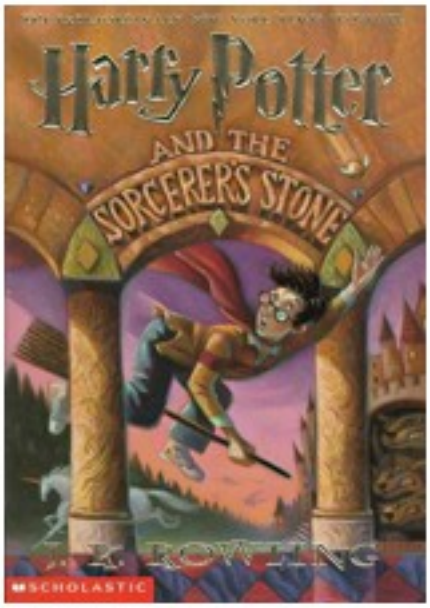
More recursion

What we are doing

- Intro to Markov (by request)
- More recursion
  - recursion()
    - recursion()



- Harry -> Potter
- Dark -> Lord -> Arts
- Magic -> Wand

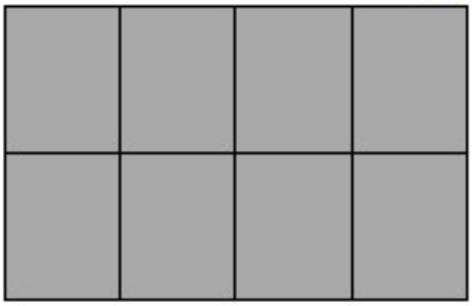
gy independence on something firmer, and more honest in our battlefields may be Democrats and Republican nominee, John McCain, I will stop giving them to companies stop discriminating against those wi

Brute-Force

- seed = random k-character substring from the training text --- the *initial seed*
- repeat N times to generate N random letters
  - for each occurrence of seed in training text
    - record the letter that follows the occurrence of seed in a list
  - choose a random element of the list as the generated letter C
  - print or store C
  - seed = (last k-1 characters of seed) + C

Brute-Force

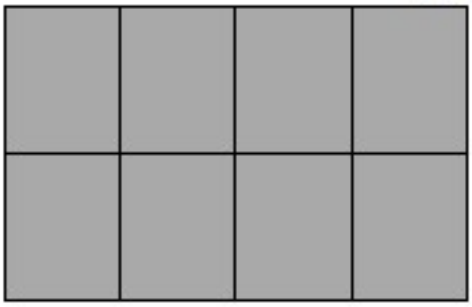
bbbabbabbbbbaba



```
public void brute(int k, int numLetters) {

        // pick random k-character substring as initial seed
        int start = myRandom.nextInt(myString.length() - k + 1);
        String seed = myString.substring(start, start + k);

        // copy first k characters to back to simulate wrap-around
        String wrapAroundString = myString + myString.substring(0,k);

        StringBuilder build = new StringBuilder();
        ArrayList<Character> list = new ArrayList<Character>();

        for (int i = 0; i < numLetters; i++) {
            list.clear();
            int pos = 0;
            while ((pos = wrapAroundString.indexOf(seed, pos)) != -1 &&
pos < myString.length()) {
                char ch = wrapAroundString.charAt(pos + k);
                list.add(ch);
                pos++;
            }
            int pick = myRandom.nextInt(list.size());
            char ch = list.get(pick);
            build.append(ch);
            seed = seed.substring(1) + ch;
        }
    }
```

Look at code on your computer so you can read it!!!!!!

Brute-Force



bbbabbabbbbaba

```
public void brute(int k, int numLetters) {

        // pick random k-character substring as initial seed
        int start = myRandom.nextInt(myString.length() - k + 1);
        String seed = myString.substring(start, start + k);

        // copy first k characters to back to simulate wrap-around
        String wrapAroundString = myString + myString.substring(0,k);

        StringBuilder build = new StringBuilder();
        ArrayList<Character> list = new ArrayList<Character>();

        for (int i = 0; i < numLetters; i++) {
            list.clear();
            int pos = 0;
            while ((pos = wrapAroundString.indexOf(seed, pos)) != -1 &&
pos < myString.length()) {
                char ch = wrapAroundString.charAt(pos + k);
                list.add(ch);
                pos++;
            }
            int pick = myRandom.nextInt(list.size());
            char ch = list.get(pick);
            build.append(ch);
            seed = seed.substring(1) + ch;
        }
    }
```

Look at code on your computer so you can read it!!!!!!

Brute-Force

bbbabbabbbbaba

```
public void brute(int k, int numLetters) {

        // pick random k-character substring as initial seed
        int start = myRandom.nextInt(myString.length() - k + 1);
        String seed = myString.substring(start, start + k);

        // copy first k characters to back to simulate wrap-around
        String wrapAroundString = myString + myString.substring(0,k);

        StringBuilder build = new StringBuilder();
        ArrayList<Character> list = new ArrayList<Character>();

        for (int i = 0; i < numLetters; i++) {
            list.clear();
            int pos = 0;
            while ((pos = wrapAroundString.indexOf(seed, pos)) != -1 &&
pos < myString.length()) {
                char ch = wrapAroundString.charAt(pos + k);
                list.add(ch);
                pos++;
            }
            int pick = myRandom.nextInt(list.size());
            char ch = list.get(pick);
            build.append(ch);
            seed = seed.substring(1) + ch;
        }
    }
```

Look at code on your computer so you can read it!!!!!!

