1. Log in
2. Unit 4
   - Writing More Methods
   - Introduction to Recursion
   - Classes & Objects

Oct 16-7:52 AM

Remember, we can write our own methods ...

```
package unit04;

public class unit2Day2 {

   public static double myMethod(int a, int b)
   {
      return (double)b/a;                          //review data types!
   }

   public static void main(String[] args)
   {
      System.out.println("Whenever I want to use myMethod, I just " +
            "call it like this...\n");
      System.out.println("myMethod(int,int)\n");
      System.out.println("For example, myMethod(2,5) = "+ myMethod(2,5));
   }
}
```

Nov 5-8:02 AM

Recursion: The process that occurs when a method calls itself. Beware, this could lead to an endless loop if not careful!

Other ways to think about recursion ...

- The need to repeatedly do the same thing.
- In order to continue the method needs to use itself
- Very powerful (yet dangerous) looping occurs!

Nov 4-7:21 AM

First Example of Recursion ...

Follow EXACTLY what count(2) does:
1. Do you see recursion?
2. Do you know the output?

```
public class unit2Day2 {
   public static int count(int a)
   {
      if(a==10)                                    //BASE CASE - what makes it stop!
         return 10;
      else
      {
         System.out.print(a+" ");
         return count(a+1);
      }
   }
   public static void main(String[] args) {
         System.out.println(count(2));
      }
}
```

Nov 4-7:21 AM

Mathematics: Factorial

5! = 5 * 4 * 3 * 2 * 1 = 120

Nov 5-8:33 AM

Mathematics: Factorial

5! = 5 * 4 * 3 * 2 * 1 = 120

*** This is recursion, but we have to think of it differently ***

Nov 5-8:33 AM

Mathematics: Factorial

5! = 5 * 4 * 3 * 2 * 1 = 120

*** This is recursion, but we have to think of it differently ***

5! = 5 * 4!
5! = 5 * (5-1)!

Nov 5-8:33 AM

Mathematics: Factorial

5! = 5 * 4 * 3 * 2 * 1 = 120

*** This is recursion, but we have to think of it differently ***

5! = 5 * 4!
5! = 5 * (5-1)!

*** and then ***

5! = 5 * 4 * 3!
5! = 5 * 4 * (4-1)!

Nov 5-8:33 AM

Mathematics: Factorial

5! = 5 * 4 * 3 * 2 * 1 = 120

*** So, in general, recursion thinking is ***

n! = n * (n-1)!

Nov 5-8:33 AM

Mathematics: Factorial

5! = 5 * 4 * 3 * 2 * 1 = 120

*** So, in general, recursion thinking is ***

n! = n * (n-1)!

*** To get factorial, multiply by another factorial ***

*** In other words, a factorial method is going to need itself ***

Nov 5-8:33 AM

Here's a factorial class with method ...

```
public class factorial {
  public static int factorial(int a)
  {
     if(a==1)                          // Base Case - makes it stop!
        return 1;
     else
        return a*factorial(a-1);
  }
   public static void main(String[] args) {
      System.out.println(factorial(4));
   }
}
```

Nov 5-8:49 AM

Here's a factorial class with method ...

```
public class factorial {
  public static int factorial(int a) {
     if(a==1)
        return 1;
     else
        return a*factorial(a-1);
  }
   public static void main(String[] args) {
      System.out.println(factorial(4));
}}
```

factorial(4) = 4 * factorial(3)

Nov 5-8:49 AM

Here's a factorial class with method ...

```
public class factorial {
  public static int factorial(int a) {
    if(a==1)
      return 1;
    else
      return a*factorial(a-1);
  }
  public static void main(String[] args) {
    System.out.println(factorial(4));
}}
```

factorial(4) = 4 * factorial(3)

factorial(3) = 3 * factorial(2)

Nov 5-8:49 AM

Here's a factorial class with method ...

```
public class factorial {
  public static int factorial(int a) {
    if(a==1)
      return 1;
    else
      return a*factorial(a-1);
  }
  public static void main(String[] args) {
    System.out.println(factorial(4));
}}
```

factorial(4) = 4 * factorial(3)

factorial(3) = 3 * factorial(2)

factorial(2) = 2 * factorial(1)

Nov 5-8:49 AM

Here's a factorial class with method ...

```
public class factorial {
  public static int factorial(int a) {
    if(a==1)
      return 1;
    else
      return a*factorial(a-1);
  }
  public static void main(String[] args) {
    System.out.println(factorial(4));
}}
```

factorial(4) = 4 * factorial(3)

factorial(3) = 3 * factorial(2)

factorial(2) = 2 * factorial(1)

factorial(1) = 1

*** JAVA finally finds an end to the looping! ***

!!!!! IMPORTANT !!!!!

