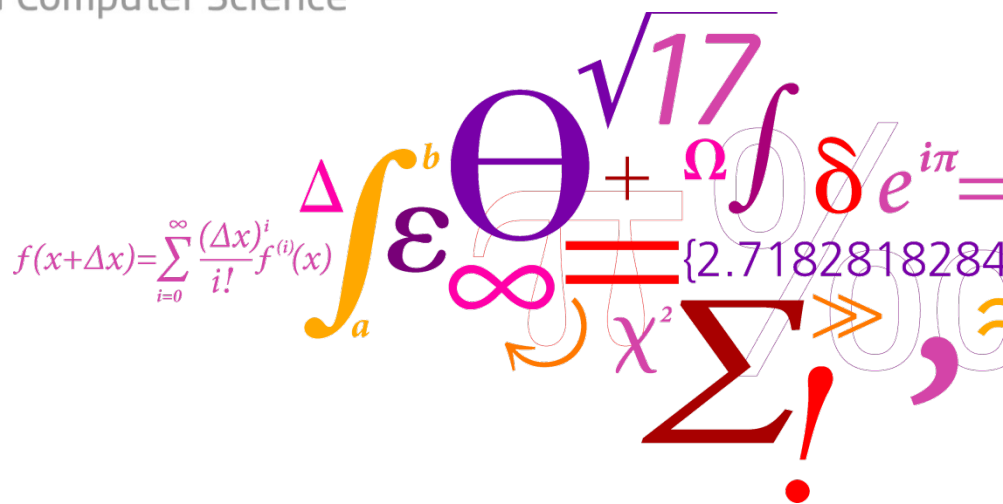


The ePNK: Hands-on / Project

Ekkart Kindler

DTU Compute

Department of Applied Mathematics and Computer Science



runtime-EclipseApplication - /resource/org.pnml.tools.epnk.tutorials.pn-mbse.testit/model/second.pnml/#pg1 - Eclipse Platform

File Edit Diagram Navigate Search Project Run Window Help

Segue UI 9 B I A 125%

Project Explorer

- JETEmitters
- org.pnml.tools.epnk.tutorials.pn-mbse.testit
 - JRE System Library [jdk-12.0.1]
 - Plug-in Dependencies
 - src
 - META-INF
 - model
 - first.pnml
 - second.pnml
 - .classpath
 - .project
 - build.properties

second.pnml Page (id): pg1

int x = 0;

t0: x = 0;

t1: x < 5; System.out.println("x = " + x); x++;

t2: System.out.println("x = " + x);

t4: (unlabeled)

Places: p0, p1, p2

Palette

- Place
- Transition
- Arc
- Page
- RefPlace
- RefTransition
- Label
- Link Label
- Page Label

Tasks Properties Problems Console ePNK: Applications

Label

Core	Property	Value
Appearance	Label	◆ Action System.out.println("x = " + x);x++;
	Object	◆ Transition Code t1
	Text	▣ System.out.println("x = " + x);...

Outline

The screenshot shows the Eclipse IDE interface with a Petri net diagram open in the editor. The diagram consists of places p0, p1, and p2, and transitions t0, t1, t2, and t4. Place p0 contains one token and is labeled with the code `int x = 0;`. Transition t0 is labeled with `x = 0;`. Transition t1 is labeled with `x < 5` and contains an action box with the code `System.out.println("x = " + x);` and `x++;`. Transition t2 is labeled with `System.out.println("x = " + x);`. Transition t4 is labeled with `x++;`. The diagram shows a flow from p0 to p1, then from p1 to t1, t2, and t4, and finally from t1, t2, and t4 back to p2.

A red circle highlights the action box in transition t1. Another red circle highlights the 'Action' property in the Properties view, which is set to `System.out.println("x = " + x);` and `x++;`. The Properties view also shows the 'Transition Code t1' property set to `System.out.println("x = " + x);...`.

The Project Explorer on the left shows the project structure, including the `second.pnml` file. The Outline view at the bottom left shows a smaller version of the Petri net diagram.

runtime-EclipseApplication - /resource/org.pnml.tools.epnk.tutorials.pn-mbse.testit/model/second.pnml/#pg1 - Eclipse Platform

File Edit Diagram Navigate Search Project Run Window Help

Segue UI

Project Explorer

- JETEmitters
- org.pnml.tools.epnk.tutorials.pn-mbse.testit
 - JRE System Library [jdk-12.0.1]
 - Plug-in Dependencies
 - src
 - META-INF
 - model
 - first.pnml
 - second.pnml
 - .classpath
 - .project
 - build.properties

second.pnml Page (id): pg1

```
int x = 0;
```

Diagram elements:

- Place p0: `x = 0;`
- Transition t0: `x = 0;`
- Place p1: `1`
- Transition t1: `x < 5` (highlighted with a red circle)
- Transition t2: `System.out.println("x = " + x);`
- Transition t4: `System.out.println("x = " + x);`
- Place p2: `System.out.println("x = " + x);`

Palette:

- Place
- Transition
- Arc
- Page
- RefPlace
- RefTransition
- Label
- Link Label
- Page Label

Tasks Properties Problems Console ePNK: Applications

Label

Core	Property	Value
Appearance	Label	◆ Action System.out.println("x = " + x);x++;
	Object	◆ Transition Code t1
	Text	System.out.println("x = " + x);...