Brute-Force

bbbabbabbbbbaba

bba			

```
public void brute(int k, int numLetters) {

        // pick random k-character substring as initial seed
        int start = myRandom.nextInt(myString.length() - k + 1);
        String seed = myString.substring(start, start + k);

        // copy first k characters to back to simulate wrap-around
        String wrapAroundString = myString + myString.substring(0,k);

        StringBuilder build = new StringBuilder();
        ArrayList<Character> list = new ArrayList<Character>();

        for (int i = 0; i < numLetters; i++) {
            list.clear();
            int pos = 0;
            while ((pos = wrapAroundString.indexOf(seed, pos)) != -1 &&
pos < myString.length()) {
                char ch = wrapAroundString.charAt(pos + k);
                list.add(ch);
                pos++;
            }
            int pick = myRandom.nextInt(list.size());
            char ch = list.get(pick);
            build.append(ch);
            seed = seed.substring(1) + ch;
        }
    }
```

Look at code on your computer so you can read it!!!!!!

Brute-Force

bbbabbabbbbbaba

bba	b		

```
public void brute(int k, int numLetters) {

        // pick random k-character substring as initial seed
        int start = myRandom.nextInt(myString.length() - k + 1);
        String seed = myString.substring(start, start + k);

        // copy first k characters to back to simulate wrap-around
        String wrapAroundString = myString + myString.substring(0,k);

        StringBuilder build = new StringBuilder();
        ArrayList<Character> list = new ArrayList<Character>();

        for (int i = 0; i < numLetters; i++) {
            list.clear();
            int pos = 0;
            while ((pos = wrapAroundString.indexOf(seed, pos)) != -1 &&
pos < myString.length()) {
                char ch = wrapAroundString.charAt(pos + k);
                list.add(ch);
                pos++;
            }
            int pick = myRandom.nextInt(list.size());
            char ch = list.get(pick);
            build.append(ch);
            seed = seed.substring(1) + ch;
        }
    }
```

Look at code on your computer so you can read it!!!!!!

Brute-Force

bbbabbabbbbaba

bba	b		

```
public void brute(int k, int numLetters) {

        // pick random k-character substring as initial seed
        int start = myRandom.nextInt(myString.length() - k + 1);
        String seed = myString.substring(start, start + k);

        // copy first k characters to back to simulate wrap-around
        String wrapAroundString = myString + myString.substring(0,k);

        StringBuilder build = new StringBuilder();
        ArrayList<Character> list = new ArrayList<Character>();

        for (int i = 0; i < numLetters; i++) {
            list.clear();
            int pos = 0;
            while ((pos = wrapAroundString.indexOf(seed, pos)) != -1 &&
pos < myString.length()) {
                char ch = wrapAroundString.charAt(pos + k);
                list.add(ch);
                pos++;
            }
            int pick = myRandom.nextInt(list.size());
            char ch = list.get(pick);
            build.append(ch);
            seed = seed.substring(1) + ch;
        }
    }
```

Look at code on your computer so you can read it!!!!!!

Brute-Force

bbbabbabbbbaba

bba	b	b	


```
public void brute(int k, int numLetters) {

        // pick random k-character substring as initial seed
        int start = myRandom.nextInt(myString.length() - k + 1);
        String seed = myString.substring(start, start + k);

        // copy first k characters to back to simulate wrap-around
        String wrapAroundString = myString + myString.substring(0,k);

        StringBuilder build = new StringBuilder();
        ArrayList<Character> list = new ArrayList<Character>();

        for (int i = 0; i < numLetters; i++) {
            list.clear();
            int pos = 0;
            while ((pos = wrapAroundString.indexOf(seed, pos)) != -1 &&
pos < myString.length()) {
                char ch = wrapAroundString.charAt(pos + k);
                list.add(ch);
                pos++;
            }
            int pick = myRandom.nextInt(list.size());
            char ch = list.get(pick);
            build.append(ch);
            seed = seed.substring(1) + ch;
        }
    }
```

Look at code on your computer so you can read it!!!!!!

Brute-Force

bbbabbabbbbaba

bba	b	b	

```
public void brute(int k, int numLetters) {

        // pick random k-character substring as initial seed
        int start = myRandom.nextInt(myString.length() - k + 1);
        String seed = myString.substring(start, start + k);

        // copy first k characters to back to simulate wrap-around
        String wrapAroundString = myString + myString.substring(0,k);

        StringBuilder build = new StringBuilder();
        ArrayList<Character> list = new ArrayList<Character>();

        for (int i = 0; i < numLetters; i++) {
            list.clear();
            int pos = 0;
            while ((pos = wrapAroundString.indexOf(seed, pos)) != -1 &&
pos < myString.length()) {
                char ch = wrapAroundString.charAt(pos + k);
                list.add(ch);
                pos++;
            }
            int pick = myRandom.nextInt(list.size());
            char ch = list.get(pick);
            build.append(ch);
            seed = seed.substring(1) + ch;
        }
    }
```

Look at code on your computer so you can read it!!!!!!

