exam-n: exam papers

Norman Gray
(norman.gray@glasgow.ac.uk)

Version exam-n-1.4.1, 2023 November 22

Contents

This class file supports creating exam papers. This is intended as part of a slightly larger system for assembling and managing the papers, by combining questions from multiple authors.

This document is addressed to the exams convener, or someone responsible for the overall exam; it goes into complete detail about the class file, and how to customise it. There is more compact documentation for the class, addressed to the authors of individual questions, in the companion document 'Notes for Authors'.

The most up-to-date version of this class can be found at http://purl.org/nxg/dist/exam-n.

1 Description

Usage:

```
\documentclass[options...]{exam-n}
<preamble declarations>
\begin{document}
\maketitle
\begin{question}...\end{question}
...
\end{document}
```

1.1 Class options

The class takes the following options. Note that these options are processed in order, so if an option changes a default (for example, draft sets showsolutions by default), then it can be overridden by an option later in the list.

compose, draft, final When composing questions, you should give the compose option to `{exam-n}`; when assembling the paper, you should use draft; and you should use final only for the final version. The draft option switches on the showsolutions option (below), displays `\comment` remarks, and makes some mild layout changes. The compose option implies the showsolutions option, causes questions to be formatted one per page, and turns off various checks on the number of questions; the draft option should have the same pagination as the final paper and shifts the body of the paper to the left so that marginal comments are easier to write and read; the final option turns off the `\comment` command.

cmfonts, psfonts, mathptm, mathtime, mtpro2, stix2 The cmfonts option (the default) uses the Computer Modern fonts for the document, and the other options use PostScript or OpenType fonts.

The mathtime or mtpro2 options select PostScript Times Roman, Helvetica and Courier for the body text, plus either Mathtime or Mathtime Professional 2 to typeset maths (this is commercially distributed by PCTEX[1], and designed to be compatible with Times Roman). A broadly compatible alternative to this is to use the mathptm option, but although this is free, the results aren't impressive.

A final option is to use the stix2 option to choose specifically and exclusively the (free) STIX2 font set[2] for both text and maths. On this site, you will need to download and install the files in the static_otf.zip bundle. Note that the support for this option is somewhat experimental, and at present, this package simply uses (serif) STIX2 roman for all font shapes. Since the exam style doesn't use any sans serif by default, this doesn't matter, at some level. The sansserif option is not compatible with this option (the option won't produce any error, but neither will it produce any effect).

[1] https://www.pctex.com/mtpro2.html
[2] https://www.stixfonts.org

FIXME: remove this when unicode-math is working. Note: because of an unfortunate interaction between packages, the [stix2] and [siunitx] packages can produce unexpected results with some combinations of TEX engine and package version (specifically, [stix2] plus [siunitx] and `\micro` or `\circ` can produce garbled characters without LATEX reporting an error). Before TEXLive 2021, this worked correctly with XeLATEX and LuaLATEX, but not with pdfLATEX; from TEXLive 2021 this works as expected with pdfLATEX, but works with the other two *only* if you load the [siunitx] package with `\usepackage{siunitx}[=v2]`. I hope that future versions of one or other of these packages will make clear how this package should handle them, in a way which avoids this special-casing.

The handling of non-standard fonts has always been slightly fragile in LATEX. The mtpro2 and stix2 options are those currently most used by the author, and are therefore the ones most likely to get the various font nuances correct.

uprightpi By default π is set slanted, as is the usual LATEX default. When this is referring to the circular constant, however, this should (in some typesetting styles) be set as an upright letter, in fonts which support this. If the option uprightpi is present, then `\pi` is defined to produce an upright letter, and the command `\italicpi` is defined to produce the slanted version. Whether this option is supplied or not, the macro `\uppi` will produce a single upright π. Note: this is at present implemented only for the mtpro2 and stix2 options, and the combination of pdflatex+stix2 does not support `\uppi` at present.

(no)siunitx Load the [siunitx] package, or not. The default is yes. The `\units` macro, described below, still exists as an alternative, but will be removed in a forthcoming release (probably after 1.4.x).

(no)showsolutions These control whether the solutions to the questions are shown on the paper or not. showsolutions is the default when the either draft or compose options is present, and noshowsolutions in final case, though this will be overridden by one of the {no}showsolutions options later in the list.

(no)perquestionmarks and (no)showmarktotals These control whether mark totals and subtotals are tracked within the question (perquestionmarks), and whether the total available marks are displayed at the end of the question (showmarktotals). See Sect. 1.2 for more discussion (this option was called showmarks up to release 0.15).

