2019/04/10 HW 06

20150497 위희주

“Growing a Language by Guy L. Steele Jr.”

* If you want to build a language, you must start from small language and make a language larger. Thus, you need to design(= to build a thing in one’s mind but not in the real world yet) a language that can grow and to leave some choices that users can implement what they need in real work site.
* The language design is not design a thing. It is designing a pattern for growing the pattern for defining the patterns that would be used in real work and designing a tool for making more tools for them. So the parts of the language must be designed to help the task of growth. Also, we need some groups who judges and testing the new codes users made, so other users can use it with credibility. And you have to know that some parts of the programming vocabulary are just for someone who need. Thus, building a working vocabulary to make good PL.
* Don’t forget that short words work well in everywhere. Thus, the programming languages need to be more like as the language we use to speak, so the programming languages have to be short.
* Java is growing now, also it is a huge language, but user can learn it step by step if he need. So do not worry that if the user cannot user the grown language because it became too big.

<Examples of grown languages>

* + Fortran has grown and grown. Now Fortran is not same as the language they first came to know. It grew too much.
  + PL/I has not grown much
  + Pascal grew just a bit (that strings were hard to use because they had all fixed size)and after that it was used to build many large programs.
  + C grew out of a smaller language called B, and has since grown to be a larger language called C++. A language as large as C++ could not have spread so wide if it had been came out on the world all at once.
  + User will not wait for “the right thing” that small language will have after several changes; However, if language designer lets the users help do the work, growth of the language can be quick. If many persons work side by side, and the best work is added with care and good taste, a great deal can be added in a short time
  + APL was designed by one man, a smart man. But in APL, new words defined by the user do not look like language primitives at all. The name of a piece of user code is a word, but things that are built in are named by strange words.
  + However, in Lisp, new words defined by the user look like primitives and all primitives look like words defined by the user. It is not used quite as much as it used to be, but parts of it live on in other languages such as garbage collection
  + Linux: put the source code out and let all persons play with it. Have a person in charge who is a quick judge of good work and who will take it in and shove it back out fast
* A library = a vocabulary designed to be added to a programming language to make the vocabulary of the programming language larger. The new words defined by a library should look just like primitives of the language, so user can grow the language easily.
* a pattern = a plan that has some number of parts and shows you how each part turns a face to the other parts, how each joins with the other parts ,or how each part does what it does and how the other parts aid it or drag it down, and how all the parts may be grasped as a whole and made to serve as one thing, for some higher goal or as part of a larger pattern. A good pattern will say how changes can be made.
* A method = a named piece of code that is part of an object
* A generic type = a map from one or more types to a type.
* An operator = a glyph, such as a plus sign, that can be used in a language as if it were a word.
* The Java programming language has objects and classes and fields and methods and types.
* An operator can be overloaded in C++, but the operators in the Java programming language cannot be overloaded by the programmer, though names of methods may be overloaded
* Meta means that you step back from your own place. Ex) A meta foo is a foo in whose slots you can put foos. (차원이 낮은 제 1의 언어를 대상언어라고 하며, 대상언어에 대해 다시 한 번 언급하는 언어를 메타언어라고 한다. 예를 들면 그 언어의 참 거짓 반별이 있을 수 있다)