Brute-Force

bbbabbabbbbaba

bba	b	b	b


```
public void brute(int k, int numLetters) {

        // pick random k-character substring as initial seed
        int start = myRandom.nextInt(myString.length() - k + 1);
        String seed = myString.substring(start, start + k);

        // copy first k characters to back to simulate wrap-around
        String wrapAroundString = myString + myString.substring(0,k);

        StringBuilder build = new StringBuilder();
        ArrayList<Character> list = new ArrayList<Character>();

        for (int i = 0; i < numLetters; i++) {
            list.clear();
            int pos = 0;
            while ((pos = wrapAroundString.indexOf(seed, pos)) != -1 &&
pos < myString.length()) {
                char ch = wrapAroundString.charAt(pos + k);
                list.add(ch);
                pos++;
            }
            int pick = myRandom.nextInt(list.size());
            char ch = list.get(pick);
            build.append(ch);
            seed = seed.substring(1) + ch;
        }
    }
```

Look at code on your computer so you can read it!!!!!!

Brute-Force

bbbabbabbbbaba

bba	b	b	b

```
public void brute(int k, int numLetters) {

        // pick random k-character substring as initial seed
        int start = myRandom.nextInt(myString.length() - k + 1);
        String seed = myString.substring(start, start + k);

        // copy first k characters to back to simulate wrap-around
        String wrapAroundString = myString + myString.substring(0,k);

        StringBuilder build = new StringBuilder();
        ArrayList<Character> list = new ArrayList<Character>();

        for (int i = 0; i < numLetters; i++) {
            list.clear();
            int pos = 0;
            while ((pos = wrapAroundString.indexOf(seed, pos)) != -1 &&
pos < myString.length()) {
                char ch = wrapAroundString.charAt(pos + k);
                list.add(ch);
                pos++;
            }
            int pick = myRandom.nextInt(list.size());
            char ch = list.get(pick);
            build.append(ch);
            seed = seed.substring(1) + ch;
        }
    }
```

Look at code on your computer so you can read it!!!!!!

Brute-Force

bbbabbabbbbaba

bba	b	b	b
bab			

```
public void brute(int k, int numLetters) {

        // pick random k-character substring as initial seed
        int start = myRandom.nextInt(myString.length() - k + 1);
        String seed = myString.substring(start, start + k);

        // copy first k characters to back to simulate wrap-around
        String wrapAroundString = myString + myString.substring(0,k);

        StringBuilder build = new StringBuilder();
        ArrayList<Character> list = new ArrayList<Character>();

        for (int i = 0; i < numLetters; i++) {
            list.clear();
            int pos = 0;
            while ((pos = wrapAroundString.indexOf(seed, pos)) != -1 &&
pos < myString.length()) {
                char ch = wrapAroundString.charAt(pos + k);
                list.add(ch);
                pos++;
            }
            int pick = myRandom.nextInt(list.size());
            char ch = list.get(pick);
            build.append(ch);
            seed = seed.substring(1) + ch;
        }
    }
```

Look at code on your computer so you can read it!!!!!!

Brute-Force

bbbabbabbbbbaba

bba	b	b	b
bab			

```
public void brute(int k, int numLetters) {

        // pick random k-character substring as initial seed
        int start = myRandom.nextInt(myString.length() - k + 1);
        String seed = myString.substring(start, start + k);

        // copy first k characters to back to simulate wrap-around
        String wrapAroundString = myString + myString.substring(0,k);

        StringBuilder build = new StringBuilder();
        ArrayList<Character> list = new ArrayList<Character>();

        for (int i = 0; i < numLetters; i++) {
            list.clear();
            int pos = 0;
            while ((pos = wrapAroundString.indexOf(seed, pos)) != -1 &&
pos < myString.length()) {
                char ch = wrapAroundString.charAt(pos + k);
                list.add(ch);
                pos++;
            }
            int pick = myRandom.nextInt(list.size());
            char ch = list.get(pick);
            build.append(ch);
            seed = seed.substring(1) + ch;
        }
    }
```

Look at code on your computer so you can read it!!!!!!

Brute-Force

bbbabbabbbbbaba

bba	b	b	b
bab	b		

```
public void brute(int k, int numLetters) {

        // pick random k-character substring as initial seed
        int start = myRandom.nextInt(myString.length() - k + 1);
        String seed = myString.substring(start, start + k);

        // copy first k characters to back to simulate wrap-around
        String wrapAroundString = myString + myString.substring(0,k);

        StringBuilder build = new StringBuilder();
        ArrayList<Character> list = new ArrayList<Character>();

        for (int i = 0; i < numLetters; i++) {
            list.clear();
            int pos = 0;
            while ((pos = wrapAroundString.indexOf(seed, pos)) != -1 &&
pos < myString.length()) {
                char ch = wrapAroundString.charAt(pos + k);
                list.add(ch);
                pos++;
            }
            int pick = myRandom.nextInt(list.size());
            char ch = list.get(pick);
            build.append(ch);
            seed = seed.substring(1) + ch;
        }
    }
```

Look at code on your computer so you can read it!!!!!!

Brute-Force

bbbabbabbbbaba

bba	b	b	b
bab	b		

```
public void brute(int k, int numLetters) {

        // pick random k-character substring as initial seed
        int start = myRandom.nextInt(myString.length() - k + 1);
        String seed = myString.substring(start, start + k);

        // copy first k characters to back to simulate wrap-around
        String wrapAroundString = myString + myString.substring(0,k);

        StringBuilder build = new StringBuilder();
        ArrayList<Character> list = new ArrayList<Character>();

        for (int i = 0; i < numLetters; i++) {
            list.clear();
            int pos = 0;
            while ((pos = wrapAroundString.indexOf(seed, pos)) != -1 &&
pos < myString.length()) {
                char ch = wrapAroundString.charAt(pos + k);
                list.add(ch);
                pos++;
            }
            int pick = myRandom.nextInt(list.size());
            char ch = list.get(pick);
            build.append(ch);
            seed = seed.substring(1) + ch;
        }
    }
```

Look at code on your computer so you can read it!!!!!!

