Example Project A (C++)

This project will demonstrate:

1. Creating an object to encapsulate and abstract data. (Rectangle.h, Rectangle.cpp)
2. Defining default constructor and overloading constructors.
3. Overloading functions.
4. Calling multiple constructors at once. (Java equivalent of super() and this().)
5. Inheritance by subclassing the Rectangle class in Room. (Room.h, Room.cpp)
6. Association of objects, e.g. House 'uses' Room by calling getArea() and getPerimeter().
7. Composition of objects, e.g. House contains several Room objects.
8. Using 'cin' and 'cout' to output to and receive input from the user.
9. Creating a pseudo-Interface IShape and abstract class Rectangle when IShape is declared and not implemented.
10. Dynamic memory allocation and deallocation using C++ destructor with delete keyword.
11. Operator overloading by overloading the assignment operator '='.
12. Implementing the Java equivalent of toString() by using the stream operator '<<'.
13. Use of the friend keyword for implementation of the stream operator.

Activities

1. Modify the `string Rectangle::getName() const` method to display "no name" when the name has not been set.

SOLUTION `return (name.empty()) ? "no name" : name;`
CAVEAT `return (name == NULL) ? "no name" : name;`

In Java, when a member field is not instantiated it is null. However in C++ all member objects are automatically instantiated by the default constructor if it is not a primitive type. The 'name' member is never null. Note: In C++ null refers to a binary expression, i.e. a comparison between null and a pointer. In this case we are illegally comparing between null and a non-pointer. Instead, we call the empty() method in the string class.

2. Create a few Rectangle objects experimenting with both the name constructor and default constructor.

You will notice that when we use the string constructor and call `getWidth()` or `getLength()` it returns an unexpected value. i.e., `6.95322e-310`

Define two constants, for width and length respectively, that all new instances of Rectangle classes will have when a width or length are not explicitly defined.

SOLUTION `const double DEFAULT_WIDTH = 5.0;`

```
const double DEFAULT_LENGTH = 5.0;
```

These should be declared in the Rectangle.h file. We need to also implement a change to the string constructor and default constructor.

```
Rectangle(double w=DEFAULT_WIDTH, double l=DEFAULT_LENGTH);
```

In the implementation of `Rectangle(const string &name)` we need to add:

```
width = DEFAULT_WIDTH;
length = DEFAULT_LENGTH;
```

3. Sometimes we may want to create copies of objects. In Java, you typically define a copy constructor. In C++ we can define a copy constructor and use a technique called operator overloading. We can tell the compiler that when a Rectangle object is set equal to another Rectangle object it should copy it.

Example Java

```
Rectangle rect = new Rectangle(10.0, 11.0);
Rectangle copyRect = new Rectangle(rect);
```

Example C++

```
Rectangle rect(10.0, 11.0);
Rectangle copyRect(rect);
```

We can alternatively say:

```
Rectangle copyRect;
Rectangle copyRect = rect;
```

To do this we need to implement the following instance methods:

```
Rectangle::Rectangle(const Rectangle& other)
Rectangle& Rectangle::operator=(const Rectangle& rhs)
```

SOLUTION

```
Rectangle::Rectangle(const Rectangle& other) {
    width = other.width;
    length = other.length;
    name = string(other.name);
}
Rectangle& Rectangle::operator=(const Rectangle& rhs) {
    if (this == &rhs) return *this;
    width = other.width;
    length = other.length;
    name = string(other.name);
    return *this;
}
```

Note: `if (this == &rhs) return *this;` is designed to handle the special case of self assignment. Although this particular example may not require it, consider it best practice.

4. We would like to create a House class that utilizes the Rectangle class to define the size of each room. However we would also like to define some windows to go along with each room. Lets create a Rectangle subclass that is closer to our domain called Room.

5. Many OOP languages offer a construct called 'Interface' and 'Abstract class.' C++ implements this idea with purely virtual functions. Functions are purely virtual when the `virtual` keyword is used and the function is set to zero. Create an interface called `IShape` that will require subclasses implementing the interface to provide methods for getting the area and perimeter of a shape.

   Note: In this project the prefix `I` will be used to denote an interface. This is an arbitrary naming convention. You can pick which how you prefer to name interfaces and abstract classes. Prefix I, for example, is used in Microsoft's C#. Some C++ libraries use the suffix `Interface` or `Abstract`. E.g. `ShapeInterface`

Example C++ `virtual void foo() = 0;`

This indicates to the compiler that any subclass must define an implementation for `foo()`. It also turns the class that contains the definition and any subclass that does not define an implementation for `foo()` into an abstract class. An abstract class cannot be instantiated.

SOLUTION
```
virtual double getArea() const = 0;
virtual double getPerimeter() const = 0;
```
Declare Rectangle a subclass of `IShape`. This will then make Rectangle abstract and any attempt to instantiate a Rectangle class will fail. To solve this provide implementations for `IShape`.
```
class Rectangle : public IShape
```
Note: Don't forget to include the `IShape` header.
```
public:
    double getArea() const { return length*width; }
    double getPerimeter() const { return (length*2.0)+
     (width*2.0); }
```

6. Use `dynamic_cast<type>` to cast a shape as an IShape to access getArea() and getPerimeter() methods.

7. Create a House class. The House should then print a result to the screen using the 'cout' stream operator.

- unsigned int count
- Room ** rooms
+ bool addRoom(Room *)
+ unsigned int getRoomCount() const
+ static unsigned int getMaxRooms()
+ unsigned int getWindows() const
+ double getTotalArea() const

8. Test the House class by adding rooms. It should output the total number of windows in the house, and area as well as the name of each room. It should then deallocate the room objects from memory and delete the allocated space for the array.

9. Provide a C++ version of Java's toString() method for the Rectangle class. Call the C++ toString() method on the Room objects created in Activity 8 by using a `static_cast<type>`.