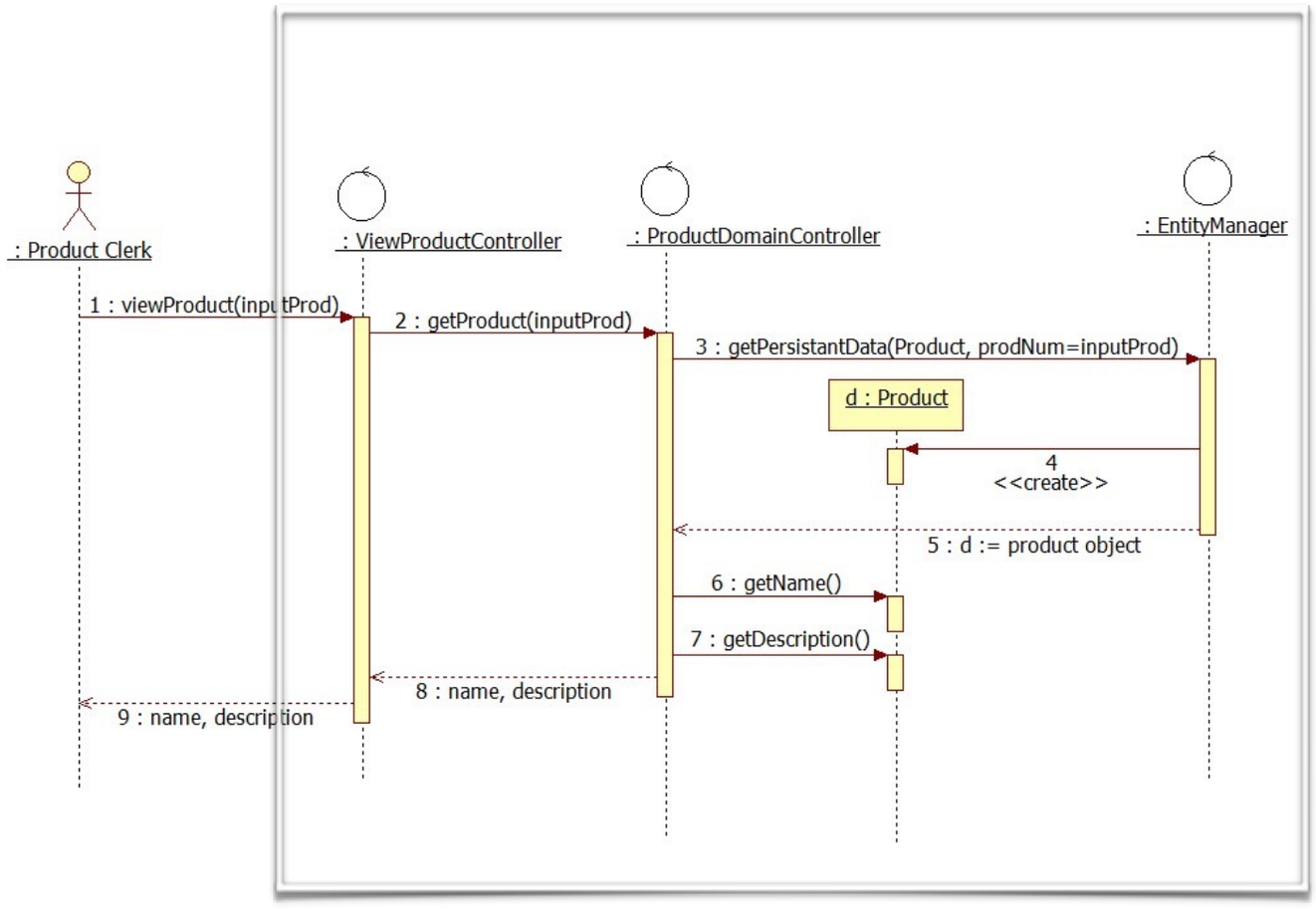
SYS466
Analysis and Design

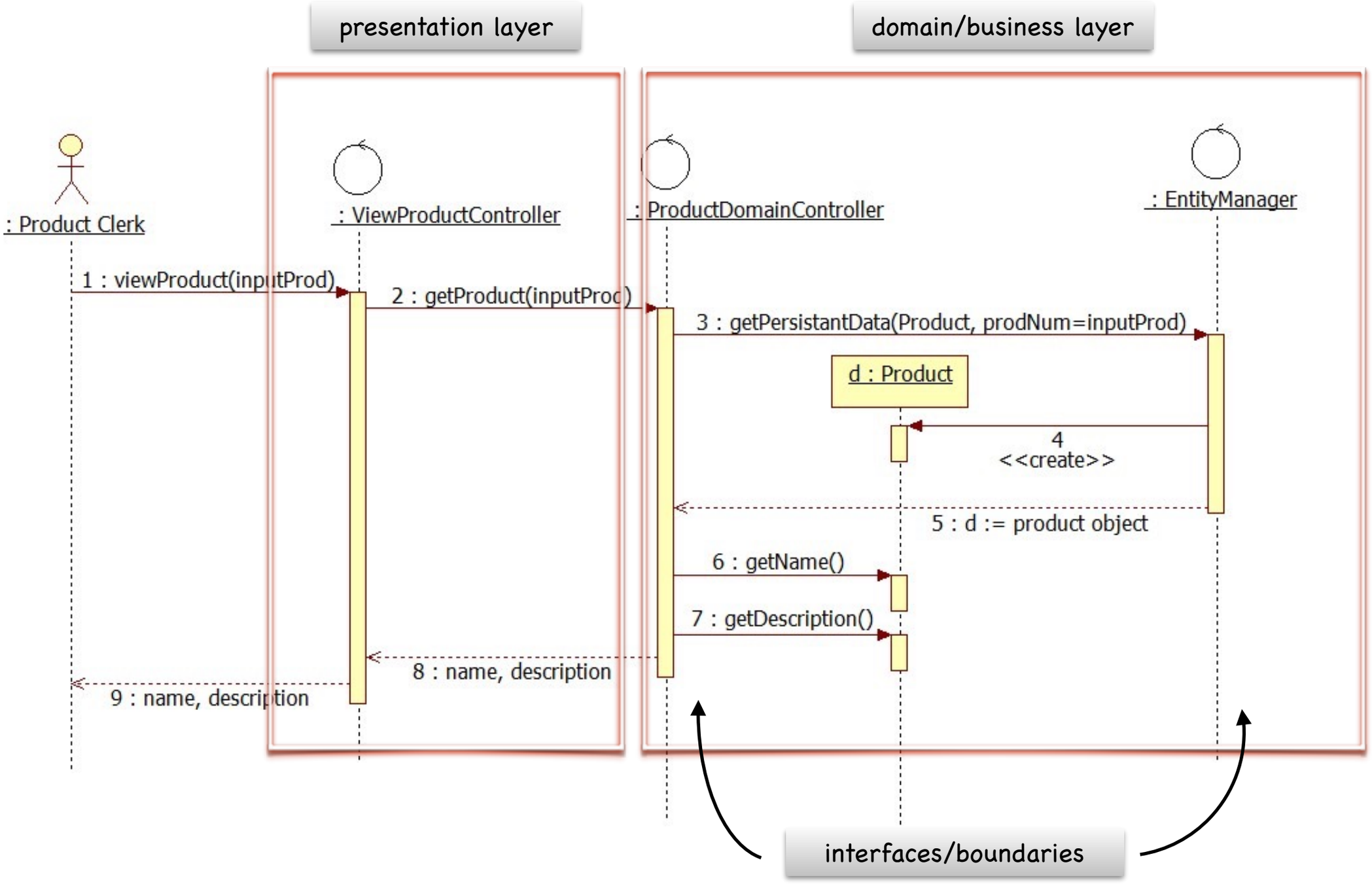
Lecture 7 - Object Sequence Diagrams II
School of Information and Communications Technology
Seneca College

