# **Practical work No.1 Project structure**

## Project Overview

The practical works represent the development of a single project, its evolution from a simple class to a full-featured user application. The work topics include tasks that a user can create for himself/herself, such as "Bowling with friends on Wednesday", or "Running 3 km every day at six o'clock". The project that is the subject of these practical works is an application intended to help the user store, view, and remind to complete the tasks.

## Objects: Tasks

The main objects that the application will work with are tasks.

Tasks have some text that describes the *details of the task*, such as "Cleaning the room."

In addition, tasks can be *active or inactive* . For example, during the holiday, task "Morning run" may be inactive and temporarily not performed.

Integers will be used to describe the start time, indicating, for example, the number of hours that have elapsed since the start of the countdown (for example, from 00:00 on February 1, 2000), and the number 36 will mean 12:00 on February 2, 2000.

Tasks can be scheduled to be performed *once*, for example "Meeting in a cafe on June 26 at 18:00":

Time

“Meeting in a café”

0

June 26

Or the task can be scheduled to be performed *regularly* over a period of time at a given interval (in hours), such as "Morning run from June 1 to June 5 every day at 8:00":

Time

"Morning run"

0

Start time: 08:00   
June 1

End time: 00:00   
June 5

Repetition interval 24 hours

Thus, the goal of the first practical work will be to create a Task object class.

**Details on performing the work**

When performing this work, you must design your class files as described in the Java Code Conventions, which can be found at <https://www.oracle.com/technetwork/java/codeconventions-150003.pdf>. In addition, all public elements must have javadoc, which briefly explains the purpose and use of the element. When writing such documentation, it is not necessary to describe things that are easy to understand from the declaration, such as types of parameters and results. It is necessary to specify non-obvious things: restrictions on parameter values, special resulting values, possible exceptional situations.

## Task 1. Creating a base class

Create a Task class in the ua.sumdu.j2se.studentName.tasks package (replace studentName with your name) using the following public methods:

* The Task (String title, int time) constructor constructs an inactive task to run at a specified time without repeating with a given name.
* The Task (String title, int start, int end, int interval) constructor constructs an inactive task to run within the specified time range (including the start and the end time) with the set repetition interval and with a given name.
* Methods for reading and setting the task name: String getTitle(), void setTitle(String title).
* Methods for reading and setting the task status: boolean isActive(), void setActive(boolean active).
* Methods for reading and changing execution time for non-repetitive tasks:
  + int getTime() – if the task is a repetitive one, the method must return the start time of the repetition;
  + void setTime(int time) – if the task was a repetitive one, it should become non-repetitive.
* Methods for reading and changing execution time for repetitive tasks:
  + int getStartTime() – if the task is a non-repetitive one, the method must return the start time of the execution;
  + int getEndTime() – if the task is a non-repetitive one, the method must return the end time of the execution;
  + int getRepeatInterval() – if the task is a non-repetitive one, the method must return 0;
  + void setTime(int start, int end, int interval) – if the task is a non-repetitive one, it should become repetitive.
* To check the task for repeatability use the boolean isRepeated() method.

Compile the project. It should be compiled without compilation errors, and preferably without design errors.

## Task 2. Checking the next task execution

An important operation when working with tasks is to find the next moment of the task execution.

Yes, if the task is not active, it is never executed, if the task is active and executed only once, then the next moment of execution will be either this single moment, or never (if the task has already been completed):

Time

"Meeting in a cafe"

0

June 26

Regarding this moment,   
the next execution is on June 26

Regarding this moment,   
the task will not be executed anymore

June 23

June 27

If the task is active and repetitive, then similarly the next moment of the task execution is found relative to the given time:

Time

"Morning run"

0

Start time:  
 June 1

End time:  
 August 1

Repetition interval: 24 hours

Regarding these moments, the following ones are marked

Regarding this moment,   
the task will not be executed anymore

You should add the int nextTimeAfter (int current) method to the Task class that returns the next start time of the task execution after the current time. If after the specified time the task is not executed anymore, the method must return -1.