Aggregation

An aggregate is a class that contains objects of class type

* everything in Java is either a primitive or an object
* when a class contains objects rather than primitives, you must pay attention to the fact that objects are stored by reference

Equals method in aggregation

* Need to check that
  + Both objects are equal … do this with the object’s own equals method
  + One or the other object could be null – equality is satisfied if BOTH objects are null
  + The problem:

a.equals(b)

results in a runtime error if a is null

* + Need to check two cases:
    - If a is null, check whether both are null
    - If a is not null, check a.equals(b)

Aggregation vs. composition

* Container object = the aggregation (i.e., the object that contains another object – Portfolio)
* Within Portfolio we have Stock and BankAccount (inside the container)
* Two kinds of relationships between container and the objects inside it:
  + Aggregation – container keeps a reference, but it’s okay if changes happen to the object in the outside world
  + Composition – container “owns” the object inside itself, and it’s not okay if the outside world makes changes
* The importance of this is seen because objects are stored by reference
  + Aggregation: it’s okay to just keep whatever reference we are given, because changes are fine
  + Composition: must make “defensive copies” when objects are moved into or out of the container
* In our portfolio example, we “own” the BankAccount, so that is a composition
* However, we need to keep track of the changing stock price – it’s okay (and in fact essential) that the outside world make changes to this object – this is an aggregation