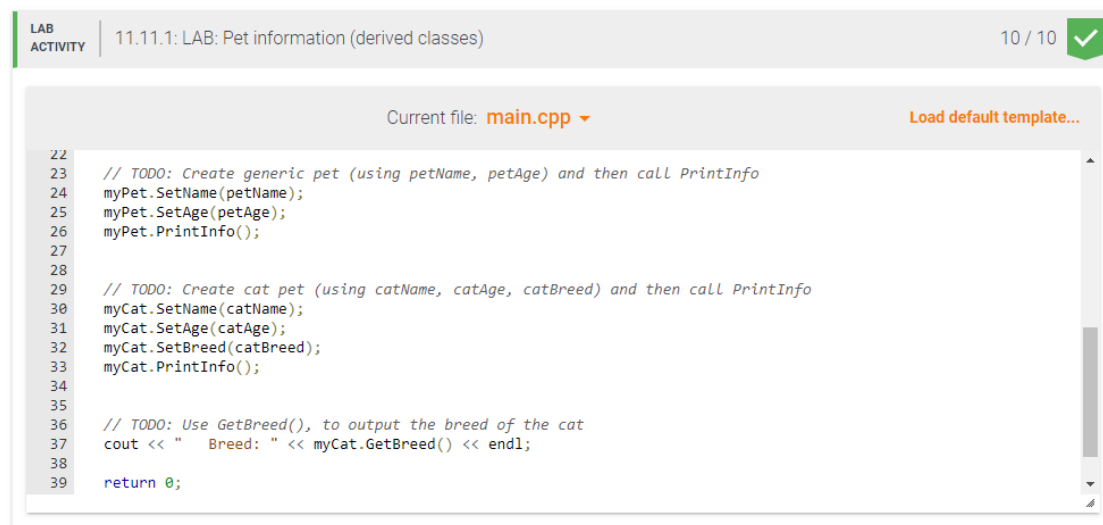


# CSCI 140 PA 10 Submission

Due Date: 5/16/23

Name: Ali Mortada

Exercise 1 – **zyBook 11.11 LAB: Pet information** -- check if completely done in zyBook \_\_X\_\_ ; otherwise, discuss issues below  
Include a screenshot of current status/score

