CSCI 200 - Fall 2023
Foundational Programming Concepts & Design

Lab 5A - Escape Room

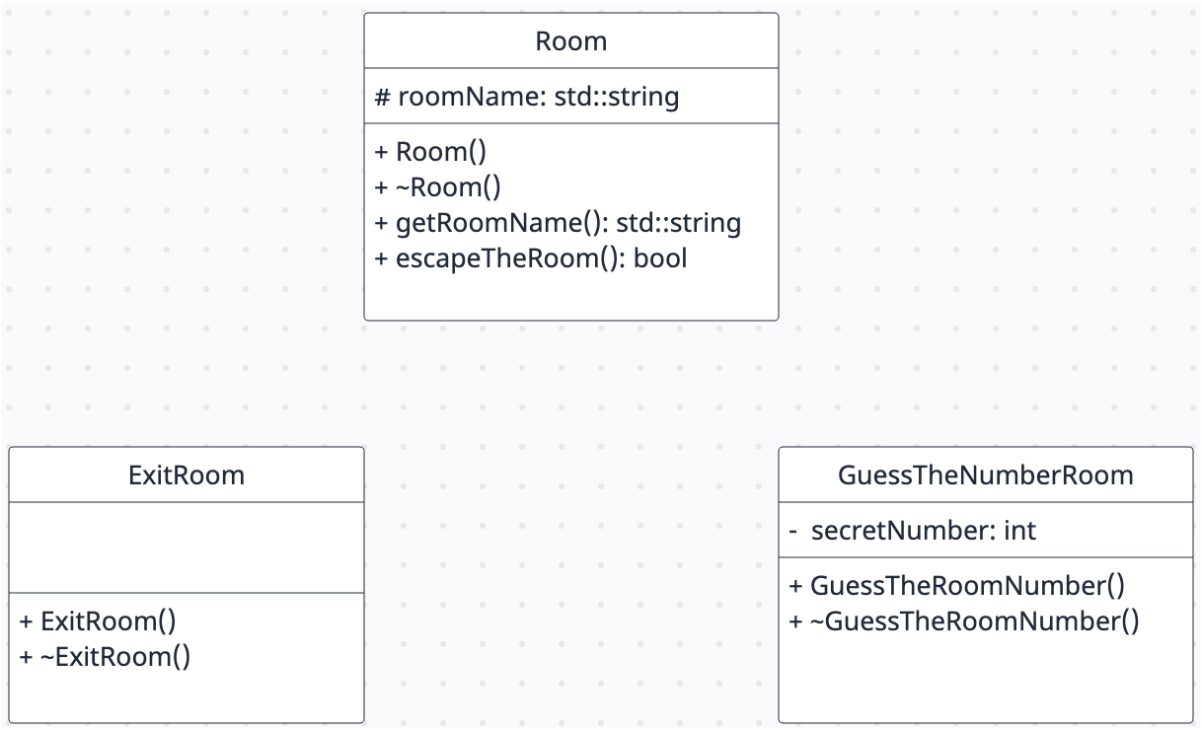
This lab is due by Tuesday, November 14, 2023 11:59 PM.
As with all labs you may, and are encouraged, to pair program a solution to this lab. If you choose to pair program a solution, be sure that you individually understand how to generate the correct solution.

Jump To: **Rubric Submission**

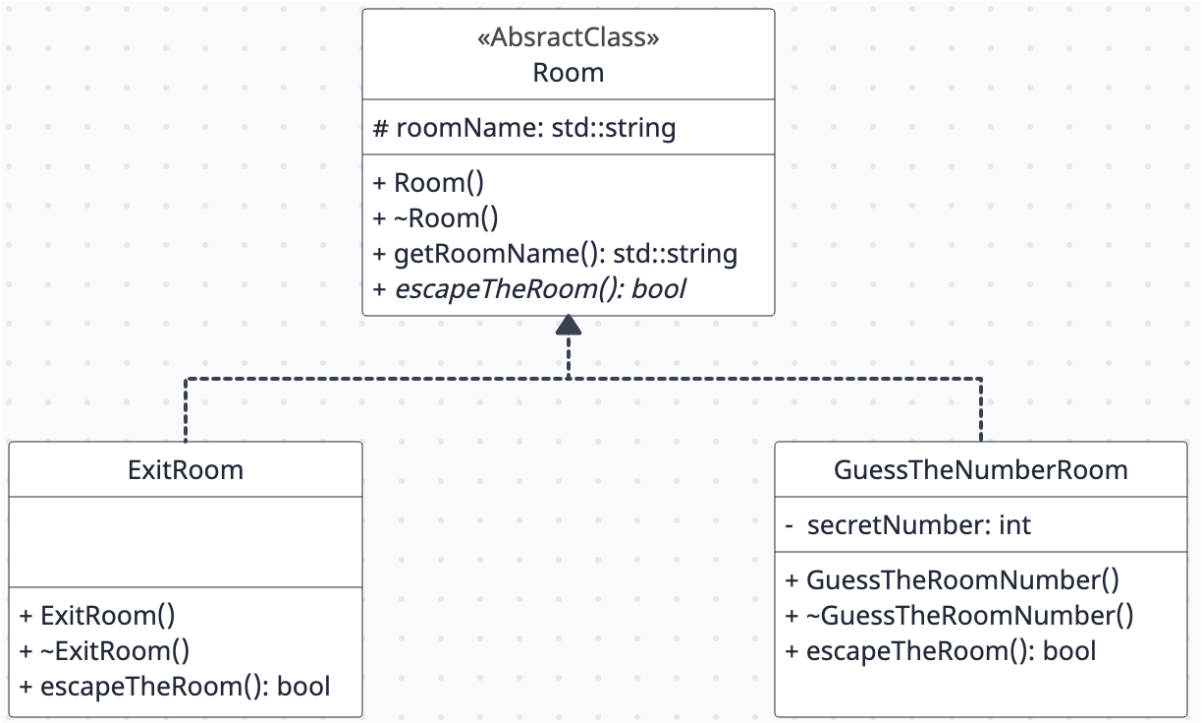
You awaken and find yourself in a room with no doors. You're trapped! You need to get out! Download the **Escape Room starter pack**. The program will build out of the box, but upon running (go ahead and try it) you're stuck in an infinite loop! Your task - create a way to get out.

