ReferenceVsValueTypes Assignment

Research over the weekend further into Reference types and Value types outside what has been discussed in class.

Write a small one page paper on:

-What each is (reference and value type)

-how are they different

-how they are utilized in iOS, Swift, and ObjC

-pros and cons to either

Write a flow chart on when it is better to use one vs the other.

-Recommendation for this is to create a UML diagram flow chart

-Look up Lucid Chart, it is a free diagram creating tool that is useful, but others can be used as well. (Lucid Chart just integrates seamlessly with common task tracking tools such as JIRA)

-There will be times you will be expected to desing a system, so it is wise to familiarize yourself with UML diagrams early rather then on the job

1.

Value-types is a category where every instance keeps a unique copy of its data, such as a struct or enum. Reference-types is the category where instances share a single copy of the data, like a class.

One of the most important differences is that copying a value type creates an independent instance with its own unique copy of its data. Conversely, copying a reference value type creates a shared instance.

In iOS, Swift and ObjC they are used as structs and enums for value types, and as classes for reference types.

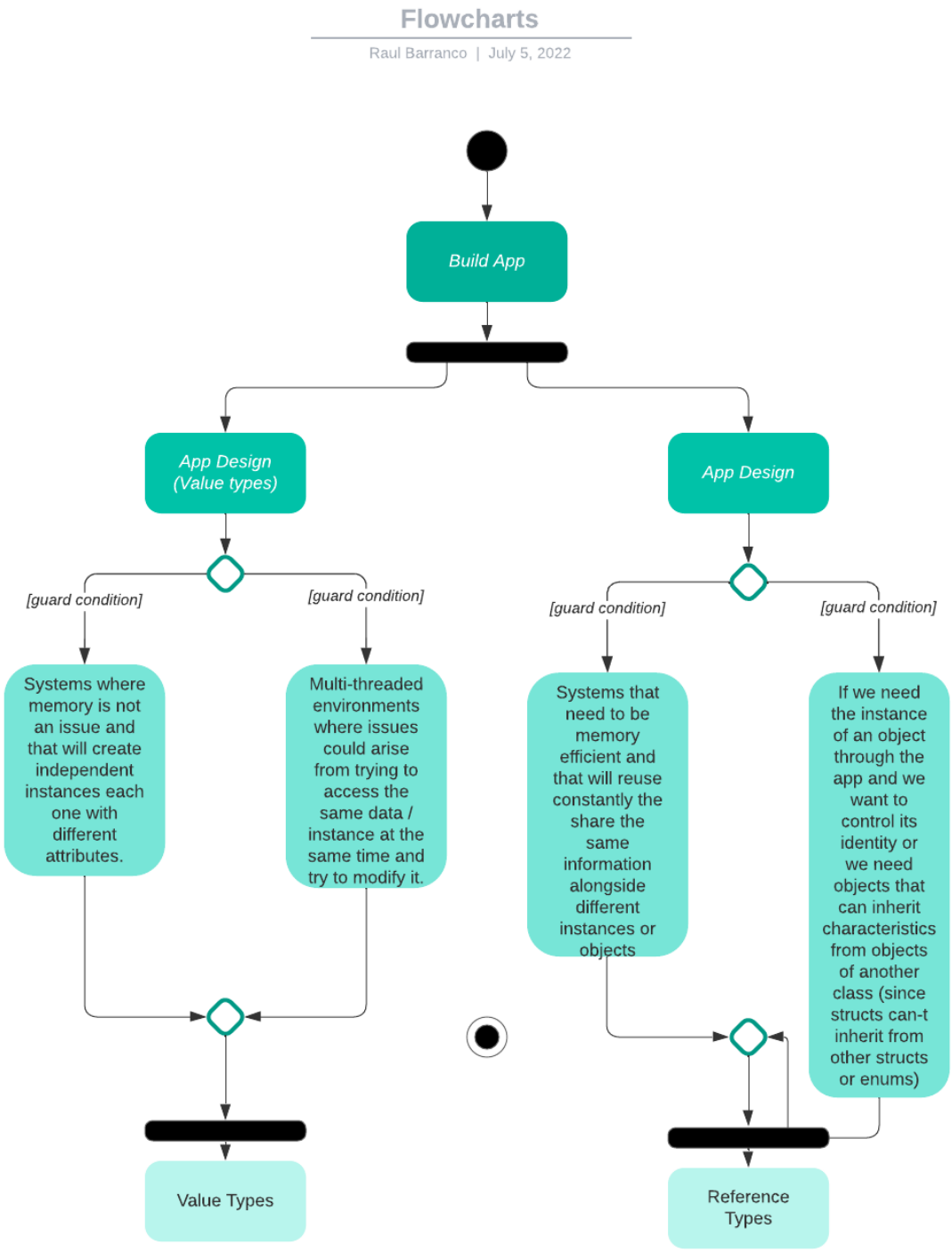
The pros of value types are that they are speed effective, since they can access a value directly instead of having to allocate it first. Additionally, there is no risk for a given instance to have changed the data accidentally the way it can change for reference types. This can be especially helpful in multi-threaded environments where a different thread could alter the data unexpectedly.

The cons of value-types are that they cannot be modified and they will always create a new copy of an instance, making value types memory-expensive. Value-types, in the form of structs and enums, don't support inheritance.

The pros of reference types are that they don't create a new or extra copy of the data, because they reference the same data that is already created, hence making them memory effective with dynamic memory allocation. Also, reference types, in the form of classes in Swift, offer all the benefits of inheritance, whilst structs and enums don't support inheritance.

The cons of reference types can be that they are speed-expensive in terms of their performance. Moreover, when implementing reference-types data could be accidentally modified, making it more prone to unexpected errors and harder to debug. Also, Swift does not currently provide any language mechanism to enforce class immutability the way it enforces immutability on struct and enum.

2.


Flowcharts
Raul Barranco | July 5, 2022
Build App
App Design (Value types)
App Design
[guard condition]
[guard condition]
[guard condition]
[guard condition]
Systems where memory is not an issue and that will create independent instances each one with different attributes.
Multi-threaded environments where issues could arise from trying to access the same data / instance at the same time and try to modify it.
Systems that need to be memory efficient and that will reuse constantly the share the same information alongside different instances or objects
If we need the instance of an object through the app and we want to control its identity or we need objects that can inherit characteristics from objects of another class (since structs can-t inherit from other structs or enums)
Value Types
Reference Types