Instructor: Dr. Jie Shen
Release date: April 03, 2021
Due date: April 14, 2021

Student Name: Demetrius Johnson

*Special note: see my lab uploads of the .CPP and .EXE files for ease of access and testing any of the programs for any question.

Table of Contents

Question 1 (30 points) Polymorphism

Source code (USED C++ COMPILER on Microsoft Windows 10)

```
// CIS-200-LAB 10 -DemetriusJohnson.cpp : This file contains the 'main' function. Program execution begins and ends there.
//

/*
//Author: Demetrius E Johnson
//Date: 12 April 2021
//Last Modification Date: 04-12-2021
//Purpose: Demonstrate the use of Polymorphism and Dynamic Binding: one of the corner stones of object-oriented programming in C++

*/

/*
Question 1:

Write a program that uses inheritance and polymorphism. Make a Pet class and let it have a kind of food (string).
You have a feed() function and a speak() function. Both functions should be virtual functions.
Child classes will have different things depending on what it is.
Inside the four child classes, you should override the feed( ) and speak( ) functions to generate the specified output.


//notes:

//Dynamic binding (i.e., polymorphism) is one corner stone of the object-oriented programming.
//In C++, we rely on virtual functions and pointers to a base class and its derived classes to implement the polymorphism.

**it gives the capabaility of using one pointer to point to several different but similarly derived objects but with their own unique characteristics
//and take advantage of being able to accessing all of the functions and data members because
//of dynamic binding (polymorphism of the base class, from which other classes are dereived so as to take on another form of the base class object)

*/

#include <iostream>
#include "Pet.cpp"
//#include<assert.h>
using namespace std;


//FUNCTION DECLARATIONS



//FUNCTION DECLARATIONS
```

```
int main()
{

   cout << "---WELCOME: This program demonstrates Polymorphism through Dynamic Binding with virtual functions\n--BY Demetrius Johnson\n\n";

            Pet* nick, * jeff, * chris, * sam;

            nick = new Cat("Meow Mix", "red");
            jeff = new Monkey("Banana", true);
            chris = new Lizard("Flies", 5);
            sam = new Turtle("Lettuce", 0.5);

            nick->feed(); jeff->feed(); chris->feed();
            sam->feed();
            nick->speak(); jeff->speak(); chris->speak();
            sam->speak();


   cout << endl << endl << "The program has finished execution....now exiting...thank you....\n\n";
   system("pause");

   return 0;
}


//FUNCTION DEFINITIONS BELOW THIS LINE

//PET.CPP BELOW:
/*
//Author: Demetrius E Johnson
//Date: 12 April 2021
//Last Modification Date: 04-14-2021
//Purpose: Demonstrate the use of Polymorphism and Dynamic Binding: one of the corner stones of object-oriented programming in C++

*/

#include <string>
#include <iostream>
using namespace std;

class Pet
{
protected:
   string Kind_of_Food = "Eats generic food";
   string Kind_of_Sound = "Makes an animal noise";
public:

   Pet() { //default constructor

      //cout << "Got a pet" << endl;
      //special noteL: I had to comment out the above line because I learned while stepping through the program that
      //all derived class constructors will always first call the base class constructor before executing any of its code lines
```

```
    }

    virtual void feed(void) {

        cout << Kind_of_Food << endl;
    }

    virtual void speak(void) {

        cout << Kind_of_Sound << endl;

    }
};

//note: good practice for when overiding a virtual function is to use keyword 'override'

class Cat : public Pet
{

private:
    string color;
public:
    Cat() {//default constructor --  good practice to define default constructor when any other consturctor is
defined
            //and to make sure no paramter creation of an object is supported

        Kind_of_Sound = "Meow!";
        color = "~unspecified color";
        cout << "Got a " << color << " cat" << endl;
    }

    Cat(string inputFood, string inputColor) { //parametized constructor needed based on calls made from
Lab's Main function

        Kind_of_Sound = "Meow!";
        Kind_of_Food = inputFood;
        color = inputColor;
        cout << "Got a " << color << " cat" << endl;

    }
    void feed(void) override {

        cout << "Eats " << Kind_of_Food << endl;

    }
    void speak(void) override {

        cout << Kind_of_Sound << endl;

    }


};

class Monkey : public Pet
```

```
{
private:
   bool hasTail;
public:
   Monkey() {//default constructor --  good practice to define default constructor when any other
consturctor is defined
            //and to make sure no paramter creation of an object is supported

      Kind_of_Sound = "*Scratches pit*";
      cout << "Got a monkey (unknown if it has a tail)" << endl;
   }

   Monkey(string inputFood, bool tail) { //parametized constructor needed based on calls made from Lab's
Main function

      Kind_of_Sound = "*Scratches pit*";
      Kind_of_Food = inputFood;
      hasTail = tail;
      if (hasTail) {
         cout << "Got a monkey with a tail" << endl;
      }
      else {
         cout << "Got a monkey without a tail" << endl;
      }

   }
   void feed(void) override {

      cout << "Eats " << Kind_of_Food << endl;

   }
   void speak(void) override {

      cout << Kind_of_Sound << endl;

   }

};

class Lizard : public Pet
{
private:
   unsigned short int Length_of_Tongue;
public:
   Lizard() {//default constructor --  good practice to define default constructor when any other consturctor
is defined
             //and to make sure no paramter creation of an object is supported

      Kind_of_Sound = "Grrrrrrr~";
      cout << "Got a lizard (unknown length of tongue)" << endl;
   }

   Lizard(string inputFood, int inputTongueLength) { //parametized constructor needed based on calls made
from Lab's Main function

      Kind_of_Sound = "Grrrrrrr~";
```

