Objectives: Build contacts app (OO programming); recursion intro;

Up next: Quiz this week; MP6 out - due in one week; Pre-lecture: readings and Touring's Craft drop off, online exercises take over.

1. **Factorial - the 'Hello World' of recursion:**

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

5! = 5 * 4!

5! = 5 * 4 * 3!

5! = 5 * 4 * 3 * 2!

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

5! = 5 * 4 * 3 * 2 * 1 * 0!

woah! 0! = 1 by definition

5! = 5 * 4 * 3 * 2 * 1 * (1)

5! = 5 * 4 * 3 * 2 * (1)

5! = 5 * 4 * 3 * (2)

5! = 5 * 4 * (6)

5! = 5 * (24)

5! = (120)

```
int myFactorial = factorial(5);
```

```
public static int factorial(int n) {
   fact = 1;
   for (int i = 5; i > 0; i++)
      fact = fact * i;

   return fact;
}
```

```
int myFactorial = factorial(5);
```

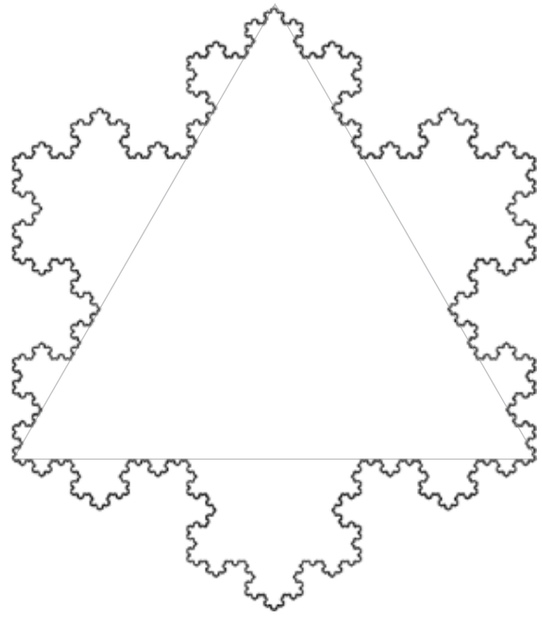
```
public static int factorial(int n) {




}
```

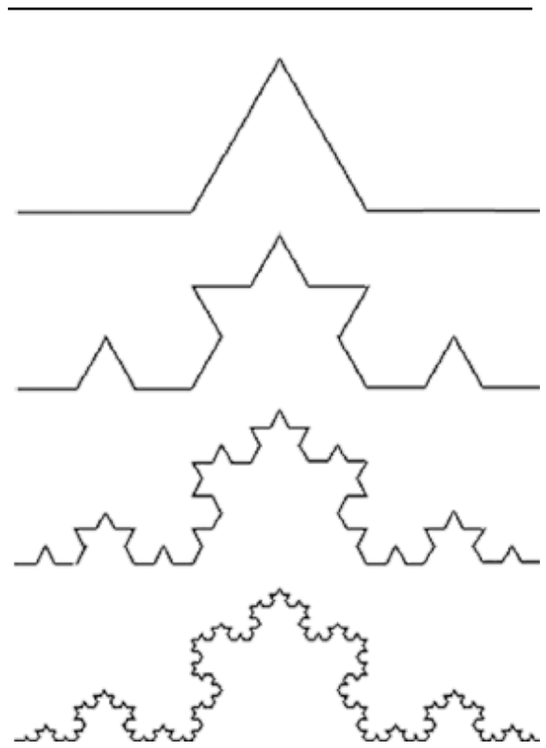
```
import java.util.Date;
public class Friend extends Person {

   // instance variable - friendsSince of type Date


   // constructor that takes a name and phone number
   // sets the date of becoming friends to now




   // constructor that takes a Person




   // toString method




   // printDescription


}
```

```
import java.util.Date;
public class SoulMate extends Friend {




}
```

1. **Recursion: allows us to define a function that calls itself to solve a problem by breaking it into simpler cases - "divide and conquer"**

recursion: method of defining problems in computer science; method of designing solutions to problems in computer science.





Learning recursion

Static: definitions. In code: magic!

Dynamic: taking the magic out.

2a. **Iteratively**

```
public class Example001 {

   public static void countdown(int n) {
      for (int i=n; i > 0; i--) {
         System.out.println(i);
      }
      System.out.println("Blast Off!");
   }

   public static void main(String[] args) {
      countdown(5);
   }
}
```

2b. **Recursively**

```
public class Example002 {

   public static void countdown(int n) {
      if (n == 0) {
         System.out.println("Blast Off!");
      } else {
         System.out.println(n);
         countdown(n-1);
      }
   }

   public static void main(String[] args) {
      countdown(5);
   }
}
```

Recursive algorithms composed of two cases:

1) **recursive case** calls the recursive procedure on a simpler case (usually a part of the input).
2) **base case** is <u>necessary</u> in recursion; it determines when the procedure returns a value (or terminates), rather than continuing the recursive process.

3. x^y

Raising a number to a power in java? x^y ???

No! Write two versions: iterative, recursive:

$$x^y = \underbrace{x * x * x * \ldots x}_{\text{y times}}$$

3a. **Iteratively**

```
public class MyMath {

 public static int power(int base, int exp)
  {









  }

  public static void main(String[] args) {
     System.out.println(power(10,3));
  }
}
```

3b. **Recursively**

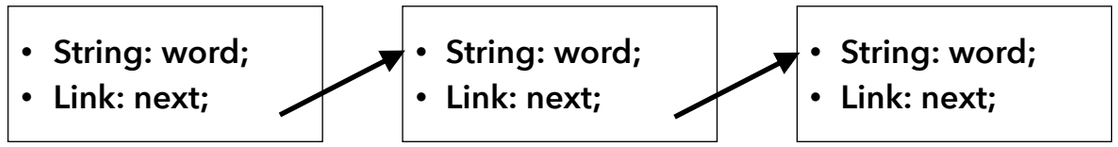
```
public class MyMath {

public static int power(int base, int exp) {









 }

 public static void main(String[] args) {
   System.out.println(power(10,3));
 }
}
```

$$x^y = x * x * x * \ldots x$$

y-1 times

4d. **Write a recursive instance method that returns the length of the list.**

Linked List



4a. **Write a java class to create a linked list.**

Write a java class to create a linked list.
Each Link object contains: i) an String 'word'
ii) a reference 'next' to refer to the next link in the chain.

ANSWER:

```
public class Link {
   private String word;
   private Link next;

   public Link(String w, Link n) {
      word = w;
      next=n;
   }
}
```

4b. **Write a recursive instance method that returns the word contained in the last link. (Hint: The last link's next reference is *null*):**

```
public String getLastValue() {
   if(next == null) // BASE CASE



      else



}
```

4c. **A main method to create a list and display the last link:**

```
public static void main(String[] args) {
   Link head = new Link("One", new Link("Two", new Link("Three",
      new Link("Four", new Link("Five", new Link("Six", null))))));
   String lastValue = ____________________________________;
   System.out.println(lastValue);
}
```

4e. **Write a recursive instance method to print a string representation of the list:**

4f. **Write a recursive instance method to return a string with all the words concatenated together:**