Assignment 1 — Trains

T-106.420 Concurrent programming

Jan Lönnberg

Department of Computer Science
Helsinki University of Technology

18th October 2005

Introduction

The Story

- A small railway company wants to automate their train control.
- You have been tasked with implementing the train control software.
- Proof of concept: write control software for two trains on a simulated track.

Goal

- Write a control program for the trains that lets them travel between two stations safely.
- Trains must be independent; each train is controlled by a separate thread.

Task

You get:

- A simulator called tsim.
- A track with two trains.
- Java interface code for tsim.
- Example control code that drives a train from one station to another.

You should:

- Add sensors to the track.
- Write control code that drives two trains back and forth between two stations.
- Describe your control code in a brief report.

Task

Doing the assignment

- Group size: 1–2.
- Submission before 2005-11-16 03:00.
- Submit a tar.gz archive containing:
  - Control code source.
  - Track with added sensors.
  - Report in PDF form.
- Full instructions will be on the course home page.
- Grading: fail/pass/pass with honours.

Using tsim

The tsim simulator

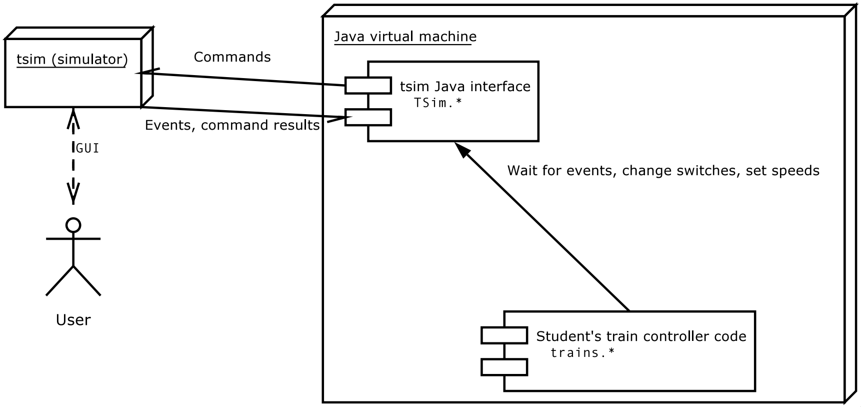
- tsim is a simple train simulator.
- GUI allows you to:
  - Edit tracks and trains.
  - Control trains.
  - View simulation.
- Trains can be controlled through an external program.

Controlling tsim with another program

Connecting tsim to a control program

- tsim connects to a control program using standard input and output.
- You write control code in Java.
- Control code accesses tsim through the Java tsim interface (`TSim.*`).
- Control code has one thread per train.

Controlling tsim with another program



Controlling tsim with another program

Java interface

- Trains controlled through class `TSim.TSimInterface` (get one instance from `TSim.TSimFactory.getTSimInterface()`):
  - `public void setSpeed(int trnId, int speed)`
    - Set train speed (measured in pixels/second).
  - `public void setSwitch(int xPos, int yPos, int switchPos)`
    - Set the state of the switch at `(xPos, yPos)` to `switchPos` (`TSimInterface.SWITCH_LEFT` or `TSimInterface.SWITCH_RIGHT`).
  - `public SensorEvent getSensor(int trnId)`
    - Waits for a sensor event from train `trnId`.
    - Generated when train enters or exits a sensor square.
    - Sensor events: sensor position, event type and train id.

Running tsim

Getting tsim

- tsim (and associated code) can be downloaded from the assignment page.
- Requires a Unix-like system with X (e.g. Linux, Windows with Cygwin).
- Preinstalled on Niksula.

Running tsim

Starting tsim alone (for editing)

- `cd tsim-0.7; ./tsim ../track`
- `~jlonnber/tsim/tsim track` in Niksula.

Starting tsim with a control program

- tsim includes a program called 2 that starts two programs and connects their standard input and output.
- `./2 "cd tsim-0.7; ./tsim ../track-sensors" "java trains.Train"`
- `~jlonnber/tsim/run` in Niksula.

Semaphores

Semaphores

- Semaphores are the traditional synchronisation mechanism for trains.
- Your train code should use semaphores for communication between trains.
- You will use the `sync.Semaphore` class with the following operations:
  - `public void acquire()`
  - `public void release()`
  - `public boolean tryAcquire()`

Conclusion

Conclusion

- Assignment is intended to help learn mutual exclusion, semaphores, basics of threads in Java and interprocess communication.
- Assignment description will be linked from course home page.
- Technical questions and clarification requests to the newsgroup.
- Clarifications (if any) will be posted to the newsgroup.
- Questions not intended for the public to `jlonnber@cs.hut.fi`.