Object System Sequence Diagrams

- opens up the “black box”
- documents interaction for a single scenario
- shows how objects collaborate to fulfil a request
- used to illustrate ordered sequence of messages between objects
- not good at describing exact behaviour



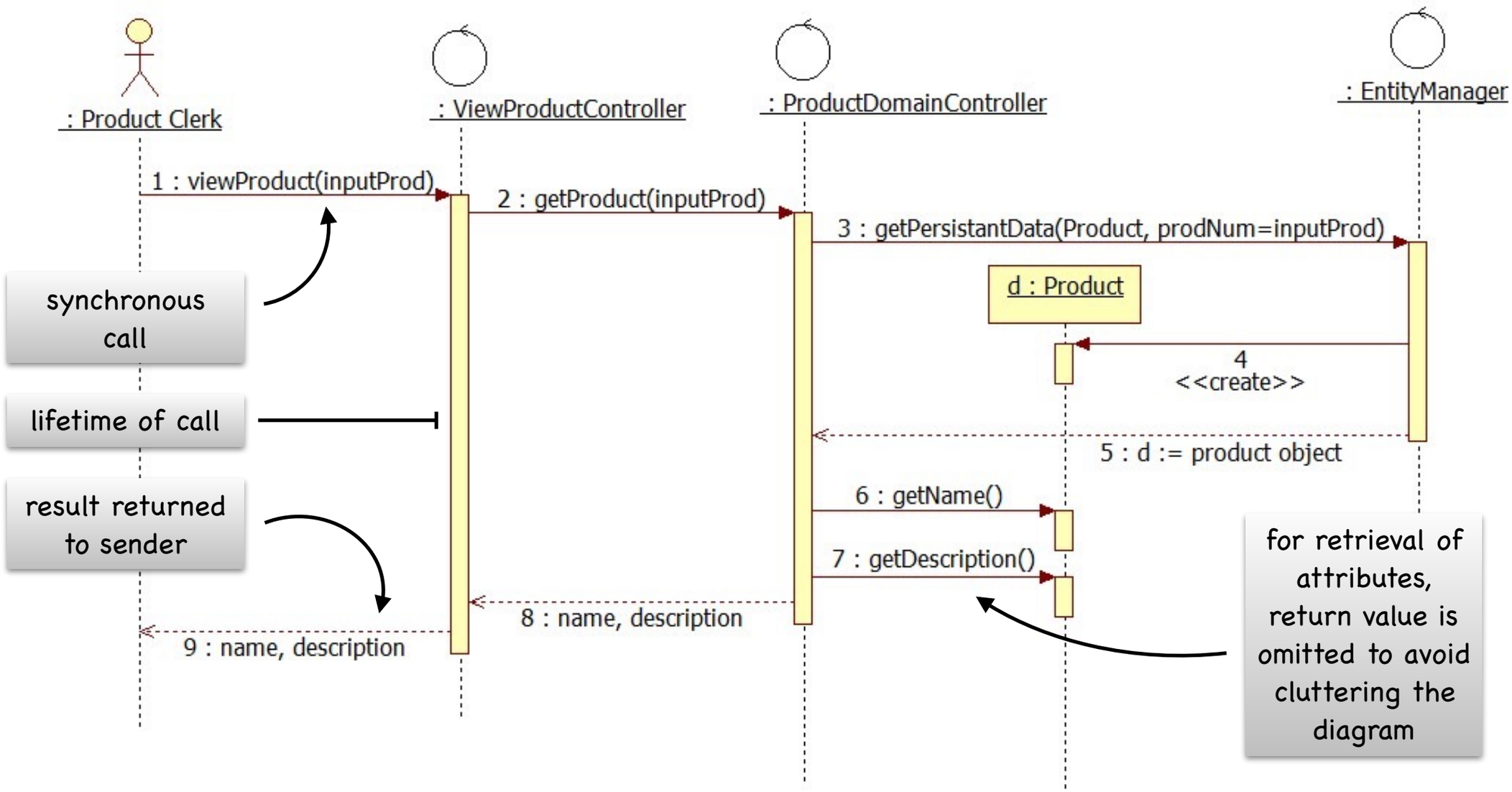
System Details



Layered Pattern Collaborations