(no)pageperquestion If the option pageperquestion is present, then each question is on a separate page. Option nopageperquestion is the default. This is most often set by default by one of the other options, or by a `.clo` file.

oneside, twoside These control whether the document is formatted for one- or twosided printing. This is a standard option, which is redundant in this case, since in this document style there is no difference in formatting. You may in principle use other article options, though you are encouraged not to.

fleqn Display equations flush-left, rather than centred. You generally won't set this in the `\documentclass` options – it's here so that it can be set in a `.clo` file.

sloppydescription Disables the requirement that all exam metadata be present. See the discussion of `\universitycoursecode` below.

mono, colour The university logo is typeset in colour by default, as is the solution text; the mono option causes everything to be in mono instead. The colour option does nothing, but is present for symmetry.

sansserif Use a sans serif font for the body text. This is plug-ugly, not least because it doesn't match the maths font; also, it may be easier or harder for dyslexic students to read (there seems to be a variety of advice here, in both directions, ranging from confident to dogmatic, but remarkably little solid evidence either way).

largefont, or hugefont Produce a version of the paper in a 'big' font (18pt) or 'huge' font (36pt) for the benefit of students with visual impairments. See notes below.

Any other options are interpreted as an instruction to read in a `.clo` file, containing course-, department- or group-specific style modifications. Most typically, these changes will affect the rubric, and the sheet of physical constants. The only generic style option is myclass (which is included as an example of how such a file is written). See Sect. 2 below.

In some circumstances – for example when processing exam scripts under the control of a Makefile or script – it can be convenient to control package options from outside the package. If there is a file called `exam-n.config` in the input path (most likely in the same directory as the exam paper), then this is read in when any exam script is processed, and will supplement any options in the `\documentclass` line. For example, if this file were present and contained the line `\ExecuteOptions{showsolutions}`, then the typeset exam would include the solutions. Note that this will supplement, *but not override*, options in the input file's `\documentclass`; it is therefore useful only for switching options away from the default.

The exam-n class includes the [amsmath] package, so you can make immediate use of amsmath features if you wish (see `http://www.ams.org/tex/amslatex.html` for discussion).

The change of font size with the hugefont option requires a couple of minor layout changes. You may need to force some similar changes in the exam paper, in this case. You can do that by bracketing the adjusted text in `\ifbigfont`⟨*big font material*⟩`\else`⟨*normal font material*⟩`\fi`; the `\ifbigfont` flag is true in the case of hugefont, but not largefont. If you want the `\ifbigfont` flag to apply to largefont, too, then you may set `\bigfonttrue` in a suitable place. The [amsmath] `multline` environment can be useful here. The exams convener should review the result carefully: a few judicious `\ifbigfont\newpage\fi` insertions can make the result look less awful. The largefont and hugefont options apply to the content of solutions if the showsolutions option is present; this may or may not be the optimal choice.

At the bottom of each page, you see a faint identification code, such as 'QM/123-456'. This consists of an exam identifier, extracted from the exam preamble, plus a code which changes each time LaTeX is run. This helps you avoid collation accidents, and to distinguish between slightly different versions of the printed document. The identifier is based on the date and time, so changes each time

you run LaTeX on the file (that is, it identifies a printing, rather than a source-file version).

1.2 The Question environment

Within the document, you include questions within a `{question}` environment, within which you may further have `{questiondata}` and `{solution}` environments.

`question` (*env.*) The `{question}` environment delimits a single exam question.

Usage, with perquestionmarks:

```
\begin{question}[⟨questionnumber⟩]{⟨marks⟩}
…
\end{question}
```

Or, with noperquestionmarks:

```
\begin{question}[⟨questionnumber⟩]
…
\end{question}
```

There are two variants of this environment, depending on whether the perquestionmarks option is present or not. If it is present (the default) then the `{question}` environment takes a non-empty argument showing the total marks available for a question. In addition, the class checks that the `\partmarks` commands within the question (see below) add up to this declared goal mark. If the showmarktotals option is present, then this (expected and checked) total is displayed at the end of the question.

If the perquestionmarks option is *not* present, then questions have no individual marks, the environment takes no marks argument, and no marks are shown (this is usual in essay-question exams, which typically comprise a sequence of short equal-valued questions with an explanatory rubric). Also, in this case the `\partmarks` command cannot be used within the question.

The `{question}` environment takes an optional argument, giving the expected question number. In compose mode, this is used as the question number (unsurprisingly). In the two other modes, this is compared with the question number which would be generated based on the position in the sequence of `{question}` environments in the file, so that the first `{question}` environment is for question one, the second for question two, and so on. If these do not match, the exam class displays a warning in draft mode, and produces a fatal error in final mode. The intention is that this can act as a check that all expected questions are present; see also the `\numquestions` command.

`\QuestionNumberChecksOff` In some cases, the questions are not numbered in this straightforward fashion, so that you might have questions '2A' and '2B'. In this case, the check is not meaningful, and you must suppress it by calling the macro `\QuestionNumberChecksOff` in the preamble. After that, you must provide a question-number argument (in square brackets) for every question.

