Tasksheet to practice use of TUDaExercise class

Marei Peischl

TECHNISCHE UNIVERSITÄT DARMSTADT

Spring 2042
Sheet 5

Task 5.1: Title

TUDaExercise does not support the `titlepage` option. To save space the setting is enforced to `titlepage=false`. The title material supports some additional commands, a full example is shown below:

```
\title[Short title to be used in running headers]{Title}
\subtitle{Subtitle}
\author{Author/Professor}
\term{Term}
\date{Date}
\sheetnumber{Sheetnumber}
```

All fields but the title may be kept empty. In case no sheetnumber is used the tasks will be numbered starting at 1.

Task 5.2: Coloring

The coloring uses the same mechanisms as the other classes of the TUDa-CI bundle. The options `color`, `accentcolor`, `textaccentcolor`, `identbarcolor` work as usual [see 1]. It is also possible to deactivate the colored title block. The option `colorback=false` makes it possible to set the toner-saving variant for versions for printing

Task 5.3: Create a task

Tasks are defined in tudaexercise using the `task` environment. The required argument is used as a title. It may remain empty.

```
\begin{task}[<options>]{title}
  Text describing the task
\end{task}
```

Cross-references between the tasks are possible as usual (`\label` and `\ref`).

Task 5.4: Subtasks

5.4a) Basic structure

tudaexercise provides the `subtask` environment to subdivide tasks into subtasks. Currently inly one level is supported. For further subdivision, it is also possible to divide the document into sections using `\section`.

```
\begin{subtask}[title={title},points=5]

\end{subtask}
```

In contrast to tasks, subtasks do not require a heading. Therefore, the syntax for the transfer differs from that of the task environment. However, there is a variant to avoid this distinction, see task 5.10.

5.4b) Layout modification of the headings

From version 3.13, tudaexercise also supports switching the layout of subtask headings. There are three option values for the document class: subtask=ruled subtask=plain subtask=runin If very small tasks are set, the break after the title can be prevented . In this case, the user is responsible for ensuring that any necessary visual separation between the title and task text takes place, e.g. by highlighting the title text Ab Version 3.13 unterstützt tudaexercise zusätzlich eine Umschaltung bezüglich des Layouts der Subtask-Überschriften. Es existieren drei Optionswerte für die Dokumentenklasse:

subtask=ruled Default setting, the headings are separated from the body text by lines (As shown in this document).

subtask=plain No rules around the headings

subtask=runin If very small tasks are set, the break after the title can be prevented . In this case, the user is responsible for ensuring that any necessary visual separation between the title and task text takes place, e.g. by highlighting the title text.

Task 5.5: Headline customization

The mechanism for the headers has been adapted so that its content can be freely selected.

```
\ConfigureHeadline{
headline={content of the headline}
}
```

There is now a distinction between left and right pages (in case `twoside=true`).

```
\ConfigureHeadline{
even={content of the header for left pages},
odd={content of the header for right pages},
oneside={content of the header in single-page mode}
}
```

Settings that affect the same pages overwrite each other.

5.5a) Custom content for headlines

The content can be any LaTeX code which will be placed within a box as wide as the text. Content will be typeset left aligned. It is possible to place graphics or tables within this box.

There are some ready-made elements for the headers that can be placed in the header.

```
\ShortTitle
\StudentID
\StudentName
```

5.5b) Predefined header options

In addition, the `headline` option acceps some pre-defined values which will place the entries entries according to the self-explanatory naming scheme: `title-name-id`, `title-name`, `title`, `name-id`, `name`

5.5c) Enable the header for the title page

As the header may be used to specify student data, it is possible to use the class option `headontitle`. This will enable the header for the title page.

Task 5.6: Solutions

tudaexercise has a mechanism for managing solution proposals within the files. The `solution` environment exists for this purpose. To enable the output of solutions it's possible to use the class option (global)

```
\documentclass[...,
   solution=true,
]{tudaexercise}
```

or configure it locally for individual tasks:

```
\begin{task}[solution=true]{Heading}
Task description
   \begin{solution}
   Solution
   \end{solution}
\end{task}
```

The default setting is solution=false.

Solution:

This is an example for a solution block.

There also exists a `solution*` environment. Which behaves the same way, but does not automatically add a title to the block.

