MAKING IT GENERIC

JON-TAIT BEASON

@BUGKRUSHA

GLOWFORGE

GENERICS: ABSTRACTION & SPECIALIZATION

ABSTRACTING AWAY PROGRAM DIFFERENCES TO GET A SINGLE UNIFIED GENERIC PROGRAM

GENERICS: FLEXIBILITY

```
struct Resource<Attribute> {
    let attribute: Attribute

    ///
}
```

GENERICS: TYPES

- VALUE
- FUNCTION
- TYPE

GENERICS BY VALUE

HOPEFULLY WE ARE ALL DOING THIS 😬

GENERICS BY VALUE: ASCII ART

```
*
**
***
****
*****
```

```
func drawAsciiArt() {
    print("*")
    print("**")
    print("***")
    print("****")
    print("*****")
}
```

GENERICS BY VALUE:ASCII ART

```swift
func drawAsciiTriangle(height: Int, ascii: Character) {
    for row in 1...height {
        var line = ""
        for _ in 1...row {
            line.append(ascii)
        }
        print(line)
    }
}
```

GENERICS BY VALUE: ASCII ART

```
drawAsciiTriangle(height: 7, ascii: "😘")
```

```
😘
😘😘
😘😘😘
😘😘😘😘
😘😘😘😘😘
😘😘😘😘😘😘
😘😘😘😘😘😘😘
```

GENERICS BY FUNCTION

WHOO

GENERICS BY FUNCTION

```
func makeLowerCase(list: [String]) -> [String] {
    var lowerCaseList: [String] = []

    for string in list {
        lowerCaseList.append(string.lowercased())
    }
    return lowerCaseList
}
let lowercased = makeLowerCase(list: ["ONE", "LOVE", "mi", "bredda"])
// ["one", "love", "mi", "bredda"]
```

GENERICS BY FUNCTION

```swift
func evenOdd(list: [Int]) -> [Bool] {
    var evenOddBools: [Bool] = []

    for number in list {
        evenOddBools.append(number % 2 == 0)
    }
    return evenOddBools
}
let evenOddList = evenOdd(list: Array(1...5))
// [false, true, false, true, false]
```

GENERICS BY FUNCTION: MAP

TRAVERSING A LIST
APPLYING A TRANSFORM FUNCTION

```
// make lower case
lowercased()

// even odd
number % 2 == 0
```

GENERICS BY FUNCTION: MAP

```
extension Array {
    func map<T>(_ transform: (Element) -> T) -> [T] {
        var result: [T] = []
        for x in self {
            result.append(transform(x))
        }
        return result
    }
}
```

MAP: FLEXIBLE AND SAFE

```
let lowercased = ["ONE", "LOVE", "mi", "bredda"].map { $0.lowercased() }
// ["one", "love", "mi", "bredda"]

let evenOddList = Array(1...5).map { $0 % 2 == 0 }
// [false, true, false, true, false]
```

GENERICS BY TYPE: STACK

CONTROLLING TYPE EXPANSION

GENERICS BY TYPE: STACK

```
list // Can be an array
push(value: Int) // Adds an element to the list
pop() -> Int? // Removes the last element from the list.
```

GENERICS BY TYPE: EMOJI STACK

```
class Stack {
    private var list: [Character] = []

    func push(value: Character) {
        list.append(value)
    }

    func pop() -> Character? {
        return list.popLast()
    }
}
let emojiStack = Stack()
emojiStack.push(value: "🤣")
```

GENERICS BY TYPE: INT STACK? 😬

```
let intStack = Stack()
intStack.push("23")

// Cannot convert value of type Int to expected argument type Character
```

GENERICS BY TYPE: INT STACK

```
class IntStack {
    private var list: [Int] = []

    func push(value: Int) {
        list.append(value)
    }

    func pop() -> Int? {
        return list.popLast()
    }
}
let intStack = IntStack()
intStack.push(value: 23)
```