Outline

The screenshot shows the Eclipse IDE interface with a Petri net diagram open in the editor. The diagram consists of three places (t0, p1, p2) and three transitions (t1, t2, t4). Place t0 contains one token and is connected to transition t1. Transition t1 has a guard condition $x < 5$ and an action `System.out.println("x = " + x); x++;`. Transition t2 has an action `System.out.println("x = " + x);`. Transition t4 is connected to places p1 and p2. Place p1 contains one token. A red circle highlights the text `int x = 0;` in the editor, and a red box labeled "Declaration" points to it. The Properties view at the bottom shows the details for the selected transition.

Property	Value
Core	
Appearance	
Label	
Object	◆ Action System.out.println("x = " + x);x++;
Text	◆ Transition Code t1 System.out.println("x = " + x);...

- Implement a new Petri net type extending PTNets (a PNTD for what we call PNCode) by adding
 - Action(label)s to transitions
 - Condition(label)s to transitions
 - Declaration(label)s to pages
- These concepts should be text in Java syntax
- For now, it is not necessary to check syntactical correctness of this syntax
- A basic project with a set up of a simple model is provided to you (see slides on details later)

runtime-EclipseApplication - org.pnml.tools.epnk.tutorials.pn-mbse.testit/model/second.pnml - Eclipse Platform

File Edit Navigate Search Project ePNK Editor Run Window Help

Project Explorer

- JETEmitters
- org.pnml.tools.epnk.tutorials.pn-mbse.testit
 - JRE System Library [jdk-12.0.1]
 - Plug-in Dependencies
 - src
 - META-INF
 - model
 - first.pnml
 - second.pnml
 - .classpath
 - .project
 - build.properties

second.pnml Page (id: pg1)

- Resource Set
 - platform:/resource/org.pnml.tools.epnk.tutorials.pn-mbse.testit/model/second.pnml
 - Petri Net Doc
 - PN Code
 - Name Second
 - Page Code pg1

Save Java code...

Please select a folder to which the generated java code should go.

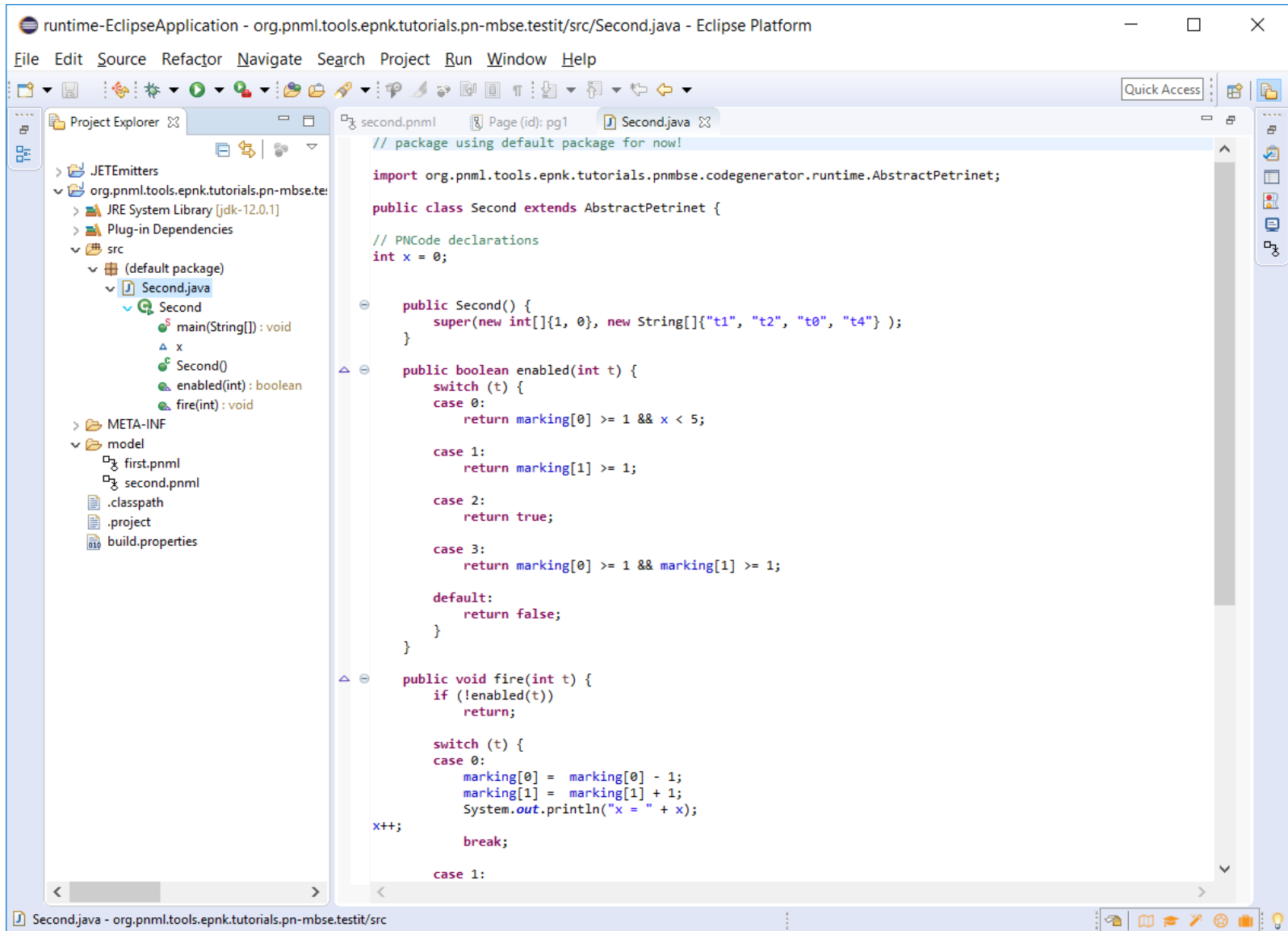
- JETEmitters
- org.pnml.tools.epnk.tutorials.pn-mbse.testit
 - bin
 - META-INF
 - model
 - src

