Course 02158 Concurrent Programming
Fall 2011, DTU Informatics

Solution to the mandatory assignment

Due Date : 16/11/2011

Group 19

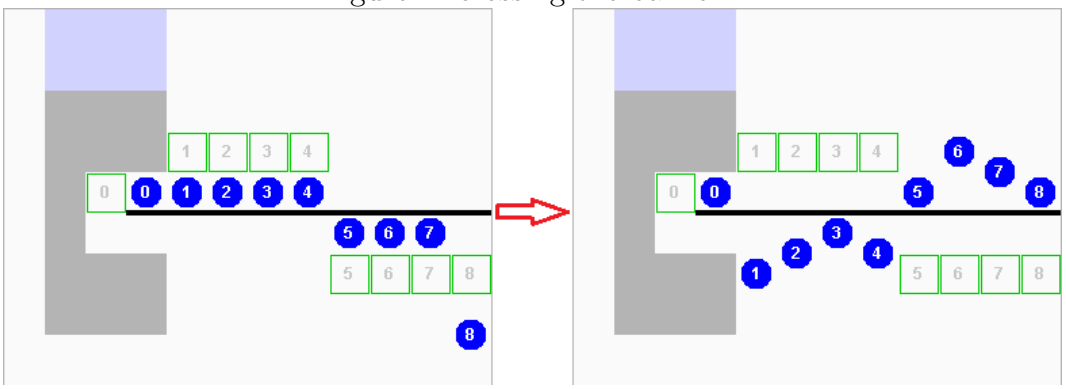
Jakub Judas s110845
Nicolas Silva s111781
Robin Deléarde s111873

1 Introduction: the cars control problem

2 Step 3: the Barrier

The purpose of this step is to implement a barrier on the playground in order to prevent the cars from making more rounds than the others. This is done by making every car stop at a barrier line, and wait for all the others before starting a new round. This synchronization issue can be solved with semaphores, which make it possible to check if all the cars are present at the barrier, and to make them wait while that's not the case, since a car is nothing more than a thread.
In order to stop those Car-threads, each car is assigned a semaphore, whose value is 0 while the car is not allowed to pass the barrier (the starting value is 0), and 1 either, thanks to a V operation triggered by the last car awaited. Those 9 semaphores are stored in a table called **barrier**.

Figure 1: crossing the barrier



A barrier is defined by the Barrier class as:

- a table of 9 semaphores as described previously;
- a boolean **ison** to control the state of the barrier;
- a semaphore **access** and a boolean **pass**, used in the method *sync* responsible for the access control at the barrier.

Actually this last semaphore acts as a counter allowing 7 cars to pass (the initial value is 7) and stopping the 8th one. Associated with the boolean **pass** whose value tells us where a car has been stopped in the program (i.e. by which semaphore: its own or the **access** one), it allows us to know when there's only one car left running on the playground. Then this car can release

all the others when it arrives at the barrier.
The code of the synchronization class is given below, where **no** is the number of the car reaching this step:

```
public void sync(int no) {
   if (!pass) {
   pass = true;
   //only 7 cars may access this zone, the 8th one is stopped here
   try { access.P(); } catch (InterruptedException e) {}
   pass = false;
   //the cars are stopped by the barrier
   try { barrier[no].P(); } catch (InterruptedException e) {}
   }
   //the last car may pass here and releases everyone
   else {
   pass = false;
   for (int i=0; i<9; i++) {
   if (i!=no) {
   access.V();
   barrier[i].V();
   }
   }
   }
}
```