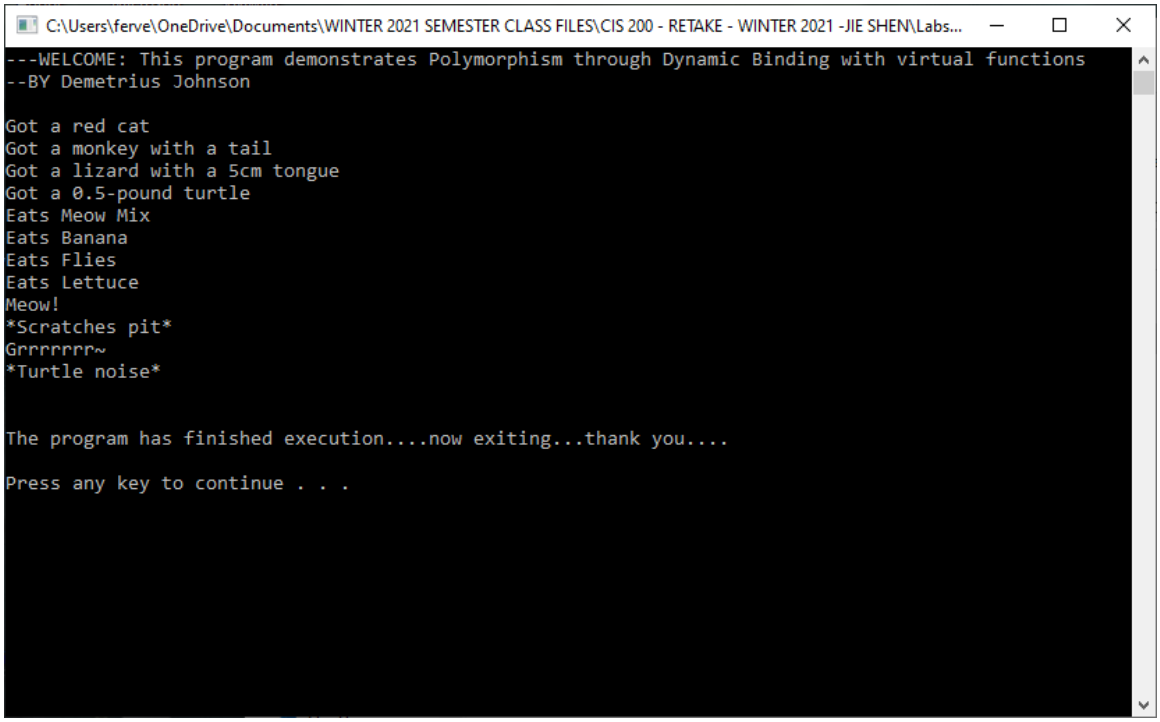
```
        Kind_of_Food = inputFood;
        Length_of_Tongue = inputTongueLength;
        cout << "Got a lizard with a " << Length_of_Tongue << "cm tongue" << endl;

    }
    void feed(void) override {

        cout << "Eats " << Kind_of_Food << endl;

    }
    void speak(void) override {

        cout << Kind_of_Sound << endl;

    }

};

class Turtle : public Pet
{
private:
    float weight;
public:
    Turtle() {//default constructor --  good practice to define default constructor when any other consturctor
is defined
            //and to make sure no paramter creation of an object is supported

        Kind_of_Sound = "*Turtle noise*";
        cout << "Got a turtle (weight unknown)" << endl;
    }

    Turtle(string inputFood, float inputweight) { //parametized constructor needed based on calls made from
Lab's Main function

        Kind_of_Sound = "*Turtle noise*";
        Kind_of_Food = inputFood;
        weight = inputweight;
        cout << "Got a " << weight << "-pound turtle" << endl;

    }
    void feed(void) override {

        cout << "Eats " << Kind_of_Food << endl;

    }
    void speak(void) override {

        cout << Kind_of_Sound << endl;

    }

};
```

Test data and expected results

Test Table:

Test #	Valid / Invalid Data	Description of test	Input Value	Expected Output	Actual Output	Test Pass / Fail
1	valid	Show that my program was able to correctly output the specified output required from the lab using dynamic binding and polymorphism	Main function from lab question 1	Output specified by lab	See screenshot	pass

TEST 1:



***note: you can drop my code in visual studio and you will see the above results are from my source code that uses dynamic binding

Question 2 CRC Card

In this question, design a CRC model for all the classes in Question 1. Show five CRC cards, which correspond to the Pet, Cat, Monkey, Lizard, and Turtle classes. Use "super classes" and "subclasses" to represent the inheritance relationship between these five classes.

