COMSC-200
Lab 3

Ryan Jacoby

13 September 2020

1 Base and Derived Classes

a. Base class: Employee
   Derived class: Manager

b. Base class: Polygon
   Derived class: Triangle

c. Base class: Student
   Derived class: GraduateStudent

d. Base class: Person
   Derived class: Student

e. Base class: Employee
   Derived class: Professor

f. Base class: BankAccount
   Derived class: CheckingAccount

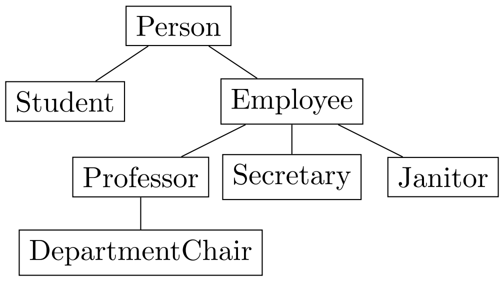
g. Base class: Vehicle
   Derived class: Car

h. Base class: Vehicle
   Derived class: Minivan

i. Base class: Car
   Derived class: Minivan

j. Base class: Vehicle
   Derived class: Truck

2 Inheritance Relationships



3 Program

What does the following program print?

```cpp
#include<iostream>

using namespace std;

class B {
public:
    void print(int n) const;
};

void B::print(int n) const {
    cout << n << endl;
}

class D : public B {
public:
    void print(int n) const;
};

void D::print(int n) const {
    if (n <= 1) { B::print(n); }
    else if (n % 2 == 0) { print(n / 2); }
    else { print(3 * n + 1); }
}

int main() {
    D d;
    d.print(3);
    return 0;
}
```

This program outputs 1.

4 Person, Student, and Instructor Classes

•• **E10.7** Implement a base class `Person`. Derive classes `Student` and `Instructor` from `Person`. A person has a name and a birthday. A student has a major, and an instructor has a salary. Write the class definitions, the constructors, and the member functions `display` for all classes.

```cpp
// Ryan Jacoby

#include<iostream>

#include"Person.h"
#include"Student.h"
#include"Instructor.h"

using namespace std;

int main() {
    Person p = Person("Ryan");
    Student s = Student("Fred", "", "CS");
    Instructor i = Instructor("Ethan", "", 5);

    p.display();
    cout << '\n';
    s.display();
    cout << '\n';
    i.display();

    return 0;
}
```

Listing 1: main.cpp

```cpp
// Ryan Jacoby

#ifndef Person_h
#define Person_h

#include<string>

using namespace std;

class Person {
private:
    string name;
    string birthday;
public:
    Person();
    Person(string name);
    Person(string name, string birthday);

    string getName() { return this->name; }
    string getBirthday() { return this->birthday; }

    void setName(string name) { this->name = name; }
    void setBirthday(string birthday) { this->birthday = birthday; }

    void display();
};

#endif
```

Listing 2: Person.h

```cpp
// Ryan Jacoby

#include<iostream>
#include<string>

#include"Person.h"

using namespace std;

Person::Person() {
    this->name = "";
    this->birthday = "";
}

Person::Person(string name) {
    this->name = name;
    this->birthday = "";
}

Person::Person(string name, string birthday) {
    this->name = name;
    this->birthday = birthday;
}

void Person::display() {
    cout << "Name: " << this->name << '\n'
         << "Birthday: " << this->birthday << '\n';
}
```

Listing 3: PersonImp.cpp

```cpp
// Ryan Jacoby

#ifndef Student_h
#define Student_h

#include<string>

using namespace std;

class Student : public Person{
private:
    string major;
public:
    Student() :Person() {}
    Student(string name) :Person(name) {}
    Student(string name, string birthday) :Person(name, birthday) {}
    Student(string name, string birthday, string major);

    string getMajor() { return this->major; }

    void setMajor(string major) { this->major = major; }

    void display();
};

#endif
```

