Name: Maitra Patel (00984222)

Class: CIS 200 (Computer Science II)

Assignment: Project 3 – Data Abstractions & UML – Section 01 (Assigned – 3/21/2018)

Date: 4/1/2018

Semester: Winter 2018

1. **Problem Statements**
   1. Problem 1
      1. Create a parent class named Mammal with provided attributes & operations in which you set the type of mammal, weight & age. Along with that, create two child classes inherited from the parent class in which you have specific types of mammals, cattle & cat. These two classes have all functions of a mammal & certain additional unique functions. Using mutators & accessors, test the following classes and create a UML class diagram to represent relationship between them.
2. **Lab Status**
   1. Problem 1: There are no issues with this program.
3. **Source Code:**

Compiler used for coding: Microsoft C/C++ compiler -Visual Studio Community 2017.

* 1. **Source Code for Problem 1:**
     1. **Source Code for Main file:**

#include "Mammal.h"

#include <iostream>

using namespace std;

#include <string>

#include "Cat.h"

#include "Cattle.h"

int main()

{

Mammal x;

cout << "Initial values for x: " << endl;

cout << "Age = " << x.getAge() << " Weight = " << x.getWeight() << endl;

x.setAge(10);

x.setWeight(123);

cout << "Modified values for x: " << endl;

cout << "Age = " << x.getAge() << " Weight = " << x.getWeight() << endl;

Cattle y("dairy");

cout << endl << endl;

cout << "Initial values for y: " << endl;

cout << "Age = " << y.getAge() << " Weight = " << y.getWeight() << " Type = " << y.displayType() << endl;

y.setAge(12);

y.setWeight(127);

y.setType("beef");

cout << "Modified values for y: " << endl;

cout << "Age = " << y.getAge() << " Weight = " << y.getWeight() << " Type = " << y.displayType() << endl << endl;

Cat z("wild");

cout << "Initial values for z: " << endl;

cout << "Age = " << z.getAge() << " Weight = " << z.getWeight() << " Type = " << z.displayType() << endl;

z.setAge(13);

z.setWeight(128);

z.setType("domestic");

cout << "Modified values for z: " << endl;

cout << "Age = " << z.getAge() << " Weight = " << z.getWeight() << " Type = " << z.displayType() << endl << endl;

Mammal w(x);

cout << "Modified values for w: " << endl;

cout << "Age = " << w.getAge() << " Weight = " << w.getWeight() << endl;

w.sound();

cout << endl << endl;

Cat v(z);

cout << "Modified values for v: " << endl;

cout << "Age = " << v.getAge() << " Weight = " << v.getWeight() << " Type = " << v.displayType() << endl;

v.sound();

system("pause");

return 0;

}

* + 1. **Source Code for Mammal.h:**

#ifndef MAMMAL\_H

#define MAMMAL\_H

class Mammal

{

public:

Mammal();

Mammal(const Mammal &x);

void setAge(int);

void setWeight(float);

int getAge();

float getWeight();

void sound();

~Mammal();

private:

int mammalAge;

float mammalWeight;

};

#endif !MAMMAL\_H

* + 1. **Source Code for Cattle.h:**

#ifndef CATTLE\_H

#define CATTLE\_H

#include <string>

#include "Mammal.h"

using namespace std;

class Cattle: public Mammal

{

public:

Cattle();

Cattle(string);

Cattle(Cattle&);

void sound();

void setType(string);

string displayType();

~Cattle();

private:

string type;

};

#endif !CATTLE\_H

* + 1. **Source Code for Cat.h:**

#ifndef CAT\_H

#define CAT\_H

#include <string>

#include "Mammal.h"

using namespace std;

class Cat : public Mammal

{

public:

Cat();

Cat(string);

Cat(Cat&);

void sound();

void setType(string);

string displayType();

~Cat();

private:

string type;

};

#endif !CATTLE\_H

* + 1. **Source Code for Mammal.cpp:**

#include <iostream>

#include "Mammal.h"

using namespace std;

//Constructer

Mammal::Mammal()

{

mammalAge = 0;

mammalWeight = 0;

}

//Copy constructor

Mammal::Mammal(const Mammal &x)