Brute-Force

bbbabbabbbbaba

bba	b	b	b
bab	b	b	

```
public void brute(int k, int numLetters) {

        // pick random k-character substring as initial seed
        int start = myRandom.nextInt(myString.length() - k + 1);
        String seed = myString.substring(start, start + k);

        // copy first k characters to back to simulate wrap-around
        String wrapAroundString = myString + myString.substring(0,k);

        StringBuilder build = new StringBuilder();
        ArrayList<Character> list = new ArrayList<Character>();

        for (int i = 0; i < numLetters; i++) {
            list.clear();
            int pos = 0;
            while ((pos = wrapAroundString.indexOf(seed, pos)) != -1 &&
pos < myString.length()) {
                char ch = wrapAroundString.charAt(pos + k);
                list.add(ch);
                pos++;
            }
            int pick = myRandom.nextInt(list.size());
            char ch = list.get(pick);
            build.append(ch);
            seed = seed.substring(1) + ch;
        }
    }
```

Look at code on your computer so you can read it!!!!!!

Brute-Force

bbbabbabbbbaba

bba	b	b	b
bab	b	b	

```
public void brute(int k, int numLetters) {

        // pick random k-character substring as initial seed
        int start = myRandom.nextInt(myString.length() - k + 1);
        String seed = myString.substring(start, start + k);

        // copy first k characters to back to simulate wrap-around
        String wrapAroundString = myString + myString.substring(0,k);

        StringBuilder build = new StringBuilder();
        ArrayList<Character> list = new ArrayList<Character>();

        for (int i = 0; i < numLetters; i++) {
            list.clear();
            int pos = 0;
            while ((pos = wrapAroundString.indexOf(seed, pos)) != -1 &&
pos < myString.length()) {
                char ch = wrapAroundString.charAt(pos + k);
                list.add(ch);
                pos++;
            }
            int pick = myRandom.nextInt(list.size());
            char ch = list.get(pick);
            build.append(ch);
            seed = seed.substring(1) + ch;
        }
    }
```

Look at code on your computer so you can read it!!!!!!

Brute-Force

bbbabbabbbbaba

bba	b	b	b
bab	b	b	a

```
public void brute(int k, int numLetters) {

        // pick random k-character substring as initial seed
        int start = myRandom.nextInt(myString.length() - k + 1);
        String seed = myString.substring(start, start + k);

        // copy first k characters to back to simulate wrap-around
        String wrapAroundString = myString + myString.substring(0,k);

        StringBuilder build = new StringBuilder();
        ArrayList<Character> list = new ArrayList<Character>();

        for (int i = 0; i < numLetters; i++) {
            list.clear();
            int pos = 0;
            while ((pos = wrapAroundString.indexOf(seed, pos)) != -1 &&
pos < myString.length()) {
                char ch = wrapAroundString.charAt(pos + k);
                list.add(ch);
                pos++;
            }
            int pick = myRandom.nextInt(list.size());
            char ch = list.get(pick);
            build.append(ch);
            seed = seed.substring(1) + ch;
        }
    }
```

Look at code on your computer so you can read it!!!!!!

Questions?

Factorial

```
long fact(long i) {
   if (i == 1) {
      return 1;
   }

   long c = fact(i-1);
      return i * c;
}
```

Factorial

```
long fact(long i) {
  if (i == 1) {
    return 1;
  }

  long c = fact(i-1);
    return i * c;
}
```

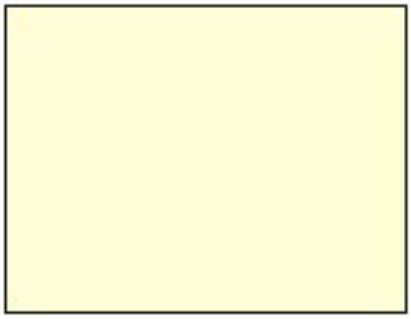
fact(5);

Factorial

```
long fact(long i) {
  if (i == 1) {
    return 1;
  }

  long c = fact(i-1);
    return i * c;
}
```

fact(5);

Call Stack

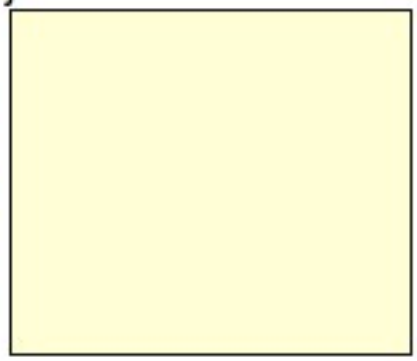


Factorial

```
long fact(long i) {
   if (i == 1) {
      return 1;
   }

   long c = fact(i-1);
      return i * c;
}
```

fact(5);

Call Stack

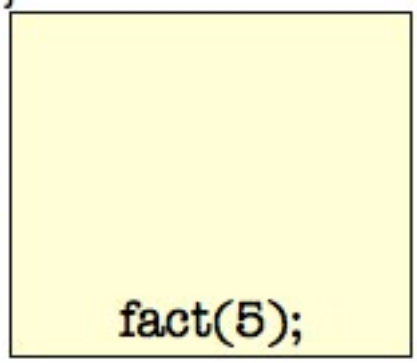


Factorial

```
long fact(long i) {
   if (i == 1) {
      return 1;
   }

   long c = fact(i-1);
      return i * c;
}
```

fact(5);