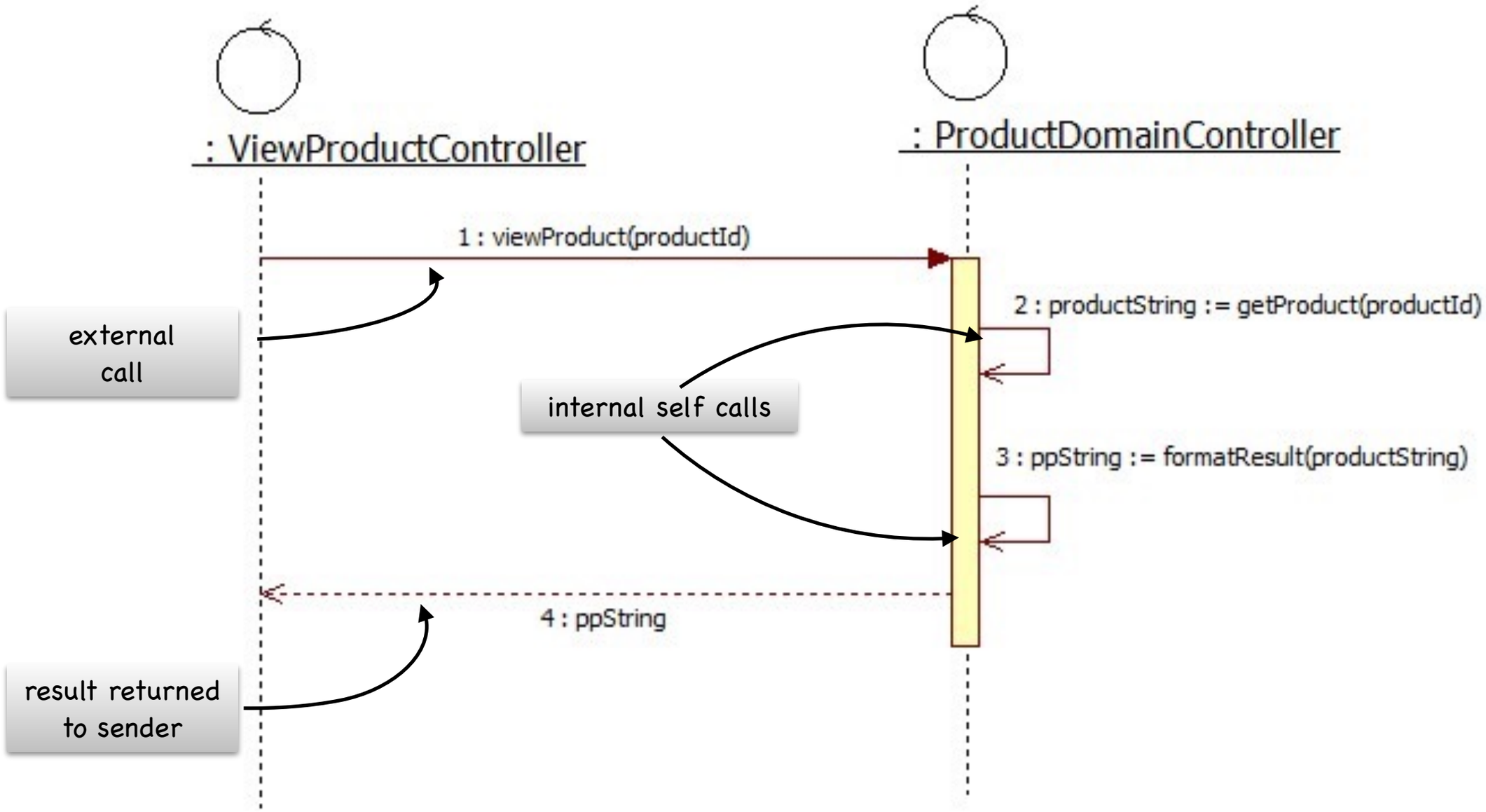
Objects across layers collaborate to provide services

Controllers provides interfaces to access to neighbouring layers



Operations

Models invoking a class method on an object or triggering a event



Operations

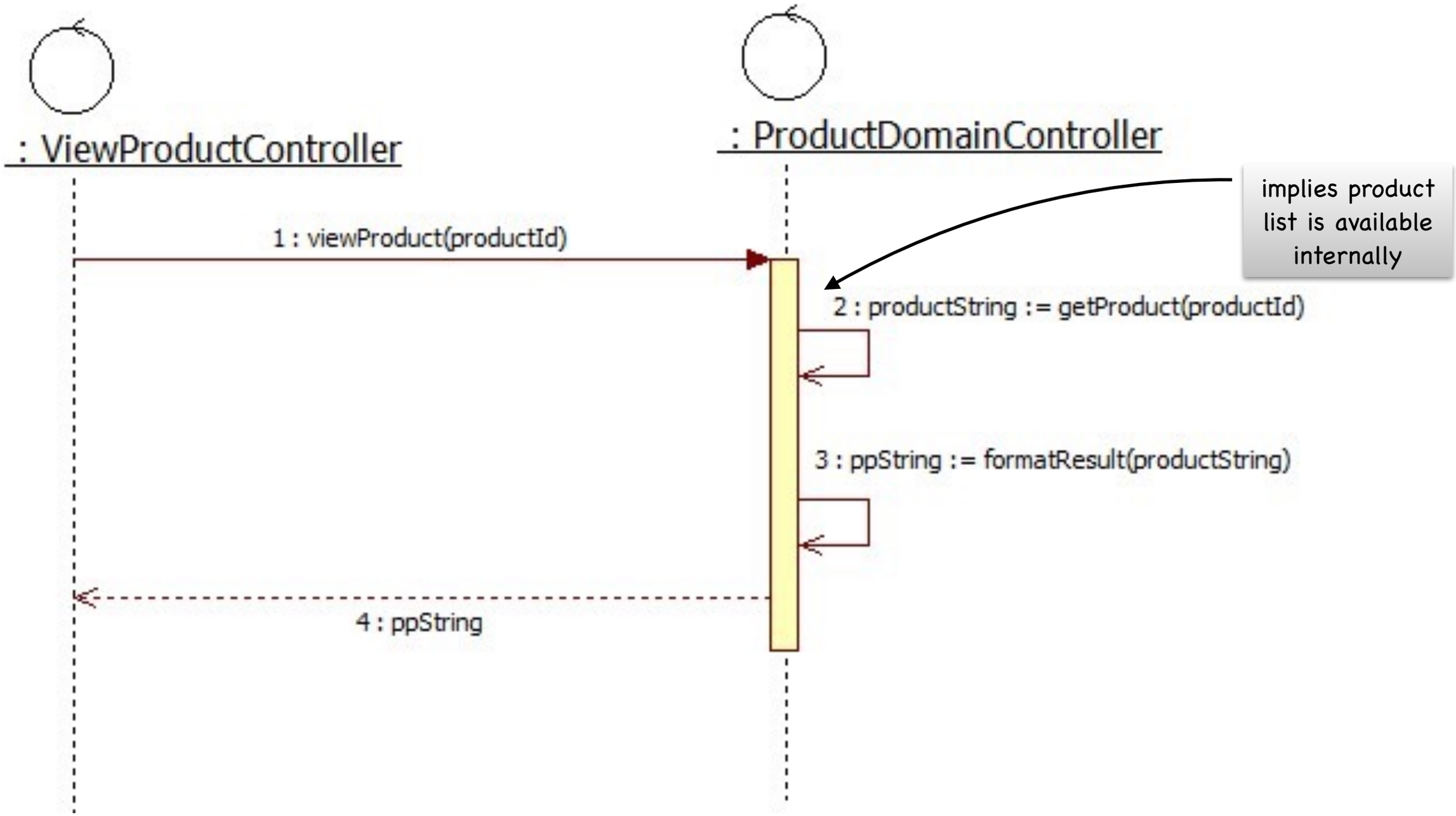
Receiver must have an operation to handle message
Objects can send messages to itself

