**1.7.1 Compilation**

The compiler method translates programs into machine language that can be directly executed on the computer. This has the advantage of being able to execute the program very fast after it is translated. The most common languages that use this method are C, COBOL and C++. The translated language is called the source language and takes place in several phases. I have gone over these phases in a three page paper during week one. The disadvantage of this method is that there are so many steps to it. In my opinion errors can occur during any of these steps and it leaves lots of room for error. But this does not often happen. The speed of compilation is the key factor of this method and the only constituent is the architecture of the computer.

**1.7.2 Pure Interpretation**

Pure interpretation is the opposite of compilation. Programs here are interpreted by the interpreter and there is no translation whatsoever. The interpreter acts as a software simulation of a machine whose cycle deals with the high level language program statements instead of machine code. The software simulation is a virtual machine for the language. The advantage of this code is that it makes debugging simpler and in real time. It does this because the errors can only be from source level units. The disadvantage here is that it is 10 to 100 times slower than compilation. The slowness comes from interpreting complex and high level language statements instead of simple machine code. It also has to be decoded everytime which is a common bottleneck between the processor and the memory. Another disadvantage is that it often requires more space. The symbol table must be present during all interpretations. The source code must be present for easy modification instead of minimization like in compilation.

**1.7.3 Hybrid Implementation Systems**

Some language implementation is split between compilation and interpretation. The compilation step makes the code into a form that is easily interpreted. This is faster than pure interpretation because the source code only needs to be decoded once. This method is called the hybrid implementation system. Java was initially hybridized. The intermediate code is called bytecode and an advantage of this is that it can be portable to any machine. There are many different hybrid forms. Some translate the intermediate code to machine code. Some fix errors in different points of the process.

I think that the interpreter is my favorite way. It is the least complex but takes the longest. The next would be hybrid because this is often how Java runs and I really like using Java. I feel as though the most experience I have is with compilers and they have worked great for me so far but I do easily understand the process. There are so many comparison steps and analyzers and tables it feels too difficult to think about. All of these methods have their pros and cons but if you're looking for a certain aspect like speed or simplicity you'd have to choose between the first two. If you want a combination, the last one is the best.