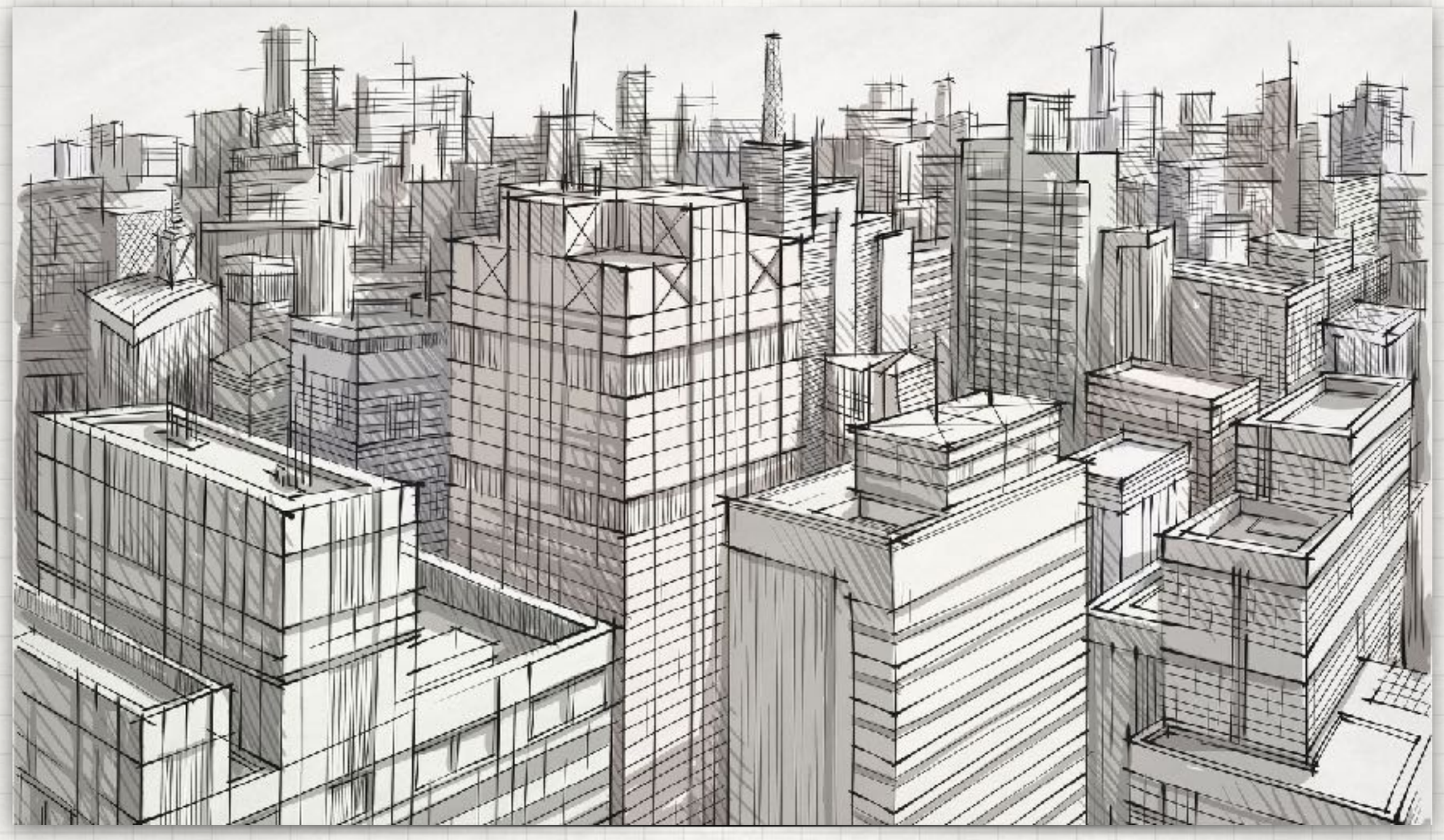
SWIFT ENUMS

“COMMENT J’AI DÉCIDÉ DE POLYMORPHISER”



ENUMS

"C'EST PAS DES INTS?"

En C (et héritiers), les enums sont effectivement des ints

```
typedef enum {
  left = 0,
  right = 1,
  top,
  bottom
} pad_direction;
```

ENUMS

“C’EST PAS DES INTS?”

```
void print_pad_dir(pad_direction p) {
  switch(p) {
    case left: printf(“left”);
      break;
    case right: (…)
  }
}

print_pad_dir(0); // ça marche

print_pad_dir(18); // ça marche
```

en théorie, les valeurs successives d’une enum valent des ints qui se suivent

ENUMS

"C'EST PAS DES INTS?"

Sucre syntaxique pour des constantes et une série de if/else if/else if/else

Ça aide un peu, quand même

ENUMS

“C’EST PAS DES INTS!”

```
enum PadDirection {
  case left;
  case right;
  case top;
  case bottom;
}
```

```
func printPadDir(p: PadDirection) {
  switch(p) {
    case .left: print(“left”)
    case .right: (…)
  }
}
```

ENUMS

“C’EST PAS DES INTS!”

```
print_pad_dir(0); // ça marche pô
```

le typage fort permet de vérifier que tous les cas sont pris en compte

en fait, on pourrait plutôt voir ça comme des *sous-types*

ENUMS

“OUI MAIS ÇA PEUT ÊTRE DES INTS… ?”

