In this exercise we study basic principles of object oriented programming.

Excercise 2 (Using a class 1p)

We need to write an application, that reads two times (times are represented by two int numbers: hours and minutes). Then the program finds out which time is later time. After that it calculates the time difference between these times. Finally the program displays the smaller (earlier) time and the time difference (duration) in the format

```
starting time was 11:22
duration was     1:04
```

The main function that does these things looks as follows:

```
int main(void) {
    Time time1, time2, duration;
       time1.read("Enter time 1");
       time2.read("Enter time 2");
       if (time1.lessThan(time2)) {
     duration = time2.subtract(time1);
     cout << "Starting time was ";
     time1.display();
    }
    else {
     duration = time1.subtract(time2);
     cout << "Starting time was ";
     time2.display();
    }
    cout << "Duration was ";
    duration.display();
    return 0;
}
```

Now you need to define and implement class Time so that the program becomes working. As can be seen from the main function Time has the following member functions:
read that is used to read time (minutes and hours) from the keyboard.

1. `lessThan` that is used to compare two times.
2. `subtract` that is used to calculate time difference between two times.
3. `display` that is used to display time in the format hh:mm.

Extra exercises (Counter, Dice and Inspector, 0,25p)

We used counter, dice and inspector objects in our example program that demonstrated object oriented thinking. The program generates 100 random numbers between (0.0, 1.0]. It counts how many of those numbers are between (0.0, 0.5] and how many between (0.5,1]. The program we used is as follows

```
void main(void) {
     Dice dice;
```

```
          Counter counter1, counter2, i;
          Inspector inspector;
          dice.initialize();
          counter1.reset(); counter2.reset(), i.reset();
          inspector.setLimits(0.0, 0.5);
          while (i.getCount() < 100) {
               if (inspector.isInLimits(dice.roll()))
                    counter1.increment();
               else
                    counter2.increment();
               i.increment();
          }
          cout << counter1.getCount() << endl;
          cout << counter2.getCount() << endl;
     }
```

No class definitions nor the class implementations were given. Do them now so that the main function above becomes a working program

Remark 1. You can use the main function presented in the class as such and only define and implement the three classes: Dice, Counter and Inspector so that the program is really working and gives correct results on the screen.

Remark 2. You don't have to implement constructors in this phase.