Internal/External Operations

```
class Store { Employee e; Warehouse w; …. bool isOpen(); }

Product* Store::sellProduct(int productID) {
  if (w.isProductAvailable() and e.isAuthorizedSeller() and isOpen() )
     Product p = w.removeProductFromInventory();
     p.setSoldBy(e);
     return p
  }}

bool Store::isOpen() { return open == true;}

main() {
  Store s = new Store();
  …
  s.signIn(new Employee(“Jack”));
  s.addWarehouse(new Warehouse(“Toronto Depot”) );
  …
  Product p = s.sellProduct(345);}
```

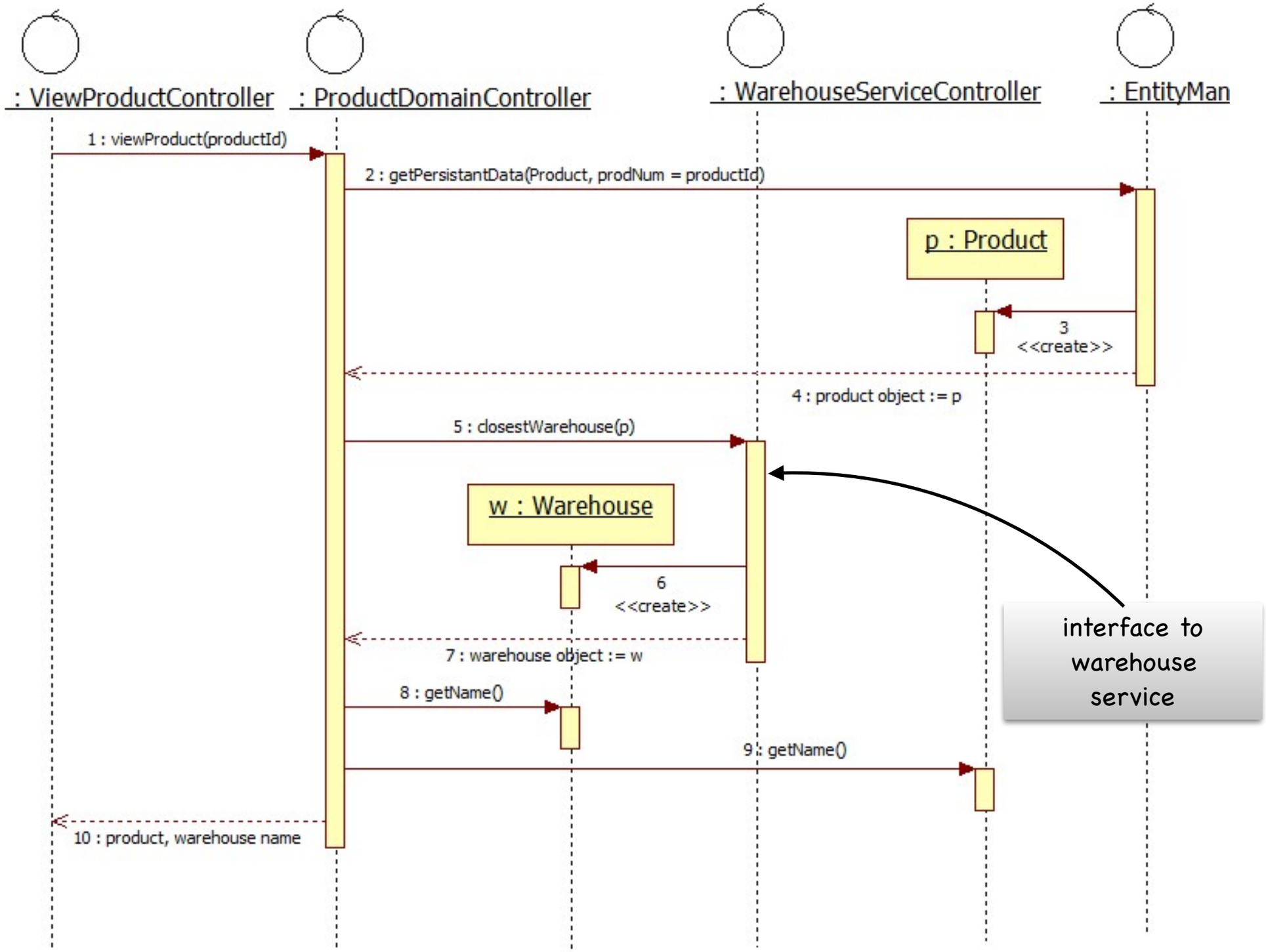
internal self call

external call



Centralized Control Approach

Object handles most of the responsibility of handling a incoming message



Distributed Control Approach

Object delegates most of the responsibility of handling a incoming message to other objects