Using `\label` within a question sets a label for the question number; using it within a `\part` (see below) labels the part number.

In some odder circumstances, you might not want to have any question numbers at all; for example, you might want to require examinees to attempt *all* of

the questions, and so simply have a mark for the whole exam. There isn't a mode for this as such, but if you use the `\QuestionNumberChecksOff` macro, and give `[\space]` as the optional 'question number' at the beginning of the `{question}` environment, then this will have the desired effect.

`solution` (*env.*) The `{solution}` environment, contained within the `{question}` environment, contains the solution to the question, or other notes. It is displayed by default in compose mode, and suppressed by default in the others, though this behaviour may be overridden in either case with the (no)showsolutions option. You can have one `{solution}` at the end of your question, or have multiple ones scattered throughout it. You may use the `\partmarks` macro within the solution, to indicate the distribution of marks within (this part of) the solution – these, of course, do not count towards the total mark for the question. You must not have a solution inside a solution.

`questiondata` (*env.*) At the end of a question, it is frequently useful to include further information, such as extra equations, or numerical data. These should be included within a `{questiondata}` environment, in order for them to be formatted appropriately. You may include multiple paragraphs, equations, and displays in this environment, as appropriate. Typically, you will have only one such environment per question, appearing at the end, but may have several of them if you really wish to.

`mcq` (*env.*) Some exams include multiple-choice questions rather than extended-answer ones. These are numbered in the same sequence as the other questions, but are formatted and marked-up differently.

```
\begin{mcq}
In 1908, where was there an airburst `impact'?
\answer Tunguska
\item Arizona
\item Off the Mexican coast
\item Egypt
\end{mcq}
```

That is, the `{mcq}` environment contains a list of possible answers, all of which are indicated by `\item`, except precisely one correct answer, indicated by `\answer`.

`\multiplechoiceanswers` All multiple-choice questions must have the same number of possible answers, which is declared with the command `\multiplechoiceanswers{`$\langle n \rangle$`}` The `{mcq}` environment is permitted only after `\multiplechoiceanswers`. It's OK to have a `{solution}` within an `{mcq}` environment, which might provide further commentary on the correct answer.

There are various other commands which you may or should use within the document.

`\includequestion` It may be convenient to split your exam into a number of separate source files, such as having one `.tex` file for each question. You can include these various source files using the usual `\input` command.

If the separate source files have the simple form:

```
\documentclass[compose]{exam-n}
\usepackage{graphicx} % for example
\begin{document}
\begin{question}
...
\end{question}
```

```
\end{document}
```

then they can be LaTeXed separately, for example by the authors of different questions, but cannot be `\input` unedited, as described above. If, however, they have *only* these structures (that is, only the `\documentclass` command, the `{document}` environment, and zero or more `\usepackage` or `\RequirePackage` commands), then you can most conveniently import them unedited using the `\includequestion` command.

```
\includequestion{dynamics2}
```

This acts like the `\input` command, but disables the listed structures. It also puts the included command into a group, so that any (re)definitions of commands are made local-only.

Recall that the definition of the `\includequestion` command means that any `\usepackage` commands will be ignored. If you, as a question author, need certain packages to be present for your question, you will have to make sure that whoever is assembling the master file includes those packages there, too.

That is, we don't try too hard to support including just any old LaTeX, here: any complicated preamble requirements in an included file should probably be managed by the exams convener transplanting them into the preamble of the main document. This may be an overly simple-minded approach, and may change in future versions.

The `\includequestion` command takes an optional argument which overrides the question number. This caters for the case where question authors have (unhelpfully) included question numbers in the files' `{question}` environments, and the case where questions are not numbered in a straightforward sequence, for example '1', '2A', '2B', and so on.

`\section` `\subsection`

Some exams are divided into sections, or have other structure which needs to be spelled out. These are described with the `\section` command, in a form such as `\section{II}`. You can also add smaller headers before individual questions with something like `\subsection{Second semester questions}`. These can appear only between questions; it is an error to include one of these commands within a `{question}` or `{mcq}` environment.

`\part`

Questions may be subdivided into parts, such as (a), (b), (c)..., or (i), (ii), (iii), and so on. Precede each of these with this `\part` command. The formatting of the part numbers is controlled by the exam style, as customised in Sect. 2. This macro starts a new paragraph.

You can use the `\part` macro within solutions: this is useful if you have the entire `{solution}` environment at the end of the question, but distracting if you intersperse the `{solution}` environments between question parts. The `\part` macro within solutions increments separately from the increments within the question, so if you use this in the solutions, you must have as many `\part`s in the solution as there are parts in the question.

In some (rare) cases, you may want to force a particular part number. You can do that with an optional argument `\part[99]` which overrides the auto-incremented numbering. Note that this skips the auto-increment of the number but doesn't cancel it, so if you do this for one part number you should probably do it for all of them.