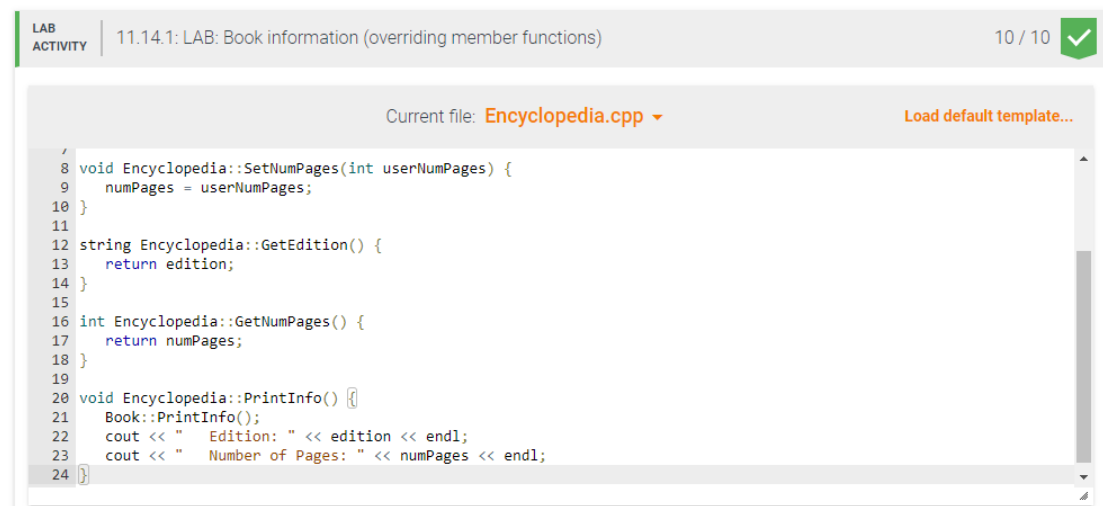


LAB ACTIVITY | 11.11.1: LAB: Pet information (derived classes) | 10 / 10 ✓

Current file: **main.cpp** ▾ | Load default template...

```
22
23 // TODO: Create generic pet (using petName, petAge) and then call PrintInfo
24 myPet.SetName(petName);
25 myPet.SetAge(petAge);
26 myPet.PrintInfo();
27
28
29 // TODO: Create cat pet (using catName, catAge, catBreed) and then call PrintInfo
30 myCat.SetName(catName);
31 myCat.SetAge(catAge);
32 myCat.SetBreed(catBreed);
33 myCat.PrintInfo();
34
35
36 // TODO: Use GetBreed(), to output the breed of the cat
37 cout << " Breed: " << myCat.GetBreed() << endl;
38
39 return 0;
```

Exercise 2 – **zyBook 11.14 LAB: Book information** -- check if completely done in zyBook \_\_X\_\_ ; otherwise, discuss issues below  
Include a screenshot of current status/score

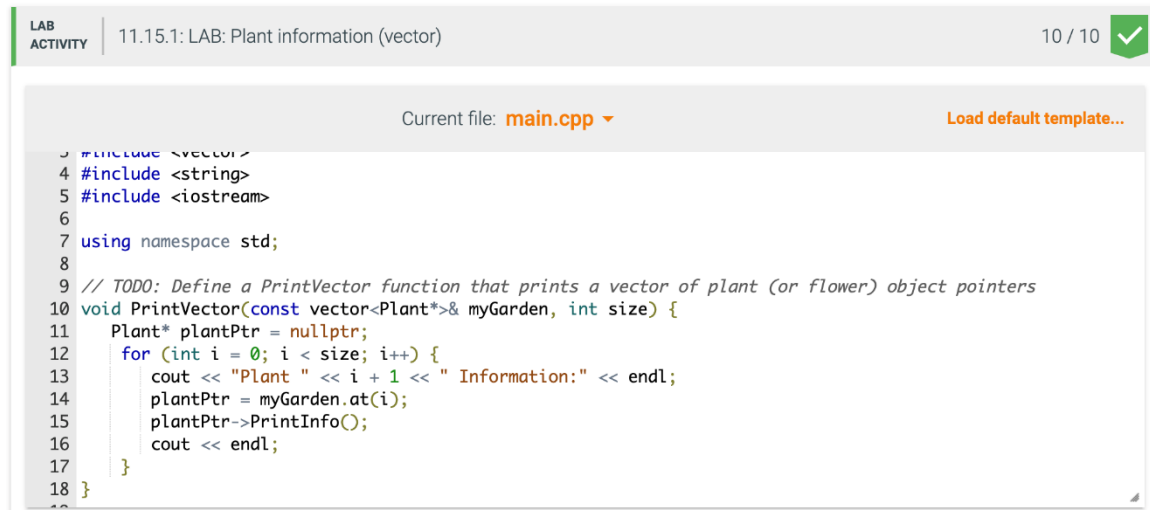


LAB ACTIVITY | 11.14.1: LAB: Book information (overriding member functions) | 10 / 10 ✓

Current file: **Encyclopedia.cpp** ▾ | Load default template...

```
8 void Encyclopedia::SetNumPages(int userNumPages) {
9     numPages = userNumPages;
10 }
11
12 string Encyclopedia::GetEdition() {
13     return edition;
14 }
15
16 int Encyclopedia::GetNumPages() {
17     return numPages;
18 }
19
20 void Encyclopedia::PrintInfo() {
21     Book::PrintInfo();
22     cout << " Edition: " << edition << endl;
23     cout << " Number of Pages: " << numPages << endl;
24 }
```

Exercise 3 – **zyBook 11.15 LAB: Plant information (vector)** -- check if completely done in zyBook \_\_X\_\_ ; otherwise, discuss issues below  
Include a screenshot of current status/score



```
1 // LAB ACTIVITY
2 11.15.1: LAB: Plant information (vector)
3 10 / 10
4
5 Current file: main.cpp
6 Load default template...
7
8 #include <vector>
9 #include <string>
10 #include <iostream>
11
12 using namespace std;
13
14 // TODO: Define a PrintVector function that prints a vector of plant (or flower) object pointers
15 void PrintVector(const vector<Plant*>& myGarden, int size) {
16     Plant* plantPtr = nullptr;
17     for (int i = 0; i < size; i++) {
18         cout << "Plant " << i + 1 << " Information:" << endl;
19         plantPtr = myGarden.at(i);
20         plantPtr->PrintInfo();
21         cout << endl;
22     }
23 }
```

