Test 1

* all held in the lab
* programming part and a written part
* programming part: given an API, implement it
* written part: given questions, type your answers
* Material: the course notes (that I am posting online) and the course programming examples are the best thing to study
* You MUST go to your assigned lab
* You MUST have a PRISM lab account

Aggregation and Composition

* Aggregation is a class that contains an object as a field
* Composition is an aggregation where the class “owns” the object that it contains

Aggregation

* All you have to do is keep the reference – no special handling of the reference is needed
* Make sure that the internal reference is consistent with that in the outside world – you may want to keep track of outside changes (e.g., the share price in Stock)

Composition

* You “own” the object, therefore, you want to protect the object from any outside changes out of your control
* E.g. BankAccount – you own the bank account, so its balance should not change unless you agree
* Passing an object in to a composition: Remember that the client may keep a reference to the object; therefore, the client may be able to manipulate this object (sometimes called a “privacy leak”)
* To isolate the object inside the composition from the client, make a copy when it is passed in (sometimes called a “defensive copy”)
* Some classes don’t have copy constructors
	+ String – copy is unneeded because String is immutable. Immutable classes don’t need to be copied because they can never change after construction.
	+ Clone() – if clone() is provided but not a copy constructor, you can use clone().
* Must do the same when you pass an object out – before returning a reference to the internal composition object, make a copy and pass that copy to the client

Copying aggregations and compositions

* When writing a copy constructor for an aggregation or a composition, you must respect the status of each object
* Aggregation: make sure the copy has the same reference as the original
* Composition: Create a new copy of the internal object in the copied container. If you don’t do this, the copy will contain a reference that is not specific to the composition.

In a composition, each instance has the only reference to its contained object.

* Writing the copy constructor for Portfolio – create a copy of BankAccount for the new Portfolio [composition], but NOT a copy of Stock [aggregation].
* When you make a copy, respecting the status of each object inside the container, this is called a “deep copy”.