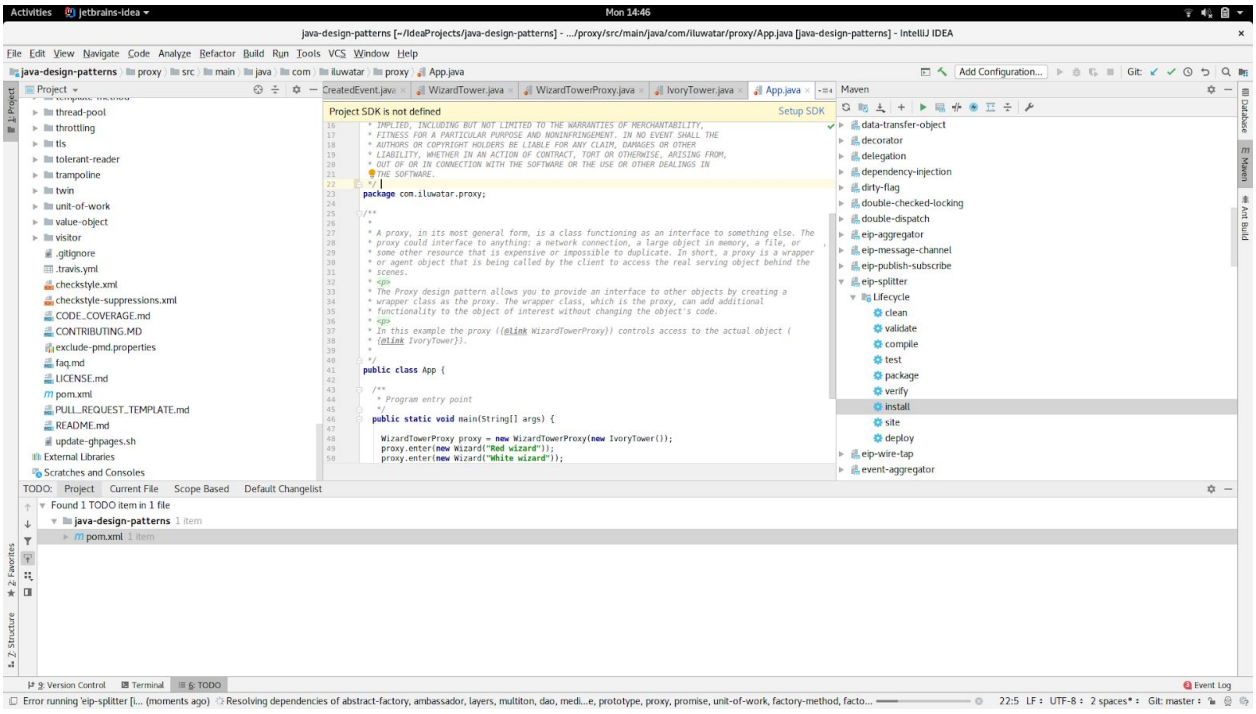
IDE Intelligent Tutorials (IDE-IT)
David Thien and John Barcellos

Summary
An IDE plugin that provides intelligent tutorials to the user. These tutorials would only show up when the user was attempting to manually complete a task that the IDE already has built in as a feature. The tutorial information is delayed until it is relevant to the user.

Motivation
IDEs can be immensely helpful to developers. They aren't the right tool in every situation, but they can provide more support to a developer and automate tasks that they would otherwise have to do manually in a text editor. One of the challenges when using an IDE is how cumbersome they can be. They can have dozens or hundreds of great features, but without properly communicating those features to the user, they are essentially worthless.
Most IDEs (Visual Studio, Intellij, Eclipse, etc) have some form of tips (tips of the day usually) that try and convey some of their features to the user. However, those tips are commonly irrelevant to the user and are more of an annoyance than a help.



Example of the complicated layout of IDEs

Tutorials for IDEs from the IDE developers seem to be more in the form of API documentation than a fleshed out tutorial. Third party tutorials are published in the form of long videos, cumbersome webpages made to get the user to click to the next page, or "Top 20 best features for…." lists.
It's difficult to know a feature exists in an IDE for something that a developer would find incredibly useful, unless they stumble upon it, happen to read one of the tips of the day and

remember it, are told by a friend / colleague, or get frustrated with doing a tedious task and start googling for an easier way to accomplish it.

Approach

IDE Intelligent Tutorials would be a plugin available to the IDE for users. The most important part we are trying to achieve is providing relevant information at relevant times. That is to say, the plugin will detect when you are attempting to accomplish a task manually (or through the slow tedious way), when there is a feature that automates most of the process already. When certain actions from the user trigger the tutorial, it will pop up on screen and provide information to the user on a feature that would be beneficial to the user at that time.

For example, if a user wants to rename a variable and they start going through their code and manually renaming the variable each time it appears. After the second variable renamed, a pop up window could appear explaining about how you can quickly refactor the code to rename the variable throughout the program instead of searching for each occurrence manually. Or if a user uses the find functionality to locate where a method is called throughout their program, the IDE Intelligent Tutorial would detect that and provide information regarding how to look up all occurrences of the method throughout their project. Or a user could be debugging and manually stepping through many lines of code, then the tutorial could inform the user about conditional breakpoints.

A longer term goal would be to include this plugin as an easy to modify tutorial system, so others could easily add in their own tutorial information with the associated trigger to have that information appear. This would allow tutorials on new IDE features to be added, as well as other plugin tutorial information.

Challenges and Risks

The single most serious challenge in developing the product on schedule would be the limitations involved in how much information the IDE can provide to plug in support. Can we keep track of the user's clicks, searches, and other behavior in a low resource manner? Most of the tutorial triggers will need to be set up in clever ways as well. Not everyone uses an IDE the same way and everyone has different ways they like to code, develop, and debug.
A less immediate challenge, but still relevant, is creating tutorial information that is not annoying and cumbersome. Tutorial pop up windows can be annoying and headache inducing. We want to focus on the approach of providing relevant information at the relevant time to provide a benefit to the user.

Time Spent

This takes into consideration the time both partners spent
10-12 hours brainstorming and thinking of ideas
3-5 hours creating the documentation, powerpoint, and practicing the presentation