

# Meilstein Omega

Antoine Scheffold 2556024  
Roman Tabachnikov 2565209

July 2018

## 1 EDIT

As was noted in our last meeting - our implementation of `reserveRoute` in `Trainservice` was lacking; the tryLocks on route reservation for the parking and the lock on the method itself lead to a bottleneck. We solved this problem by simply writing a new method; `reserveRoute2` that uses direct locks (instead of the trylocks) and avoids the lock on method. Additionally, the unlocks that lead to the bottleneck by using the signal - we deleted the signal to avoid this confusion.

```
...
//VERYIMPORTANTEDIT
void waitingforReservedRoute(List <Connection> connections, Location currentLocation, int id)
    throws InterruptedException {
    reserveRoute2(connections, currentLocation, id);

    //DEPRECATED
    //lock.lock();
    //waitingRouteFree.await(10, TimeUnit.MILLISECONDS);
    //lock.unlock();

}

//VERYIMPORTANTEDIT
boolean reserveRoute2(List <Connection> connections, Location currentLocation, int id){
    List <Connection> alreadyReservedConnection = new LinkedList<>();
    List <Location> alreadyReservedLocation = new LinkedList<>();
    //get all locations on the route
    List <Location> locationsToReserve = locationsOnRoute(connections, currentLocation);

    int[] connectionsIds = new int[connections.size()];
    int[] locationIds = new int[locationsToReserve.size()];
    int i = 0;
    //get all ids of the asked connections
    for(Connection c : connections) {
        connectionsIds[i] = c.getRomanAndAntoineID();
        i++;
    }

    i = 0;
    //get all ids for the asked locations
    for(Location l : locationsToReserve) {
        locationIds[i] = l.getRomanAndAntoineID();
        i++;
    }
    //sort the ids in ascending order
    Arrays.sort(connectionsIds);
```

```

        Arrays.sort(locationIds);

        //FORCE(instead of try) to lock all connections on the route in ascending order of their ids
        for(i = 0; i < connectionsIds.length; i++) {
            allConnections.get(connectionsIds[i]-firstConnectionId).getLock().lock();
            //previously we used a signal that would wake the sleeping Thread on each unlock
        }
        //FORCE(instead of try) to lock all locations on the route in ascending order of their ids
        for(i = 0; i < locationIds.length; i++) {
            allLocations.get(locationIds[i]-firstLocationId).getLock().lock();
            //previously we used a signal that would wake the sleeping Thread on each unlock
        }
        return true;
    }
    ...

```

Next we solved the new exceptions-tests by adding a new catch `AssertionError` case in `Simulator` as-well catching the case of empty schedules and connections/locations. The new `idsNotStartingAtZeroAndNotContinuous` forced us to write our own implementation of `ids` both for the `Location` and the `Connection` (`romanAndAntoineID`) to avoid the pitfall of badly initialized `Position` elements - their getter and setter was changed accordingly.

```

    ...
    public static boolean run(final Problem problem, final Recorder recorder) {
    ...
        //IN SIMULATOR
        //VERYIMPORTANTEDIT
        if(schedules.isEmpty() || map.locations().isEmpty() || map.connections().isEmpty()) {
            try {
                recorder.done();
                return true;
            }catch (Exception e) {
                print("empty problem, and calling recorder was bad");
                return false;
            }
        }
        //VERYIMPORTANTEDIT
        catch (AssertionError assss) {
            print("wtffffffff!!!! ");
            return false;
        }
    }
    ...
    //initialize locations & connections
    int ourID = 0;
    for( Location l : map.locations()) {
        l.setRomanAndAntoineID(ourID);
        ourID++;
    }
    ourID = 0;
    for (Connection c : map.connections()) {
        c.setRomanAndAntoineID(ourID);
        ourID++;
    }
    ...
}
...
//IN LOCATION/POSITION
//VERYIMPORTANTEDIT
public int getRomanAndAntoineID() {
    return romanAndAntoineID;
}

```

```

}

public void setRomanAndAntoineID(int romanAndAntoineID) {
    this.romanAndAntoineID = romanAndAntoineID;
}

private int romanAndAntoineID;

...

```

Finally we fixed the elusive error on `parkingSimple` by adjusting our `reserveRoute` in `Trainservice` as was discussed in our last meeting; `reserveRoute` now returns a `Position` if there's no way to reserve the route, otherwise (on a successful reserve) we return `null`.

```

/**
 * Can be run by multiple Threads! i.e. Trains may ask to reserve routes while others are trying
 * On success reserve will lock all connections and locations on the route
 * @param connections of the asked route
 * @param currentLocation of the asking Train
 * @param id of the asking train (debugging info)
 * @return Position, if failed, or {@code null} wenn reserved
 */
//VERYIMPORTANTEDIT
Position reserveRoute(List <Connection> connections, Location currentLocation, int id){

```



Figure 1: Sincerely, Your Train-managers; Antoine and Roman