Call Stack

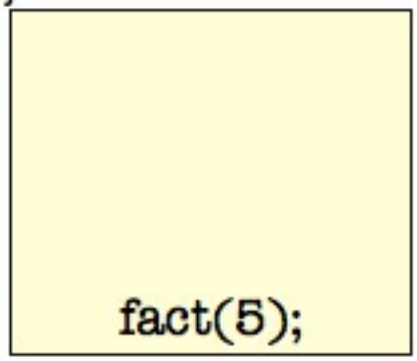


Factorial

```
long fact(long i) {
   if (i == 1) {
      return 1;
   }

   long c = fact(i-1);
      return i * c;
}
```

fact(5);

Call Stack

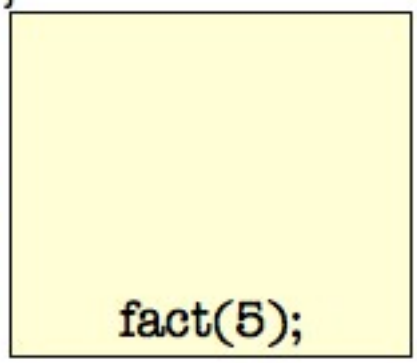


Factorial

```
long fact(long i) {
   if (i == 1) {
      return 1;
   }

   long c = fact(i-1);
      return i * c;
}
```

fact(5);

Call Stack

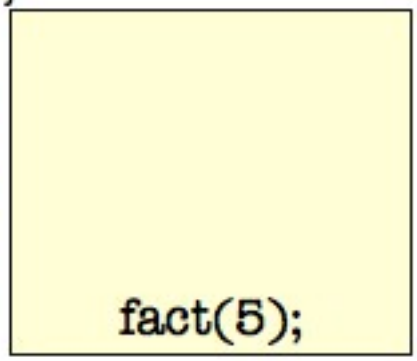


Factorial

```
long fact(long i) {
  if (i == 1) {
    return 1;
  }

  long c = fact(i-1);
    return i * c;
}
```

fact(5);

Call Stack



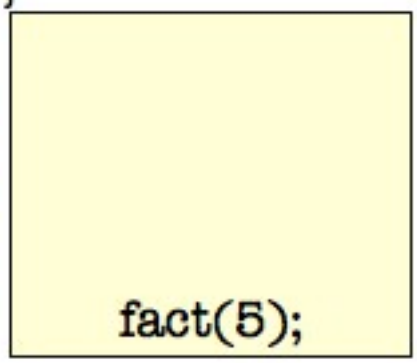
Factorial

```
long fact(long i) {
  if (i == 1) {
    return 1;
  }

  long c = fact(i-1);
    return i * c;
}
```

fact(5);

c = fact(4);

Call Stack

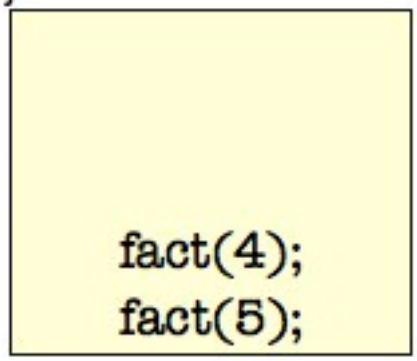


Factorial

```
long fact(long i) {
  if (i == 1) {
    return 1;
  }

  long c = fact(i-1);
    return i * c;
}
```

```
fact(5);
    c = fact(4);
```

Call Stack

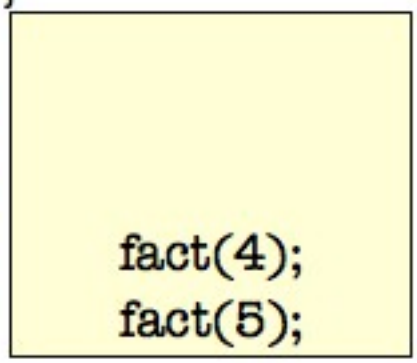


Factorial

```
long fact(long i) {
  if (i == 1) {
    return 1;
  }

  long c = fact(i-1);
    return i * c;
}
```

```
fact(5);
    c = fact(4);
```

Call Stack

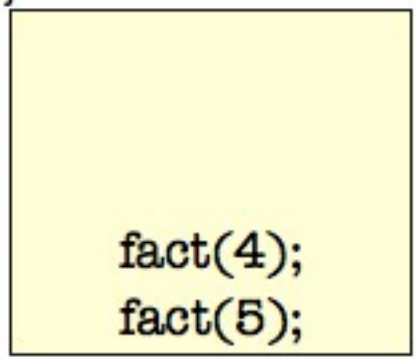


Factorial

```
long fact(long i) {
  if (i == 1) {
    return 1;
  }

  long c = fact(i-1);
    return i * c;
}
```

```
fact(5);
    c = fact(4);
```