{

mammalAge = x.mammalAge;

mammalWeight = x.mammalWeight;

}

//Destructor

Mammal::~Mammal()

{

mammalAge = 0;

mammalWeight = 0;

}

//Sets mammal's age

void Mammal::setAge(int x)

{

mammalAge = x;

}

//Sets mammal's weight

void Mammal::setWeight(float x)

{

mammalWeight = x;

}

int Mammal::getAge()

{

return mammalAge;

}

float Mammal::getWeight()

{

return mammalWeight;

}

void Mammal::sound()

{

cout << "Generic sound" << endl;

}

* + 1. **Source Code for Cattle.cpp:**

#include <iostream>

#include <string>

#include "Cattle.h"

using namespace std;

Cattle::Cattle()

{

type = "beef";

}

Cattle::Cattle(string x)

{

type = x;

}

Cattle::Cattle(Cattle &x) :Mammal(x)

{

type = x.type;

}

Cattle::~Cattle()

{

type = "beef";

}

void Cattle::sound()

{

cout << "Moo" << endl;

}

void Cattle::setType(string x)

{

type = x;

}

string Cattle::displayType()

{

return type;

}

* + 1. **Source Code for Cat.cpp:**

#include <iostream>

#include <string>

#include "Cat.h"

using namespace std;

Cat::Cat()

{

type = "domestic";

}

Cat::Cat(string x)

{

type = x;

}

Cat::Cat(Cat &x) : Mammal(x)

{

type = x.type;

}

Cat::~Cat()

{

type = "domestic";

}

void Cat::sound()

{

cout << "Mew" << endl;

}

void Cat::setType(string x)

{

type = x;

}

string Cat::displayType()

{

return type;

}

1. **Executable Module Instructions:**
   1. There are no specific module instructions required to run the codes.
2. **Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| Value used for test and what function(s) | Reason for Test | Expected Result | Actual Result |
| x.getAge() & x.getWeight()  Note: x is an object of type mammal | To ensure that the age & weight are 0 before setting them to a value. | x.getAge & x.getWeight should return 0 | x.getAge & x.getWeight returns 0 when printed using “cout <<” |

|  |  |  |  |
| --- | --- | --- | --- |
| Value used for test and what function(s) | Reason for Test | Expected Result | Actual Result |
| x.getAge(10) & x.getWeight(123)  Note: x is an object of type mammal | To ensure that the age & weight are setting & returning properly. | x.getAge() should return 10 and x.getWeight() should return 123. | x.getAge() returns 10 & x.getWeight() returns 123. |

|  |  |  |  |
| --- | --- | --- | --- |
| Value used for test and what function(s) | Reason for Test | Expected Result | Actual Result |
| y.getAge() & y.getWeight()  and  y(“dairy”)  Note: y is an object of type Cattle | To ensure that the age & weight are returning 0 and the type of cattle is set to dairy. | y.getAge() & y.getWeight should return 0 and y.displayType() should return dairy. | y.getAge() & y.getWeight returns 0 and y.displayType() returns dairy. |

