[Handout for L1P2]

Prime Your Primary Weapon: Introducing IDEs

IDEs (Integrated Development Environments)

Professional software engineers rarely use notepad-like primitive editors. They use more sophisticated tools called *Integrated Development Environments (IDEs)*. The aim of an IDE is to support all development-related work within the same tool environment.

An IDE generally consists of:

- A source code editor – includes features such as syntax coloring, auto-completion, easy code navigation, error highlighting, and code-snippet generation.
- A compiler and/or an interpreter together with other build automation support – allow us to compile/link/run/deploy a program easily.
- A debugger – allows to locate errors.

In addition, IDEs can include features such as automated testing support, drag-and-drop UI building, version management support, and modeling support.

Examples of popular IDEs: Eclipse, NetBeans, Visual Studio, DevC++, Dr Java, XCode

Note: Some experienced developers, those from a UNIX background in particular, prefer lightweight yet powerful text editors with scripting capabilities (e.g., Emacs http://www.gnu.org/software/emacs/) over heavier IDEs.

Debugging

Debugging is the process of discovering defects in the program. There are several approaches to debugging.

- **By inserting temporary print statements**: This is an ad hoc approach in which we insert print statements in the code to print out internal information such as the value of a variable. While this approach is simple, it takes too much extra work of inserting and removing print statements. Modifying code unnecessarily increases the risk of introducing errors accidentally and runs the risk of forgetting to remove some of those print statements which could then make their way to the production version. Therefore, this approach is not recommended.
- **By manually tracing through the code**: Otherwise known as 'eye-balling', this approach has its merits. However, it is difficult, time consuming, and error-prone technique. For example, the same faulty concept that made you create the bug in the first place will often make you miss it when you are looking for it later.
- **Using a debugger**: A debugger tool lets us pause the execution, execute it one statement at a time, while examining internal state as necessary. Most IDEs come with an inbuilt debugger. This is the recommended approach for debugging.

---End of Document---