

Mr G's Java Jive

#19: Dealing With Dates - Part 2

In the previous handout, I introduced you to getting dates, showing dates, and formatting dates. But at the time doing math with dates was beyond our abilities. There were a number of reasons for this. One was that dates were **objects** and so couldn't be worked on like normal **primitives**. Another was that the time information inside a Date object is stored in **milliseconds**, which requires the use of a **long** data type. But the third, and most important reason is that computers don't quite do date math the same way that we do intuitively.

Enough Problems for Several Months

If I ask you how long it was from **June 19, 2000** to **November 20, 2000**, you'll probably think for a few seconds and then say **five months and one day**. That's because to you it's one month from June 19 to July 19, a second one from July 19 to August 19, a third one from August 19 to September 19, a fourth one from September 19 to October 19, a fifth one from October 19 to November 19, and then one day from November 19 to November 20. It doesn't matter to you that June and September only have 30 days in them. **To humans, a month is the distance from a certain date in the first month to the same date in the next.**

To computers it's different. Once we were able to do some relative simple date math, Java would probably tell us that it was **154 days**. Well, that's correct, but how many **months** is that? That's what most people want to know.

How many months? Well, that depends on how big your months are. In 30-day months it would be **five months and four days**. In 31-day months (which there are more of) it would be **four months and 30 days**. Neither of these answers is the **five months and a day** that most humans would expect.

Getting the Time Span with TimeSpan

So now that you've seen the problem, let's solve it, using the `gatling.timeSpan` method in a new program called **SecondDate**. As usual, it will be based on **BigLoop**, so we can run it over and over again. And as before, we'll create a special method called **datemain** to do the work for us and be called over and over again from the **main** method. Here's what it looks like:

```
public static void datemain()
{
    //start datemain
    //variables
    Date date1 = new Date();
    Date date2 = new Date();
    String d1string;
    String d2string;
    int span;

    //date formats
    DateFormat lg = DateFormat.getDateInstance(DateFormat.LONG);

    //get input
    System.out.print("This program will tell you the amount of time ");
    System.out.println("between two dates.");
    System.out.print("\tPlease enter the first date: ");
    date1=gatling.getDate();
    System.out.print("\tPlease enter the second date: ");
    date2=gatling.getDate();
}
```

Code continued on next page

```

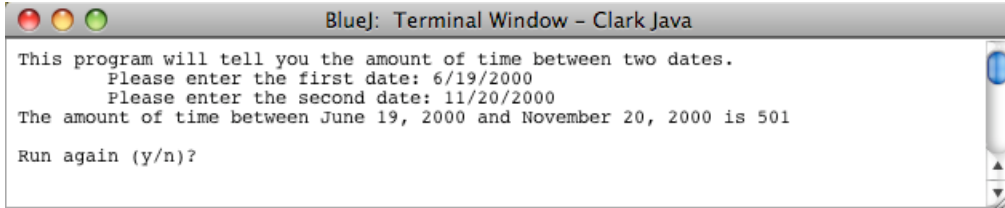
//do calculations and formatting
span=gatling.timeSpan(date1,date2);
d1string=lg.format(date1);
d2string=lg.format(date2);

//show output
System.out.print("The amount of time between "+d1string+" and "+d2string);
System.out.println(" is "+span);

} //end datemain

```

Now **compile** and **run** the program with the dates of **6/19/2000** and **11/20/2000**. Your output should look like the example below:

