|  |
| --- |
|  |
| JPeNS – Group 5 |
| Java Petri Network Simulator |
|  |
| **Chris Hobbs** |
| **Jonathan Babbitt** |

April 7, 2012

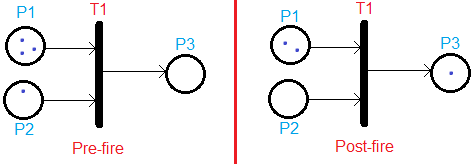
|  |
| --- |
|  |

**What is JPeNS**

Our Java Petri Network Simulator, or JPeNS for short, is a program that aids in the visual representation and learning of Petri Networks in a visual way. It is meant to give the user both a hands-on and visual experience while learning about these networks. It is designed to be interactive and displays real-time feedback when you fire a transition.

Petri Networks are a way to visual represent or layout a descriptive network for a trigger or action based system. These networks are drawn as directed bipartite graphs. Bipartite means that the nodes in the graph can be divided into two disjoint sets with edges connected nodes in opposite sets but not connecting to nodes within the same set. A directed graph means that the edges go in one direction, like one way streets. In the case of Petri Network diagrams, the places and transitions are split into the two sets. That is to say, a place can only connect to a transition and a transition can only connect to a place. Places cannot connect to places and transitions cannot connect to transitions. However, multiple places can connect into one transition and one transition can connect out into multiple places. In a Petri Network graph the places are represented as circles and the transitions are represented as vertical bars.

The actions of the Petri Network function based off of tokens located in the places. Each transition requires one token from each precondition place to fire. If one or more of the preconditions does not have a token, the transition cannot fire. After the transition is fired, one token is generated in each of the post-condition places. The preconditions for a transition are all the places with directed edges (arrows) coming from the place into the transition. The post-conditions for the transition are all the places with directed edges coming from the transition into a place. [See figure 1.1]

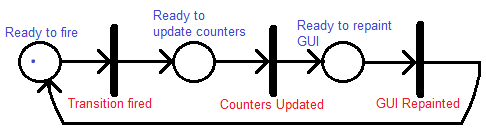


**Figure 1.1**

In the pre-fire state you can see that P1 contains three tokens, P2 contains one token, and P3 is empty. P1 and P2 are the preconditions for T1 and P3 is the post-condition. The post-fire state shows that T1 takes one token from both P1 and P2 and places one token in P3. This network cannot fire again as P2 is now empty so T1’s preconditions are no longer met.

JPeNS enables an easy way to create and distribute Petri Networks. It is very simple to make an XML file that defines the desired network and the file can be easily sent to anyone who wishes to view it. All they would need it JPeNS and the XML file; from there they can import it and view the network. This would be ideal for educating people on Petri Networks as the instructor could distribute premade XML files to the students for interactive examples.

**Why JPeNS**

 We chose our topic as Petri Networks because they are a very interesting and useful topic. They have a wide range of applications such as software design, workflow management, process modeling, data analysis, diagnosis, and simulation. See figure 1.2 as an example of a Petri Network for software design.

**Figure 1.2**

This is an example of a Petri network depicting software design. This network is the basic structure of how JPeNS’ simulation mechanism works.

As you can see, Petri Networks are a priceless tool when it comes to explaining, planning, or diagraming action based systems. However, even though it is a valuable tool, it can be a difficult one to learn. For people who do not have a background in computing or engineering typically have a more difficult time grasping this concept. Since Petri Networks have such a broad range of possible uses, it’s a shame that it may be under utillised in other fields. This is where JPeNS comes in. It will make the education and understanding of this topic more intuitive so that more people may be inclined to learn and use Petri Networks in their field.

**System Description**

**Results**

**Conclusion**