Call Stack

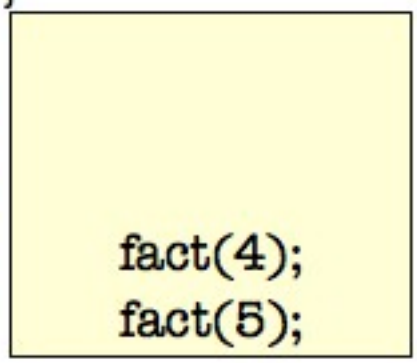


Factorial

```
long fact(long i) {
  if (i == 1) {
    return 1;
  }

  long c = fact(i-1);
    return i * c;
}
```

```
fact(5);
    c = fact(4);
```

Call Stack

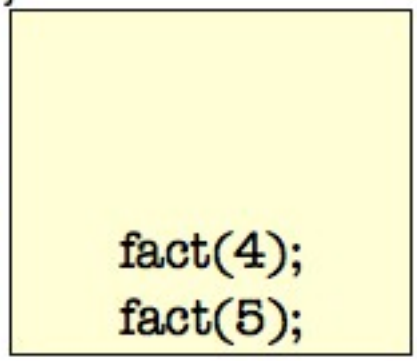


Factorial

```
long fact(long i) {
  if (i == 1) {
    return 1;
  }

  long c = fact(i-1);
    return i * c;
}
```

```
fact(5);
    c = fact(4);
      c = fact(3);
```

Call Stack

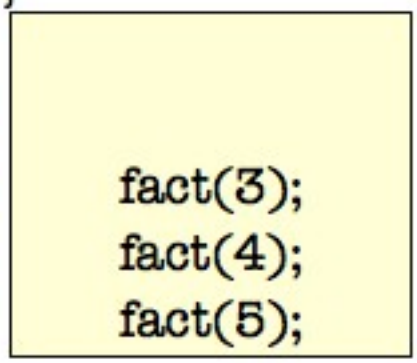


Factorial

```
long fact(long i) {
  if (i == 1) {
    return 1;
  }

  long c = fact(i-1);
    return i * c;
}
```

```
fact(5);
    c = fact(4);
      c = fact(3);
```

Call Stack



Factorial

```
long fact(long i) {
   if (i == 1) {
      return 1;
   }

   long c = fact(i-1);
      return i * c;
}
```

```
fact(5);
    c = fact(4);
       c = fact(3);
```

Call Stack

fact(3);
fact(4);
fact(5);

Factorial

```
long fact(long i) {
   if (i == 1) {
      return 1;
   }

   long c = fact(i-1);
      return i * c;
}
```

```
fact(5);
    c = fact(4);
       c = fact(3);
```

Call Stack

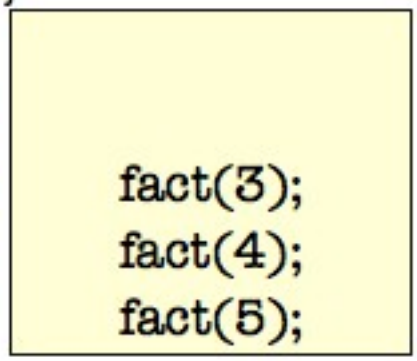
fact(3);
fact(4);
fact(5);

Factorial

```
long fact(long i) {
  if (i == 1) {
    return 1;
  }

  long c = fact(i-1);
    return i * c;
}
```

```
fact(5);
    c = fact(4);
      c = fact(3);
```

Call Stack

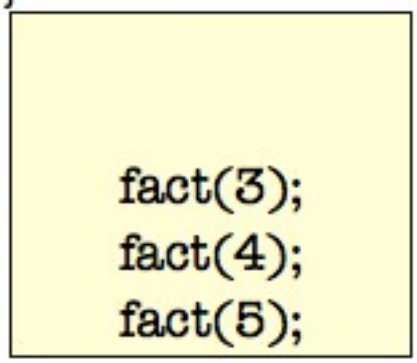


Factorial

```
long fact(long i) {
  if (i == 1) {
    return 1;
  }

  long c = fact(i-1);
    return i * c;
}
```

```
fact(5);
    c = fact(4);
      c = fact(3);
        c = fact(2);
```

Call Stack



Factorial

```
long fact(long i) {
  if (i == 1) {
    return 1;
  }

  long c = fact(i-1);
    return i * c;
}
```

```
fact(5);
    c = fact(4);
      c = fact(3);
        c = fact(2);
```

Call Stack

fact(2);
fact(3);
fact(4);
fact(5);

Factorial

```
long fact(long i) {
  if (i == 1) {
    return 1;
  }

  long c = fact(i-1);
    return i * c;
}
```

```
fact(5);
    c = fact(4);
      c = fact(3);
        c = fact(2);
```

Call Stack

fact(2);
fact(3);
fact(4);
fact(5);

Factorial

```
long fact(long i) {
  if (i == 1) {
    return 1;
  }

  long c = fact(i-1);
    return i * c;
}
```

```
fact(5);
    c = fact(4);
      c = fact(3);
        c = fact(2);
```

Call Stack

```
fact(2);
fact(3);
fact(4);
fact(5);
```


Factorial

```
long fact(long i) {
  if (i == 1) {
    return 1;
  }

  long c = fact(i-1);
    return i * c;
}
```

```
fact(5);
    c = fact(4);
      c = fact(3);
        c = fact(2);
```

Call Stack

```
fact(2);
fact(3);
fact(4);
fact(5);
```

Factorial

```
long fact(long i) {
  if (i == 1) {
    return 1;
  }

  long c = fact(i-1);
    return i * c;
}
```

```
fact(5);
  c = fact(4);
    c = fact(3);
      c = fact(2);
        c = fact(1);
```

