Initial Research: Options for the Plugin

- Getting the Stack Trace
  - The org.eclipse.ui.console package offers some Console classes we can utilize to get stderr output (and likely all other printing that occurs during a run in eclipse). If the trace was printed during the run, we can get it. This, however, doesn't let us trace an exception that was made whilst running in a different configuration (e.g. JUnit test run). The package doesn't show up in regular Eclipse distribution for some reason (not with the PDE plugin either), so we need "Eclipse for RCP and RAP Developers" if we want to use it.
  - User Input; this option is relatively easy to implement, since it only requires opening a window into which the user will paste the stack trace (meaning there's no need to extract it from different plugins and consoles). This should be implemented anyway, since it adds functionality when other methods of trace extraction fail. The drawback is that user participation and technical know-how required for this method to work (for example, to get the stack trace of a failed JUnit test, you need to find the test in the JUnit output, select it, find the relevant Exception in the Failure Trace tool, right click it and choose "Copy Trace").
  - Creating a different Run Configuration which uses our plugin instead of the usual configuration.
    - All subclasses of Throwable recognize void printStackTrace() (and a number of similar functions), which can be used to obtain the stack information we need, if we have an object of class Exception or Error.
    - Creating a different run configuration solves the main issue here, which is is reaching this object, since usually the environment would have already handled it in some way during the program run (e.g. its trace was printed to the error stream, or it was added to a different database, like the Failure Trace tool offered in JUnit output, meaning you can't get it using a simple try-catch).

    - Our system should run the program, surrounding it with a non-specific try-catch block, and get the Throwable this way. Is it possible to get this while debugging?

- Which use cases are we implementing?
    - Basic: The user pastes a stack trace into a window
    - Normal: The users run their programs regularly, if it fails, they run the plugin, which collects the stack dump independently and sends it to the website. This should work when debugging as well.
    - Complicated: The user sets up the plugin to run the program itself, if it fails, automatically sends the user to the relevant page in the website.
- Note: When an Exception is caught and gets rethrown, the stack trace will contain the term “Caused by:” (maybe we should extract the root cause and search specifically for it).