```
public String MyArray[]; this is the string array for the file

public String InFile; string for reading the file one line to the next

public static Type InstructionType; for command type

public int AtLine; counter for file lines

public static int symbol=16; the symbol value
```

Parse(String OurFile)

Takes in the file as a string and starts the parsing process. The extra symbols that are not needed and then It places the file into a string array (`MyArray[]`) that will be used for the parsing process.

```
public Type Type()
```

does not require a input, returns the instruction type found so L_COMMAND, A_COMMAND, or C_COMMAND. You can use this to find the type of instruction given the return. An example of how to call this would be `parseobject.Type()== parse.Type.A_COMMAND` this could be used to tell if a command type is an A instruction.

```
public String symbol()
```

This is called if command is A or L, no input needed. It returns the string lable which is the command without the extra symbols like @ for an A instruction or replaces `\\((.*?)\\)` with re for Lables

```
public String Jump()
```

no input needed returns null if no jump is in the current command and if there is it returns a decimal that needs to be converted into binary instructions.

```
public String Destination()
```

no input needed returns null if no destination is in the current command and if there is it returns a decimal that needs to be converted into binary instructions.

```
public String Computation()
```

no input needed returns null if no computation is in the current command and if there is it returns a decimal that needs to be converted into binary instructions.

```
public boolean Commands()
```

returns true if there are more commands in the array from the file to walk through.

```
public  void Continue()
```

Increases the current line number by one to move onto the next command. Use alongside Commands as youll only want to call this method if more commands are found.