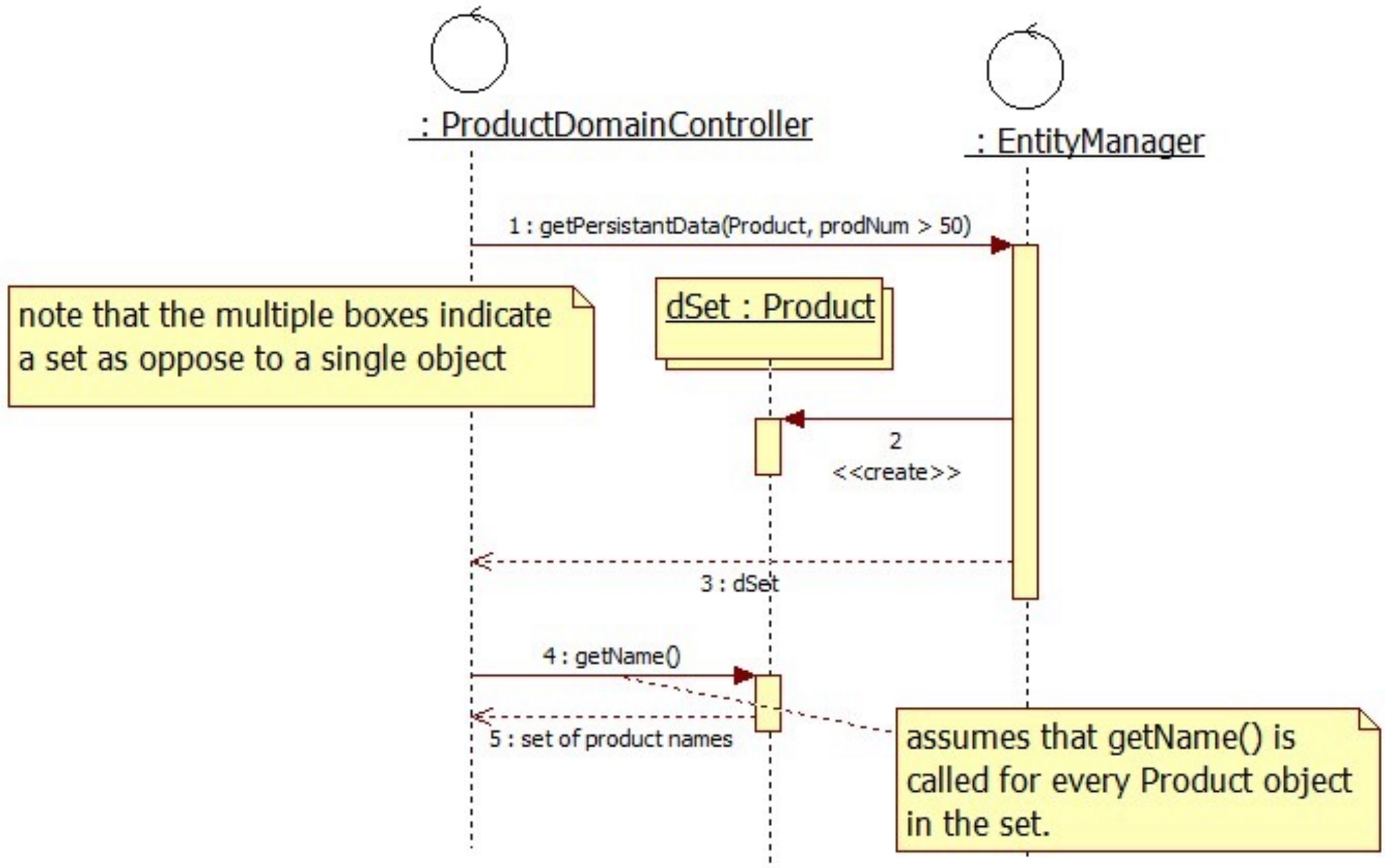
Design Considerations

Centralized Control

- simplified design (less need to worry about complexity of communication between objects)
- efficiency (can reduce overhead/latency by removing time needed to communicate between objects)

Distributed Control

- flexible design (less coupling between objects, so changes can be made to one without affecting the others)
- efficiency (throughput can be increased by allowing multiple object “clones” to service requests - multiple requests can be serviced at the same time)