\partmarks The \partmarks{⟨*num*⟩} macro announces the number of marks associated with the current part of a question. The class checks that the number of marks here does add up to the number declared at the beginning of the question environment. You should have only one \partmarks per \part (the current version of the class does not enforce this, but may do so in future, in order to preserve the 1–1 relationship between a question part and the marks associated with it).

The \partmarks command will most naturally go at the end of a paragraph, but it may also appear inside an equation (that is, in \[...\]; don't use $$...$$), inside one or other [amsmath] display or {equation} environments, or in a list or other environment. If it appears inside an environment, the indicator will appear at the *end* of the environment, independent of where in the environment the command was typed (which implies that you can't have more than one inside an environment). The alignment of the partmarks indicator is not currently as good as I'd like it to be, in the case of [amsmath] alignments, but this turns out to be hard to improve.

\partmarks* The starred version is almost identical, except that it places its indicator a little upwards of the location where it is invoked, by a heuristic amount which is about right when it is used after an equation. It may not be used inside an equation or other environment. It's useful as a fallback, if a particularly complicated displayed equation prevents the unstarred version from working properly (some amsmath environments can cause trouble here), or occasionally after a list.

The unstarred version should only be used at the end of a paragraph, and in fact forces a paragraph end; the starred version should on stylistic grounds generally be used only at the end of a paragraph, but it doesn't force one.

The \partmarks command has an optional argument which indicates the category of the question, thus 'bookwork', 'unseen', and so on. If this is present, then the category is included in the marks indicator.

The \partmarks category/comment will typically be only one or two words long, and can sit comfortably in the margin. If an author wants to write more here, then it will be turned into a footnote on the page. This will obviously change the layout of the page, though since this text appears only in showsolutions mode, that shouldn't be a problem.

\defaultpartmarkscategory If the exams convener wishes to *oblige* people to include such a category, then they might call \defaultpartmarkscategory{category?} in a package-options (.clo) file, to default the category with a highlighted remark to the question setter.

\comment \comment{⟨*text*⟩} associates a comment with a part of the text. This is ignored in final mode, but appears in the margin in the other modes.

\author The \author command is a convenience. Used within a {question} environment – most naturally just after the \begin{question} – it creates a comment with the author's name. Its functionality may be expanded in future, so you should use this command, rather than a generic \comment, when noting the authorship of a question.

\shout If there is part of a question which is, for example, incomplete, and which needs a more prominent callout than 'comment', then you should \shout{...} it. Shouts appear in all modes (*including* final) and appear whether or not the class is showing solutions. This makes a prominent remark in the text, and also in a list of shouts at the end of the text. Your co-authors, or the exam proof-checker, really have no excuse for missing it after that.

\leftnudge For various reasons, most often because of printing problems, it can be useful

to nudge the textblock left or right a little. You should call the `\leftnudge` command to do this, rather than fiddling with the underlying LaTeX dimensions yourself. Give the command a dimension argument such as `\leftnudge{1cm}` to nudge the textblock leftwards by 1cm. You can give a negative dimension to nudge it rightwards instead. The draft option automatically nudges the text block leftwards, to create a larger right-hand margin for notes.

`\questionpreamble` If `\questionpreamble` is called, then its contents are displayed just before the start of the next question. This is useful for text like `\questionpreamble{And one of\dots}` which might reinforce information in the examination rubric.

1.3 Preamble

`\exambanner` The `\exambanner` macro supplies text like 'Examination for the degrees of…'. Since the contents of this command is automatically uppercased in some styles, and there are per-department specifics about the punctuation of abbreviations, you should use the commands `\BSc`, `\MSci`, and friends (see Sect. 1.4) to set the degree names appropriately.

`\universitycoursecode` `\degreedescriptions` `\coursetitle` Declare the identity of the exam with `\universitycoursecode`, `\schoolcoursecode`, `\coursetitle` and `\degreedescriptions`. The distinction between these is as follows:

University course code This is the code for the course (and thus for the paper) as it appears in university information systems, and is a university-unique code such as 'PHYS3031'.

Course title This is just a textual name for the course, for example 'Quantum Mechanics'.

Degree descriptions This is a textual description of the qualifications that the students doing this exam are heading for. This text has little formal weight, but might help a lost student realise they're in completely the wrong exam room…. This is something like `{Physics 3\\Chemical Physics 3}` (separate each description using `\\`).

All of these are required elements, and the LaTeX compilation will halt if they are absent. If for some reason the exam paper does not need these to be present – perhaps it is a class test, for example – then give the class option sloppydescription, and the checks for these elements (and for `\rubric`) are suppressed. The layout may end up looking a little funny.

`\schoolcoursecode` There is also a command `\schoolcoursecode`, which is a more informal, but possibly more recognisable, code for the course/paper, as it is generally recognised within the school; for example, the honours Quantum Mechanics course is known within the school as P304H. Its use is optional.

