CS 365 — Programming Languages

Exceptions

Mar 2, 2012

```c
#include<stdio.h>
#include<math.h>


void sqrtMean(int array[], int length);

int main()
{

  int array[10] = { 5, 12, 13, 3, 4, 5, 21, 42, 63, 7};

  sqrtMean(array, 10);

  array[3] = -47;

  sqrtMean(array, 10);

  return 0;
}

void sqrtMean(int array[], int length)
{
  float sqrtSum = 0;
  int i;
  for (i = 0; i < length; i++)
    if (array[i] < 0) goto error;
    else sqrtSum += sqrt(array[i]);

  printf("%f\n", sqrtSum / length);
  return;

 error:
  printf("found negative in array.\n");
}
```

```
public class Continue {
    public static void main(String[] args) {

        int[][] array = { {45,76,34,56,879,24,46,57,789,234},
                    { 26,7,58,52,-236,84,48,14,25,95},
                    { 734, 28,  75, 34, 88, 53} };

        outer:
        for (int i = 0; i < array.length; i++) {
            double sqrtSum = 0;
            for (int j = 0; j < array[i].length; j++)
                if (array[i][j] < 0) {
                    System.out.println("There's a negative in this row, yuck.")
                    continue outer;
                }
                else sqrtSum += Math.sqrt(array[i][j]);
            System.out.println("" + (sqrtSum / array[i].length));
        }
    }
}
```

```
public class Goto {
    public static void main(String[] args) {
        f(25);
    }
    static int f(int x) {
        int a;
        int i;
        @ static int g(int y) {
            int j;
            @ static int h(int z) {
                 @ static int m(int k) {
                       if (k == 12) goto exit;
                       else if (k >= x) return;
                       else {
                           a = a + z;
                           m(k + 3);
                       }
                 }

                 m(z);
             }
            for (j = y; j > 0; j = j - 7)
                h(j);
        }
        for (i = 0; i < x; i = i + 1)
            g(i);
   exit:
        return a;
  }
}
```

```
public class A { }

...

public static { int, String, A} m(int x) {
     if (x < 0)
           return 5;
     else if (x = 0)
           return "hello";
      else
           return new A();
}
```

```
typecase(m(z)) {
case int {int i} : ...

case String {String s} : ...

case A {A a} : ...
}
```

```
datatype a = ...;

datatype multi = Int of int | String of string | A of a;

case m(z) of
  Int(i) => ...
| String(s) => ...
| A(aa) => ...
```

```
public class E1 extends Exception { public int i; }
public class E2 extends Exception { public String s;}
public class E3 extends Exception { public A a; }

public static void m(int x) throws E1, E2, E3{
     if (x < 0)
           throw new E1(5);
     else if (x = 0)
            throw new E2("hello");
      else
           return new E3(new A());
}


try { m(z)
} catch (E1 e) {
    int i = e.i;
    ...
} catch (E2 e) {
    String s = e.s;
    ...
} catch (E3 e) {
    A s = e.a;
   ...
}
```

```
try {
    ...  // A

    x = y / z;

    ...   // B
} catch(ArithmeticException ae) {
   ... // C
}
```

```
try {
    ...  // A

    x = y / z;

    ...   // B
} catch(ArithmeticException ae) {
   ... // C
}
```

```
boolean exception = false;

... // A

if (z == 0) {
    exception = true;
    goto CATCH;
}

x = y / z;

... // B

CATCH:
if (exception) {
   ... // C
}
```

Statement	$\rightarrow$	; | *Block* | *Assignment* | *IfStatement* | *WhileStatement* | *PrintStatement* | *CallStatement* | *ReturnStatement* | *TryCatch*
TryCatch	$\rightarrow$	`try` *Block* *Catch*+
Catch	$\rightarrow$	`catch (` *ExceptionType* `ID )` *Block*
ExceptionType	$\rightarrow$	`ArithmeticException` | `ArrayIndexOutOfBoundsException` | `NullPointerException`

```
  ExceptionType:!(AE "ArithmeticException",
                  AIOOBE "ArrayIndexOutOfBoundsException",
                  NPE "NullPointerException")
Statement => Skip | Block | Assignment | Conditional | Loop | Print |
                CallStmt | ReturnStmt | TryCatch ;

TryCatch -> Block Catch*;

Catch -> ExceptionType Block;
```