GENERICS BY TYPE

```
class Stack<Element> {
    private var list: [Element] = []

    func push(value: Element) {
        list.append(value)
    }

    func pop() -> Element? {
        return list.popLast()
    }
}
```

GENERICS BY TYPE

```swift
let intStack = Stack<Int>()
intStack.push(value: 23)

let emojiStack = Stack<Character>()
emojiStack.push(value: "🤣")

// No shade from the compiler
```

JAVASCRIPT OBJECT NOTATION: JSON

PARSING HETEROGENEOUS DATA

JSON

```json
{
    "firstName": "Jah Zie",
    "lastName": "Tini",
    "age": 35,
    "address": {
        "streetAddress": "247 My Bag",
        "city": "Seattle",
        "state": "WA",
        "postalCode": "98104"
    }
}
```

JSON & GENERICS

```
struct Person: Codable {
    let firstName: String
    let lastName: String
    let age: Int
    let address: Address
}

let decoder = JSONDecoder()
let jaz = decoder.decode(Person.self, from: jsonData)
```

GENERICS: JSON API

```json
{
    "meta": {
        "type": "identifier",
        "id": "0b6a12ec-343d-4830-b029-4ed648e4c5d7",
        "resource_type": "print"
    },
    "data": {
        "type": "print",
        "id": "1c871eec-44e7-4123-b14e-3e51646f6d5c"
    }
}
```

JSON AT GLOWFORGE

```
struct Printer: Codable {
    let id: String

    ///
}

struct PrintActivity: Codable {
   let timeRemaining: Double

   ///
}
```

META

```swift
enum ResourceType: String, Codable {
    case printActivity
    case printer
    /// ...
}

struct Meta: Codable {
    let type: MetaType
    let id: String
    let resourceType: ResourceType
}
```

DATA: RESOURCE PACKET

```swift
struct ResourcePacket<Resource: Codable>: Codable {
    let type: ResourceType
    let id: String

    func incomingRequest() -> APIRequest<Resource> {
        return APIRequest(id: id, type: type)
    }
}
```

API REQUEST

```swift
struct APIRequest<Resource: Codable> {
    let id: String
    let type: ResourceType

    var url: String {
        var dns = "www.glowforge.com"
        switch type {
        case .printActivity:
            dns += "/prints/\(id)"
        case .machine:
            dns += "/printers/\(id)"
        }
        return dns
    }
}
```

RESOURCE IDENTIFIER

```
struct ResourceIdentifier<Resource: Codable> {
    let meta: Meta
    let data: ResourcePacket<Resource>

    var request: APIRequest<Resource> {
        return data.incomingRequest()
    }
}
```

NETWORK CALL

```
class API {
    func perform<Resource>(_ request: APIRequest<Resource>, completion: @escaping(Resource) -> Void) {
        ///...
        if let resource = try? decoder.decode(Resource.self, from: data) {
            completion(resource)
        }
    }
}
```

OBJECT MANAGER

```
class Manager {
    func updateReceived<Resource>(identifier: ResourceIdentifier<Resource>,
                                  resourceReturned: @escaping(Resource) -> Void) {
        api.perform(identifier.request) { resource in
            resourceReturned(resource)
        }
    }
}
```

SOCKET MESSAGE RECEIVED

```swift
func socketMessageReceived(meta: Meta, data: Data) throws {
    let decoder = JSONDecoder()
    switch meta.resourceType {
    case .printer:
        let packet = try decoder.decode(ResourcePacket<Printer>.self, from: data)
        let identifier = ResourceIdentifier(meta: meta, data: packet)
        printerManager.updateReceived(identifier: identifier) { printer in
            /// Use printer here
        }
    case .printActivity:
        /// Handle print activity here
    }
}
```

TAKE AWAY

HIGHLY REUSABLE COMPONENTS MUST BE BUILT WITH A MINIMUM SET OF REQUIREMENTS

@BUGKRUSHA