`\paperident` It can be convenient to add some identification to each page, if for no other reason than to double-check that you haven't inserted a field theory question into an 'astronomy for poets' exam. The command `\paperident` allows you to declare some text which appears at the bottom of each page of the exam. It will typically repeat some of the text in the `\schoolcoursecode` or `\degreedescriptions` arguments. This is generally not necessary, however, as in its absence the class generates an identifier. This identifier contains the name of the exam, plus a pair of counters (for example *QM2/98-1177*). The function is two-fold: (i) since the

counters increase monotonically (they actually encode the date and time when the document was LaTeXed), you can tell which of two superficially similar documents is the later; and (ii) if you drop a sheaf of papers on the photocopier floor, you can work out which one is which.

`\examdate` `\examtime` Give the date and time of the exam with `\examdate` and `\examtime`. Sometimes an exam may have different time limits for different qualifications: this case, separate the various times with `\\`, as in `\examtime{9.30am -- 12 noon\\ (or) 9.30am -- 1.45am}`.

`\rubric` `\norubric` `\baserubric` The rubric, provided unsurprisingly by the command `\rubric`, may contain more than one paragraph, delimited by the usual blank line. Any emphasised words should be marked with `\emph` – they are typically emphasised with a bold font. The class checks that a rubric has been specified (unless sloppydescription is present); if you really wish to suppress this rubric – perhaps because the `\baserubric` is sufficient – then give the command `\norubric`. As well as this exam-specific rubric, the style produces an additional boilerplate rubric, containing the usual material such as 'Do not on any account attempt to write on both sides of the paper at once. Calculations may be done on the fingers, but candidates should avoid counting on their toes unless special permission has been obtained beforehand.' You will typically not have to change this, but if you do for some reason, you can override it with the `\baserubric` command.

Note that the `\baserubric` command is typically used within a `.clo` file, within the argument to `\OverrideFormatting`. If you wish to further override this on a per-exam basis, then you will need to do so after `\OverrideFormatting` has done its work, and thus immediately after the `\begin{document}`.

`\numquestions` Finally, declare the number of questions which are to be in the paper with `\numquestions`. The class issues a warning if we don't have this number, in draft or final mode. This is optional – no check is done if this isn't present.

1.4 Other useful commands

`\BSc and friends` Macros `\BSc`, `\MSci`, `\MSc`, `\MA`, `\MEng` and `\BEng` are used within the preamble macros to give appropriately capitalised and punctuated versions of the degree types.

`\vec` Macro `\vec` is redefined to give bold-font vectors, rather than vectors with arrows, which is the (weird) LaTeX default. This should work for bold greek as well as roman.

`\dd` Macros `\dd` and `\ddd`: `\dd` is a roman d, as used for differentials; `\ddd` is the same with a preceding thinspace, as used within integrals; for example

$$\texttt{\textbackslash int f(x)\textbackslash ddd x = \textbackslash int f(x)\textbackslash,\textbackslash dd x} = \int f(x)\,\mathrm{d}x$$

`\Diffl` You can typeset derivatives neatly:

`\Diffl{a}{b}`	$\dfrac{\mathrm{d}a}{\mathrm{d}b}$
`\Diffl[2]{a}{b}`	$\dfrac{\mathrm{d}^2a}{\mathrm{d}b^2}$
`\Diffl*{a}{b}`	$\mathrm{d}a/\mathrm{d}b$
`\Diffl*[2]{a}{b}`	$\mathrm{d}^2a/\mathrm{d}b^2$

`\e`	$e^{i\pi} = -1$	the exponential is typeset in an upright rather than italic shape, as in `$\e^{i\pi}=-1$`.

Table 1: Miscellaneous symbols

The unstarred versions are for displayed equations, the starred ones for inline maths. There is analogous support for partial derivatives with `\Partial`.

`\Partial`

`\units`

You should generally type units, and numbers with units, using the [siunitx] package, which is loaded by the siunitx option (which is now enabled by default). However this class currently also supports a basic `\units` command, described below. This macro will be removed in a future version of the class.

Macros `\units`, `\units*`: you can typeset physical units in `\rm`, with tilde or dot acting as a separator between units. Since this is typeset in maths mode, all other spacing is ignored. The unstarred version includes leading `\thinspace`, as in `$v=10\units{m.s^{-1}}$`, giving $v = 10\,\mathrm{m\,s^{-1}}$. The starred version can be used when referring to the unit by itself (eg axis is `$B/\units*T$`, or B/T), and is not qualifying a number.

For other useful symbols, see table 1.[3]

2 Customising the exam style

As described in Sect. 1.1 above, any unrecognised options are interpreted as an instruction to search for and include a class options file, formed from the name of the unrecognised option, suffixed with `.clo`, which can be anywhere in the TEX include path. This options file has a fair amount of leeway to override and adjust the layout of the exam.