Listing 4: Student.h

```
// Ryan Jacoby

#include<iostream>
#include<string>

#include"Person.h"
#include"Student.h"

using namespace std;

Student::Student(string name, string birthday, string major) :Person(name,
    birthday) {
    this->major = major;
}

void Student::display() {
    Person::display();
    cout << "Major: " << this->major << '\n';
}
```

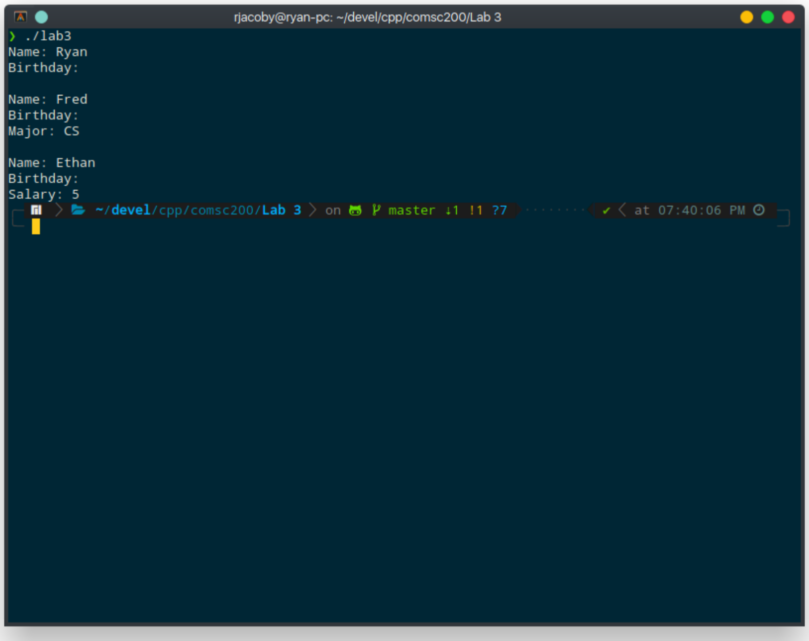
Listing 5: StudentImp.cpp

```
// Ryan Jacoby

#ifndef Instructor_h
#define Instructor_h

#include<string>

using namespace std;

class Instructor : public Person{
private:
    int salary;
public:
    Instructor() :Person() {}
    Instructor(string name) :Person(name) {}
    Instructor(string name, string birthday) :Person(name, birthday) {}
    Instructor(string name, string birthday, int salary);

    int getSalary() { return this->salary; }

    void setMajor(int salary) { this->salary = salary; }

    void display();
};

#endif
```

Listing 6: Instructor.h

```cpp
// Ryan Jacoby

#include<iostream>
#include<string>

#include"Person.h"
#include"Instructor.h"

using namespace std;

Instructor::Instructor(string name, string birthday, int salary) :Person(name, birthday) {
    this->salary = salary;
}

void Instructor::display() {
    Person::display();
    cout << "Salary: " << this->salary << '\n';
}
```

Listing 7: InstructorImp.cpp



```
rjacoby@ryan-pc: ~/devel/cpp/comsc200/Lab 3
❯ ./lab3
Name: Ryan
Birthday:

Name: Fred
Birthday:
Major: CS

Name: Ethan
Birthday:
Salary: 5
 ~/devel/cpp/comsc200/Lab 3 ❯ on  master ⇣1 !1 ?7 ········ ✓ ❮ at 07:40:06 PM
```

5 Clock

•• **P10.1** Implement a class `Clock` whose `get_hours` and `get_minutes` member functions return the current time at your location. To get the current time, use the following code, which requires that you include the `<ctime>` header:

```
time_t current_time = time(0);
tm* local_time = localtime(&current_time);
int hours = local_time->tm_hour;
int minutes = local_time->tm_min;
```

Also provide a `get_time` member function that returns a string with the hours and minutes by calling the `get_hours` and `get_minutes` functions. Provide a derived class `WorldClock` whose constructor accepts a time offset. For example, if you live in California, a `new WorldClock(3)` should show the time in New York, three time zones ahead. Which member functions did you override? (You should not override `get_time`.)