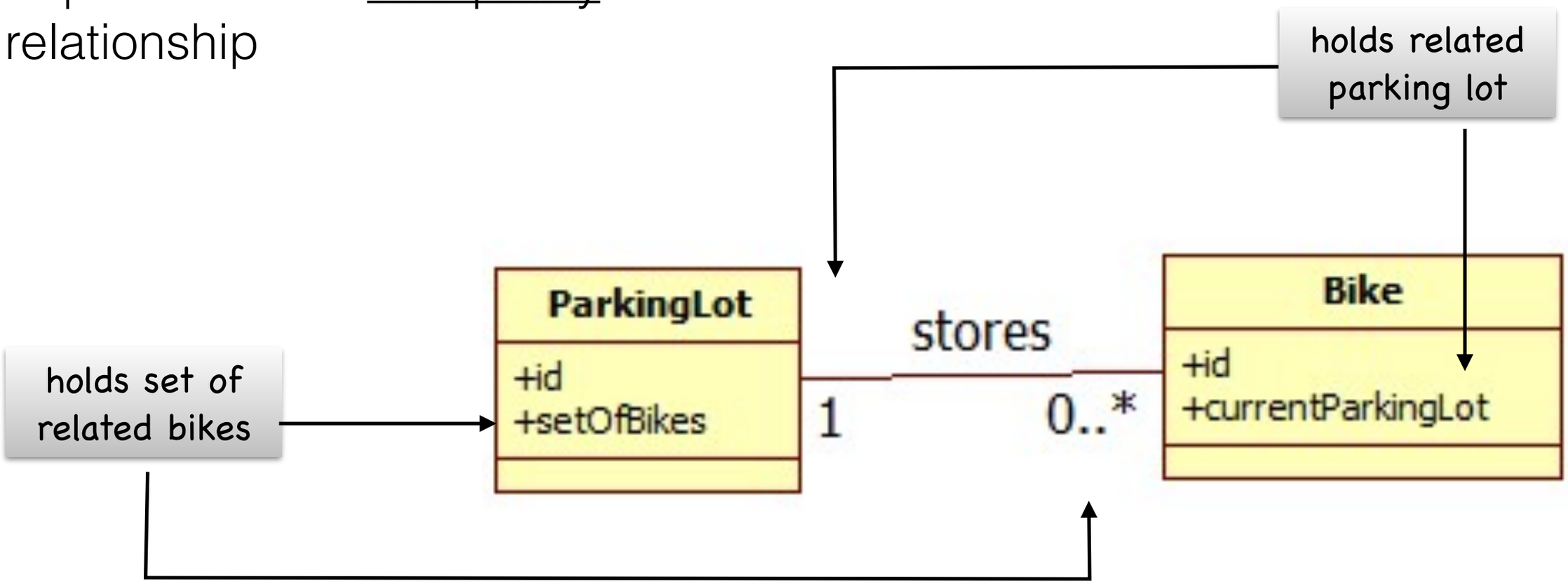
Handling Sets

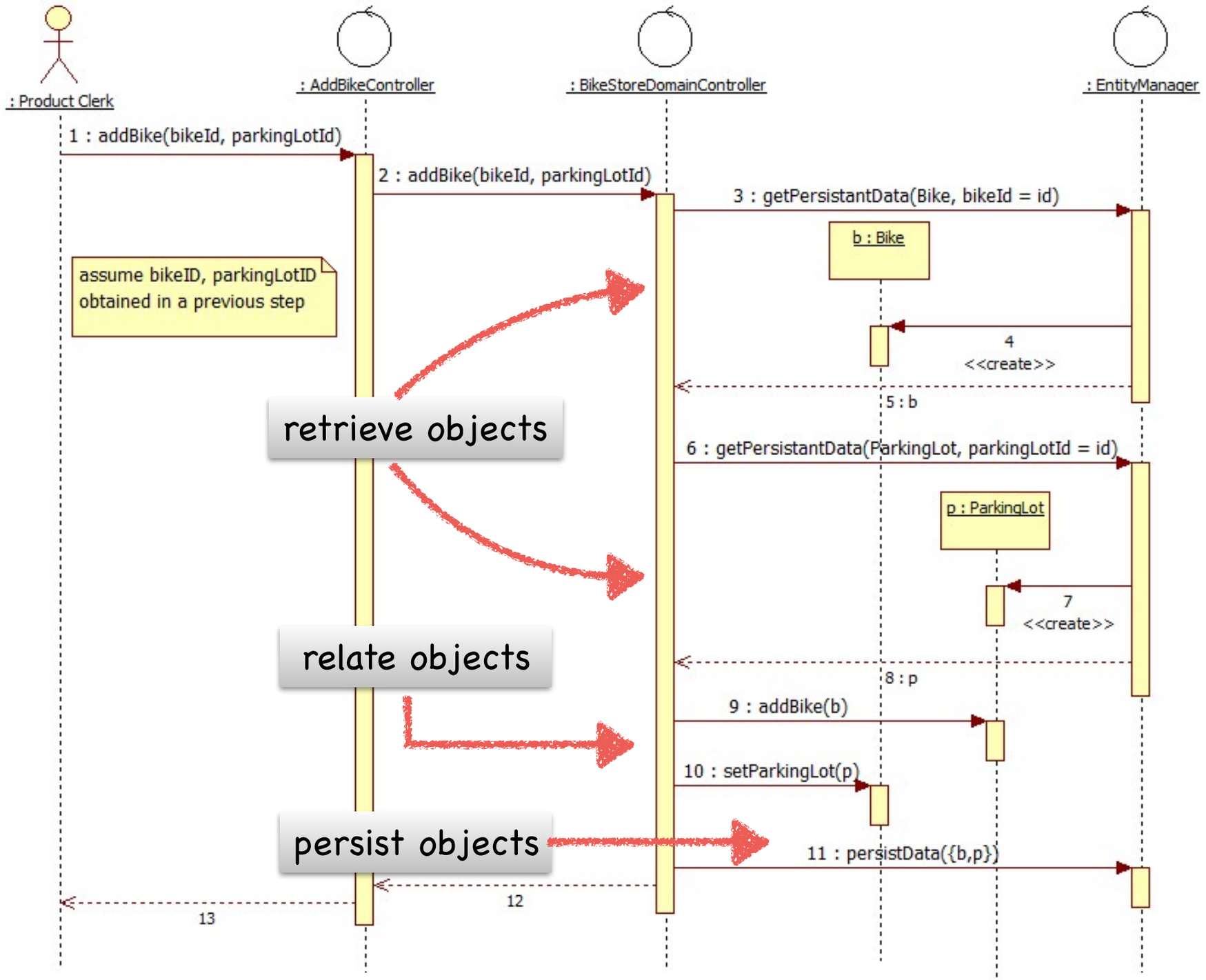
Multiplicity notation is added so that objects and object sets can be handled uniformly

Association Attributes

- attributes store references to objects they are associated with

- set or single object references depends on the multiplicity of relationship





Maintaining Associations

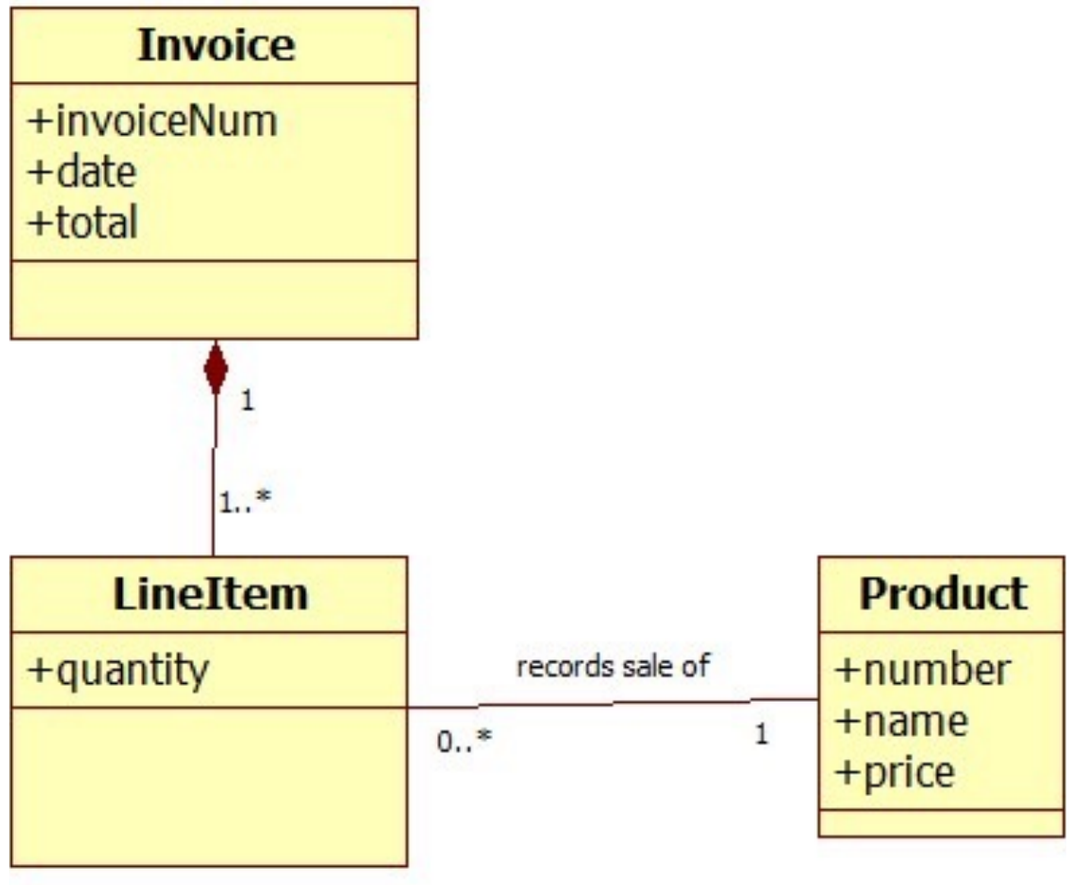
In order to establish/modify associations, related objects are retrieved, association attributes are set, and objects are stored