The most typical changes here will be to adjust the exam rubric for a particular class, with the command `\baserubric`, and to change the sheet of physical constants, with the command `\constantssheet`. See the sample file `myclass.clo` for examples.

Examine this sample `SpecialExam.clo` file:

```
\typeout{Physics Special exam options, for Special people}
\ExecuteOptions{pageperquestion}

\OverrideFormatting{
\renewcommand\FormatPartMarks[1]{\{#1\}}
\renewcommand\FormatPartNumber
  {\hbox to 0pt{\hss (\StylePartNumber{partnumber})\hskip1em}}
\let\StylePartNumber\roman % as opposed to \alph
\renewcommand\FormatQuestionNumber
  {\hbox to 0pt{\hss \textbf{\@currentquestion}\hskip2.5em}}
}

\constantssheet{
   \begin{center}
   $E=mc^2$ and $c=3\times10^8 {\rm m\,s^{-1}}$
```

[3] The package used to support an `\au` macro, for astronomical unit, and `\lambdabar` for Compton wavelength, but these have since been removed. The former is available via [siunitx].

```
    \end{center}
}
```

This announces itself, then invokes the exam style's pageperquestion option.

It then includes a number of formatting adjustments, enclosed within the `\OverrideFormatting` command; the formatting hooks are described below.

Then it declares a 'constants sheet', which is a display of constants or equations, or indeed anything else, which is to be displayed on the second page of the exam.

The available formatting hooks are as follows:

FormatPartMarks This formats the indication of the marks carried by a particular part of a question. In this case, we have chosen to have the marks contains inside curly brackets, rather than the default square brackets. By default, the part-marks text will be placed at the end of the paragraph it completes, flush right, and with at least 2em of space before it. You can change this default space with `\@partmarksspace`; as a special case, you can have the text sitting in the margin instead, by having `\FormatPartMarks` generate a zero-width box, and setting `\@partmarksspace=0pt`.

FormatPartNumber This overrides how to format the various `\part` markers within a question, using the `partnumber` counter. In this case, the markers will jut into the left-hand margin, rather than being run-in.

StylePartNumber This overrides how to style the `\part` markers. The default is `\alph`, but you might prefer, for example, `\Roman` or `\arabic`. Override this with `\let\StylePartNumber\Roman`.

FormatQuestionNumber If you adjust the part marks, you should probably adjust the formatting of the question number also.

If you really want to go to town on reformatting, you can redefine the command `\maketitle`, which formats the front-page title. When formatting this, you have access to each of the fragments of text described in Sect. 1.3, via a macro named after the corresponding command. Thus the argument of the `\exambanner` command is available in the macro `\@exambanner`. The exception is `\numquestions`. If you find yourself needing to do this, it might be worth having a discussion with the style's maintainer – there may be a simpler way to get what you want.

`\CheckExamMetadata`

`\RequiredMetadata`

One of the things you may change within the `\maketitle` is the checking of exam metadata – which fields are required and which are optional. You can change these from the default (in Sect. 1.3) by defining a command `\CheckExamMetadata`. This macro takes no arguments, and should use the command `\RequiredMetadata{`⟨*field*⟩`}{`⟨*description*⟩`}{`⟨*help-text*⟩`}` to perform its tests. Here ⟨*field*⟩ is the metadata key, such as `{examdate}`, which corresponds to the macro `\examdate`, ⟨*description*⟩ is a brief description of the field, such as `{exam date}`, and ⟨*help-text*⟩ is a longer bit of explanation. If the corresponding data is missing, then LaTeX stops with an error, and the user can examine the ⟨*help-text*⟩ by pressing the `h` key. You can adjust the test by examining the value of `\iffussydescription ... \fi`, which is set to `\false` if the sloppydescription option was provided.

No more clues. If you want to hack at this, see the definition of `\maketitle` in the class file `exam-n.cls`. Aspire not to break things.

3 A work-flow

This class file was developed as part of a move to 'LaTeXify' the production of exams in the School of Physics and Astronomy, in the University of Glasgow. The work-flow which emerged was, roughly, as follows:

1. The exam convener (the person in charge of assembling the exam for a given course or lecture series) creates a Sharepoint directory which contains a master file (which consists mostly of `\includequestion` macros), plus subdirectories each containing a short template file for the benefit of question authors, and a copy of the `notes-for-authors.pdf` instructions.

2. Each of those 'question' directories is made visible to, and read-writable by, the lecturer responsible for that question. They write and LaTeXed their question, and upload it back to Sharepoint.

3. The exam convener, who has access to the whole tree of questions, downloads or network-mounts the Sharepoint folder, and either edits the questions or negotiates with the authors, until the exam LaTeXs successfully. The result can then be snapshotted, and sent off to the external examiner.

This provides a centralised and easily manageable repository for a previously fiddly and error-prone process.

