Recitation 3

2D Arrays, Exceptions

2D arrays

Many applications have multidimensional structures:

- Matrix operations
- Collection of lists
- Board games (Chess, Checkers)
- Images (rows and columns of pixels)
- ...

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

1D Array Review

```
Animal[] pets= new Animal[3];
```

pets → ~~null~~ Array@0x10

```
pets.length  is 3
pets[0]=  new Animal();
pets[0].walk();
```

Array@0x10

0	null
1	null
2	null

Why is the following illegal?

```
pets[1]  = new Object();
```


Java arrays vs Python lists

Java arrays do not change size!

b → ~~A@0xab~~ A@0x12

bBig → A@0x12

A@0xab

0	"Cornell"
1	"Ithaca"

A@0x12

0	"Cornell"
1	"Ithaca"
2	
3	

```
String[] b= {"Cornell", "Ithaca"};
String[] bBig=  Arrays.copyOf(b, 4);
b= bBig;
```


Java array initialization

Instead of

```
int[] c= new int[5];
c[0]= 5; c[1]= 4; c[2]= 7; c[3]= 6; c[4]= 5;
```

Use an array initializer:

```
int[] c= new int[] {5, 4, 7, 6, 5};
```

Note: The length of c is the number of values in the list.

Exercise 1: Looping over an array

```
/** Return  index  of  occurrence  number  n  of
 *  t in b.
 *  Precondition:  n >= 1.
 *  Return  -1  if  not  found.  */
public  static  int  get(int[]  b,  int  n,  int  t)  {
   ...
}

get(new  int[]{2110,  0,  1,  2110,  2110},  2,  2110);
```

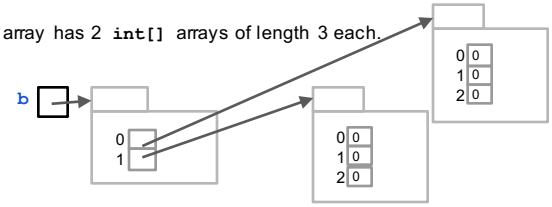
would return 3

2D arrays: An array of 1D arrays.

Java only has 1D arrays, whose elements can also be arrays.

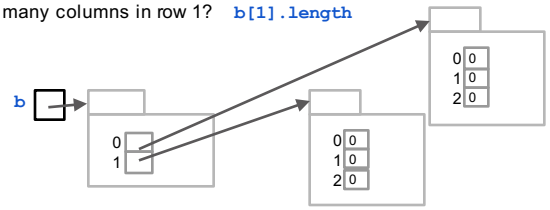
```
int[][]  b= new int[2][3];
```

This array has 2 `int[]` arrays of length 3 each.



2D arrays: An array of 1D arrays.

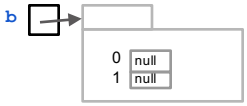
How many rows in `b`? `b.length`
How many columns in row 0? `b[0].length`
How many columns in row 1? `b[1].length`



2D arrays: An array of 1D arrays.

```
int[][]  b= new int[2][];
```

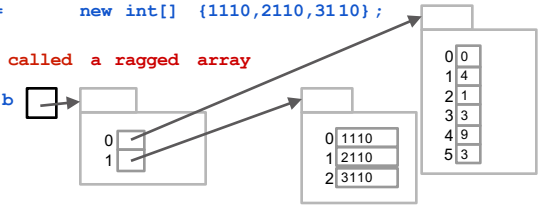
The elements of b are of type `int[]`.



2D arrays: An array of 1D arrays.

```
int[][]  b= new int[2][];
b[0]=       new int[]  {0,4,1,3,9,3};
b[1]=       new int[]  {1110,2110,3110};
```

`b is called a ragged array`



Exercise 2: Transpose Matrix

$A^T[i][j]$ is $A[j][i]$

Exceptions

Exceptions make your code crash

```
public static void main(String[] args) {
      System.out.println(args[0]);
}

public static void main(String[] args) {
      System.out.println(8 / 0);
}

public static void main(String[] args) {
      System.out.println(null.toString());
}
```

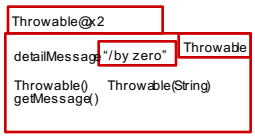

What could happen without exceptions?

```
public static double getAverage(double[] b) {
      double sum = 0;
      for (int i = 0; i < b.length; i++) {
            sum += b[i];
      }
      return sum / b.length;
}
```

If b.length is 0, what should be returned?
- Infinity
- "special" int - Integer.MAX_VALUE? 2110? 0?

