1. Log in
2. Unit 4
   - Writing Methods
   - Recursion - MORE
   - Classes and Objects

Oct 16-7:52 AM

Analyze this user-defined method: addEm(4) = ?

Answer: [redacted]

```
public static int addEm(int n)
{
   if(n==0)
      return 15;
   else
      return n + addEm(n-1);
}
```

Nov 5-8:02 AM

Step-by-Step Analysis ...

```
public static int minus(int n)
{
   if(n==1)
      return 1;
   else
      return n - minus(n-1);
}
```

minus(4) = ?

Nov 5-12:51 PM

Step-by-Step Analysis ...

```
public static int minus(int n)
{
   if(n==1)
      return 1;
   else
      return n - minus(n-1);
}
```

minus(4) = 4 - minus(3)
↓
3 - minus(2)

Nov 5-12:51 PM

Step-by-Step Analysis ...

```
public static int minus(int n)
{
   if(n==1)
      return 1;
   else
      return n - minus(n-1);
}
```

minus(4) = 4 - minus(3)
↓
3 - minus(2)
↓
2 - minus(1)

Nov 5-12:51 PM

Step-by-Step Analysis ...

```
public static int minus(int n)
{
   if(n==1)
      return 1;
   else
      return n - minus(n-1);
}
```

minus(4) = 4 - minus(3)
↓
3 - minus(2)
↓
2 - minus(1)
↓
1

Now the computer works backward (note: answer = 2)

Nov 5-12:51 PM

Here's a tricky one ...

```
public static void divide(int n){
    if(n<=1)
        System.out.print(n);
    else{
        divide(n/2);
        System.out.print(","+n);
    }
}
```

divide(9) =

Nov 5-12:56 PM

Here's a tricky one ...

```
public static void divide(int n){
    if(n<=1)
        System.out.print(n);
    else{
        divide(n/2);
        System.out.print(","+n);
    }
}
```

divide(9) = divide(4) **and then** print " ,9" //n=9

Nov 5-12:56 PM

Here's a tricky one ...

```
public static void divide(int n){
    if(n<=1)
        System.out.print(n);
    else{
        divide(n/2);
        System.out.print(","+n);
    }
}
```

divide(9) = divide(4) **and then** print " ,9" //n=9

divide(2) **and then** print ", 4" //n=4

Nov 5-12:56 PM

Here's a tricky one ...

```
public static void divide(int n){
    if(n<=1)
        System.out.print(n);
    else{
        divide(n/2);
        System.out.print(","+n);
    }
}
```

divide(9) = divide(4) **and then** print " ,9" //n=9

divide(2) **and then** print ", 4" //n=4

divide(1) **and then** print ", 2" //n=2

Nov 5-12:56 PM

Here's a tricky one ...

```
public static void divide(int n){
    if(n<=1)
        System.out.print(n);
    else{
        divide(n/2);
        System.out.print(","+n);
    }
}
```

divide(9) = divide(4) **and then** print " ,9" //n=9

divide(2) **and then** print ", 4" //n=4

divide(1) **and then** print ", 2" //n=2

print "1" //n=1

Remember, JAVA now compiles this "bottom-up"

Nov 5-12:56 PM

Here's how the divide( ) method gets compiled ...

Remember, JAVA now compiles this "bottom-up"

OUTPUT

divide(9) = divide(4) **and then** print " ,9"

divide(2) **and then** print ", 4"

divide(1) **and then** print ", 2"

print "1" 1

Nov 5-12:56 PM

Here's how the divide( ) method gets compiled ...

Remember, JAVA now compiles this "bottom-up"

OUTPUT

divide(9) = divide(4) **and then** print " ,9"

divide(2) **and then** print ", 4"

1 **and then** print ", 2" 1,2

Nov 5-12:56 PM

Here's how the divide( ) method gets compiled ...

Remember, JAVA now compiles this "bottom-up"

OUTPUT

divide(9) = divide(4) **and then** print " ,9"

1,2 **and then** print ", 4" 1,2,4

Nov 5-12:56 PM

Here's how the divide( ) method gets compiled ...

Remember, JAVA now compiles this "bottom-up"

OUTPUT

divide(9) = 1,2,4 **and then** print " ,9" 1,2,4,9

Final Output for divide(9) = 1,2,4,9

Nov 5-12:56 PM

Mathematics, The World Around Us, and Nature ...

Fibonacci: 1, 1, 2, 3, 5, 8, 13, 21, 34, ...

Golden Ratio ...