Call Stack

```
fact(2);
fact(3);
fact(4);
fact(5);
```


Factorial

```
long fact(long i) {
  if (i == 1) {
    return 1;
  }

  long c = fact(i-1);
    return i * c;
}
```

```
fact(5);
  c = fact(4);
    c = fact(3);
      c = fact(2);
        c = fact(1);
```

Call Stack

```
fact(1);
fact(2);
fact(3);
fact(4);
fact(5);
```

Factorial

```
long fact(long i) {
  if (i == 1) {
    return 1;
  }

  long c = fact(i-1);
    return i * c;
}
```

```
fact(5);
  c = fact(4);
    c = fact(3);
      c = fact(2);
        c = fact(1);
```

Call Stack

```
fact(1);
fact(2);
fact(3);
fact(4);
fact(5);
```


Factorial

```
long fact(long i) {
  if (i == 1) {
    return 1;
  }

  long c = fact(i-1);
    return i * c;
}
```

```
fact(5);
  c = fact(4);
    c = fact(3);
      c = fact(2);
        c = fact(1);
```

Call Stack

```
fact(1);
fact(2);
fact(3);
fact(4);
fact(5);
```

Factorial

```
long fact(long i) {
  if (i == 1) {
    return 1;
  }

  long c = fact(i-1);
    return i * c;
}
```

Call Stack

fact(1);
fact(2);
fact(3);
fact(4);
fact(5);

fact(5);
  c = fact(4);
    c = fact(3);
      c = fact(2);
        c = fact(1);

Factorial

```
long fact(long i) {
  if (i == 1) {
    return 1;
  }

  long c = fact(i-1);
    return i * c;
}
```

Call Stack

fact(2);
fact(3);
fact(4);
fact(5);

fact(5);
  c = fact(4);
    c = fact(3);
      c = fact(2);
        c = fact(1);
          fact(1) = 1

Factorial

```
long fact(long i) {
  if (i == 1) {
    return 1;
  }

  long c = fact(i-1);
    return i * c;
}
```

Call Stack

fact(2);
fact(3);
fact(4);
fact(5);

fact(5);
c = fact(4);
c = fact(3);
c = fact(2);
c = fact(1);
fact(1) = 1

Factorial

```
long fact(long i) {
  if (i == 1) {
    return 1;
  }

  long c = fact(i-1);
    return i * c;
}
```

Call Stack

fact(2);
fact(3);
fact(4);
fact(5);

fact(5);
c = fact(4);
c = fact(3);
c = fact(2);
c = fact(1);
fact(1) = 1

Factorial

```
long fact(long i) {
  if (i == 1) {
    return 1;
  }

  long c = fact(i-1);
    return i * c;
}
```

Call Stack

fact(3);
fact(4);
fact(5);

fact(5);
c = fact(4);
c = fact(3);
c = fact(2);
c = fact(1);
fact(1) = 1
fact(2) = 2*1

Factorial

```
long fact(long i) {
  if (i == 1) {
    return 1;
  }

  long c = fact(i-1);
    return i * c;
}
```

Call Stack

fact(3);
fact(4);
fact(5);

fact(5);
c = fact(4);
c = fact(3);
c = fact(2);
c = fact(1);
fact(1) = 1
fact(2) = 2*1

Factorial

```
long fact(long i) {
  if (i == 1) {
    return 1;
  }

  long c = fact(i-1);
    return i * c;
}
```

Call Stack

fact(3);
fact(4);
fact(5);

fact(5);
c = fact(4);
c = fact(3);
c = fact(2);
c = fact(1);
fact(1) = 1
fact(2) = 2*1

Factorial

```
long fact(long i) {
  if (i == 1) {
    return 1;
  }

  long c = fact(i-1);
    return i * c;
}
```

Call Stack

fact(4);
fact(5);

fact(5);
c = fact(4);
c = fact(3);
c = fact(2);
c = fact(1);
fact(1) = 1
fact(2) = 2*1
fact(3) = 3*2*1

Factorial

```
long fact(long i) {
  if (i == 1) {
    return 1;
  }

  long c = fact(i-1);
    return i * c;
}
```

Call Stack

fact(4);
fact(5);

```
fact(5);
  c = fact(4);
    c = fact(3);
      c = fact(2);
        c = fact(1);
        fact(1) = 1
      fact(2) = 2*1
    fact(3) = 3*2*1
```


Factorial


```
long fact(long i) {
  if (i == 1) {
    return 1;
  }

  long c = fact(i-1);
    return i * c;
}
```

Call Stack

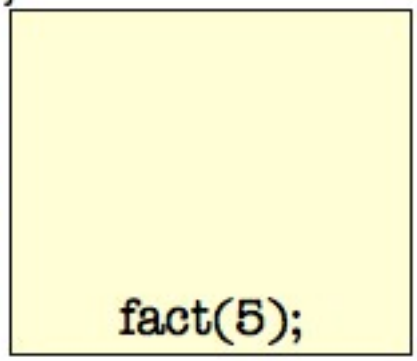
fact(4);
fact(5);

```
fact(5);
  c = fact(4);
    c = fact(3);
      c = fact(2);
        c = fact(1);
        fact(1) = 1
      fact(2) = 2*1
    fact(3) = 3*2*1
```

Factorial

```
long fact(long i) {
  if (i == 1) {
    return 1;
  }

  long c = fact(i-1);
    return i * c;
}
```

