CS 2110: Exceptions + Prelim Review

Not all statement or expressions execute "normally."

Ex/ `String s = null; s.length();` // NullPointerException
`2/0` // ArithmeticException.
`int[] a = {2,3}; a[2]` // ArrayIndexOutOfBoundsException.

runtime ↑

Code can generate exceptions using **throw**

```
throw new NullPointerException();
throw new MyException();
```

Can catch exceptions with try-catch stmt.

```
try {
    ⋮
    f()  ← generates AIOOBE
    ⋮
    a[i]
    ⋮
} catch (ArrayIndexOutOfBoundsException exc) {
    ⋮
    // handle exception
    ⋮
}
```

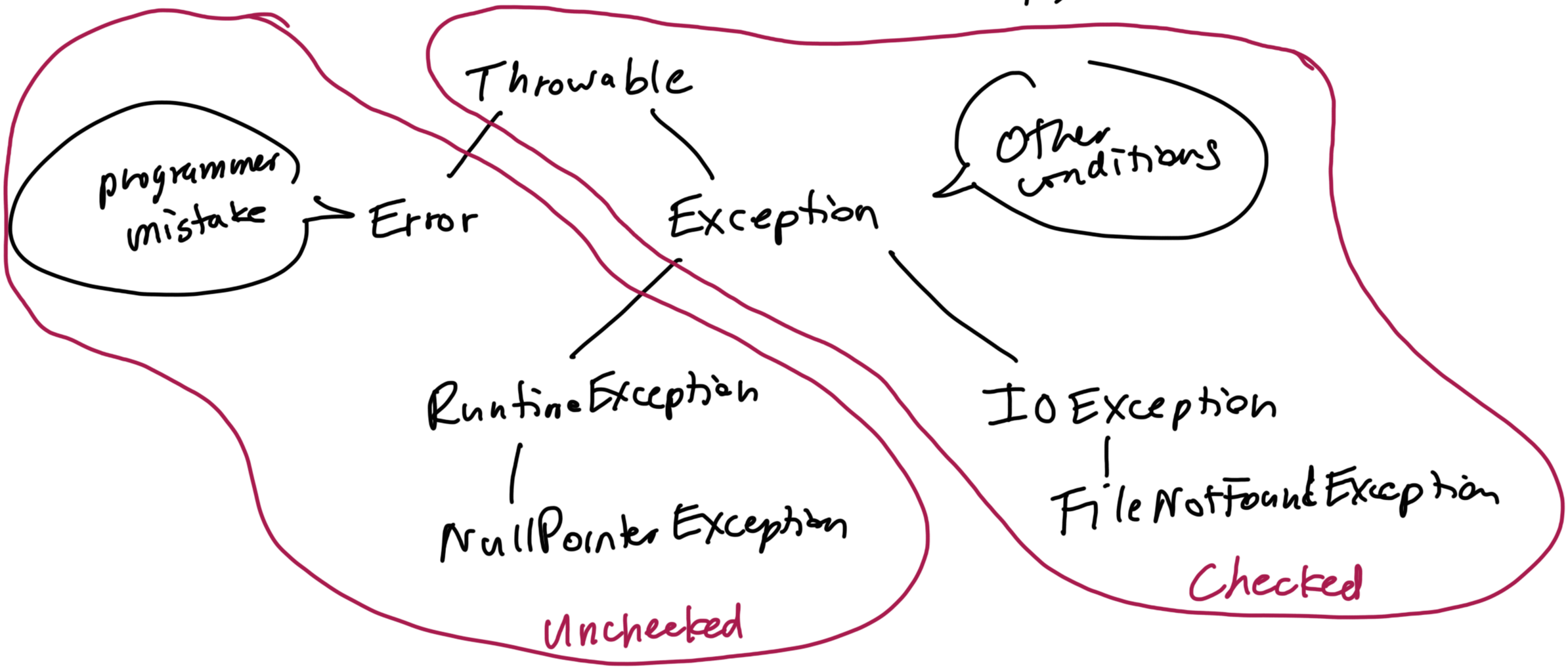
skipped

exception

Bound to exception object

Exception terminates current execution of expression, statement, propagates up call stack until matching ^catch clause found.

