Recuerdo (Memento)

- *Patrón de Comportamiento*

- *Propósito*

  Permite capturar y extraer el estado interno de un objeto, sin violar su encapsulación, de tal modo que se puede restaurar su estado a posteriori

- *Motivación*

  - Operaciones de deshacer, transacciones...

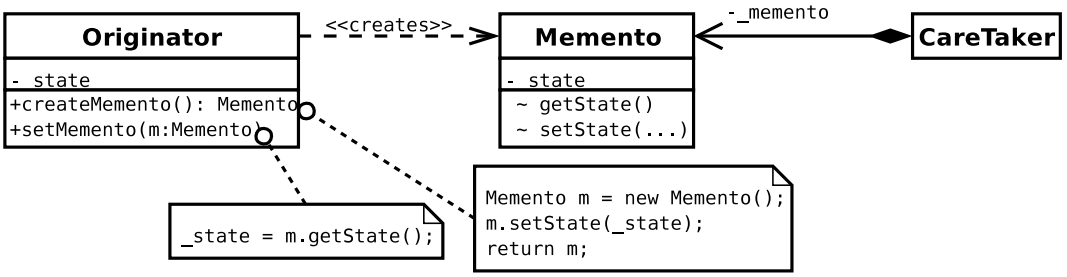
- *Aplicabilidad*

  - Cuando se dan las dos siguientes condiciones:

    - Una imagen del estado del objeto (o un subconjunto) debe ser almacenado de tal forma que pueda ser restaurado más tarde

    - Una interfaz directa para obtener el estado expondría detalles de implementación, violando la encapsulación del objeto

- *Estructura*



- *Participantes*
  - Recuerdo (Memento)

    Almacena el estado interno de la clase Originante

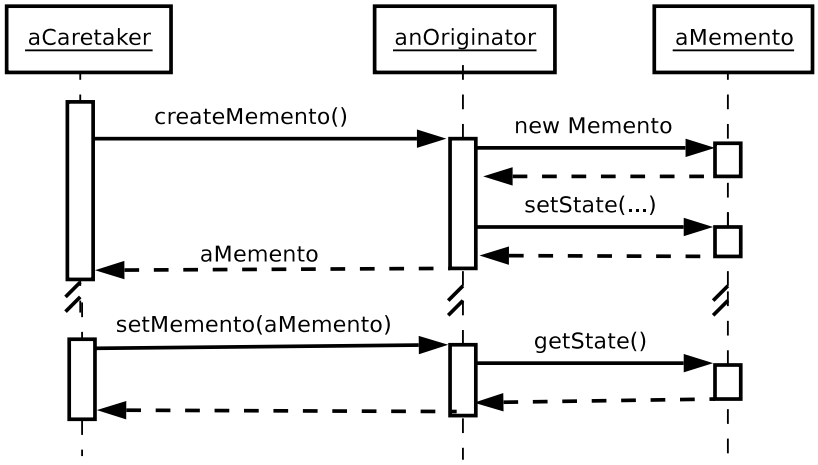
    Sólo permite el acceso al estado almacenado a la clase Originante
  - Originante (Originator)

    Crea un Recuerdo con la imagen de su estado actual

    Usa un Recuerdo para restaurar su estado interno
  - Cuidador (Caretaker)

    Responsable de almacenar los Recuerdos

    No opera o examina el contenido de los Recuerdos
- *Colaboraciones*
  - El cuidador solicita un recuerdo del originante, lo almacena por un tiempo, y si es necesario, lo devuelve al originante



  - Los recuerdos son *pasivos*

- *Consecuencias*
  - Evita la exposición de información que sólo debe gestionar el originante, pero que debe guardarse fuera de éste
  - Simplifica el originante, al separar la gestión de los recuerdos
  - Puede resultar costoso
  - Difícil en algunos lenguajes el soporte de dos visibilidades distintas
  - Oculta el coste de almacenamiento del recuerdo
- *Implementación*
  - Soporte de dos visibilidades diferentes
  - Almacenamiento de cambios incrementales
    - Particularmente útil en la implementación de operación deshacer dado que se conoce la secuencia de cambios realizados (histórico)