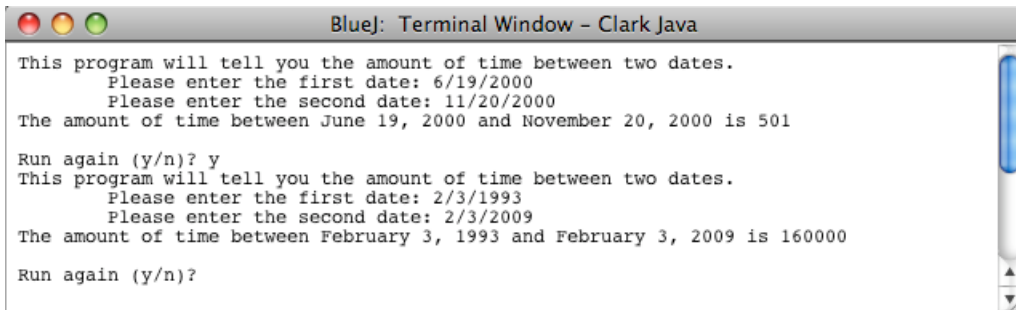


```

BlueJ: Terminal Window - Clark Java
This program will tell you the amount of time between two dates.
Please enter the first date: 6/19/2000
Please enter the second date: 11/20/2000
The amount of time between June 19, 2000 and November 20, 2000 is 501
Run again (y/n)?

```

The amount of time given is **501**. But 501 of what? It can't be 501 days, because we know that it's 154 days. It can't be 501 months or years either, because they're both way too much time. What is it 501 of? Let's try some more dates to find out. Now try **2/3/1993** and **2/3/2009**.

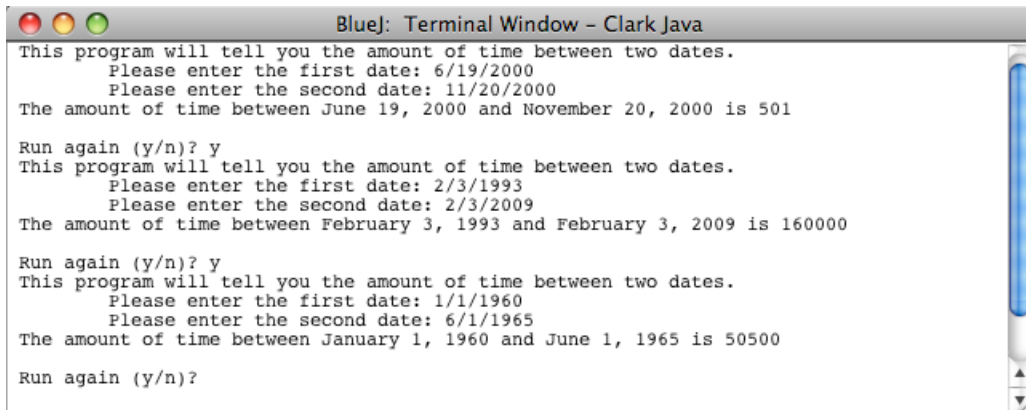


```

BlueJ: Terminal Window - Clark Java
This program will tell you the amount of time between two dates.
Please enter the first date: 6/19/2000
Please enter the second date: 11/20/2000
The amount of time between June 19, 2000 and November 20, 2000 is 501
Run again (y/n)? y
This program will tell you the amount of time between two dates.
Please enter the first date: 2/3/1993
Please enter the second date: 2/3/2009
The amount of time between February 3, 1993 and February 3, 2009 is 160000
Run again (y/n)?

```

This time it gives a time of **160000**. But again, what is it? It can't be 160,000 years. Again, that's obviously way too long. Let's do one more set. Try from **1/1/1960** to **6/1/1965**.



```

BlueJ: Terminal Window - Clark Java
This program will tell you the amount of time between two dates.
Please enter the first date: 6/19/2000
Please enter the second date: 11/20/2000
The amount of time between June 19, 2000 and November 20, 2000 is 501
Run again (y/n)? y
This program will tell you the amount of time between two dates.
Please enter the first date: 2/3/1993
Please enter the second date: 2/3/2009
The amount of time between February 3, 1993 and February 3, 2009 is 160000
Run again (y/n)? y
This program will tell you the amount of time between two dates.
Please enter the first date: 1/1/1960
Please enter the second date: 6/1/1965
The amount of time between January 1, 1960 and June 1, 1965 is 50500
Run again (y/n)?

```

This time it gives us **50500**. What do these numbers represent?

Deceptively Simple

That's right, the answer is deceptively simple. We knew when we got started that there were **five months and a day** between 6/19/2000 and 11/20/2000. What answer did the program give us? **501**.

