CRITERIA REGARDING CHARACTERISTIC OF CLASSES

Description

Classify classes into one of the following role stereotypes:

- Information Holder
- Structurer
- Service Provider
- Controller
- Coordinator
- Interfacer

Role stereotypes:

Information Holder

An object designed to know certain information and provide that information to other objects.

Criteria:

- May contains “Java library provided object” type or primitive type attributes and also getter/setter methods or methods for accessing those attributes.
- May contains persistence feature(s), e.g. saving to database or implementing Java’s Serializable interface.
- May be represented as enum class.
- Can be an interface, if its methods are only setters and getters (in general: giving access to its attributes)

Structurer:

An object that maintains relationships between objects and information about those relationships. Complex structures might pool, collect, and maintain groups of many objects; simpler structures maintain relationships between a few objects.

Example: Java HashMap, relates key to values

Criteria:

- May contain user defined object type as attributes
- May extend/implement Java’s Collection framework or equivalent
- Has method(s) to maintain relationships between objects
    + methods that manipulate the collection such as *sort(), compare(), validate(), remove(), updates(), add(), etc.*

+ methods that give access to a collection of objects such as get(index), next(), hasNext(), etc.

Service provider:

An object that **performs specific works** and **offers services** to others on demand.

Criteria:

- Class name may end with "-er" (eg. Provider) or "-or" (eg. Creator, Detector)
- Might has methods and attributes that are easily accessed by other classes (often static and public, or protected, not private)
- Could be realization of an Interface
    - Contains overloading or overriding method (for abstract- or sub- classes)
- The methods perform specific work
    - May use other methods to help doing the work (e.g. internal/private methods in the same class).
    - May overload or override methods from its parent classes
- The method might make decision itself to perform the tasks.
    - The decision should be at a very basic level, only to support a specific work (could be a call to library functions)
- Contains or read configurations

Controller:

An object designed to **make decisions** and **control complex task**.

Criteria:

- Class name may ended with "Controller" or "Manager"
- Should have access to information holders, coordinators, or service provider
- Its main responsibility is to make decision to control the flow of the application
    - Should contain condition statements (e.g. IF, IF ELSE, SWITCH CASE, x : ?)[1][2]
    - The decision should be at the higher level than decision made at Service Provider/Coordinator.

Coordinator:

An object that doesn't make many decisions but, in a rote or mechanical way, **delegates work to other objects**.

Features:

- Its main responsibility is to coordinate the works among other stereotypes.
    - Passing and gathering information and requests
    - Might have some logics to break down the work and selecting suitable executing units
- Called by other classes (could be from a controller, an interfacer)
- Should delegate the work by calling other object's methods and passing parameters

[1] https://docs.oracle.com/javase/tutorial/java/nutsandbolts/if.html
[2] https://docs.oracle.com/javase/tutorial/java/nutsandbolts/switch.html

Interfacer:

An object that transforms information or requests between distinct part of the system. The edges of an application contain user-interfacer objects that interact with the user and external interfacer objects, which communicate with external systems. Interfacer also exist between subsystems.

Translate requests and convert from one level of abstraction to another

Types:

1. User interfacer
    a. transmits user requests for action or display information that can be updated
    b. typically collaborate only with objects in other non-UI parts of the application to signal events or changes in the information display
2. Internal interfacer
    a. provide outsiders a limited view into an object neighbourhood
    b. serve as the "storefront" to services offered to outsiders
    c. convey requests to objects hidden from the view
    d. collaborates by delegating external requests to objects in its neighbourhood
    e. whom it collaborates with and how it does depend on how transparently it packages the services it offers
3. External interfacer
    a. usually do not collaborate with many other application objects
    b. may delegate to service providers the responsibility to format or convert information that they push or pull from their external partners, mostly just encapulate non object oriented APIs

Criteria:

- May contains Java Swing[3], AWT[4], and other UI components → User interfacer
    - In Android Applications: contains UI Android classes, such as Activity class
- Manage user interface and handle user interaction
    - In Android apps, this extends Activity classes (define how the class react to activity - for example onCreate())
- Encapsulates functions or objects in the system by providing an Interface or an abstract class that can be used outside of the system

[3] https://docs.oracle.com/javase/tutorial/uiswing/start/about.html
[4] https://docs.oracle.com/javase/7/docs/api/java/awt/package-summary.html