Marco Karier
Pre-Task 2

Regular Grammars:

Basically a regular grammar is a set of rules and associations. Rules similar to B-→bA even though those variables are not yest defined it is a whole set of associations used to create a form of a language. The definition is not as useful as understanding the actual point of this concept. The point of creating these association rules to make a language that other people can repeat and actually use. For example, when a little kid makes up it's own language it will speak gibberish and claims that the gibberish means something in his native language. However, that kid won't repeat the words and meaning therefore it wont become a real language. Regular grammars are used to make sure that strings or words will be created in the correct way that it can actually be used. This is mostly for programming languages because in order for computer code to be parsed and run on the computer, it has to be written in the correct way for computers to process it. For example, Java is based on c and uses some c libraries but Java is an object oriented language where as c is an assemble language. In order for those 2 to communicate when you write in Java, Java to parse that string and translate it to a c language string so the computer can run the language. That is why we have a compiler. The compiler makes sure that the strings in the file are written in the correct way that it can be used properly.

Context-free Grammars:

Is a recursive system to rewrite code. There are 4 components of Context free grammars. Terminal symbols, non terminal symbols, productions, and start symbol. Context-free grammars will start at the start symbol in the code take an expression stop when it reaches something like +,-,*, and / symbols. The symbols have to be able to be represented by numbers so that the context-free grammar and process the information. An expression for example can be represented by number then combine with the mathematic termination symbol to create a whole new expression that is represented by a number. This is how a computer is able to perform the math calculations even though it is very basic math. But this is how computer code like Java can be translated into computer language. However, it is mostly when writing a mathematical formula in a code language you have to be able to use parenthesis to group processes together otherwise the number returned will be the wrong number. For example, in math multiplication or division will be performed before addition or subtraction. However, there are other times when you want to have addition and subtraction performed before multiplication and division so to make that happen you have to use parenthesis.

Domain (or application)-specific languages:

Is a specialized computer language to for a particular application. This includes languages like mark up languages like HTML which are designed only for web pages. For example, I could use c to create a website server and have it out put a website code, or I could use c to make a calculator or anything else my mind can come up with. Alternatively, HTML can only be used for web pages and nothing more. I wouldn't be able to create a calculator out of HTML code, because it lacks most the features that a normal computer language could use to make a calculator. Domain specific languages are created to solve a certain problem. When the Internet was first created, every website had their own languages and only certain web browser could load certain website. To standardize the websites web developers standardize HTML as the only language for websites so everyone could use it and that all web browsers could open any web page. That is why HTML can only do one thing and nothing more. Otherwise web browsers would have to start using accepting different features and then you would be back to where you started. MATLAB is another example of DSL because it is designed to solve a specific problem for people, in engineering/science fields. Another is the different Game Maker

languages because some games have to be compatible with a game engine therefor, you want to limit the user to a specific language or features to make sure that things work.

JavaCC

It is a lexicographical and parser that was written in Java so you can then compile your own language that you have created. As we looked at the different types of grammars and domain specific languages, each one is very unique and not always widely used. In fact some of those languages aren't used out side that project. However, you still need to be able to use compile that new language and run it. That is why JavaCC was created. It is a Java language program that lets you create a compiler and you can then create use it to compile the code that you have created. The alternative to JavaCC is Yacc which is a bottom up parser and JavaCC is a top down compiler. This way you can create code rules of associations and how it will be used. JavaCC allows the users to tokens and start/stop characters and even create a look ahead so that the compiler will look to ahead in the code to see how the variable is going to be used to make sure that the code is properly wriftten. However, JavaCC can't do left recursive associations. This means you can't create rules like $b \rightarrow a$ and $a \rightarrow c$ where to make this work you need b to create a before you can perform the second rule.

Reference:

https://www.cs.montana.edu/ross/theory/contents/chapter02/green/section05/page04.xhtml
https://www.cs.rochester.edu/~nelson/courses/csc_173/grammars/cfg.html
https://en.wikipedia.org/wiki/Domain-specific_language
http://cs.lmu.edu/~ray/notes/javacc/