# Formatter for Text Fields

## A Programming Task

**innoWake international inc.**
innovative | sustainable | flexible

1101 S Capital of Texas Hwy Bldg J
Ste 100 | Austin, TX 78746

http://www.innowake.com

# 1 Requirements

Please develop a small framework that offers components for receiving data via text fields, validation and conversion in a defined format.

Please develop a generic formatter interface that offers methods such as:

- a method that converts text into an object (parse),
- a method that converts an object into text (format),
- a method for validation of text (isValid)

Regarding naming, signature, or numbers of methods you are free to choose any approach.

Possible implementations of the interface are: date, time, currency, number and floating-point number, email addresses, social security no., phone no., bank account no., etc.

There should not be any dependency on graphics UI tools like Swing, AWT etc. The formatter instances must be open for flexible usage with different frontend technologies as well as servers.

The framework should take into account that two different types of developers will use the API:

- *API-user*: they will instantiate the formatter objects and use them in their own applications, for example a Swing app, an Eclipse plug-in, or a web application.
- *API-Extension user*: those who extend the framework with new formatters.

Developers may fit either role.

The formatter must allow international usage, e.g. input and output must be open for different languages and locale specifics.

The API must be open for tolerant formatting regarding the data input, e.g. date input as "900" shall be identified as "9:00". The input shall then be converted to an explicit output format. Please take into account that potential extensions may not follow this strict rule.

# 2 Optional Extensions

- Certain formatters support gradual validation. The validation offers three different results: "error", "warning" or "ok"
    - "warning" means: formatting is possible but the data input is "dirty" or violates certain conventions.
    - "error" and "warning" result in an output that informs the user about the incorrectness of the data input.
- The API must allow for certain formatters the possibility to define patterns like regular expressions, Scanf/Printf-style, date and time patterns, etc.
- It must be possible to offer formatters that support incremental validation, e.g. while the input is done the validation will be done – item by item. Please consider the special validation state "the input is valid (so far), but the input is incomplete".

# 3 Task Description

- Create a design of the framework using a UML class diagram. If you have no access to any UML tool please feel free to scribble it with pen and paper and then please scan the UML graphic.
- Develop any interfaces and API classes in Java (at least version 5).
- Develop through the perspective of the API extension user two examples of implementations of the formatter: date and email address.
- Last but not least develop through the perspective of the API user a small runnable showcase.

# 4 Annotation

The design should be easy to read, modular and the API intuitive in usage and easy to maintain. If helpful please use design patterns and highlight them in the UML diagram.

Please send us the source codes per email as a zip archive. We do not need the binary files. In case there are dependencies on open source libraries, please also add these to the zip archive or provide us with the corresponding Maven pom.xml file.

Please don't upload your source to any public repository.

Good luck!