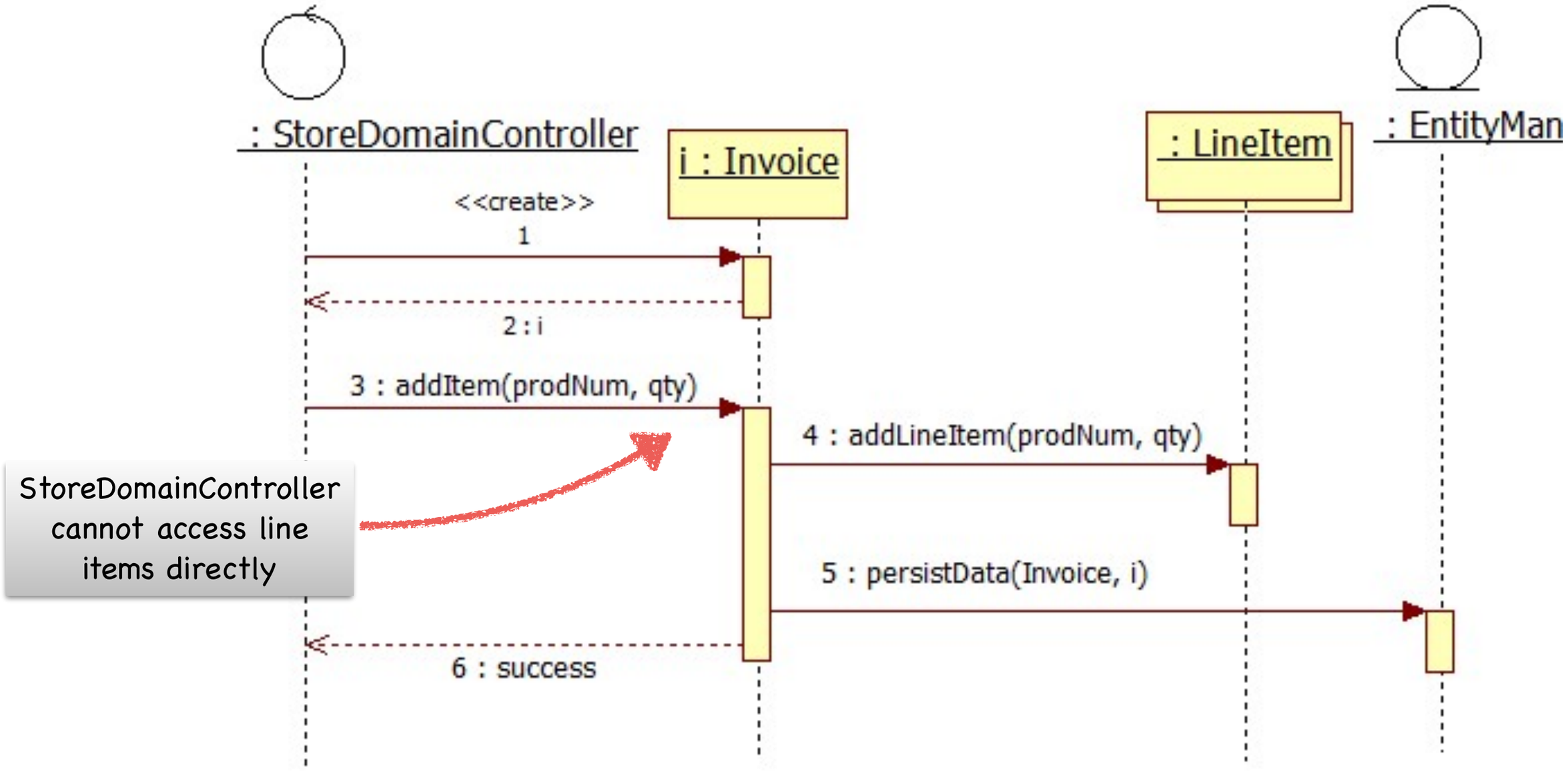
Composition

- when an container object <u>owns</u> other objects

- owned objects can not exist without container

- other objects are <u>only aware of the container object</u>





Composition in Sequence Diagrams

Calls to contained objects must go through the container

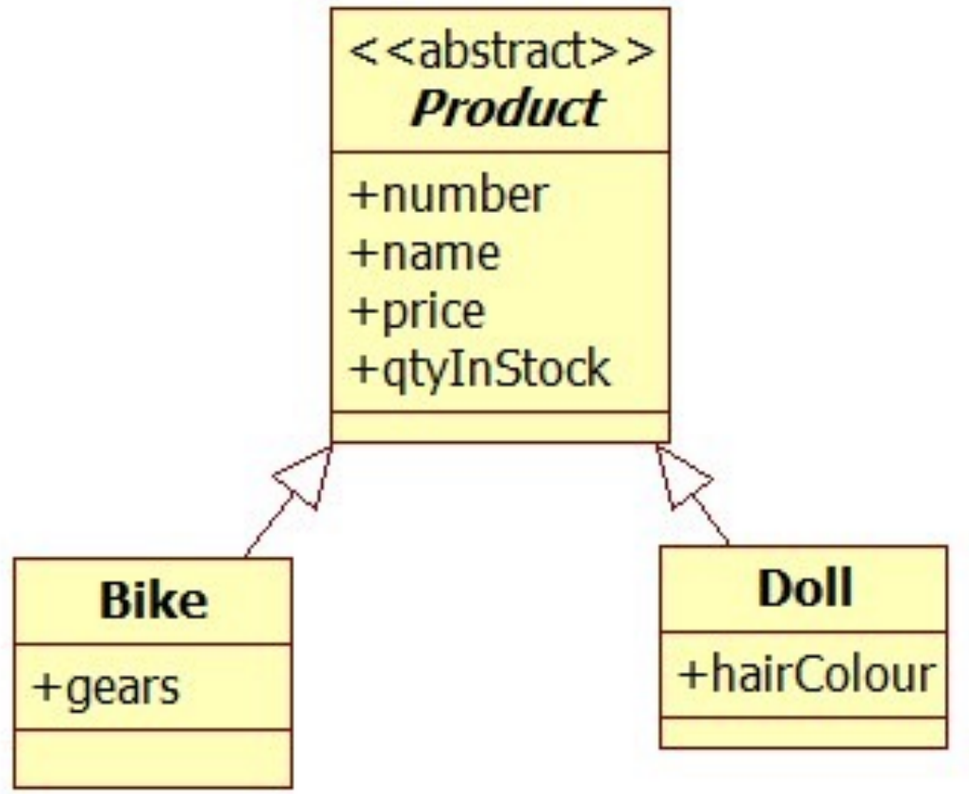
Generalization

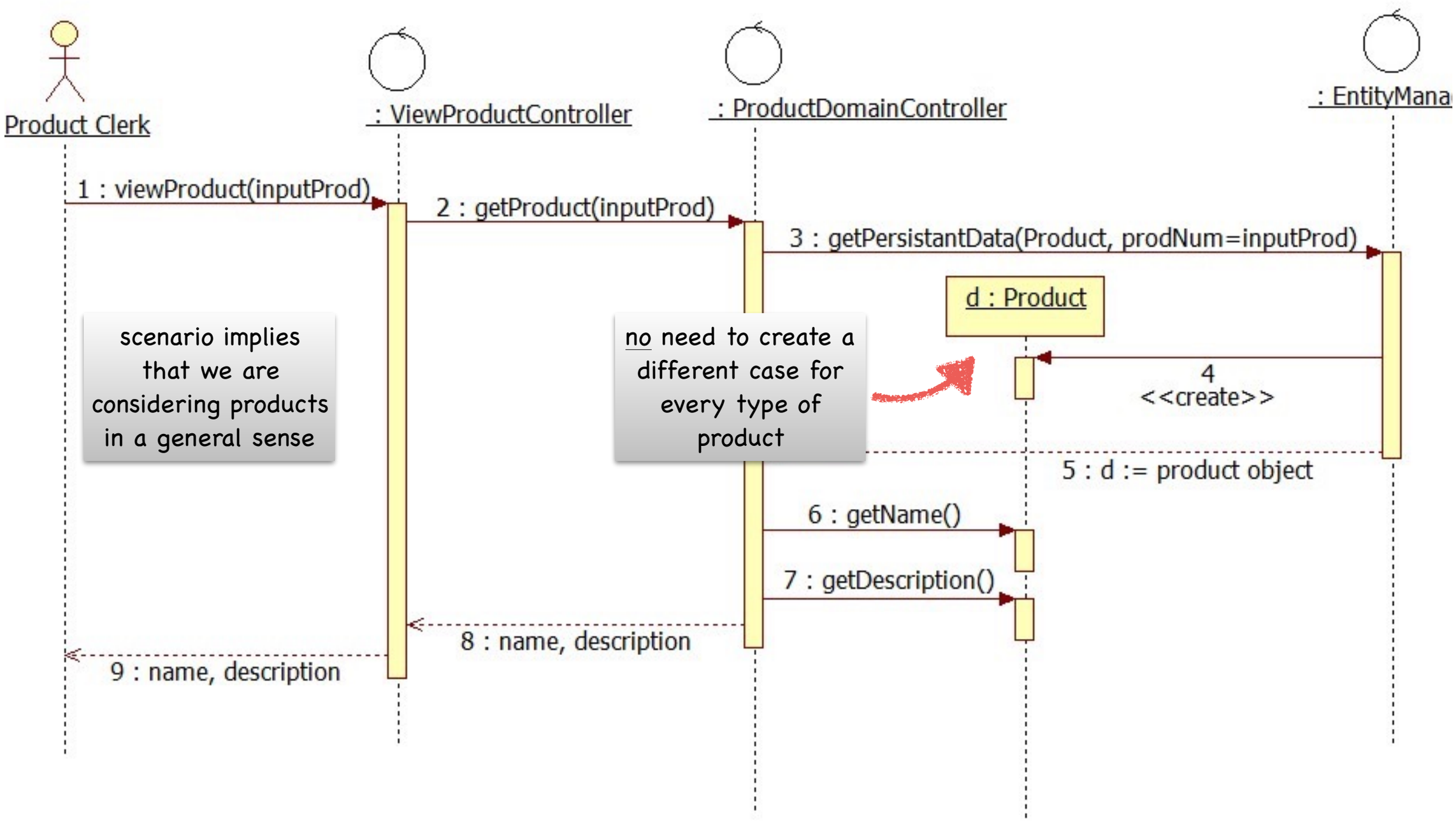
- when multiple classes share certain properties, they may be “grouped” into a conceptual hierarchy