New Folder...

OK Cancel

Selected Object: PN Code

Right-click (on type PNCode):
→ ePNK
→ Generate Java Code from PNCode



```
// package using default package for now!  
  
import org.pnml.tools.epnk.tutorials.pnmbse.codegenerator.runtime.AbstractPetrinet;  
  
public class Second extends AbstractPetrinet {  
  
    // PNCode declarations  
    int x = 0;  
  
    public Second() {  
        super(new int[]{1, 0}, new String[]{"t1", "t2", "t0", "t4"});  
    }  
  
    public boolean enabled(int t) {  
        switch (t) {  
            case 0:  
                return marking[0] >= 1 && x < 5;  
  
            case 1:  
                return marking[1] >= 1;  
  
            case 2:  
                return true;  
  
            case 3:  
                return marking[0] >= 1 && marking[1] >= 1;  
  
            default:  
                return false;  
        }  
    }  
  
    public void fire(int t) {  
        if (!enabled(t))  
            return;  
  
        switch (t) {  
            case 0:  
                marking[0] = marking[0] - 1;  
                marking[1] = marking[1] + 1;  
                System.out.println("x = " + x);  
                x++;  
                break;  
  
            case 1:  

```


Right-click:
→ Run As
→ Java Application

```
import org.pnml.tools.epnk.tutorials.pnmbse.codegenerator.runtime.AbstractPetriNet;  
  
public class Second extends AbstractPetriNet {  
  
    // PNCode declarations  
    int x = 0;  
  
    public Second() {  
        super(new int[]{1, 0}, new String[]{"t1", "t2", "t0", "t4"});  
    }  
  
    public boolean enabled(int t) {  
        switch (t) {  
            case 0:  
                return marking[0] >= 1 && x < 5;  
  
            case 1:  
                return marking[1] >= 1;  
  
            case 2:  
                return true;  
  
            case 3:  
                return marking[0] >= 1 && marking[1] >= 1;  
  
            default:  
                return false;  
        }  
    }  
  
    public void fire(int t) {  
        switch (t) {  
            case 0:  
                marking[0] -= 1;  
                marking[1] += 1;  
                x++;  
  
            case 1:  
                marking[1] -= 1;  
  
            case 2:  
                marking[0] += 1;  
  
            case 3:  
                marking[0] -= 1;  
                marking[1] -= 1;  
                x++;  
  
            default:  
                return;  
        }  
    }  
  
    public void main(String[] args) {  
        Second s = new Second();  
        s.run();  
    }  
}
```

Second: 1
Stop t1 t2 t0 t4

x = 0
x = 1

- Implement an action that generates Java code from such PNCode running as a Java application
- The GUI and “runtime” environment as well as all to the set up the action and initiate the code generation is provided to you
- Also a generator template for the basic Petri net (not taking actions, conditions and declarations into account) is provided to you
- You can focus on **extending the template and skeleton for generating the code for actions, conditions and declarations**