In step one, the template file is just one with a `[compose]` option, following the brief example on p.6. We had excellent secretarial help to manage this step; it was fiddly the first couple of times, but largely mechanical thereafter. In Sharepoint, this step can in principle be automated, though we concluded this was unnecessary in our case.

Sharepoint (possibly slightly surprisingly) worked smoothly for us; in other circumstances a group-writable source code repository, or an institutional CMS, or even just a shared drive, would work just as well. The advantage of Sharepoint, from our point of view, was that it integrated with the university's existing authentication framework, and was (axiomatically) acceptably secure for the process of creating exams.

The majority of our authors were habitual users of LaTeX for writing journal articles, and with subsequently submitting them to journals or the arXiv; and when the process was presented in those terms, with the same advantages, it was readily accepted.

Thanks to Graham Woan and Rachael MacLaughlan for the numerous iterations of this process until both it and the class file were stable.

4 Example

Here is a short example file. There are further examples in the `sample/` directory of the distribution.

```
⟨*example⟩
%%%START example (Makefile strips out this block)
\documentclass{exam-n}  % standard final version
%%\documentclass[draft,showsolutions]{exam-n}  % draft style, showing solutions
%%\documentclass[compose]{exam-n}  % compose (author's) style

```

```
\examdate{Wednesday, 18 May 2005}
\examtime{9.30am -- 12 noon\\(or) 9.30am -- 1.45am}

\exambanner{Examination for the Degrees of \BSc(Science) and
  \MSci\ on the Honours Standard}
\schoolcoursecode{P304D and P304H}
\universitycoursecode{PHYS3031 and PHYS4025}
\coursetitle{Quantum Mechanics}
%\degreedescriptions{Physics 3, Chemical Physics 3, Physics with
%  Astrophysics 3, Theoretical Physics 3M, Joint Physics 3}
\degreedescriptions{Physics 3\\Chemical Physics 3\\Physics with
  Astrophysics 3\\Theoretical Physics 3M\\Joint Physics 3}
\paperident{GR/P304}

\rubric{Candidates for examination in \emph{Quantum Mechanics} should
  answer question 1 (16 marks) and \emph{either 2A or 2B} (24 marks each)}

\numquestions{3}

\begin{document}
\maketitle

\section{I}

\begin{question}{20}
\part At various points in the development of the mathematical theory of
General Relativity, we pick a coordinate system in which
differentiation is simple, and do a calculation using non-covariant
differentiation, indicated by a comma.  We then immediately deduce the
covariant result, replacing this comma with a semicolon.

Separately, the strong equivalence principle is sometimes
referred to as the `comma goes to semicolon' rule.

Explain the logic of each of these replacements of a comma with a
semicolon, putting particular stress on the distinction between
them.\partmarks{10}

\part The radial and angular coordinates, $r$ and $\phi$ respectively,
of a test particle moving in the Schwartzschild metric exterior to a
star of mass $M\ll r$, are related by the equation
\[
  r = \frac{h^2}M \left(
        1 + e\cos\phi + \frac{3M^2}{h^2}e\phi\sin\phi
        \right)^{-1},
\]
where $h$ and $e$ are constants.  Show that this equation takes the
form of a precessing ellipse, of semi-latus rectum $l=h^2/M$, in which
the pericentre line advances each orbit by an amount
$\Delta=6\pi M^2/h^2$, stating clearly any assumptions that
you make.\partmarks{6}

The solar-mass star HD83443 has a 0.35 Jupiter-mass planet that
follows a circular orbit of period 2.986 days and radius \SI{0.038}{\astronomicalunit}.
```

```
Calculate the rate of precession, in arcseconds per year, of the
pericentre line of the planet's orbit.\partmarks{4}

[Schwartzschild radius of the Sun: \SI{3e3}m; $\SI1\astronomicalunit = \SI{1.5e11}m$]

\begin{solution}
  In the first type of calculation, we do a calculation in the LIF, in
which~$\Gamma^i_{jk}=0$, so that single partial differentiation is the
same as covariant differentiation.  If this process produces a
geometrical object such as a scalar or a tensor, then we know that the
result is frame-invariant.  If the result involves only single partial
differentiation -- that is, no second derivatives -- then since
partial differentiation is the same as covariant differentiation in
these coordinates, we cannot distinguish partial and covariant
derivatives, and can replace the commas by semicolons.  Since these
are now manifestly covariant derivatives, so that the result is a
tensor, and thus frame-invariant, the same expression would be true in
any frame.

The second situation is the statement that the expressions of physical
laws in SR, such as the conservation equation
$T^{\mu\nu}_{,\nu}=0$, must take the same \emph{form} when written
as a covariant equation in GR, crucially without any curvature
coupling.  The slogan `comma goes to semicolon' is just a mnemonic for
this.

The distinction is that the first is a mathematical trick, of sorts,
whereas the second is a version of the equivalence principle, and thus
a statement with deep physical content.

They don't have to explain things at this length or with this
coherence (?) to get quite a few marks.  They just have to show they
have a clue.
\end{solution}
\end{question}

\end{document}
%%%END example
```
⟨/example⟩

5 Release notes

Recent release notes are below. For older notes, see the source distribution.