3/2 = 1.5

5/3 = 1.67

8/5 = 1.6

13/8 = 1.625

21/13 - 1.61538

Independent research if this intrigues you ...

Sequence found by adding the last two numbers in the pattern...

Nov 5-1:42 PM

Fibonacci: 1, 1, 2, 3, 5, 8, 13, 21, ...

I want to calculate the 4th number ...

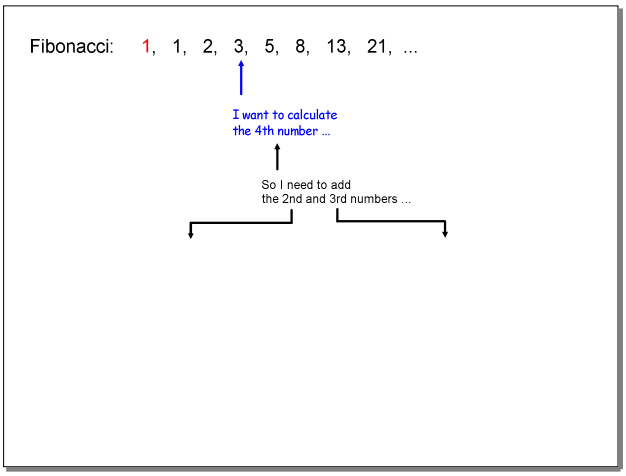
Nov 5-1:47 PM

Fibonacci: 1, 1, 2, 3, 5, 8, 13, 21, ...

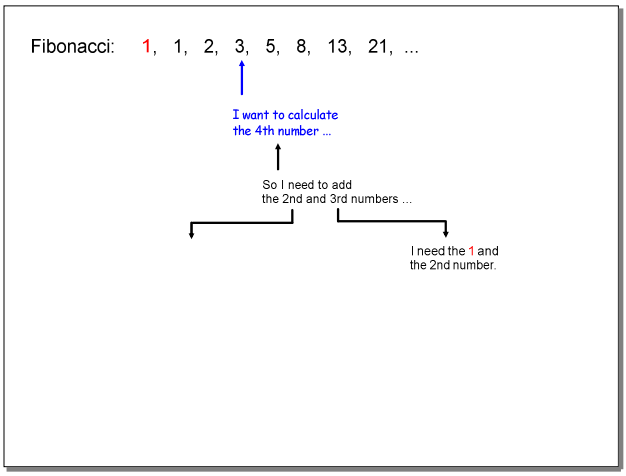
I want to calculate the 4th number ...

So I need to add the 2nd and 3rd numbers ...

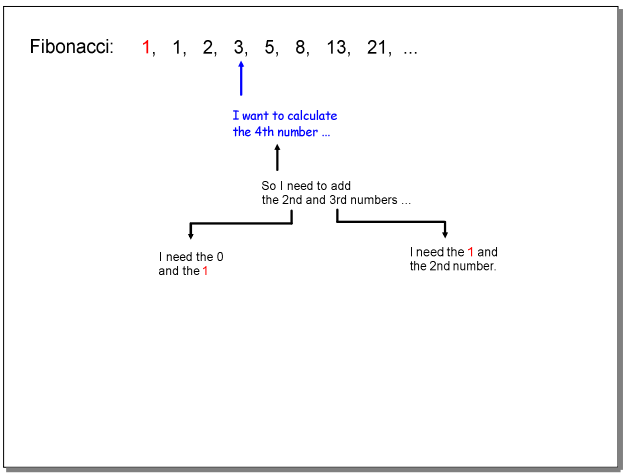
Nov 5-1:47 PM



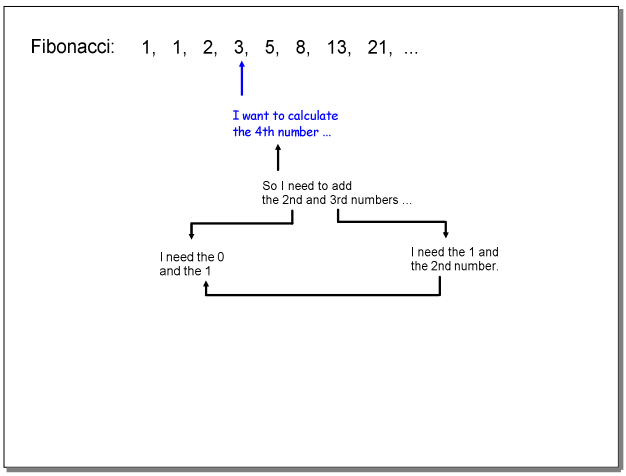
Nov 5-1:47 PM



Nov 5-1:47 PM



Nov 5-1:47 PM



Nov 5-1:47 PM

Here's a user-defined fibonacci method( ) ...

```
public static int fib(int n){
    if(n==0){
        return 0;
    }
    else if(n==1){
        return 1;
    }
    else{
        return fib(n-1)+fib(n-2);
    }
}
```

fib(4) = fib(3) + fib(2)