- Exceptions named by classes
  - Catch clause catches any subclass of named class (like instanceof)



Checked exceptions : must be handled
or declared to be thrown.

Ex/ class Negative extends Exception {} ← Checked

```
double sqrt(double x) throws Negative, ...
  {
    :
  }

double f(...) throws Negative
  try {
    sqrt(x) + sqrt(y)
  } catch (Negative exc) { err.println("no");}
}
```

Checked exceptions ⇒ Compiler helps ensure exceptions are handled ⇒ more reliable code.

Unchecked exceptions ⇒ no compile-time checking
⇒ usu. for programmer mistakes.
⇒ terminates program.

Specifying

- Legal behavior (poss. unusual)?
  in postcondition ⇒ Returns clause.
- Only result of programmer error?
  - Requires clause / Checks clause

Null

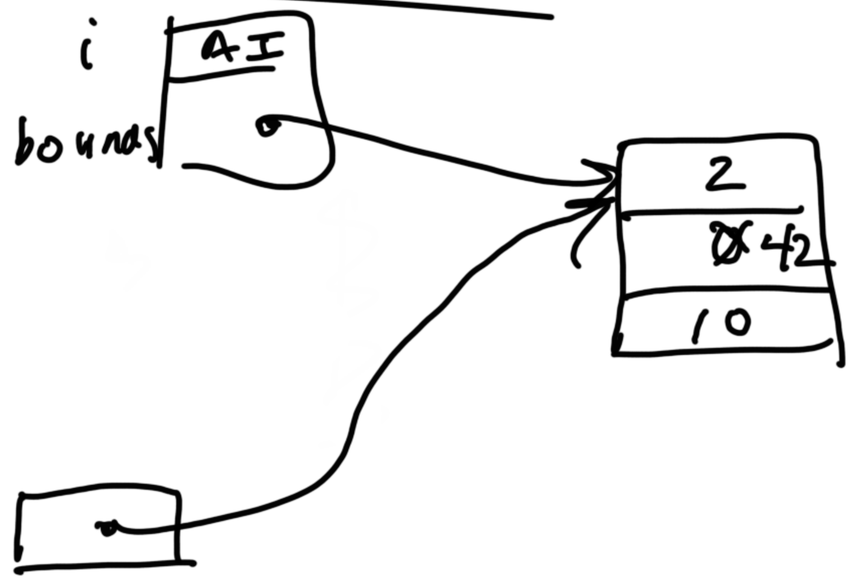
NPEs are big source of bugs.
unchecked ⇒ compiler doesn't warn
⇒ use null only when necessary
– Design patterns (e.g. Optional)

```
class ArrayInterval {
    int[] bounds;

    int[] bounds() { return bounds; }
}
```

```
ArrayInterval i;
int[] a = i.bounds();
a[0] = 42;
```

i.bounds and a are aliases



Classes

Interfaces

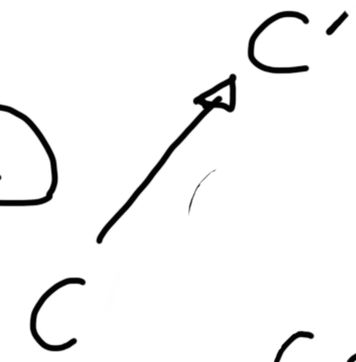
class C implements I { ... }

C <: I

no inheritance

(just subtyping)

class C extends C' { ... }



C <: C'

inheritance

Rep exposure? return type is...

A: only arrays
B: other types too
C: neither

~~int[]~~

```
Object o = new int[]{1,2,3};
String s = (String) o;   // compile-time: ok.
                         // run time:
                         //   ClassCastException
```

```
if (o instanceof String s){
    :
    //use s
} else {

}
```

o.getClass()
↳ class object

false && ——
↑
not evaluated.