Call Stack

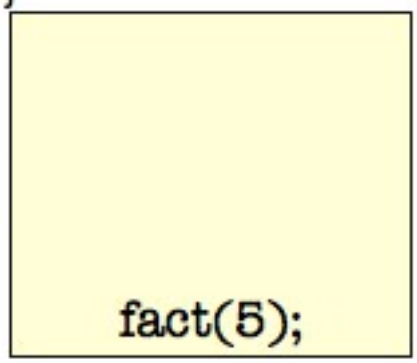



```
fact(5);
  c = fact(4);
    c = fact(3);
      c = fact(2);
        c = fact(1);
        fact(1) = 1
      fact(2) = 2*1
    fact(3) = 3*2*1
```


Factorial

```
long fact(long i) {
  if (i == 1) {
    return 1;
  }

  long c = fact(i-1);
    return i * c;
}
```

Call Stack

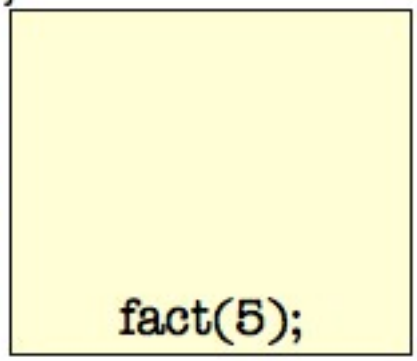



```
fact(5);
  c = fact(4);
    c = fact(3);
      c = fact(2);
        c = fact(1);
        fact(1) = 1
      fact(2) = 2*1
    fact(3) = 3*2*1
  fact(4) = 4*3*2*1
```

Factorial

```
long fact(long i) {
  if (i == 1) {
    return 1;
  }

  long c = fact(i-1);
    return i * c;
}
```

Call Stack

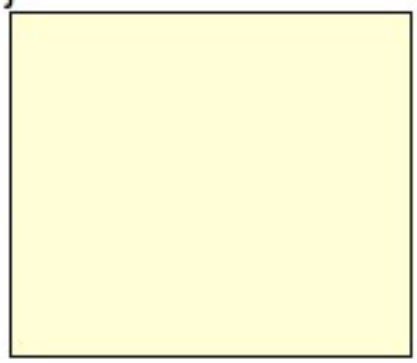



```
fact(5);
  c = fact(4);
    c = fact(3);
      c = fact(2);
        c = fact(1);
        fact(1) = 1
      fact(2) = 2*1
    fact(3) = 3*2*1
  fact(4) = 4*3*2*1
```


Factorial

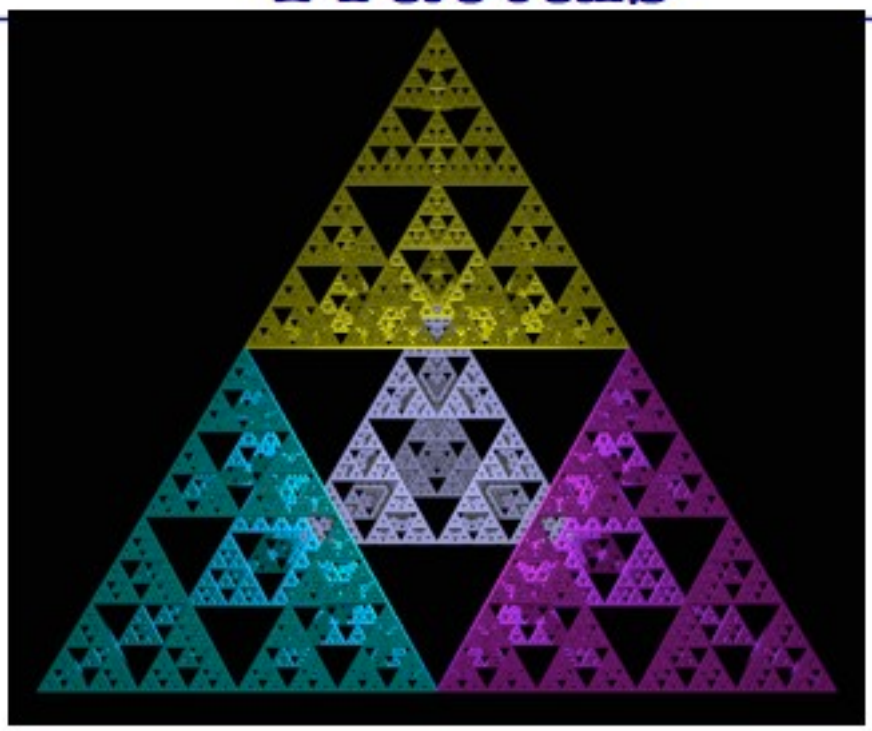
```
long fact(long i) {
  if (i == 1) {
    return 1;
  }

  long c = fact(i-1);
    return i * c;
}
```

Call Stack



```
fact(5);
  c = fact(4);
    c = fact(3);
      c = fact(2);
        c = fact(1);
        fact(1) = 1
      fact(2) = 2*1
    fact(3) = 3*2*1
  fact(4) = 4*3*2*1
fact(5) = 5*4*3*2*1
```

Fractals



fractals.chat.ru

Code time

Practice

- http://codingbat.com/java/Recursion-1
  - triangle
  - sumDigits
  - countHi
    - hint: look at Java String functions

- Put all three methods in ONE java file names Recitation5.java and submit via ambient.