En un double dispatch (DD), ¿qué información aporta cada uno de los dos llamados?

En el primer llamado, el objeto receptor se envía a sí mismo como colaborador, hacia su colaborador externo, luego desde el segundo llamado, el objeto receptor realiza alguna operación con el colaborador externo recibido.

Con lo que vieron y saben hasta ahora, ¿donde les parece mejor tener la lógica de cómo instanciar un objeto? ¿por qué? ¿Y si se crea ese objeto desde diferentes lugares y de diferentes formas? ¿cómo lo resuelven?

Dentro del mismo objeto, ya que si se hiciese de otra forma probablemente se violaría el encapsulamiento. Si se crea de diferentes formas, se pueden definir métodos de clase en el mismo objeto que reciban distintos tipos de colaboradores, y devuelvan un nuevo objeto instanciado en base a estos.

Con lo que vieron y trabajaron hasta ahora, ¿qué criterio están usando para categorizar métodos?

Los separamos en privados y públicos de acuerdo a si pueden ser accedidos por otros objetos, y en métodos de acceso si es que están hechos para acceder a atributos del objeto. También hicimos una categoría para los métodos usados en el double dispatch, ya que no son métodos privados (se usan en las colaboraciones entre enteros y fracciones), pero al mismo tiempo no se supone que el usuario los use.

Si todas las subclases saben responder un mismo mensaje, ¿por qué ponemos ese mensaje sólo con un “self subclassResponsibility” en la superclase? ¿para qué sirve?

Para indicar que cualquier clase hija nueva que se cree debe tener su propia implementación de ese mensaje, y que el mensaje en la superclase es abstracto.

¿Por qué está mal/qué problemas trae romper encapsulamiento?

Debido a que, si los métodos o atributos que se utilizan rompiendo encapsulamiento cambian, habría que modificar los llamados a estos métodos o atributos que se hayan hecho por fuera del objeto, pudiéndose tener que realizar varias modificaciones en distintos objetos y causando bugs.