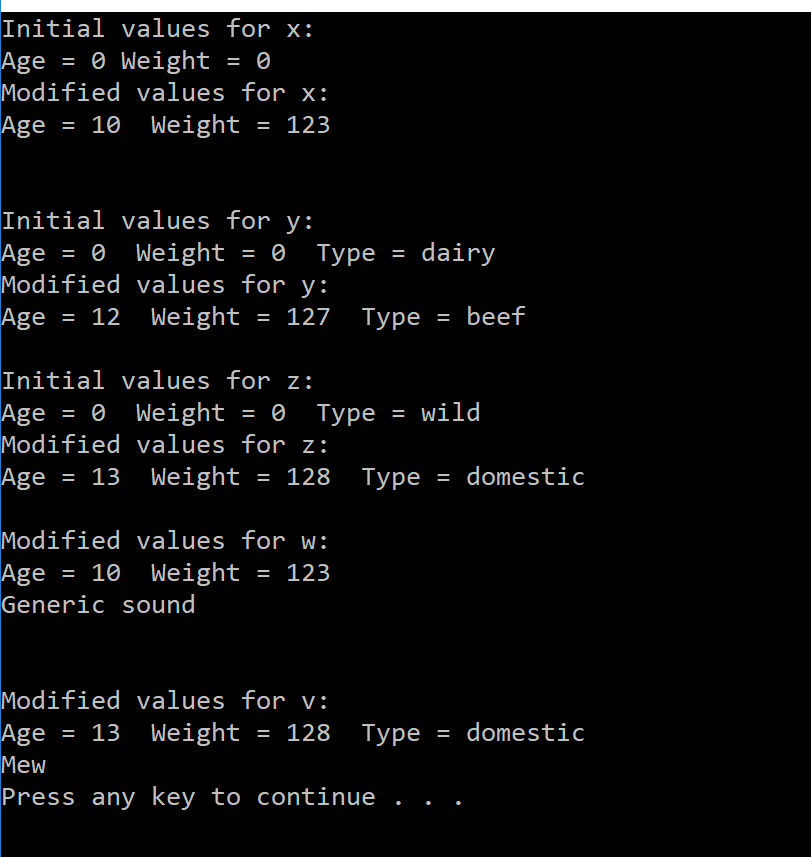
|  |  |  |  |
| --- | --- | --- | --- |
| Value used for test and what function(s) | Reason for Test | Expected Result | Actual Result |
| y.setAge(12) & y.setWeight(127)  and  y.setType(“beef”)  Note: y is an object of type Cattle | To ensure that the age, weight, and type are setting and returning the proper values. | y.getAge() should return 12, y.getWeight() should return 127 and y.displayType() should return beef. | y.getAge() returns 12, y.getWeight() returns 127 & y.displayType() returns beef. |

|  |  |  |  |
| --- | --- | --- | --- |
| Value used for test and what function(s) | Reason for Test | Expected Result | Actual Result |
| z.getAge() & z.getWeight()  and  z.setType(“wild”)  Note: z is an object of type Cat | To ensure that the age, weight, and type are 0 and the type of Cat is set to wild. | y.getAge() should return 0, y.getWeight() should return 0 and y.displayType() should return wild. | y.getAge() returns 0, y.getWeight() returns 0 & y.displayType() returns wild. |

|  |  |  |  |
| --- | --- | --- | --- |
| Value used for test and what function(s) | Reason for Test | Expected Result | Actual Result |
| z.getAge(13) & z.getWeight(128)  and  z.setType(“domestic”)  Note: z is an object of type Cat | To ensure that the age, weight, and type are set properly. | y.getAge() should return 13, y.getWeight() should return 128 and y.displayType() should return domestic. | y.getAge() returns 13, y.getWeight() returns 128 & y.displayType() returns domestic. |

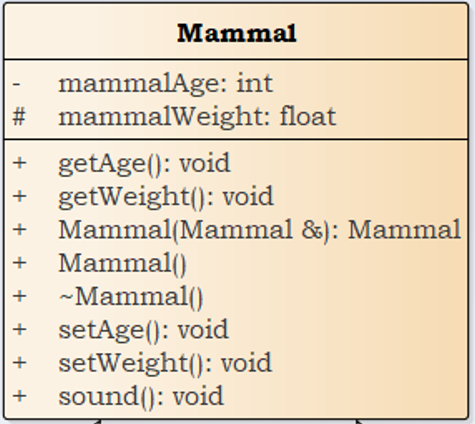
|  |  |  |  |
| --- | --- | --- | --- |
| Value used for test and what function(s) | Reason for Test | Expected Result | Actual Result |
| w(x)  Note: w is an object of type Mammal & x is an object of type Mammal which was already instantiated before. | To ensure that the age, weight, and type are copied properly as per the object that was passed to copy constructor. | w.getAge() should return 10, w.getWeight() should return 123 and w.sound() should return “Generic sound” | w.getAge() returns 10, w.getWeight() returns 123 & w.sound() returns “Generic sound”. |

|  |  |  |  |
| --- | --- | --- | --- |
| Value used for test and what function(s) | Reason for Test | Expected Result | Actual Result |
| v(z)  Note: v is an object of type Cat & x is an object of type Cat which was already instantiated before. | To ensure that the age, weight, and type are copied properly as per the object that was passed to copy constructor. | v.getAge() should return 13, v.getWeight() should return 128 and v.sound() should return “Mew” | v.getAge() returns 13, w.getWeight() returns 128 & w.sound() returns “Generic sound”. |