Task 5.7: Query the `solution` setting

As of version 3.04, it is also possible to query the solution setting using the following structures:

```
\IfSolutionT{<if solution=true>}
\IfSolutionF{<if solution=false>}
\IfSolutionTF{<if solution=true>}{<if solution=false>}
```

In the current setting (solution=false), the above block generates the following output:

<if solution=false> <if solution=false>

Task 5.8: Points (10)

Points can be specified using the `credit` key.

using the macro (o)ption key. The value is placed after the task title within parentheses. This is internally done using the `\creditformat` command which can be redefined to archieve different formats.

Alternatively, there is the `points` key which, in addition to the specification, also adds the text `\PointName` or `\PointsName` depending on whether the value equals 1. The points key therefore only accepts numerical values (from version 3.13 floating point numbers are supported). The decimal separator here is a period. Correction to a comma is possible by redefining `\pointformat`.

5.8a) Example for the using `points` (1 Point)

As older versions of TUDaExercise did not allow option processing for subtask, an additional mechanism is included here. If both a title and specific information are required, the title key is also included here. A check is made to see whether the argument contains an equals sign; if this is the case, it is assumed that options are passed according to the key=value structure.

Task 5.9: Automatisms for grading tables (Total: 13 Points)

At version 3.13 tudaexercise received the functionality for automatic grading tables and references to the point data.

5.9a) Activate references (3 Points)

tudaexercise has generally deactivated this functionality, but it can be activated via the class option `points=true` be switched on.

The referencing mechanism works in a similar way to the object references in LaTeX. This makes it necessary to compile the document twice. If tudaexercise detects different values between begin and end of the compilation process, a corresponding warning is generated.

Two macros are available for the reference. One works by calling up the task number, the other via labels:

```
\getPoints{5.9}
\refPoints{task:refPoints}% requires the \label{task:refPoints}, see source code
```

In this case, both macros return the same value: 13. As \refPoints works independently of hte structure of the task numbers, this variant is generally preferable. In addition to references to individual tasks, `\getPointsTotal` returns the total of all tasks of the last compilation.

5.9b) Summation among subtasks (2 Points)

The new mechanism also allows automatic calculation of the total points of all subtasks within a task. To enable this mechanism the option `points=auto` has to be set. If it's set as a class option, it will be activated for all tasks. If the class option points=auto is set, this is automatically activated for all tasks.

However, the automatic value can be overwritten by manually entering a value, e. g. if a task does not onldy consist of subtasks.

There is currently no automatic check that the sum of the subtasks matches the total.

There is another internal variant of the macro `\creditformat` to allow a distinction between summed values and manually set values.

```
\newcommand*{\creditformatsum}[1]{\creditformat{#1}}
```

By default it is the same but could be overwritten like the following example:

```
\renewcommand*{\creditformatsum}[1]{\creditformat{Total: #1}}
```

5.9c) Automatic grading tables (3 Points)

tudaexercise also provides a list of all points with the option of referencing the task values. As the structure of a grading table varies greatly depending on personal preferences, the macro is designed in a very abstract way.

```
\mapPoints{<code to be executed for each task>}
```

`\mapPoints` iterates over all tasks and allows any code elements as arguments. Within the code, `#1` (Task Number) and `#2` (associated point value) can be used. Tasks with a value of zero are skipped by default.

For example:

```
\begin{tabular}{@{}lr@{}}
  \toprule
  \mapPoints{Task #1&#2 Points\\}
  \midrule
  Total&\getPointsTotal{} Points\\
  \bottomrule
\end{tabular}
```

creates the following output:

Task 5.9	13 Points
Total	13 Points

5.9d) Detailed grading tables including subtasks (5 Points)

With version 4.00 there is also the option using `\mapPoints*`. This variant also iterates over subtasks. In addition, the mechanism has been expanded to include a more general variant. This can be used to interact with all tasks. The documentation will be extended!

Task 5.10: Uniform syntax for task/subtask

By default, TUDaExercise distinguishes between task and subtask. This is due to the fact that subtask does not require a title to be specified.

The need for a uniform syntax was expressed (see `https://github.com/tudace/tuda_latex_templates/issues/189`).

5.10a) Starred variants of the task/subtaks environments

The task* and subtask* variants implemented with version 3.0 make this possible:

```
\begin{task*}{Task heading}
\begin{subtask*}{subtask heading}
Description
  \end{subtask*}
\end{task*}
```

5.10b) Swapping both variants

The class option `match-task-subtask-syntax` allows subtask and subtask* to be swapped. The task* and task environments are identical anyway.