Recreation

lds a JUnit test:

```
oid mogrifyTest() {
tEquals("mogrify fails", new int[] { 2, 4, 8, 12
         MyClass.mogrify(new int[] { 1, 2, 4, 6 }));
```

ays seems to fail, no matter what `mogrify` does. Why?

es this in an autograder log:

```
roj0/game2048 directory.
```

y to be the problem?

oes not see his proj0 submission under the Scores tab.
the problem?

CS61B Lecture #12: Exceptions

What to do About Errors?

t of any production program devoted to detecting and
o errors.

are external (bad input, network failures); others are
rs in programs.

d has stated precondition, it's the client's job to comply.

to detect and report client's errors.

throw exception objects, typically:

SomeException (*optional description*);

re objects. By convention, they are given two construc-
h no arguments, and one with a descriptive string argu-
the exception stores).

throws some exceptions implicitly, as when you deref-
pointer, or exceed an array bound.

Catching Exceptions

es each active method call to *terminate abruptly,* until
we come to a **try** block.

tions and do something corrective with **try**:

that might throw exception;
SomeException e) {
ething reasonable;
SomeOtherException e) {
ething else reasonable;

life;

Exception exception occurs in "Stuff...," we immedi-
ething reasonable" and then "go on with life."

string (if any) available as `e.getMessage()` for error
d the like.

ceptions: Checked vs. Unchecked

hrown by **throw** command must be a subtype of `Throwable`
g).

lares several such subtypes, among them

ed for serious, unrecoverable errors;

, intended for all other exceptions;

xception, a subtype of `Exception` intended mostly for
ing errors too common to be worth declaring.

exceptions are all subtypes of one of these.

of `Error` or `RuntimeException` is said to be *unchecked.*

eption types are *checked.*

Unchecked Exceptions

r

er errors: many library functions throw
rgumentException when one fails to meet a precondi-

tected by the basic Java system: e.g.,

ng `x.y` when `x` is null,

ng `A[i]` when `i` is out of bounds,

ng `(String) x` when `x` turns out not to point to a `String`.

tastrophic failures, such as running out of memory.

wn anywhere at any time with no special preparation.

Good Practice

tions rather than using print statements and System.exit

esponse to a problem may depend on the *caller*, not just
e problem arises.

w an exception when programmer violates preconditions.

good idea to throw an exception rather than let bad
t a data structure.

document when methods throw exceptions.

formation about the cause of exceptional condition, put
xception rather than into some global variable:

```
d extends Exception {          try {...
IntList errs;                  } catch (MyBad e) {
ntList nums) { errs=nums; }     ... e.errs ...
                               }
```

Checked Exceptions

indicate exceptional circumstances that are not neces-
mmer errors. Examples:

g to open a file that does not exist.

utput errors on a file.

an interrupt.

ed exception that can occur inside a method must ei-
dled by a `try` statement, or reported in the method's

,

```
d() throws IOException, InterruptedException { ... }
```

yRead (or something it calls) *might* throw `IOException`
tedException.

sign: Why did Java make the following illegal?

```
nt {          class Child extends Parent {
) { ... }         void f () throws IOException { ... }
              }
```