Lab04: Skills - Using a Debugger

***Getting Ready:*** Before going any further, you should:

# Setup your development environment.

1. Download the following files:

PhoneDriver.java

PhoneCard.java

to an appropriate directory/folder. (In most browsers/OSs, the easiest way to do this is by right-clicking/control-clicking on each [of the links above.)](https://users.cs.jmu.edu/bernstdh/web/common/labs/experimenting_unit-testing/using-objects/GiftCard.java)

1. If you don't already have one from earlier in the semester, create a project named eclipseskills.
2. Drag the file PhoneCard.java and PhoneDriver.java into the default package (using the "Copy files" option).
3. Open PhoneCard.java and PhoneDriver.java.

***Part 1.* Review*:*** This part of the lab will review a few topics related to object-oriented programming in Java.

* 1. In the main() method in the PhoneDriver class, what kind of objects are end, now, and start?

**new objects of the class "Date"**

* 1. In the main() method in the PhoneDriver class, what kind of object is card?

**A new object of the class "PhoneCard"**

* 1. Where is the code for the PhoneCard class?

**in PhoneCard.java**

* 1. Where is the code for the Date class?

**It lives in the build in java util**

* 1. Read the documentation for the Date (<https://docs.oracle.com/javase/7/docs/api/java/util/Date.html>). Make sure you find the documentation for the Date class that is in java.util. (There are several Date classes in the Java library.)

**Ok, done**

* 1. When you construct a Date object using the default constructor (i.e., the constructor that has no parameters), what properties will it have?

**It will contain the time stamp for the date when it was called, calculated down to the millisecond**

***Part 2.* Setting a Breakpoint**: One of the nice things about running an application in a debugger is that you can stop the execution at one or more pre-defined locations (called breakpoints). This part of the lab will teach you how.

1. Click on the tab containing PhoneDriver.java to make sure that it has the focus.
2. Right-click in line 33 of PhoneDriver.java and pull down to Toggle Breakpoint.
3. What happened?

**It inserted a break point on line 37**

1. Click on debug.png. This will run PhoneDriver and stop the execution at the breakpoint (i.e., line 33). Note: If prompted, allow Eclipse to enter the "Debug Perspective".
2. What happened?

**The program executed and stopped at the breakpoint.**

***Part 3. Checking State Information:*** Another nice thing about running an application in a debugger is that, once you stop the execution at a breakpoint, you can check state information (e.g., the value of attributes and variables). This part of the lab will teach you how.

1. Click on the "Variables" tab on the left side of the debug window.
2. Click on the "tree icon" next to "Locals" to expand it.
3. What is the current value of availableMillis?

**5999999**

***Part 4. Stepping Over Lines***: When running an application in a debugger, once you stop the execution at a breakpoint, you can continue the execution one "step" at a time. This part of the lab will teach you how.

1. Click on debug.png. This will run PhoneDriver again and stop the execution at the breakpoint (i.e., line 33).
2. Click on the step-over.png button.
3. What happened?

**it stepped over the breakpoint and into the next statement**

1. Click on the step-over.png button until the next if statement is highlighted.
2. What is the current value of availableMillis? (Hint: Look in the "Variables" tab. You may beed to scroll.)

**5399999**

1. Click on the resume.png button to run to the end of the application.

***Part 5: Stepping Into Lines:***So far, all of the "stepping" you have done has been in one method in one class. This is called "stepping over". You can also "step into" a line of code to see what happens there. This part of the lab will teach you how.

1. Click on debug.png. This will run PhoneDriver and stop the execution at the breakpoint (i.e., line 33).
2. Click on the step-into.png button.
3. What happened?

**It stepped into PhoneCard.java to the first variable declaration**

1. Click on the step-into.png button again.
2. What happened?

**It advanced one line**

1. Look at the call stack in the "Debug" tab. It tells you what class and method you are in and where this method was called from.
2. What method is currently being executed (and what class is it in)?

**PhoneCard.getAvailableMilliseconds() in the PhoneCard class**

1. What line is currently being executed?

**Line 71**

1. Where was this method called from?

**PhoneDriver(main) line 37**

1. Click on the "triangle icon" next to this to expand it.
2. What is the current value of balance?

**9.0**

1. Click on the step-over.png button.
2. Click on the "triangle icon" next to callNumbers to expand it.
3. What is the current value of callNumbers[0]?

**540-568-1671**

1. Click on the "triangle icon" next to callStarts to expand it.
2. What is the current value of callStarts[0]?

**Date()**

1. Why does it have that value?

**Because it is equal to the Date() object, which has multiple variables**

1. Click on the step-over.png button twice.
2. What happened?

**We came back out of PhoneCard and into PhoneDriver as the function in PhoneCard which was executing finished executing**

1. Add a breakpoint at line 46 in PhoneDriver.java (i.e., the line that constructs a Date).
2. Click on the resume.png button. This will run the application to the next breakpoint (i.e., line 46).
3. Click on the step-into.png button.
4. Why didn't the debugger step into the Date constructor?

**Date is an imported class member which isn't part of our project, so there is no code to step into so to speak**

1. Click on the resume.png button to run to the end.
2. Click on Window+Perspective+Close Perspective to close the "Debug Perspective".

***Part 6: Advanced Topics:***This part of the lab will help you use the debugger more efficiently.

1. How can you display all of the breakpoints?

**Go to the breakpoints tab in the debugger**

1. What is a conditional breakpoint?

**A breakpoint which only triggers under certain circumstances (example: when a variable = some value)**

1. How can you see variable references while debugging?

**In the variables tab of the debugger.**