Oui, ok

```
enum PadDirection : Int {
// …
}

PadDirection.left.rawValue; // == 0
```

MAIS PadDirection.init(rawValue: 12) == nil, parce que 12 ne correspond à aucune valeur “autorisée”.

Ouf.

ENUMS

"OUI MAIS ÇA PEUT ÊTRE DES INTS... ?"

Ints / Strings / ... pour mapper les valeurs à destination d'un autre langage (JSON avec Codable, par ex)

```
do {
    let d = try JSONEncoder().encode(["pad" :
PadDirection.left])
    let s = String(data:d, encoding: .utf8) ?? "can't decode"
    print(s)

    if let j = "{ \"pad\": 1 }".data(using: .utf8) {
        let o = try
JSONDecoder().decode([String:PadDirection].self, from: j)
        print(o)
    } else {
        print("no data")
    }
} catch(let e) { print("error \(e)") }
```

ENUMS

“OUI MAIS ÇA PEUT ÊTRE DES INTS... ?”

```
{"pad":0}
["pad": test.PadDirection.right]
```

traduit vers un int dans un sens, vers une des valeurs de l’enum dans l’autre

ENUMS

"ET SI C'EST PAS DES INTS?"

Attachements (types)

```
case XXX(TTT)
```

```
enum PadDirection {
  case left(Double);
  // …
}
```

Wait a minute!

ENUMS

“ET SI C’EST PAS DES INTS?”

```
enum OpMath {
  case constante(Double);
  case unaire( (Double) -> Double );
  case binaire( (Double, Double) -> Double);
}

func plus(x: Double, y: Double) -> Double {
  return x + y
}

let opPlus = OpMath.binaire(plus)
```

Hmmmmm...

ENUMS

“ET SI C’EST PAS DES INTS?”

```
func faireOp(op: OpMath, args: [Double]) -> Double? {
  switch op {
    case .binaire(let f): // on récupère l’attachement
      if args.count == 2 {
        return f(args[0], args[1]);
      } else {
        return nil
      }
    default:
      return nil
  }
}

faireOp(op: opPlus, args: [1,2]) // et oui, ça fait 3
```

ENUMS

“QUEL INTÉRÊT SI C’EST PAS DES INTS?”

tous les cas mathématiques, correctement typés

vous n’aimez pas les maths?

```
enum WebContent {
  case text(String);
  case html(AttributedString);
  case image(UIImage);
  case json([String:Any]);
  case raw(Data);
  case error(Error);
}

func download(url: URL) -> WebContent // 😎
```

ENUMS

“QUEL INTÉRÊT SI C’EST PAS DES INTS?”

```
let c = download(url: u) // c’est quoiiiiiiiiiii?
switch c {
  case .text(let s):
    // faire un truc avec s (qui est du texte brut)
  case .html(let a):
    // faire un truc avec a (qui est du texte mis en forme)
  case .json(let j):
    // par exemple, faire un truc avec j[“texte”]
  default:
    // no text, yo
}
```

ENUMS

“CLAIREMENT, C’EST PAS DES INTS”

Un exemple pratique: Kitura

“nodejs en Swift”, autrement dit un framework web permettant de répondre à des requêtes http en Swift

```
router.post(“/user/:id”) { request, response, next in
  let userId = request.parameters[“id”]
  if let b = request.body {
    switch b {
      // … c’est une enum
    }
    request.status(.OK).end() // ça aussi (OK = code 200)
  }
  request.status(.notFound).end() // et la je fais semblant
que la page n’existe pas si les args sont faux
}
```

ENUMS

“CLAIREMENT, C’EST PAS DES INTS”

body dans Kitura?

```
enum ParsedBody: text(String), json([String:Any]),
multipart([Part]), raw(Data)
en fonction du header Content-Type
```

Part?

```
struct Part {
  var name: String
  var contentType: String
  var content: ParsedBody // 😎
}
```

ENUMS

"CLAIREMENT, C'EST PAS DES INTS"

Autre cas typique: lireUnTruc(…) -> (Data?, Error?)

data == nil? error == nil? les deux? aucun?

```
enum ResultatLecture {
  case success(Data);
  case error(Error);
}

lireUnTruc(…) -> ResultatLecture
```

seulement deux cas possibles, le type associé étant *forcément bon*

ENUMS

BONUS ROUND

C’est un type comme un autre

```
enum {
  // …

  func evaluate() {
    switch self {
      case .cas1: // …
    }
  }
}
```

Fonctions *dans* des enums!

Enums dans des enums… 🤯

ENUMS

BONUS ROUND #2

Les enums marchent avec des tuples

```
switch (i % 3, i % 5) {
  case (0, 0):
    print(“Toto!")
  case (0, _):
    print(“Oeil Gauche")
  case (_, 0):
    print(“Oeil Droit")
  default:
    print(“Pourri, ce jeu! \(i)”)
}
```

ENUMS

BONUS ROUND #3

Les enums supportent le binding optionnel avec *where*

```
switch (i) {
  case let x where x % 3 == 0:
    print(“Trois!”)
  case let y where y % 5 == 0:
    print(“Cinq")
  default:
    print(“Pourri, ce jeu! \(i)”)
}
```

ou bien

```
switch (i) {
  case _ where i % 3 == 0:
    print(“Trois!”)
  case _ where i % 5 == 0:
    print(“Cinq")
  default:
    print(“Pourri, ce jeu! \(i)”)
}
```

@KRUGAZOR

ZINO@POULET.ORG

QUESTIONS

(À PART SUR LES CONTACTS OU LA COLORATION SYNTAXIQUE)