Take Advantage of powerful IDEA

Let IDEA know about your project

- Actually this page contains no useful information, only explaining why I'm not using my own project for presentation
- IDEA clearly knows projects using build tools(Gradle, Maven, …)
  but most of us do not use build tools
- MasterJH5574/Mx-Compiler will be used as an example
  instead of wu-qing-157/Rogue-Mx
  (because Rogue-Mx is not written in Java, and uses Gradle)

Let IDEA know about your project

- Set up your SDK(maybe Java 11) in project structure
  Note: language level no greater than SDK version
  in File > Project Structure
- Mark your source root, exclude irrelevant directories…
  with Right Click > Mark Directory as
- Tell IDEA where are your external libraries(maybe antlr-4.7.2-complete.jar)
  with Right Click > Add as library
  or in File > Project Structure > Libraries

Writing your code

- IDEA can perfectly handle the header part(package and imports)
- If any mistake happens, move the cursor to the part and
  use "Ctrl + F1" to see error info
  use "Alt + Enter" to quick fix (This is an extremely useful keymap)
- If you forget to cast the type, it's annoying to move the cursor here and there.
  Just type ".cast" and accept the code completion!
  Also works for ".if", ".for", ".call", ".new", ".return", ".var"…
  Find a full list by searching "Postfix Completion" in Settings
  (Using ".var", ".for" can also help you infer the variable type)

Let IDEA do some simple work

- Use Code > Generate to generate getters, setters, overriding methods…
  Keymap: Alt + Insert
  (But there is no "Insert" on my keyboard!)
- Reformat your code with "Ctrl + Alt + L"
  Configure the code style in Settings! (search for Code Style)

Learn about what you are using

- In the completion list, or move the cursor to where you need
  Use "Ctrl + Q" to have a quick view,
  use "Ctrl + Shift + I" to get detailed signature
- "Ctrl + Click" on something to jump to the definition.
  "Ctrl + Click" on a definition will show you the usage
- Click on "O"(subclass) or "I"(implementation) will jump to the corresponding definition.

Editing your code

- If you need to change the definition of something:
  first rule: check whether IDEA can do this before directly editing the code
  IDEA can find the usages, either modify the usages together, or inform you if automatic modification is impossible
- Use the options in "Right Click > Refactor", and in "Alt + Enter"
  "Shift + F6" to change the name
  "Ctrl + F6" to change the signature
  "Alt + Enter" on "," to flip two parameters

Search and multicursor

- "Ctrl + F" to search,
  select something then "Ctrl + F" to search that thing,
  use "search in selection only" to limit the search area,
  use "select all occurrences" to edit them together
- Pressing "Ctrl + Shift + Alt" when "Click" the code will add multicursor!
  (and Zoom will then be focused)

Executing your code

- First time: press the “run” button on the left of your main class
- Alternative: add a custom configuration!
  Your may change the command line arguments

Debugging your code

- Use breakpoints,
  conditional breakpoints and Java Exception breakpoints
- Step into, step over, step out
- Inspect the stack trace!
- Add watchers
  Quickly evaluate expressions with "Alt + Click"
  Warning: pay special attention to functions having side effects

Version control

- Git: IDEA supports Git and GitHub!
  Add your GitHub account in settings(search "GitHub")
  yellow files: ignored by git, red files: not tracked by git(use "Ctrl + Alt + A")
  How to commit, how to push? Maybe in "VSC (> Git)"
  (Actually I always use what is on next page to perform Git operations)
- Local History: independent of Git
  (it will consider Git operations as External Change)
  Sometimes useful because it is easy to use and saves versions automatically

Too much for me to remember!

- Use "Help > Find Action" or "Ctrl + Shift + A" to search for an action!
  Also you can search for a class, for example, when you want to learn about "java.util.ListIterator"
- Use "Help > Keymap Reference" to generate a cheat sheet for keymaps
  Also you can use the plugins in next page
  ("Editing" part is very useful)
- Search what you want in Settings instead of manually searching it

Plugin recommendations

- Statistic: LRH has written 13299 Java source lines!
- Presentation Assistant: demonstrate what you have done and the keymap of it (actually this plugin aims to help presentation like what I'm doing, so it always uses default keymap)
  Force Shortcut: force you to use keymap if available (if switched off in "Tools > Force Shortcut", still tell you the keymap) (will show the keymap according to your custom keymap)

Summary

- “Ctrl + Shift + A” to search actions!
- “Alt + Enter” to solve many problems!
- Modify definition with IDEA’s refactoring!
- “Ctrl + Q” to have a quick view!
- “Ctrl + Click” to jump to definition/usage!
- “Alt + Insert” to generate code!
- “Ctrl + Alt + L” to reformat code!
- “Alt + Click” to evaluate expressions when debugging!
- Believe IDEA is powerful!

Questions

- Any question about:
- IDEA
- Java
- Compiler Design (until IR Gen)