Java Rant #1

(A Paucity of Types)

Definition and Use of Java Pairs

```java
public class Pair {

  public int x;
  public int y;

  public Pair (int a, int b) {
    x = a;
    y = b;
  }
}
```

```java
public class User {

  public Pair swap (Pair p1) {
    Pair p2 =
      new Pair(p1.y, p1.x);

    return p2;
  }
}
```

What could go wrong?

A Paucity of Types

```
public class Pair {

  public int x;
  public int y;

  public Pair (int a, int b) {
    x = a;
    y = b;
  }
}
```

```
public class User {

  public Pair swap (Pair p1) {
    Pair p2 =
      new Pair(p1.y, p1.x);

    return p2;
  }
}
```

The input p1 to swap may be null and we forgot to check.

Java has no way to define a pair data structure that is *just a pair*.

How many students in the class have seen an accidental null pointer exception thrown in their Java code?

From Java Pairs to O'Caml Pairs

In O'Caml, if a pair may be null it is a pair option:

```
type java_pair = (int * int) option
```

From Java Pairs to O'Caml Pairs

In O'Caml, if a pair may be null it is a pair option:

```
type java_pair = (int * int) option
```

And if you write code like this:

```
let swap_java_pair (p:java_pair) : java_pair =
  let (x,y) = p in
  (y,x)
```

From Java Pairs to O'Caml Pairs

In O'Caml, if a pair may be null it is a pair option:

```
type java_pair = (int * int) option
```

And if you write code like this:

```
let swap_java_pair (p:java_pair) : java_pair =
  let (x,y) = p in
  (y,x)
```

You get a *helpful* error message like this:

```
# … Characters 91-92:
    let (x,y) = p in (y,x);;
                ^
Error: This expression has type java_pair = (int * int) option
       but an expression was expected of type 'a * 'b
```

From Java Pairs to O'Caml Pairs

```
type java_pair = (int * int) option
```

And what if you were up at 3am trying to finish your COS 326 assignment and you accidentally wrote the following sleep-deprived, brain-dead statement?

```
let swap_java_pair (p:java_pair) : java_pair =
  match p with
   | Some (x,y) -> Some (y,x)
```

From Java Pairs to O'Caml Pairs

```
type java_pair = (int * int) option
```

And what if you were up at 3am trying to finish your COS 326 assignment and you accidentally wrote the following sleep-deprived, brain-dead statement?

```
let swap_java_pair (p:java_pair) : java_pair =
  match p with
   | Some (x,y) -> Some (y,x)
```

OCaml to the rescue!

```
 ..match p with
      | Some (x,y) -> Some (y,x)
Warning 8: this pattern-matching is not exhaustive.
Here is an example of a value that is not matched:
None
```

From Java Pairs to O'Caml Pairs

```
type java_pair = (int * int) option
```

And what if you were up at 3am trying to finish your COS 326 assignment and you accidentally wrote the following sleep-deprived, brain-dead statement?

```
let swap_java_pair (p:java_pair) : java_pair =
  match p with
   | Some (x,y) -> Some (y,x)
```

An easy fix!

```
let swap_java_pair (p:java_pair) : java_pair =
   match p with
    | None -> None
    | Some (x,y) -> Some (y,x)
```

From Java Pairs to O'Caml Pairs

Moreover, your pairs are probably almost never null

Defensive programming & always checking for null is annoying

Worst of all, there just isn't always some "good thing" for a function to do when it receives a bad input, like a null pointer

In O'Caml, all these issues disappear when you use the proper type for a pair and that type contains no "extra junk”

```
type pair = int * int
```

Once you know O'Caml, it is *hard* to write swap incorrectly

```
let swap (p:pair) : pair =
   let (x,y) = p in (y,x)
```

Summary of Java Pair Rant

Java has a paucity of types
- There is no type to describe just the pairs
- There is no type to describe just the triples
- There is no type to describe the pairs of pairs
- There is no type …

OCaml has many more types
- use option when things may be null
- do not use option when things are not null
- ocaml types describe data structures more precisely
  - programmers have fewer cases to worry about
  - entire classes of errors just go away
  - type checking and pattern analysis help prevent programmers from ever forgetting about a case

Summary of Java Pair Rant

Java has a paucity of types

- There is no type to desc e just th s
- There is type to des es
- There is n d
- There is n

OCaml

- use o
- oca e precisely
  - worry t
  - entire st go ay
  - type ch g and pa ern analysis help prevent programmers from ev forgetting about a case

SCORE: OCAML 1, JAVA 0