A little quick math will tell you that it's exactly **16 years** from 2/3/1993 to 2/3/2007. What answer did the program give us? **160000**.

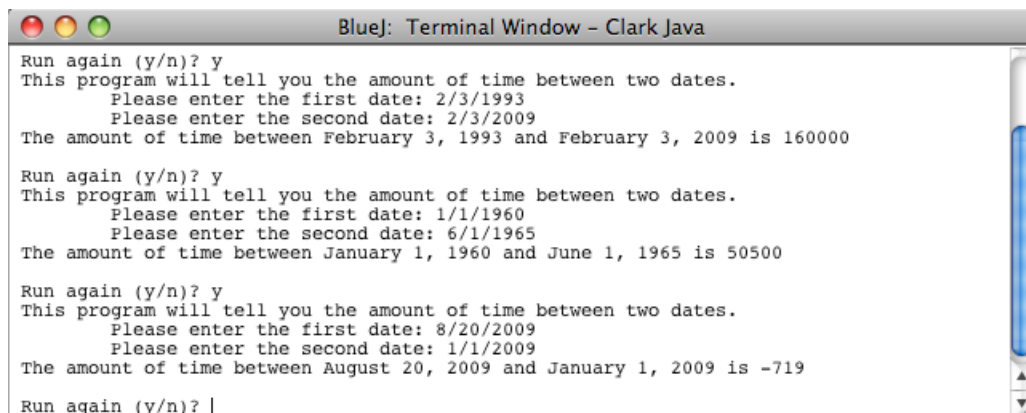
Finally, it's **five years and five months** from 1/1/1960 to 6/1/1965. The answer the computer gave us? **505000**.

The **last two** digits in the answer represent the number of **days**, the two digits before that represent the number of **months**, and any digits before that represent the number of **years** (if any).

Knowing this, it's clear to see that **501** was **five months and one day**, just what we expected it to be. **160000** was **16 years, 0 months, and 0 days**. **505000** was **five years, five months, and 0 days**. Does it make sense now?

Taking a Negative View

In the three examples we worked with so far, we started with the earlier date and went to the later one. But what happens if we start with the later date and then go to the earlier one? Let's find out with **8/20/2009** and **1/1/2009**.



```
Bluej: Terminal Window - Clark Java
Run again (y/n)? y
This program will tell you the amount of time between two dates.
Please enter the first date: 2/3/1993
Please enter the second date: 2/3/2009
The amount of time between February 3, 1993 and February 3, 2009 is 160000

Run again (y/n)? y
This program will tell you the amount of time between two dates.
Please enter the first date: 1/1/1960
Please enter the second date: 6/1/1965
The amount of time between January 1, 1960 and June 1, 1965 is 505000

Run again (y/n)? y
This program will tell you the amount of time between two dates.
Please enter the first date: 8/20/2009
Please enter the second date: 1/1/2009
The amount of time between August 20, 2009 and January 1, 2009 is -719

Run again (y/n)? |
```

This time the answer is a **negative** number. It's **-719**. What's the difference? In either case, the **absolute value** of the number is the time span between the two dates in years, months, and days. But a **positive** value means it's the time span from the first date **until** the second date, while a **negative** value is the amount of time that has **passed** from the second date to the first one. I guess if you look at everything as being a measurement of time **until** the second date, the negative number just means that you're going **backwards** in time.

More to Do (But Not Here)

So now that you know about **timeSpan**, you can finally write that program that tells the user **exactly** how old they are. But maybe that's a bit too much information. Maybe your user doesn't need to know that she's **270912**. Maybe it's just enough to tell her that she's **27 years old**. And perhaps the user who is **151119** and anxious to get his driver's license would rather have you say that he's **almost 16 years old**.

These are all things that you can do, and are all things that would make wonderful assignments for you, so you won't be seeing either of those problems solved on these pages.

What's Next?

In handout #20 we'll start working with **arrays**. They give you the ability to use groups of similar variables very slickly and efficiently.

This page intentionally left almost blank.