2/6/2006

Recursion

What is recursion? (breaking down a problem into simpler and simpler pieces) Below are some typical recursion examples.

Factorial

```
static int fact(int n) {
      if (n = = 0)
            return 1;
      else
            return n * fact(n-1);
}
```

Fibonacci

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

printList

```
static void printList(Data ptr) {
        if(ptr != null){
                System.out.print(ptr.stuff  +"->");
                printList(ptr.next);
        }
        else
                System.out.println ("");
 }
```

getEvens

```
    public static Data getEvens(Data head) {
                if(head != null){
                        if(head.stuff%2 ==0){
                                Data newHead = new Data();
                                newHead.stuff = head.stuff;
                                newHead.next = getEvens(head.next);
                                return newHead;
                        }
                        else
                                return getEvens(head.next);
                        }

                else
                        return head;
        }
```

fillArea

A non-tail recursive example

```
void fillArea(int x, int y, boolean[][] area){
        if(x < 0 || x >= area.length || y < 0 || y >= area[0].length){
                return; // out of bounds
        }

        if(area[x][y]){ // positions is true
                return;
        }
        area[x][y] = false; //fill the current spot

        fillArea(x+1, y, area); //go up
        fillArea(x-1, y, area); //go down
        fillArea(x, y+1, area); // go right
        fillArea(x, y-1, area); // go left

}
```

Context-Free Grammars

- Def: context free
  - doesn’t matter on context
  - superbowl commercial “well that killed him”
- E->e (E is non-terminal, e is a string of terminals and nonterminals)
- Super simple calculator
  - E → n
  - E → (E + E)

- (1 + (2 + 3))
    - E - > (E + E)
    - E -> (n + E)
    - E -> (1 + E)
    - E -> (1 + (E + E))
    - E -> (1 + (n + E))
    - E -> (1 + (2 + E))
    - E -> (1 + (2 + n))
    - E -> (1 + (2 + 3))
    - E -> (1 + (5))
    - E -> 6

Compute the result of the Expression

```
public int parseE (Scanner scanner) {
      if (scanner.hasNextInt()) {
            int a = scanner.nextInt();
            return a;
      }
      check(scanner, "(");
      int a = parseE(scanner);
      check(scanner, "+");
      int b = parseE(scanner)
      check(scanner, ")");
      return a + b;
}
```

Exceptions

Try-Catch block

```
try {
  parseE();
} catch (Exception exc) {
  System.out.println("Syntax error: " + exc.getMessage());
}
```

Use it for CheckToken

```
void checkToken(String expected) throws Exception {
   String s = nextToken();
   if (s.equals(expected)) return;
   throw new Exception("Got " + s + ", expected " + expected);
}
```

```
public int parseE (Scanner scanner) {
      if (scanner.hasNextInt()) {
            int a = scanner.nextInt();
            return a;
      }
      checkToken(scanner, "(");
      int a = parseE(scanner);
      checkToken(scanner, "+");
      int b parseE(scanner)
      checkToken(scanner, ")");
      return a + b;
}
```