# Lab04: Skills - Using a Debugger

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

1. Setup your development environment.

2. 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.)

3. If you don't already have one from earlier in the semester, create a project named eclipseskills.

4. Drag the file `PhoneCard.java` and `PhoneDriver.java` into the default package (using the "Copy files" option).

5. 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?

**End, now, and start are all Date objects**

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

Card is a PhoneCard object.

3. Where is the code for the `PhoneCard` class?

**The code for the PhoneCard class is in our folder along with the driver.**

4. Where is the code for the `Date` class?

**The code for the Date class is in the “util” library.**

5. 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.)

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

**When you create a date object It will contain the current date.**

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

**Repeated question?**

***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?

**A circle appeared on the line**

4. Click on 🐞. 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".
5. What happened?

**A blue arrow shows the highlighted line**

***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`?

**The current value of availableMillis is 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 🐞. This will run `PhoneDriver` again and stop the execution at the breakpoint (i.e., line 33).
2. Click on the ↷ button.
3. What happened?

**The arrow moves on to the next statement we can execute.**

4. Click on the ↷ button until the next if statement is highlighted.
5. What is the current value of `availableMillis`? (Hint: Look in the "Variables" tab. You may beed to scroll.)

**5399999**

6. Click on the ▶ 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 🐞. This will run `PhoneDriver` and stop the execution at the breakpoint (i.e., line 33).
2. Click on the ⤵ button.
3. What happened?

**The arrow moved to the next executable statement**

4. Click on the ⤵ button again.
5. What happened?

**The arrow now moves into startCall()**

6. 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.
7. What method is currently being executed (and what class is it in)?

**startCall() in PhoneCard class**

8. What line is currently being executed?

**Line 92**

9. Where was this method called from?

**This method is called from the main method of the PhoneDriver class**

10. Click on the "triangle icon" next to this to expand it.
11. What is the current value of `balance`?

**10.0**

12. Click on the ↷ button.
13. Click on the "triangle icon" next to callNumbers to expand it.
14. What is the current value of `callNumbers[0]`?

**540-568-1671**

15. Click on the "triangle icon" next to callStarts to expand it.

16. What is the current value of `callStarts[0]`?

**It should contain a reference**

17. Why does it have that value?

**Because  it can have different attributes**

18. Click on the ↩ button twice.

19. What happened?

**It returned to PhoneDriver**

20. Add a breakpoint at line 46 in `PhoneDriver.java` (i.e., the line that constructs a `Date`).

21. Click on the ▶ button. This will run the application to the next breakpoint (i.e., line 46).

22. Click on the ↷ button.

23. Why didn't the debugger step into the `Date` constructor?

**Because the debugger doesn’t have the code for the Date class.**

24. Click on the ▶ button to run to the end.

25. 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?

**By selecting the Breakpoints view**

2. What is a conditional breakpoint?

**A conditional breakpoint is a breakpoint that stops execution when a Boolean value changes**

3. How can you see variable references while debugging?

**By selecting the variables view**