Exercise 4 – **Abstract Base Class** -- check if completely done \_\_X\_\_ ; otherwise, discuss issues below

Source code below:

Class definitions:

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

class BasicShape {
public:
    BasicShape();
    BasicShape(int X, int Y);

    int getX();
    int getY();

    virtual double area() = 0;
    virtual void print() = 0;

protected:
    int x;
    int y;
};

BasicShape::BasicShape() {
```

```

        x = 0;
        y = 0;
    }

    BasicShape::BasicShape(int X, int Y) {
        x = X;
        y = Y;
    }

    int BasicShape::getX() {return x;}

    int BasicShape::getY() {return y;}

    class Circle : public BasicShape {
    public:
        Circle();
        Circle(int X, int Y, double Radius);

        double area() override;
        void print() override;
    private:
        double radius;
    };

    Circle::Circle() {
        x = 0;
        y = 0;
        radius = 1.0;
    }

    Circle::Circle(int X, int Y, double Radius) {
        x = X;
        y = Y;
        radius = Radius;
    }

    double Circle::area() {
        return 3.14 * radius * radius;
    }

    void Circle::print() {
        cout << "Circle: center point (" << x << ", " << y << ") and radius " << radius <<
endl;
    }

```

```
class Rectangle : public BasicShape {
public:
    Rectangle();
    Rectangle(int X, int Y, int Width, int Height);

    double area() override;
    void print() override;
private:
    int width;
    int height;
};

Rectangle::Rectangle() {
    x = 0;
    y = 0;
    width = 1;
    height = 1;
}

Rectangle::Rectangle(int X, int Y, int Width, int Height) {
    x = X;
    y = Y;
    width = Width;
    height = Height;
}

double Rectangle::area() {
    return width * height;
}

void Rectangle::print() {
    cout << "Rectangle: top-left point (" << x << ", " << y << "), " << width << " by "
    << height << endl;
}
```

Driver program:

```
/*
    Program: Abstract Base Class
    Author: Ali Mortada
    Class: CSCI 140
    Date: 5/16/23
    Description: Driver program for BasicShape class.
    I certify that the code below is my own work.
    Exception(s): N/A
*/

#include <iostream>
#include "BasicShape.cpp"
using namespace std;

int main() {
    cout << "Author: Ali Mortada" << endl << endl;

    Circle c1; // center (0, 0) with radius 1.0
    Circle c2(5, 7, 2.5); // center (5, 7) with radius 2.5
    Rectangle r1; // top-left (0, 0) with 1 by 1
    Rectangle r2(5, 7, 10, 15); // top-left (5, 7) with 10 by 15
    BasicShape *pShapeArray[4] = {&c1, &r1, &c2, &r2};
    for (int i = 0; i < 4; i++) {
        pShapeArray[i]->print();
        cout << "\narea: " << pShapeArray[i]->area() << endl << endl;
    }

    return 0;
}
```

Input/output below:

```
Author: Ali Mortada

Circle: center point (0, 0) and radius 1
area: 3.14

Rectangle: top-left point (0, 0), 1 by 1
area: 1

Circle: center point (5, 7) and radius 2.5
area: 19.625

Rectangle: top-left point (5, 7), 10 by 15
area: 150
```

Answer for Question 1

Function overloading is when a function is defined multiple different times so that it can take parameters of differing amounts and types. For example, a function can be defined to take in no parameters, then be overloaded to take in one parameter, two parameters, etc. Function overriding, however, is when a derived class redefines a function from a base class so it could work with the derived class's members. The function's parameters are not changed, only the body of the function is.

Answer for Question 2

To implement an Alarm class using an existing Time class, I would use composition (has-a) relationship. This is because an alarm is not a type of time; it only uses time. Thus, it would not make sense to use inheritance. However, an alarm utilizes time, meaning that there is a has-a relationship between the classes. This would mean that composition would be optimal for this situation.

Extra Credit – provide if applicable.