Java now builds the solution starting at the end!

Nov 5-8:49 AM

Here's a factorial class with method ...

```
public class factorial {
  public static int factorial(int a) {
    if(a==1)
      return 1;
    else
      return a*factorial(a-1);
  }
  public static void main(String[] args) {
    System.out.println(factorial(4));
}}
```

factorial(4) = 4 * factorial(3)

factorial(3) = 3 * factorial(2)

factorial(2) = 2 * 1

factorial(1) = 1

Java now builds the solution starting at the end!

*** Recursion is dangerous ***
It uses more run-time memory AND has the potential to infinitely loop ... look at the information that must be stored during calculation.

Nov 5-8:49 AM

Here's a factorial class with method ...

```
public class factorial {
  public static int factorial(int a) {
    if(a==1)
      return 1;
    else
      return a*factorial(a-1);
  }
  public static void main(String[] args) {
    System.out.println(factorial(4));
}}
```

factorial(4) = 4 * factorial(3)

factorial(3) = 3 * 2

factorial(2) = 2 * 1

Nov 5-8:49 AM

Here's a factorial class with method ...

```
public class factorial {
  public static int factorial(int a) {
    if(a==1)
      return 1;
    else
      return a*factorial(a-1);
  }
  public static void main(String[] args) {
    System.out.println(factorial(4));
}}
```

factorial(4) = 4 * 6

factorial(3) = 3 * 2

Solved: factorial(4) = 24

Nov 5-8:49 AM

One last recursion example ...

```
public class unit4Day2 {
   public static int beware(int a)
   {
      if(a==5)                    //base case!
         return 2;
      else
         return 2*a-beware(a-1);
   }
   public static void main(String[] args) {
            System.out.println(beware(8));
         }
}
```

beware(8) = 2*8 - beware(7)

Nov 5-9:26 AM

One last recursion example ...

```
public class unit4Day2 {
   public static int beware(int a)
   {
      if(a==5)
         return 2;
      else
         return 2*a-beware(a-1);
   }
   public static void main(String[] args) {
            System.out.println(beware(8));
         }
}
```

beware(8) = 2*8 - beware(7)

2*7 - beware(6)

2*6 - beware(5)

2

Java now builds the solution starting at the end!

Nov 5-9:26 AM

One last recursion example ...

```
public class unit4Day2 {
   public static int beware(int a)
   {
      if(a==5)
         return 2;
      else
         return 2*a-beware(a-1);
   }
   public static void main(String[] args) {
            System.out.println(beware(8));
         }
}
```

beware(8) = 2*8 - beware(7)

2*7 - beware(6)

2*6 - 2

2

Java now builds the solution starting at the end!

Nov 5-9:26 AM

One last recursion example ...

```
public class unit4Day2 {
   public static int beware(int a)
   {
      if(a==5)
         return 2;
      else
         return 2*a-beware(a-1);
   }
   public static void main(String[] args) {
            System.out.println(beware(8));
         }
}
```

beware(8) = 2*8 - beware(7)

2*7 - 10

2*6 - 2

Nov 5-9:26 AM

One last recursion example ...

```
public class unit4Day2 {
   public static int beware(int a)
   {
      if(a==5)
         return 2;
      else
         return 2*a-beware(a-1);
   }
   public static void main(String[] args) {
            System.out.println(beware(8));
         }
}
```

beware(8) = 2*8 - 4

2*7 - 10

Solved: beware(8) = 12

Nov 5-9:26 AM

Things to do ...

1. Wrap Up Unit 4 WS 01 Method Structure/Creation
2. Complete Unit 4 WS 02 Introduction to Recursion

Oct 16-9:12 AM