1.4.1, 2023 November 22 ¶ There is now a VERSION stamp file in the move-to-texmf directory. ¶ Documentation and logging clarifications.

1.4.0, 2022 October 10 ¶ The [siunitx] option is now on by default, and the `\units` macro produces a one-time-per-document warning that it will be removed in the next version. ¶ Having two `\partmarks` commands in an environment is now detected as an error (it was documented as an error, but not checked, so the extra `\partmarks` commands were simply ignored).

1.4.0-b1, 2022 August 7 ¶ The class file now depends on a version of LaTeX which is at least the 2020/10/01 release. This is so that we can use the current LaTeX hooks mechanism. ¶ Rework `\partmarks`: the `\partmarks` indicator can now go *inside* most environments, including list, quotes, unnumbered equations, and the various amsmath displays. In these cases, the indicator will automatically appear at the end of the environment. The starred variant of the command still exists, but should rarely be necessary.

1.3.5, 2022 October 10 Versions 1.3.x are now bugfix-only. Use versions 1.4 in preference.

There are no functional changes from 1.3.5-b4.

The documentation now explicitly advises against using the stix2 and siunitx options together, with a LaTeX older than 2020, but also gives some advice on dealing with that if you do have to do that.

1.3.5-b1, 2022 July 11 ¶ Fix erroneous broken line before `\partmarks`, in compose mode, when a paragraph fills the last line. ¶ Fix support for STIX2 fonts in XeLaTeX and LuaLaTeX (it was working inaccurately before). ¶ Add an optional argument for `\part` to override part numbering. ¶ Use the `xcolor` package for colour management, instead of the core `color` package (the `xcolor` package is well-known and stable, and this means that we are compatible with TikZ). ¶ `\partmarks` in solutions no longer gobble trailing space (doing so is probably right in questions, and is still the case, but `\partmarks` in solutions tend to be more interspersed with text). ¶ `\label` within a `\part` now refers to the part number (as opposed to the question number). ¶ Avoid a 'You can't use ‘’ in vertical mode' error, in certain circumstances. ¶ Add the `[siunitx]` option, indicating that we should load the siunitx[4] package (default no). This is now preferred to the `\units` macro, and the latter will be removed in a forthcoming release. ¶ Renamed `A1.clo` to clearer `myclass.clo` ¶ Repository moved from bitbucket to https://heptapod.host/nxg/exam-n[5], when bitbucket dropped support for Mercurial. This means, incidentally, that links to specific issues at bitbucket are now broken. ¶ Bold-italic maths should now work correctly with STIX. ¶ Long `\partmarks` comments now appear as footnotes. ¶ Solutions are now set `\normalsize`. ¶ Bugfix: move definition of `\defaultpartmarkscategory` so it can be invoked within a `.clo` file straightforwardly. ¶ Define the `[uprightpi]` option to set `\pi` as an upright character, as appropriate for a constant (this is implemented fully only for the `[mtpro2]` and `[stix2]` options).. ¶ Note that the `\units` macro is likely to be deprecated in a forthcoming version, and replaced by a recommendation to use the standard `[siunitx]` package.

1.3, 2018 November 21 ¶ Add a 'category' optional argument to `\partmarks`, and add `\defaultpartmarkscategory`.

1.2.1, 2018 July 2 ¶ Bugfix: ignore any content which appears after `\end{document}`, in `\includequestion` (author) files (fixes issue 6). ¶ The environments `{figure*}` and `{table*}` now produce an error. ¶ The macro `\vec` now

[4] https://www.ctan.org/pkg/siunitx
[5] https://heptapod.host/nxg/exam-n

produces correct greek bold maths. ¶ Docmentation: notes on unit formatting.

1.2, 2017 December 1 ¶ Use serif STIX2 fonts for sans and monospace cases, when using Lua- or XeLaTeX (the style doesn't use/encourage any sans-serif text, so this shouldn't matter). ¶ Fix font sizes in solutions. ¶ Adjustments to font-handling, which appear to have fixed LuaLaTeX differences. ¶ Add the `stix2` option, to use the STIX2 font set[6]. ¶ Make it possible to use `\rubric` and `\baserubric` within a `.clo` file. This was advertised as being possible, but it seems it had never been tested!

Acknowledgements

This class has greatly benefitted from comments and bug-reports from Harry Ward, Graham Woan, and Nicolas Labrosse; and it has received code contributions from Morag Casey.

[6] http://www.stixfonts.org/