Superclass of exceptions: Throwable

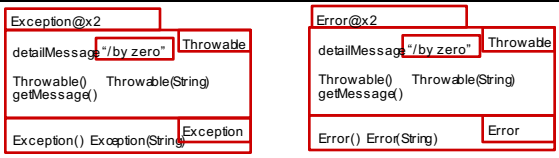
When some sort of exception occurs, an object of class `java.lang.Throwable` (or one of its subclasses) is created and "thrown" –we explain later what "throw" means.



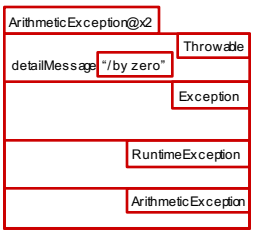
The object has
1. Field to contain an error message
2. Two constructors
3. Function to get the message in the field

Superclass of exceptions: Throwable

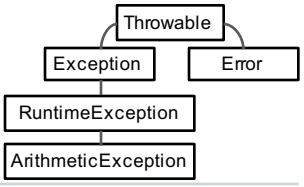
Two subclasses of Throwable exist:
Error: For errors from which one can't recover –don't "catch" them
Exception: For errors from which a program could potentially recover –it's ok to "catch" them



A Throwable instance: ArithmeticException



There are so many different kinds of exceptions we need to **organize** them.



Throwing an exception

When an exception is thrown, it is thrown to the place of call, which throws it out further to where that method was called. The code that called main will "catch" the exception and print the error message

Method call: `main(new String[] {});`

Console:

```
java.lang.AE:  / by zero
        at Ex.third(Ex.java:11)
        at Ex.second(Ex.java:7)
        at Ex.main(Ex.java:3)
```

```
1  class Ex {
2      static void main(…) {
3          second();
4      }
5
6      static void second() {
7          third();
8      }
9
10     Static void third() {
11         int c= 5/0;
12     }
13 }
```

AE

AE

AE

AE = ArithmeticException

Decoding the output from an exception

```
1  public static void main(String[] args) {
2        int div= 5/0;
3  }
```

Exception that is thrown

message

```
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at Animal.main(Animal.java:2)
```

called method

line number

Try statement: catching a thrown exception

```
try {
     code (this is the try-block)
}
catch (MyException ae) {
     code (this is the catch-block)
}

S; (code following the try statement)
```

`ae` is like a parameter. When the catch-block catches a thrown object, `ae` contains the object

To execute the try statement:

Execute the try-block. If it finishes without throwing an exception, fine.

If the try-block throws a `MyException` object, catch it (execute the catch block); else throw it out further.

If the exception was caught, execution proceeds to the code S following the try-statement.

`throw` keyword: Forcing a crash

Why might I want to crash the application?

```
parseInt("42")  -> 42
parseInt("Sid")  -> ???
```

```
class Integer {
  /** Parse s as a signed decimal integer.
    * Throw a
    * NumberFormatException if not possible
    */
    public static int parseInt(String s){

        if (can't convert to int){
            throw new NumberFormatException();
        }
        ...
    }
}
```

Demo 1: Read an Integer

- Ask the user to input an `int`
- Try to convert user input to an `int`
- If an exception is thrown, catch it and ask for more input

Exercise 3: Illegal Arguments

Create `class Person` with two fields, `name` and `age`. Throw an `IllegalArgumentException` instead of having preconditions when given a `null` name or a non-positive age.

How to write an exception class

```
/** An instance is an exception */
public class OurException extends Exception {

        /** Constructor: an instance with message m*/
        public OurException(String m) {
            super(m);
        }

        /** Constructor: an instance with no message */
        public OurException() {
            super();
        }
}
```

`throws` clause

```
public static void second() {
    …
    String line= kyboard.readLine();
    …
}
```

Unhandled exception type **`IOException`**

You may get an error message like the yellow one above. In that case, insert a throws clause as shown below.

```
public static void second() throws IOException {
    …
    String line= kyboard.readLine();
}
```


`throws` clause for checked exceptions

```
/** Class to illustrate exception handling */
public class Ex {
        public static void main() {
        try { second(); } catch (OurException e) {}
        }
        public static void second() throws OurException {
        third();
        }
        public static void third() throws OurException {
        throw new OurException("mine");
    }
}
```

If you're interested in the "controversy",
http://docs.oracle.com/javase/tutorial/essential/exceptions/runtime.html

Demo 2: Pythagorean Solver

- Given *a* and *b*: solve for *c* in $a^2 + b^2 = c^2$
- Reads in input from keyboard
- Handles any exceptions

Key takeaways

1. Java arrays do not extend!
2. A 2D array is just a 1D array of 1D arrays.
3. Thrown exceptions bubble up the call stack until they are handled by a try-catch block. In the system, the call of method main **is** in a try-catch statement, and its catch block prints out information about the thrown exception.

Alt-Text: I'm trying to build character but Eclipse is really confusing.