Questions: 9/10

Program: 27/30 36/40

Your classes were set up correctly.

For the test, in the loop, you can check what type of object you are dealing with (see pages 460 – 461). For example, you could have:

foreach (Account a in acc)

{

Console.WriteLine(a);

a.Credit(200);

Console.WriteLine(a);

a.Debit(50);

Console.WriteLine(a);

if (a is SavingsAccount)

{

decimal d = ((SavingsAccount)a).CalculateInterest();

a.Credit(d);

Console.WriteLine(a);

}

}

-Exercises 12.7

***- 1. Assigning a base-class reference to a base-class variable is straightforward***

***2. Assigning a derived-class reference to a derived-class variable is straightforward***

***3. Assigning a derived-class reference to a base-class variable is safe***

***4. Attempting to assign a base-class reference to a derived-class variable gives an error***

- In the 3trd video note (Section12.5.1), does the narrator state that it is ok to have an abstract property? ***- That’s not how he worded it. The derived class is required to build a definition of that property.*** So no, you cannot have abstract properties

- In the 4th video note (Section 12.5.2), the narrator comments out what method? ***-Earnings***

- In the 7th video note (Section 12.5.5), does the narrator state that it is ok (or not ok) to use the base.base.ToString() syntax? ***-No, it’ll only be the most recently defined version of that method.***

- In the 15th vide note (Section 12.7.4) what are some of the reasons the narrator states to use interfaces over classes and vice versa? ***-He doesn’t state why to use an interface. Seems like more work to code if the class is abstract when implementing an interface.***

Interfaces: more flexible, can implement many interfaces

Classes: can define concrete types, can inherit, can have data

- In the 16th video note (Section 12.8): True or False, in order to have the += operator work, the narrator had to code up another method. ***-False, because the rest of the code works he didn’t have any problems.***