Source

Rebecca J Wirfs-Brock, *Characterizing Classes*, IEEE Software Vol. 23 No. 2, 2006

Rebecca J Wirfs-Brock and Alan McKean, *Object Design: Roles, Responsibilities, and Classifications*, Addison-Wesley, 2002

Description

Classify classes into one of the following role stereotypes:

- Informaton 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:

- Name of the class is simple noun, basic word
- Contains "Java library provided object" type or primitive type variables
- Contains getter and setter methods for the variables
- May contains persistence feature, eg. saving to database or implements Java's Serializable interface
- **Complete information, not abstract class**

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:

- Contains other "user defined object" type as variables (Composition and Aggregation)
- Extends Java's Collections framework[1]
- **Has methods that maintaining relationships between objects or collections**
  - **methods that manipulate the collection such as sort(), compare(), validate(), remove(), updates(), add(), delete() ...**
  - **methods that give access to the objects such as get(index), next(), hasNext() ...**

Service provider:

[1] https://docs.oracle.com/javase/7/docs/technotes/guides/collections/overview.html

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

Criteria:

- Class name is ended with "-er" (eg. **Provider**) or "-or" (eg. Creator, Detector) but not **Controller**
- Has methods and attributes are easily accessed by other classes (often static and public, or protected, not private)
- **Could be realization of a Interface (the interface should be a service provider as well)**
    - **Contains overloading or overriding method**
- Contains or read configurations

Controller:

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

Criteria:

- Class name is ended with "Controller", "**Manager**"
- Should know (have access to) information holders, coordinators, service provider
- Has decision making statement (IF, IF ELSE, SWITCH CASE, x : ?)[2][3]
- When the work becomes too complicated, a controller could ask a coordinator to coordinate the work.

Coordinator:

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

Features:

- Holding connection between working objects (service provider)
- Forwarding information and requests information
- **Delegate works to other classes, by creating object of other class and calling their methods**
    - **usually when Service Provider becomes too big, it evolves into Coordinator**
- **Called by other classes (could be from a controller, an interfacer)**
    - **Passing parameters (communication) to other classes**

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

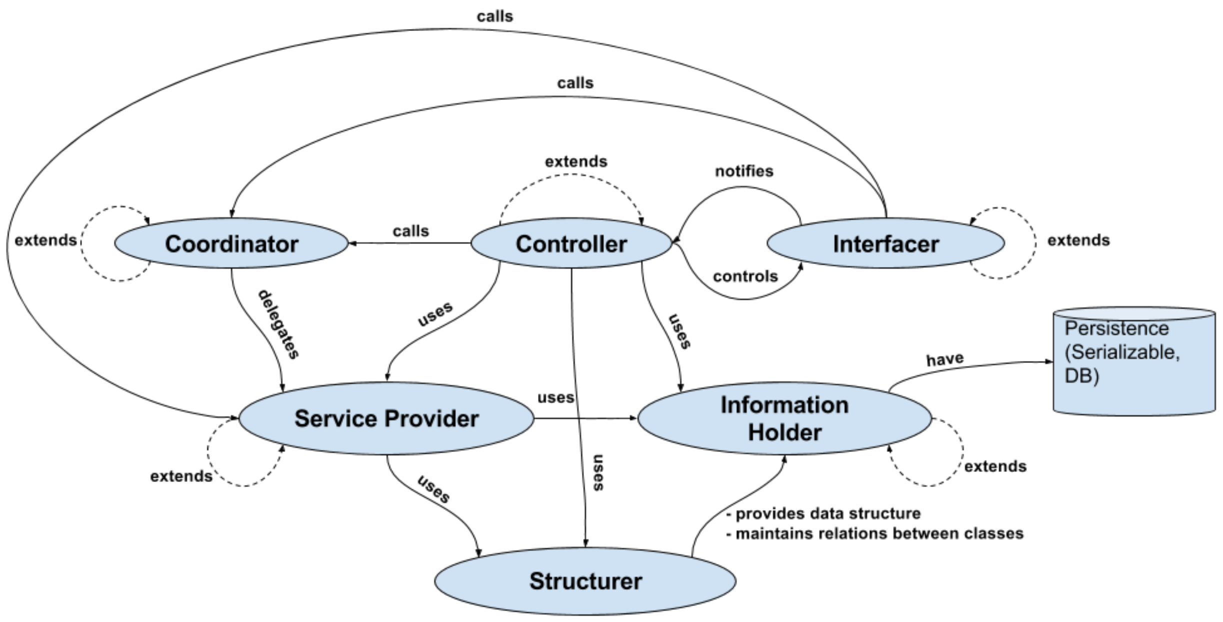
[2] https://docs.oracle.com/javase/tutorial/java/nutsandbolts/if.html
[3] https://docs.oracle.com/javase/tutorial/java/nutsandbolts/switch.html

    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:

- Contains Java Swing[4], AWT[5], and other UI components → User interfacer
    - In K9mail: contains UI Android classes, such as Activity
- Import many user defined other classes
- 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
    - If an interface is created but never implemented: may be this serves as an extension point for the system, which could be considered as "internal interfacer"

The Relationships between Roles



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