Overloading the Insertion and Extraction operators

The insertion operator, <<, is written and overloaded in the class **ostream**, while the extraction operator, >>, is written and overloaded in the class **istream**. The **ostream** class is used for output and the **istream** class is used for input. The variable or object *cout* is an instance of the class **ostream** and *cin* is an instance of the class **istream**. Given "int X = 5;" then "*cout* << X;" will display the number 5 on the screen. Both of these operators return a reference back to their corresponding streams, this will allow cascading of these operators, i.e. "*cout* << X << Y;"

When overloading any operator to work on a user-defined class and the left hand operand **is not** of the user-defined class then, then the overloaded operator function must be made a **friend** of that class. This is always the case when overloading the insertion or extraction operators, because the left hand operand is always from the class **istream** or **ostream**, and the right hand operand may be of a built in data type or a user-defined type or class.

Making a function a **friend** to a class allows the function to have access to the private data inside the class.

Syntax - prototype of overloaded insertion operator, must be put inside class specification part:

```
friend ostream& operator<< ( ostream &OutputVariableName,
                             const ClassName &ClassVariableName)
```

Syntax - prototype of overloaded extraction operator:

```
friend istream& operator>> ( istream &InputVariableName,
                             ClassName &ClassVariableName )
```

Syntax - function body for overloaded insertion operator, (extraction operator similar)

```
ostream& operator<< ( ostream &OutputVariableName,
                      const ClassName &ClassVariableName)
{
  Display the class members as needed
  return OutputVariableName;
}
```

The meaning of the & in the returned data type for the functions is whenever an operator or function returns a *stream*, you must add an & to end of name for returned type. This means the operator or function will return a reference to something as opposed to returning the value. If returned type is a stream, you cannot return its value, because value may be entire keyboard or screen. You can only return reference to this object itself as opposed to returning the value.