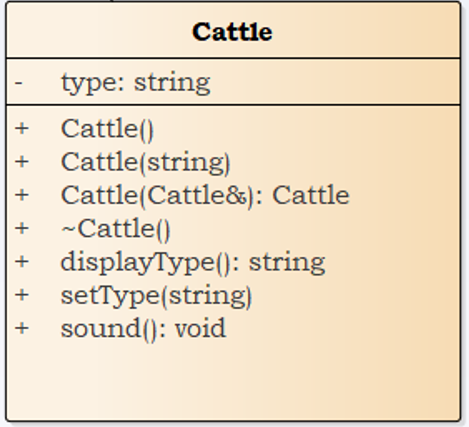
1. **Running Logs/Outputs:**
2. **Running Log of Problem 1 in Screenshot format:**

The image above shows overall results for the test driver code of Data Abstraction program. In which it shows the functionality of constructer, copy constructer, accessors & mutators and the inherited relationship among all three classes.

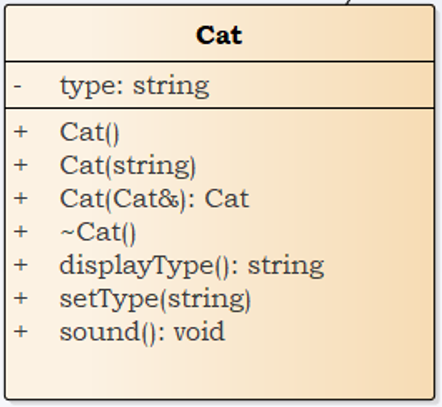
1. **Class Diagrams:**
   1. Mammal Class:

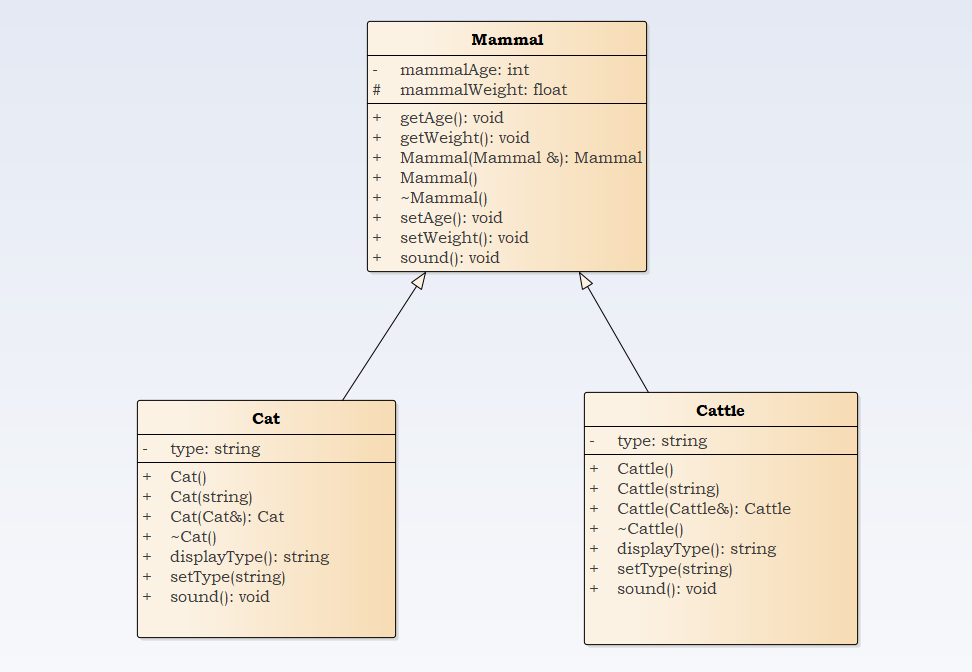


* 1. Cattle Class:



* 1. Cat Class:



1. **Generalization of Class Diagram:**

The image above shows cat and cattle classes being inherited from the parent class, mammal.