The get_hours() function is overrided. I made get_minutes() virtual so other classes could overwride it for time zones with offset minutes.

```
// Ryan Jacoby

#include<iostream>

#include"Clock.h"
#include"WorldClock.h"

using namespace std;

int main() {
    Clock c = Clock();
    WorldClock wc = WorldClock(3);

    cout << "San Francisco: " << c.get_time() << '\n';
    cout << "New York: " << wc.get_time() << '\n';

    return 0;
}
```

Listing 8: main.cpp

```
// Ryan Jacoby

#ifndef Clock_h
#define Clock_h

#include<string>

using namespace std;

class Clock {
protected:
    time_t current_time;
    tm* local_time;
public:
    Clock();
```

```cpp

    virtual int get_hours();
    virtual int get_minutes();
    string get_time();
};

#endif
```

Listing 9: Clock.h

```cpp
// Ryan Jacoby

#include<ctime>
#include<string>

#include"Clock.h"

Clock::Clock() {
    this->current_time = time(0);
    this->local_time = localtime(&current_time);
}

int Clock::get_hours() {
    return this->local_time->tm_hour;
}

int Clock::get_minutes() {
    return this->local_time->tm_min;
}

string Clock::get_time() {
    return to_string(get_hours()) + ":" + to_string(get_minutes());
}
```

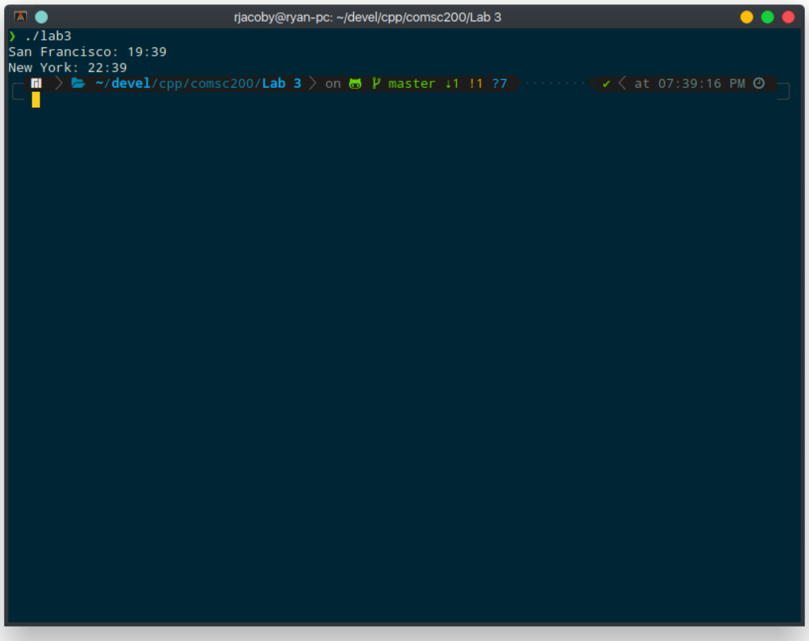
Listing 10: ClockImp.cpp

```cpp
// Ryan Jacoby

#ifndef WorldClock_h
#define WorldClock_h

using namespace std;

class WorldClock : public Clock {
private:
    int offset;
public:
    WorldClock();
    WorldClock(int offset);

    int get_hours();
};

#endif
```

Listing 11: WorldClock.h

```cpp
// Ryan Jacoby

#include<string>
```

```cpp

#include"Clock.h"
#include"WorldClock.h"

WorldClock::WorldClock() :Clock() {
    this->offset = 0;
}

WorldClock::WorldClock(int offset) :Clock() {
    this->offset = offset;
}

int WorldClock::get_hours() {
    return this->local_time->tm_hour + this->offset;
}
```
Listing 12: WorldClockImp.cpp



rjacoby@ryan-pc: ~/devel/cpp/comsc200/Lab 3

```
❯ ./lab3
San Francisco: 19:39
New York: 22:39
 > ~/devel/cpp/comsc200/Lab 3 > on  master ↓1 !1 ?7 ········ ✔ < at 07:39:16 PM
```