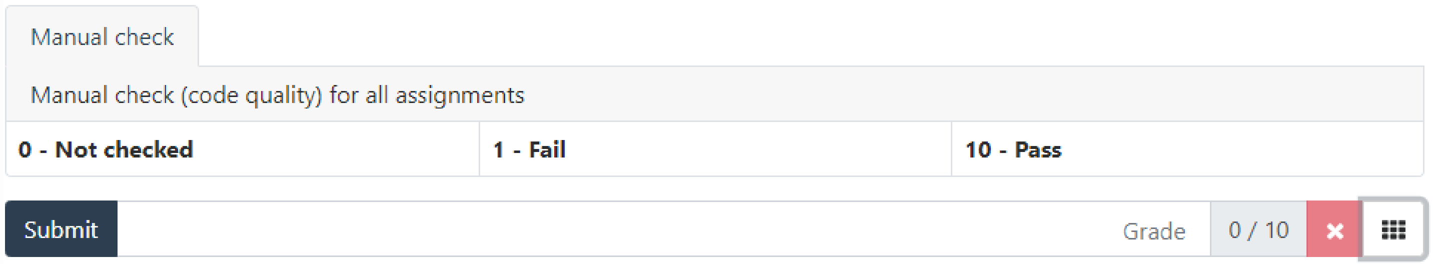
Week 3 assignments

When creating the program code, you must apply the following basic principles:

- create a separate project for each assignment;
- use name ‘assignment1’, ‘assignment2’, etcetera for the projects;
- create one solution for each week containing the projects for that week;
- make sure the output of your programs are the same as the given screenshots;

CodeGrade auto checks

This week there are no CodeGrade auto checks. The manual check will be done by the practical teacher.



Manual check (code quality) for all assignments

0 - Not checked	**1 - Fail**	**10 - Pass**

Assignment 1 ('MVC')

You probably have seen those train displays before, informing travellers which station will be next, at what time and which railway track; in each wagon there are a few of them. We will implement a system with a simple (Windows Forms) application with a few of these displays (Views), together with the necessary datastructure (Model) and a controlpanel (using a Controller). With the controlpanel the train driver can indicate that the next station has been reached. Of course, in the real world, this is done automatically with sensors in the train and on the stations, but to be able to test it without hardware, we wil use a simple controlpanel with a few buttons.

The application runs inside a train that continuously goes back and forth, for example between Den Helder and Nijmegen (you can find all stations of this journey at www.ns.nl).

Model

As model, create a class TrainStation (name, arrival track, arrival time, departure time), and a class TrainJourney (containing a list of (hardcoded) stations and the current station). Every time the current station changes, all views must be updated.

Of course there are other properties/methods needed in class TrainJourney (that implements interface 'ITrainJourney'), like AddObserver and RemoveObserver.

Controller

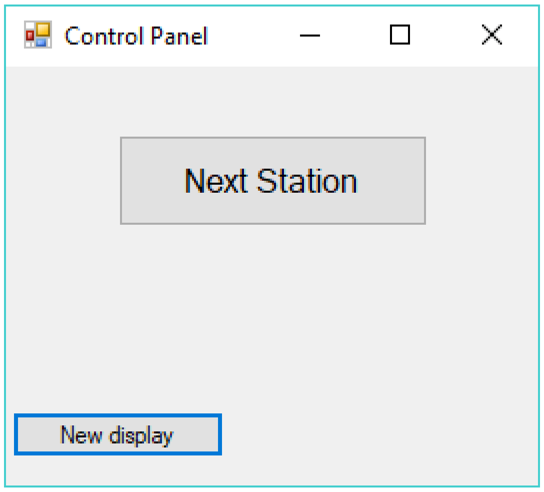
Create an interface 'ITrainController' and a class 'TrainController' (implementing the interface). Pass on a trainjourney instance (use the interface!) to the controller so the controller can pass on all actions to the (model) trainjourney.

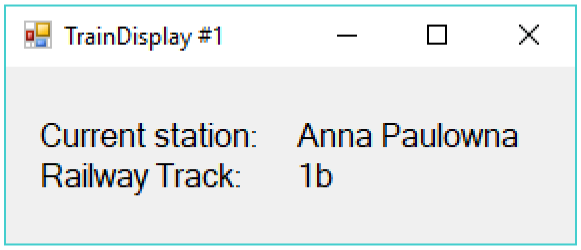
You can use the (default) main form of a Windows Form application for the control panel. The train driver can use this controlpanel to indicate that the next station has been reached, and that the return journey has started (after the end station has been reached). When the return journey starts, reverse the list of stations, or process the stations backwards. The controlpanel (main form) must communicate only with the Controller, not with the model (train journey)!

Views

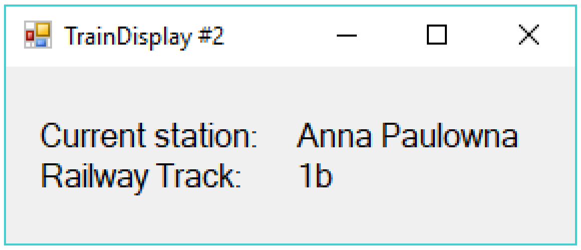
Create a separate Windows Form for the View; several views can be created from this form (all displaying the same information) with: 'ITrainDisplay display1 = new TrainDisplay()'. Display information about the next station: the name of the station, the expected arrival time, and the railway track (where the train will arrive). The information on the screen is only updated when the model (the train journey) changes.

Optionally: create a second kind of display that shows a list of all remaining stations, until the end station of the journey. *(you could check during your next train trip, how this information is displayed)*

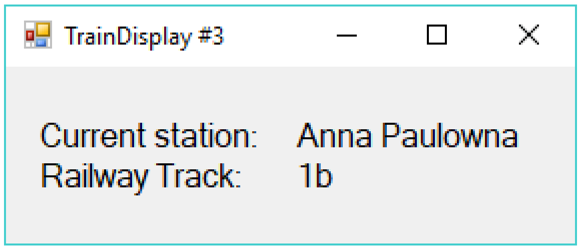




Current station: Anna Paulowna
Railway Track: 1b



Current station: Anna Paulowna
Railway Track: 1b



Current station: Anna Paulowna
Railway Track: 1b