Programming Convention Rules:

1. You must have the starting curly bracket at the start of a new line under the method name, class name, interface name, loop, or if-statement.

Example:

```
public class ExampleClass
{
        public static void exampleMethod()
        {
                (Code here)
        }
}
```

2. Always have at least one empty line between blocks of code and other blocks of code.

Example:

```
public static void methodOne()
{
        (Code here)
 }
//Empty line here
public static void methodTwo()
{
        (Code here)
}
```

3. No ternary operators (? and :). Always use the common if-statement structure.

4. Even if a method, loop, or if-statement only use one line, always use curly brackets.

Example:

```
if ( someBoolean )
{
        (Code here)
}

while ( someOtherBoolean )
{
        (Code here)
```

```
}
```

5. Return statements should return single variables with nothing else in that line.

Example:

```
public static int exampleMethod( int num )
{
        num *= 5;

        return num; // Do not add anything else to this line
}
```

6. You must comment all blocks of code. Methods, classes, constructors, if-statements, loops, and interfaces must have a separate comment directly above the block of code. All comments should have proper grammar, spelling, and punctuation. You must use javdocs for all functions, classes, and interfaces and have single or multi line comments within those. Place single and multi line comments to the side of a single line of code or place a general comment over a group of line of code. If you have to scroll sideways to view a comment, use multiple lines.
Commenting is ALWAYS required for all parts of your code.

Example:

```
/**
*Javadoc info here
*/
public class ExampleClass
{
        //Description of instance variables
        boolean someBoolean;
        boolean someOtherBoolean;

        /**
        *Javadoc info here
        */
        public static void methodOne()
        {
                //Description of loop.
                while ( someBoolean )
                {
                        (Code here) //Description of Code
                }
```

```
            //Description of if-statement.
            if ( someOtherBoolean )
            {
                  (Code here) //Description of Code
            }
      }
}
```

7. The iteration variable in for loops should be named ‘i’. If you have nested for loops, then the variable should be named ‘j’, then ‘k’, then ‘l’, etc. You probably should not have more than 3 nested *for* loops though.

8. All instance variables must be written at the top of a class, method, or interface. Group variables by their use and place comment above the group of variables to describe what the group of variables are used for or place comment on the same line as a variable to describe the variable individually. Groups of variables should be separated by a space. All primitive variables should be placed above object variables. All instance variables within a class should be private variables. **Variables should ALWAYS have descriptive names.** See naming conventions for more help.

Example:

```
public class ExampleClass
{
      //The position of the object.
      private int xPosition;
      private int yPosition;

      private int acceleration //The acceleration of the object.

      private Talon motorOne //A Object variable example
}
```

9. Do not compare a boolean to true and false using ==

Example:

```
public void someMethod()
{
      while(aBoolean)//Do NOT ever use aBoolean == true
            if(!anotherBoolean)//Do NOT use anotherBoolean == false
                  //Some Code
```

}

10. There should be space before and after assignment(+=), boolean(==), and math(-) operators and String concatenations (“ “ + “ “). There should be no space before or after incrementing operators(++ or --).

Example:

```
public class exampleClass
{
        public void doSomeMath()
        {
                int someVariable = 3
                int anotherVariable = 20;
                someVariable += anotherVariable;
                anotherVariable++;
        }
}
```

11. Different groups of code that do similar thing should be set off by empty lines and should have a comment describing them.(ie. your instance variables should be set off from your methods).

Example:

```
public static int exampleMethod( int x, int y )
{
        //The returned variable
        int sum;

        //Multiplies variables by 2
        x *= 2;
        y *= 2;

        //Adds variables together
        sum = x + y;

        return sum;
}
```

12. In all loops and if-statements, place one space between the statement and the left parenthesis.

Example:

```
if/*A single space*/( someBoolean )
{
        (Code here)
}
while/*A single space*/( someOtherBoolean )
{
        (Code here)
}
```

13. Parameters in methods and booleans in if-statements and loops should have a space in front and behind it.

Example:

```
public static void someMethod(/*A single space*/boolean someBoolean/*A single space*/ )
{
        while (/*A single space*/someBoolean/*A single space*/)
        {
                if (/*A single space*/someBoolean/*A single space*/)
                {
                        someBoolean = false;
                }
        }
}
```

14. Aim for simplicity in your code. The simpler your code is the better.

15. Try and Catch are NEVER allowed. Fix the problem, don't compensate for it.

16. With Switch statements, you have each case tabbed over once from the switch, and the statements after the case are tabbed over once from the space Also, there should be an empty line between a case's code and the case below it.

Example:

```
switch ( someInt )
{
        case 0:
                (Code here)

        case 1:
                (Code here)
}
```

Naming Conventions:

a) All constants should be written in all capital letters.

b) Always use camelCase for all names except for classes and interfaces.

c) All class and interface names are Capitalized.

d) When naming interfaces, classes, methods, and variables, use descriptive names. (i.e. Instead of naming a variable x, you would name it xPosition)

e) When coming up with names, always assume the person reading your code has no idea what the code should do.

f) Instance variables for classes and interfaces should have an underscore("_") before the actual name.