Review session: Exceptions, GUI's.

→ ~~about~~ you have in your "About the Final" handout what you need to know.

Exceptions: "~~About the Final~~ ~~says~~: ~~be able~~ ...

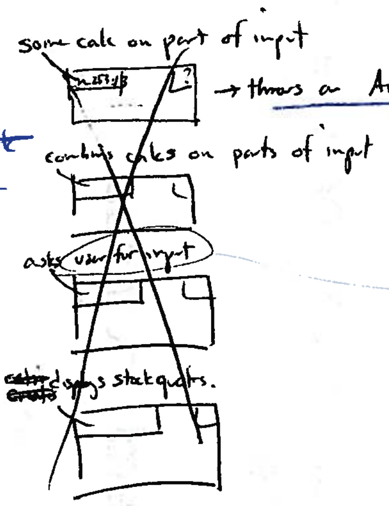
~~For signalling~~

- what are they? objects ~~the~~ serving as signals that something (unusual) has gone wrong.

~~[illegible]~~
Array/String IndexOutOfBounds - trying to access an index that doesn't exist
ArithmeticExceptn - trying to divide by zero
NullPtrExceptn - trying to access fields or methods of a null object.
numberFormatExceptn - trying to treat a String that doesn't represent it as if it did.

Exceptns can/should be caught by method that knows how to deal w/ the problem.

Example: ~~call stack~~ (see handout (0))



some calc on part of input
n=25?/3
?
→ throws an ArithMetic Exception
combines calcs on parts of input
asks user for input
~~[illegible]~~ displays stock quotes.

who should catch this?

this method, b/c it "knows" the input was bad; can ask user for new input.

(processInput).

- how can processFirstBit throw an Exceptn?
  - ~~by trying to divide by zero, access null field~~
  - system creates one (e.g., b/c of a divide by zero)
  - throw new ~~[illegible]~~ Exception(); (some pre-existing or a new Exceptn class).
    ~~Exc~~ & ↑ w/ no detail msg.
  - ~~throw new~~
  - throw new ~~Exc~~ Exceptn("processFirstBit: problem ... ") — w/ a detail msg

- how can processInput catch an Exceptn?
  inside:

```
try {
    processAllInfo();
}
catch (~~Exceptn~~ Exceptn e) {
```

what if no catch?
what if diff. exceptn?
how do we write class ExcptnExcptn?

(0) Consider the following methods:
runEverything(...): creates a GUI that responds to user requests. Calls processInput.
processInput(...): gets and processes entire input from the user. Calls calcAllInfo
preprocessAllInfo(...): collates info about result of preprocessing each piece of user input. Calls processFirstBit(...).
processFirstBit(...): preprocesses the first bit of user input.

We consider what (should) happen if processFirstBit throws an Exception.

(1) (From Spring 2010's final) We have forgotten how to find the length of a string `s`, and we are in a hurry. We do remember that `s.charAt(k)` throws a `StringIndexOutOfBoundsException` if `k` is not the index of a character `s`. So we (meaning you) write the function below, using a loop (with initialization) that successively evaluates `s.charAt(0)`, `s.charAt(1)`, `s.charAt(2)`, ... until the exception is thrown, at which time `k` will be the length! Write the body of the function. You will need a try-statement.

4 loopy q's.

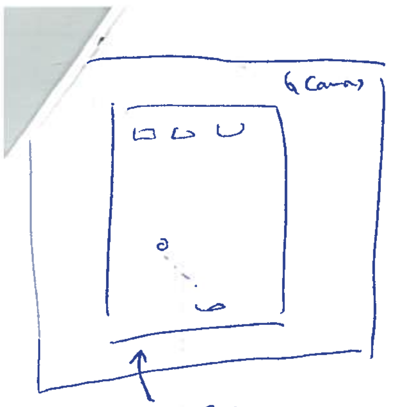
```
/** = length of string s */
public static int length(String s) {
        k= 0;
        //inv: s[0..k-1] exists.
        while (true) {
            try {
                ~~char c =~~ s.charAt(k);   // totally useless, just check if access allowed
                        ↖ just making a statement.
            } catch (StringIndexOutOfBoundsException e) {
                return k;                          // how does it terminate.
            }
          k=k+1;
        }

}
```

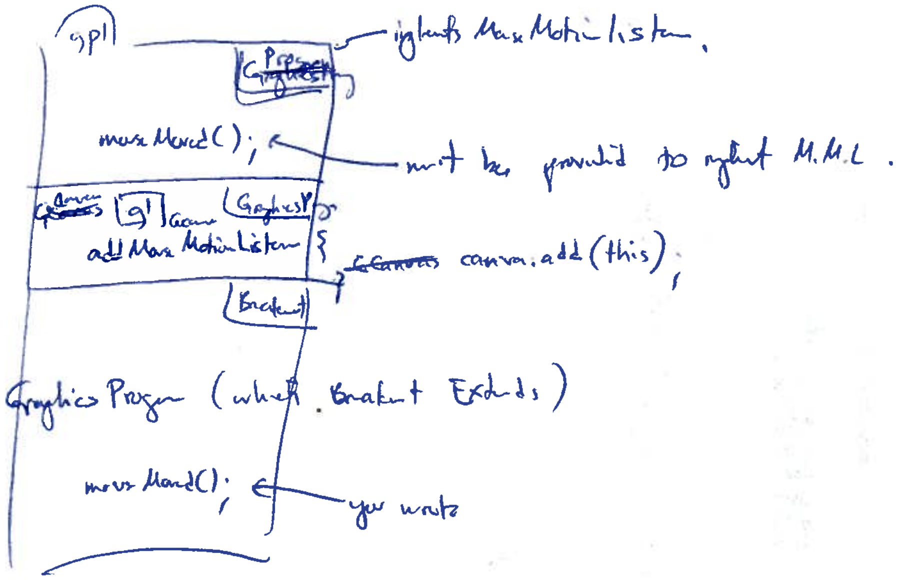
(2) How were the 3 steps for getting something to listen to an event actually implemented in the acm package we used for A7? (Remember we said that these things were done "under the hood".)

1st: recall 3 steps:
→ explain what should happen if action occurs
- some class "doer" has to contain a method ~~the~~ a~~ctionPerformed~~ mouseMoved(.) (or whatever).

→ tell Java ~~that~~ objects of this class specify what should happen
add "implements MouseMotionListener" to class header.

→ for a component when ~~an~~ the action could occur,
register a doer object (say d) as a listener.
component.addMouseMotionListener(d);





a GCanvas, part of a GraphicsProgram (which Breakout Extends)

JFrame: BorderLayout
(cp.add(button, BorderLayout.EAST);

JPanel: FlowLayout.
p.add(button);

Box: BoxLayout.
constructor: new Box(BoxLayout.X_AXIS);

- JButton
- JLabel
- JTextField
- JTextArea.