The initial class UML is shown below before any inheritance hierarchy is put in place.

RMP-H



Your task is to create the abstract `Room` class with the two concrete implementations, as shown in the finished UML diagram below.



The Room Parent

We will want to create different types of rooms. Modify the `ARoom` class to have a `virtual escapeTheRoom()` method. (For now, leave the default implementation in place.) The

`escapeTheRoom()` method will return `true` when a room has been escaped from and `false` if the room was not escaped. We need to create ways to return `true`! But not here.

The Exit Room

The start of the Exit Room is already in place, but it's incomplete. First, have `ExitRoom` publicly extend `ARoom` and in the constructor set the room name to an appropriate value. You're now forced to override the `escapeTheRoom()` method. Have the `ExitRoom::escapeTheRoom()` implementation print a message similar to `"You found the exit!"` and return `true` to signal you made it out.

Put The Exit Room In Place

Inside of `main.cpp`, there is a function called `go_to_next_room()`. It generates a random number and if it's lucky number seven, let's change the method to return an `ExitRoom` object.

Build and run the program to wait for your 10% chance to escape.

...

...

...

Wait, we're still stuck. Why? What's happening? It looks like we're making the Exit Room at some point...

```
Welcome to the Vacant Room
There's no escape
ARoom() called
~ARoom() called
Welcome to the Vacant Room
There's no escape
ARoom() called
ExitRoom() called
~ExitRoom() called
~ARoom() called
~ARoom() called
Welcome to the Vacant Room
There's no escape
ARoom() called
~ARoom() called
Welcome to the Vacant Room
```

```
There's no escape
ARoom() called
~ARoom() called
```

The issue is that we are still using compile-time polymorphism. Our function is returning an `ARoom` object. Even though we are making an `ExitRoom` object, since it is a child of `ARoom` the object is being cast to its parent type.

Runtime Polymorphism

We need to switch to runtime polymorphism where the exact type will be resolved at runtime. Change all of our `ARoom` object references to be a pointer to an `ARoom`. The `go_to_next_room()` method should return a pointer to an object on the Free Store as well.

Build and run once again, now you should find yourself out of the room!

The Abstract Room Parent

The room class is already called `ARoom` by design - it should be abstract. We can't have just a room, we need a specific type of room to be in. Modify the `ARoom::escapeTheRoom()` method to be purely virtual and abstract.

Try building and it will fail. We can no longer create our abstract room object the other 90% of the time.

The Concrete Guess The Number Room Child

The last room is stubbed out as well. Have `GuessTheNumberRoom` publicly extend `ARoom`. In the constructor, set the room name to something appropriate. Override the `escapeTheRoom()` method to contain a simple guess the number game. The class is already computing a secret number in the range `[1, 20]`. Have the method give the user five guesses to determine the number. A rough pseudocode is below to assist:

- track number of guesses made
- while number of guesses made is less than allowable number (5)
    - have user enter a guess
    - increment number of guesses
    - if guess is too low, then tell the user they are too low

    - if guess is too high, then tell the user they are too high
    - if guess is correct, then tell them they are correct and return true
- they didn't guess the number, return false

Update `go_to_next_room()` to now return a pointer to an object of type `GuessTheNumberRoom` the other 90% of the time.

Build and run your program. You now have two ways out - (1) finding the exit room (2) guessing the number.

Grading Rubric

Your submission will be graded according to the following rubric:

Points	Requirement Description
0.70	Fully meets specifications
0.15	Submitted correctly by Tuesday, November 14, 2023 11:59 PM
0.15	**Best Practices** and **Style Guide** followed
1.00	**Total Points**

Lab Submission

Always, **always**, **ALWAYS** update the header comments at the top of your `main.cpp` file. And if you ever get stuck, remember that there is LOTS of **help** available.

Zip together your `ExitRoom.h`, `ExitRoom.cpp`, `GuessTheNumberRoom.h`, `GuessTheNumberRoom.cpp`, `Room.h`, `Room.cpp`, `main.cpp`, `Makefile` files and name the zip file `L5A.zip`. Upload this zip file to Canvas under L5A.

This lab is due by Tuesday, November 14, 2023 11:59 PM.
As with all labs you may, and are encouraged, to pair program a solution to this lab. If you choose to pair program a solution, be sure that you individually understand how to generate the correct solution.

Last Updated: 10/06/23 09:36

Any questions, comments, corrections, or request for use please contact jpaone {at} mines {dot} edu.

[Jump to Top] [Site Map]