# Compiling Java Code Guide (last updated 2/18/2023)

In Java a key component of developing code is dealing with potential compilation errors that you may get while developing. As you improve you should be getting less compiler errors. For right now it is definitely gonna be an issue that you run into quite often. There are many different compiler errors that you will run into throughout the semester. This guide will not and cannot cover them all.

I have placed in the "Griffin's Course Materials" alongside this file a java file named "PhoneNumberCompile.java" it is recommended for your best practice that you load up the file and try to fix all the compiler errors prior to reading how to solve the common compiler errors in this document.

It is best to get used to fixing compiler errors now while the logic of the code you are writing is relatively simple, before long you may get frustrated when you have to deal with implementing complex logic AND fixing your compiler errors. Before going over these common compiler errors and essentially giving away the solution to the file I posted, I want to go over some ways for you to minimize the quantity of compiler errors and the frustration you have fixing them.

## 1. Incremental Development

Do not develop all your code and then click compile, this could lead to you receiving 20+ compiler errors and getting frustrated and may lead you to give up and delay the completion of the assignment. For example, rather than developing the entire CreditCard.java file at once. Develop one method at a time and click compile, this way any errors you get can be easily handled and you know where they are without reading too much of the JVM's output.

2. Have Good Style
---

I assume out of all of those reading this, about 50% of you rolled your eyes but it is true. A lot of issues arise from having bad style and the code you are debugging looks like a mess. In codio whenever you click '{' it automatically provides a '}' so all you have to do is click enter and you have proper brackets for the code block you are writing. Every time you enter a code block, click [TAB] and indent your code. See the following example, I will use both K&R along with Horstmann style since a growing number of students are preferring the latter for some reason.

```
public void check(){
      check1();
      check2();
 }

public void check()
{
     check1();
     check2();
}
```

See, in both versions the indentation is matching and the code looks clean.

[PLEASE PROCEED TO NEXT PAGE LEFT BLANK FOR FORMATTING]

## Reading the JVM Output

Now that we talked about ways to minimize the number of compiler errors and your frustration with them. Let's go over some common errors, along with how to interpret them and fix them.

Consider the following compiler error given from the Codio Terminal:

```
PhoneNumberCompile.java:38: error: illegal start of type
    if (first==0 || first==1)
    ^
PhoneNumberCompile.java:38: error: <identifier> expected
    if (first==0 || first==1)
             ^
PhoneNumberCompile.java:38: error: <identifier> expected
    if (first==0 || first==1)
                         ^
PhoneNumberCompile.java:39: error: <identifier> expected
            valid=false;
```

A common reason for this illegal start of type error is that the code presented is not within a valid code block, either you wrote the code outside the intended method or in some cases outside the intended class. Those should be your first checks to fix it. As you can see the JVM provides the line number on which the error occurred, this should help narrow the scope of where you are looking for issues.

Now consider the following compiler error given from the Codio Terminal:

```
PhoneNumberCompile.java:50: error: reached end of file while parsing
    }
     ^
1 error
```

The pretty much only reason this occurs is because you are missing a bracket, starting from the class's opening bracket if you click it and do not see its highlight that means it is missing its

corresponding closing bracket. This is how you should go about fixing that issue.

Consider the following pair of errors:

```
PhoneNumberCompile.java:27: error: cannot find symbol
        int areaCode=Integer.parseInt(n.substring(0,3));
                                      ^
  symbol:   variable n
  location: class PhoneNumberCompile
PhoneNumberCompile.java:35: error: cannot find symbol
        first=Integer.parseInt(n.substring(4,5));
        ^
  symbol:   variable first
  location: class PhoneNumberCompile
```

This is an error that I see a lot of Office Hours, if the JVM cannot find the symbol it means there is no variable declared with those names. For the latter of these two, the solution is simply to add the data type before the name so the fixed line is

```
            int first=Integer.parseInt(n.substring(4,5));
```

For the former, we know n is not declared, but a common thing people do is they reference an object's instance variables via the name given to the former parameter for the respective variable being passed into the constructor, please know that variable has local scope and is only used to help initialize the instance variables. Please use the name of the object's instance variables for its reference. The solution to those two instances are:

```
        int areaCode=Integer.parseInt(number.substring(0,3));
          int first=Integer.parseInt(number.substring(4,5));
```

Also once again the JVM tells you what lines these errors occur on, that should be the first place to look for trouble.

You will encounter more compiler errors over the course of 1004. This guide should help you become familiar with the different kinds of errors and how to resolve them. One last thing, if you are

developing code and using the compile and run button and get the following:

```
Error: Main method not found in class PhoneNumberCompile, please
define the main method as:
   public static void main(String[] args)
or a JavaFX application class must
extendjavafx.application.Application
```

This means your code has successfully compiled but there is no main method to run so execution cannot proceed and errors out as a result. If you reach this stage, go to the provided tester (or the tester you created) and run that file.

Best of luck on the future coding assignments and keep working hard on Homework 4!

-Griffin N