- a superclass represents the general concept

- subclasses represent specializations of the superclass

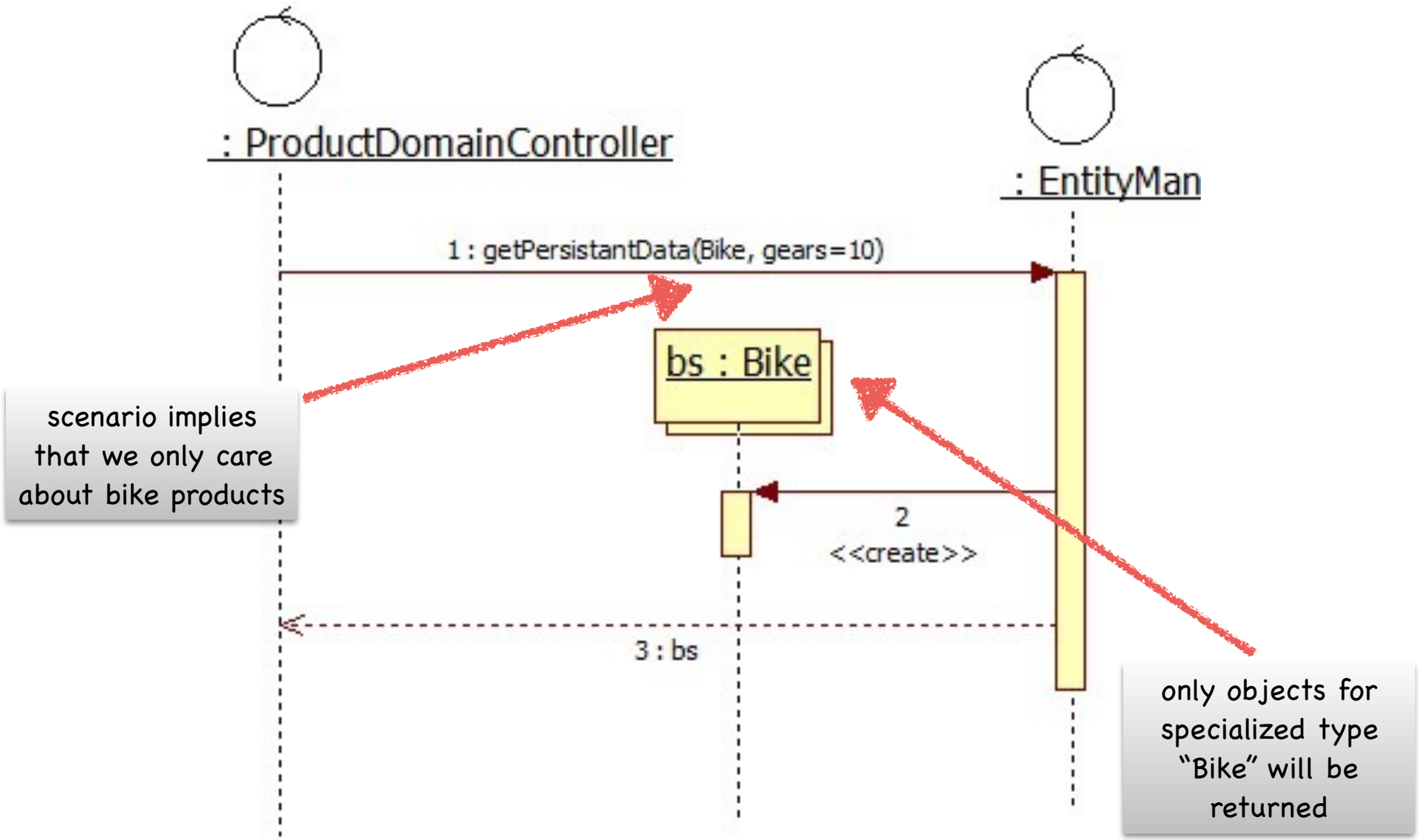
- the superclass is deemed to be abstract, if it can not be created in your conceptual model





Generalization In Sequence Diagrams

Refer to generalized class when you need to refer to their common attributes, and do not care which subclass you are referring to.



Specialization In Sequence Diagrams

Refer to specialized class when you need to refer to their unique attributes