Aggregation

So far we have seen classes that take primitives as fields (also String).

An aggregate is an object that has objects as fields.

* refer to object variables by reference – can cause difficulties with object privacy
  + if a reference to a private object is returned outside of your class, it’s called a “privacy leak”
* methods such as equals() and toString() – methods must in turn call the equals() and toString() methods of their constituent object.
* Internal object pointer could be null

Remember that if s is an object, and you say in the constructor

this.s = s;

then someone outside your class might continue to hold a reference to your private object this.s

Therefore, any changes that occur outside your object will be visible inside

Remember that objects in your aggregate may have private fields – those private fields are NOT available to methods in the aggregate

e.g. in the Portfolio example, the private fields of BankAccount are not available to the Portfolio methods – you must go through the API.

Equals:

* an aggregate variable can be null
* BUT … the two objects can still be equal if both pointers are null
* Normally you would check for equality between objects a and b by saying: a.equals(b)
* BUT … if a is null, then a.equals(b) results in a runtime error … because you can’t call a nonstatic method on a null object pointer
* First check whether a is null
  + If a is null, the aggregate can still be equal if b is null …
  + If a is not null, then use a.equals(b)