Problem Set 3

Question 1

Write an **aggregate** (composite) class definition for a **House.** A house will have the

following attributes: *address*, *type*, *price* and *owner* (of type **Person**). It should have **accessor** and **mutator** methods defined for each of its attributes as well as a **toString**() method (which

only accesses the attributes **indirectly**) that displays the attributes’ state. It should also have a no-argument constructor which sets up the attributes **indirectly** (via its mutator methods) to give House objects the following initial state:

[“No Address Specified”, “No Type Specified”, 0.0, “No owner specified”].

It should also have a second constructor that takes 6 arguments in total representing the 4 attributes and should **call the** **mutator** methods directly. You should code your class for **true encapsulation** and write a minimalistic driver program that will test all the functionality of your class. Your code should ensure that only **valid** price data is accepted – any negative price value should leave the price unchanged.

Question 2

In the UML class diagram below, Transactable and Taxable are interfaces, BankAccount is an abstract class and SavingsAccount is a concrete class.



1. Write the full code for the *Transactable* and *Taxable* interfaces.
2. The BankAccount Class includes three abstract methods appropriate to the VOPC diagram shown. Write the full code, including the class header, attributes and all appropriate methods for the BankAccount class.
3. Write the full code, including the class header, attributes, a 3-argument constructor and methods for the SavingsAccount class.
4. Write a minimal driver class that creates a number of Savings accounts.

Question 3

In the UML diagram below, IDable is an Interface. Computer, Memory and Laptop are concrete instantiable classes.



1. Write the full code for the *IDable* interface.
2. Write the full code, including the class header, attributes and all appropriate methods for the Computer class.
3. Write the full code, including the class header, attributes, and all appropriate methods for the Laptop class.
4. Write an instance method called makeLaptop that will create and return a single Laptop object. The method should prompt the user for the appropriate values. You can assume that the JOptionPane class is available.
5. Write a minimal driver class that creates an array of Computers that may include Laptops