Exercise 16: New implementation of the `Time` class

On the webpage, you can find a `Time` class. Re-implement the `Time` class such that a point in time is represented as a number of minutes since midnight (i.e. only one field is needed in the class: **private int** `min`). Change the constructor and the methods accordingly.

On the webpage, you can also find a program, `TestTime`, which you can use for testing your implementation of `Time`.

Exercise 17: Representing dates

Make a class `Date` for representing dates. The class should have a single constructor:

```
Date(int year, int month, int day)
```

`Date(2000,3,9)` creates a new `Date` object with the date 9th of March, 2000.

Think about which fields the `Date` class should have.

The `Date` class should contain the following methods (all **public**):

`String iso()`
shows the date in ISO format (e.g. 2000–03–09).

`String danish()`
the date in the usual danish format (e.g. 9/3 2000).

`String danishText()`
the date with the name of the month instead of a number (e.g. 9. marts 2000)

Write a small program that tests the class.

Exercise 18: Representing appointments

The concept of an appointment:

- an appointment has a start time and an end time (the same day)
- an appointment has a text that describes the appointment, e.g. ”lecture”
- the start time, end time, and text must be given
- an appointment may be prolonged by a number of minutes (i.e. the end time is delayed)
- an appointment may be postponed a number of minutes (i.e. both the start time and the end time is delayed the same number of minutes)
- an appointment may be formatted as a string of characters

Make a class for representing appointments. Looking at the above description, consider which fields and methods the class should have. **Hint:** Use the `Time` class.

Also write a small program that tests the class.