Labwork 1 – Object Orientation with Design Patterns 2015

This lab is worth 5% or 50 points from the total 500 points for labwork this semester

Part 1: Create a class to represent everyday object and test (10 points)

Create a class to represent a **Pet**. The Pet class should represent households pets and should have the following attributes: Name of the Pet, Age of Animal (in human years). Include a blank constructor for the Pet objects. Test the Pet class using a second class which create sample Pet objects.

Required activities and guideline for marking:

• Class structure and syntax	(2 points)
• Attritbutes (type and name for each)	(3 points)
• Constructor (blank)	(2 points)
• Create sample objects (test program with 'new')	(3 points)

Part 2: Constructors (10 points)

Modify the above **Pet** class to add two overloaded constructors to allow Pet objects to be created by passing the Pet's name (one construct) and both Pet name and age of Pet (another constructor). Also retain the blank constructor (third constructor). Create a new class to test ALL of the new constructors.

• Constructor for passing Pet name	(2 points)
• Constructor for passing Type of animal	(2 points)
• Constructor for both Pet name AND Type of animal	(2 points)
• Retain the blank constructor	(1 point)
• Test all constructors	(3 points)

Part 3: Inheritance (10 points)

Create a new class which **inherits** from the Pet class above. The class is of type Dog and must inherit directly from Pet. Add a specific attribute to the Dog class called list of tricks\commands that the dog can respond to, e.g., "sit", "walkies", "heel". Create a new constructor for Dog which includes the passing of the specific attribute (tricks\commands) but also reuses the super class constructor for Pet. Create a class to test the Dog class, in this class create three Dog objects and add them to an array of type Pet.

• New Dog class created inherits from Pet	(2 points)
• Added new specific attribute (type\name)	(2 points)
• Added new constructor (new attribute)	(2 points)
• Reused super constructor in Dog constructor	(2 points)
• Test new subclass using array of Pet types	(2 points)

Part 4: Polymorphism and dynamic binding (20 points)

Modify the Pet class above to represent the Pet's favourite food using an attribute (e.g. a String like "Pedegree Chum"). Add a method for feeding a Pet object called feedPet(); add this as a **polymorphic** method to the Pet class. Create another (in addition to the Dog class) sub-class of Pet called Cat. Write a setFavouriteFood(String foodName) mutator method to the Pet class which will set the name of the Pet's favourite food and an accessor method called getFavouriteFood() method to return the Pet's favourite food. Create a test class which contains an array of five pet objects, 3 dogs and 2 cats. Set the favourite food of each of your pets using setFavouriteFood() (each dog and cat; make sure each is individual favourite). Write a loop to output the names and favourite food of all of the animals on your list using dynamic binding and the polymorphic method feedPet() (Note!: each animal MUST use dynamic binding to output their favourite food).

• Attribute added to superclass	(2 points)
• Polymorphic method written (in super\base class)	(2 points)
• Accessor method	(2 points)
• Mutator method	(2 points)
• New Cat subclass added	(2 points)
• Polymorphic feedPet() implemented in each subclass	(2 points)
• Test class written with list of pets	(4 points)
• Dynamic binding demonstrated using for loop	(4 points)