CRC CARD – CLASS NAME: Pet	
Superclasses:	
Subclasses: • ***Cat*** • ***Monkey*** • ***Lizard*** • ***Turtle***	
Responsibilities	**Collaborators**
Provide virtual functions that will be customized at different animal subclasses	
Provide basic attributes for an animal: type of food	
Stores the result of feed assigned to an animal	

CRC CARD – CLASS NAME: Cat	
Superclasses: Pet	
Subclasses:	
Responsibilities	**Collaborators**
Specifies color of cat and animal type	
Specifies food type and speak noise	Pet class

CRC CARD – CLASS NAME: Monkey	
Superclasses: Pet	
Subclasses:	
Responsibilities	**Collaborators**
Specifies if monkey has a tail or not and animal type	
Specifies food type and speak noise	Pet class

CRC CARD – CLASS NAME: Lizard	
Superclasses: Pet	
Subclasses:	
Responsibilities	**Collaborators**
Specifies the discrete length of a lizard's tongue and animal type	
Specifies food type and speak noise	Pet class

CRC CARD – CLASS NAME: Turtle	
Superclasses: Pet	
Subclasses:	
Responsibilities	**Collaborators**
Specifies the weight of a turtle and animal type	
Specifies food type and speak noise	Pet class

Question 3 Software Modifiability

Write a half-page summary about Software Modifiability based on your Internet search.

Software Modifiability is a very important subject matter for the computer science field, particularly for software engineering. Based on my Internet search, I gather that such a subject is an absolutely essential knowledge base that a software engineer should specialize in so that they can be effective and calculated at what they do and what their job specifically requires of them. This of course refers to programmers or software developer project leaders who need to understand and be able to apply software modifiability so that a particular project can achieve a certain software quality.

In short summary, software modifiability has to do with the flexibility and adaptability of a program. It is in the best interest of the programmers, project managers, and users or customers that a program can be easily modified because it greatly improves efficiency. When a certain software application needs to be written for some specified purpose, there are often several release versions of that application for the purpose of testing and developing the software, and for modifying versions that are better suited for a specified user or environment group. This process of releasing versions of software directly relates to the time and money costs when developing a software application – and those costs are directly grounded in the modifiability of a program. It is usually most desirable to make programs that are modifiable: easy to adapt for another purpose, easy to understand, and thusly, we have a program that is very portable. The quality of modifiability is measured by such qualities as *cohesion* (how closely related variables are in a class or module) and coupling (how dependent other modules are on each other). Software Modifiability is so important, it is essentially its own field of study, where better programming tactics and project planning are researched in order to cut costs and make programs more useful and efficient.

-source: https://resources.sei.cmu.edu/asset_files/TechnicalReport/2007_005_001_14858.pdf

Question 4 Programming Ethics

Write a half-page summary about Programming Ethics based on your Internet search.

Apparently, programming ethics are a lot more important than I previous thought. Programming ethics do not only deal with good practice programming standards such as readability, modifiability, and other important aspects when developing software so that it can be understood and properly developed by other programmers, but it has to do with some of the more severe consequences when bad programming methods are adopted or when good ones are ignored. Bad programming ethics have even led to death: one example was from a software control issue that resulted in a Boeing 737 Max jet crash that killed 346 people.

Even if a programmer is developing software that is not directly related to some dangerous tasks, many times your software may find its way integrated or intermingling with another software set that does employ a potentially dangerous task – even if it is way down the chain of software relationships or intersections. It is not to say that some how you are directly responsible for some tragic even that happens way down the line that may be loosely related, but it is more pointing to the fact that if everyone employs good software standards, it becomes good programming ethics because it makes it easier for everyone to find and spot errors or potential issues with software sets.

Many of these programming ethical issue show up on project fronts. Sometimes, a project manager may forgo testing in the interest of saving time or money, or they may pressure a colleague to do the same. Often times programmers do not address programming ethics in their learning of programming or computer science, so they may not know the situation or implications of bad programming ethics. Also, some ethics involve intellectual property right violations, which involve editing code or improperly making changes to code. There are many such programming ethics not discussed with programmers, and it can be commonplace that lots of lines are crossed; thus, this is an area that needs to be a little more emphasized in the software development and computer science fields.

-source: https://simpleprogrammer.com/ethical-programming-professional/

Submission of Your Work:

The Word document should contain the following information

- Your name
- Machine type (Unix, Mac, Linux or PC machine ?)
- Compiler type
- Description of your code design and implementation
- Inclusion of your source
- A reasonable number of comment lines in your source code
- Screen shot of your test run (required)