Nov 5-1:53 PM

Here's a user-defined fibonacci method( ) ...

```
public static int fib(int n){
    if(n==0){
        return 0;
    }
    else if(n==1){
        return 1;
    }
    else{
        return fib(n-1)+fib(n-2);
    }
}
```

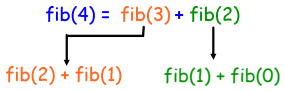
fib(4) = fib(3) + fib(2)

fib(2) + fib(1)

Nov 5-1:53 PM

Here's a user-defined fibonacci method( ) ...

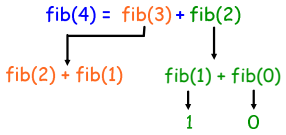
```
public static int fib(int n){
    if(n==0){
      return 0;
    }
    else if(n==1){
      return 1;
    }
    else{
      return fib(n-1)+fib(n-2);
    }
  }
```



Nov 5-1:53 PM

Here's a user-defined fibonacci method( ) ...

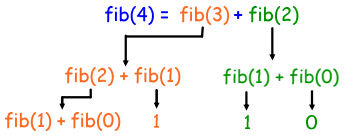
```
public static int fib(int n){
    if(n==0){
      return 0;
    }
    else if(n==1){
      return 1;
    }
    else{
      return fib(n-1)+fib(n-2);
    }
  }
```



Nov 5-1:53 PM

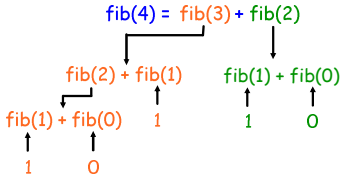
Here's a user-defined fibonacci method( ) ...

```
public static int fib(int n){
    if(n==0){
      return 0;
    }
    else if(n==1){
      return 1;
    }
    else{
      return fib(n-1)+fib(n-2);
    }
  }
```



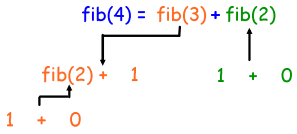
Nov 5-1:53 PM

To solve, remember JAVA compiles bottom-up ...



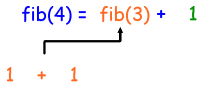
Nov 5-1:53 PM

To solve, remember JAVA compiles bottom-up ...



Nov 5-1:53 PM

To solve, remember JAVA compiles bottom-up ...



Nov 5-1:53 PM

To solve, remember JAVA compiles bottom-up ...

fib(4) = 2 + 1

So, then fib(4) = 3

Nov 5-1:53 PM

Example of recursion that doesn't work ...

Hope you have a computer with infinite memory space!

```
package unit04;

public class infiniteRecursion {
   public static int goodLuck(int a)
   {
      if(a==5)
         return a;
      else
         return a+goodLuck(a+1);
   }
   public static void main(String[] args) {
      System.out.println(goodLuck(3));          //easy, this is 12
      System.out.println(goodLuck(7));          //good luck! Infinite Math
   }
}
```

Nov 6-8:00 AM

Example of a user-defined method that doesn't return anything ...

```
package unit04;
public class voidReturnExample {
   public static void nothing(int z, int m)
   {
    if(z<m)  {
         for(int i=0 ; i<z ; i++)
            System.out.print("XO");
         System.out.println("");
         nothing(z+1,m);
    }
    else if (z==m)   {
        for(int i=0 ; i<z ; i++)
            System.out.print("-");
        System.out.println("");
        nothing(z+1,m);
    }
    else
        System.out.print("Process Complete"); //This is base case - no return
   }
   public static void main(String[] args) {
      returnNothing(7,10);                    //Notice, we aren't having
   }                                          //anything returned to the
}                                             //main method.  We are simply
                                              //having it follow directions.
```

Nov 6-8:00 AM

Things to do ...

1. Unit 4 WS 01 should be done.
2. Wrap up Unit 4 WS 02 Introduction to Recursion
3. Work on Unit 4 WS 03 More Recursion

Oct 16-9:12 AM