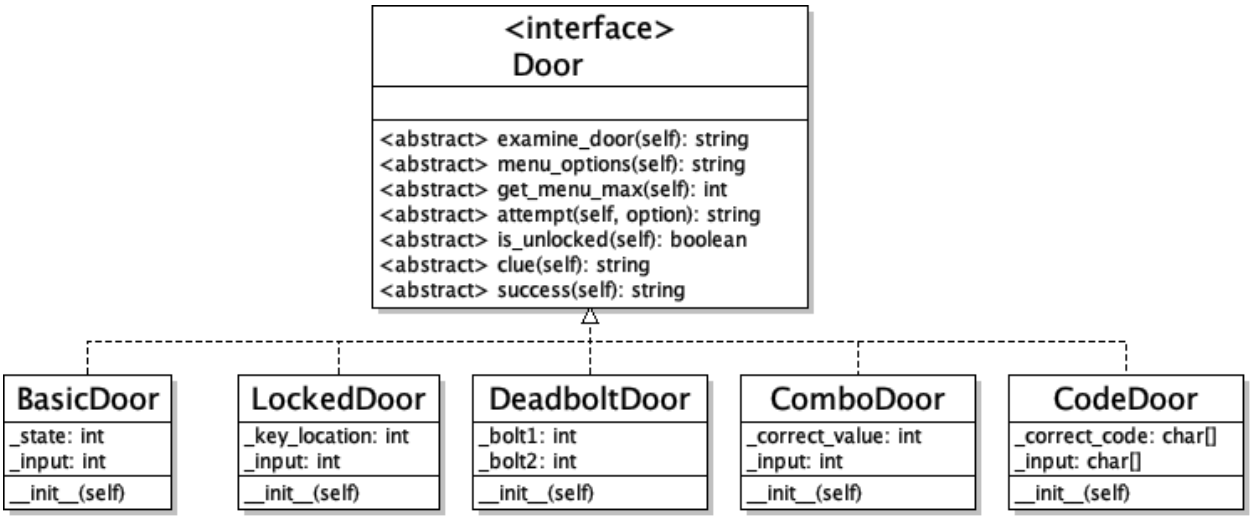
CECS 277 – Lab 9 – Interfaces

Escape Room

Write a program that simulates an escape room by having the user open a series of three doors random chosen from several different types of doors. Use the following class diagram to create your program (you must create a main, the Door interface, and Basic Door, then you can choose any two of the remaining four types of door to implement):



Door (door.py) – interface – these methods are abstract and should be implemented in each of the subclasses of door (not in door itself).

1. examine_door(self) – returns a string description of the door.
2. menu_options(self) – returns a string of the menu options that user can choose from when attempting to unlock the door.
3. get_menu_max(self) – returns the number of options in the above menu.
4. attempt(self, option) – passes in the user's selection from the menu. Uses that value to update the attributes that are needed to determine whether the user has unlocked the door (which is done in the is_unlocked method below). Returns a string describing what the user attempted.
5. is_unlocked(self) – checks to see if the door was unlocked, returns true if it is, false otherwise.
6. clue(self) – returns the hint that is returned if the user was unsuccessful at their attempt.
7. success(self) – returns the congratulatory message if the user was successful.

Subclasses of Door –

	Init	Description	Menu	Clue
BasicDoor	Randomizes the initial state of the door (push or pull).	A door that is either pushed to open, or pulled.	1. Push 2. Pull	Try the other way.

©2025 Cleary

LockedDoor	Randomizes the location of the key.	A locked door. The key is hidden nearby. Look around for the key.	1. Look under the mat. 2. Look under the flower pot. 3. Look under the fake rock.	Look somewhere else
DeadboltDoor	Randomizes the state of the two bolts (either locked or unlocked)	A door with two deadbolts. Both need to be unlocked to open the door, but you can't tell if each one is locked or unlocked.	1. Toggle bolt 1 2. Toggle bolt 2	You jiggle the door…it seems like one of the bolts is unlocked. -or- … it seems like it's completely locked.
ComboDoor	Randomize the value to a number 1-10.	A door with a combination lock. You can spin the dial to a number 1-10.	Enter # 1-10:	Too high. -or- Too low.
CodeDoor	Randomize each of the characters in the code as an 'X' or 'O'.	A door with a coded keypad with three characters. Each key toggles a value with an 'X' or an 'O'.	1. Press Key 1 2. Press Key 2 3. Press Key 3	Which characters are correct. -alternatively- The number of correct characters.

Main (main.py) –

1. open_door(door) – passes in a door object that the user will try to unlock. It should display the description of the door, and the menu, then get the user's selection, then pass that value to the attempt method and display the result. If the attempt was successful, then it should display the success message, otherwise, it should display the clue message and repeat from the menu until the user successfully opens the door.
2. main – the main should have a loop that repeats three times for the three doors that the user will try to open. Randomly construct a door from the ones that you implemented (it's fine if a particular door appears more than once) and pass it to open_door. After the user has opened all three doors, then display a congratulatory message and end the game.

Example Output:

```
Welcome to the Escape Room.
You must unlock 3 doors to
escape...

You encounter a basic door, you
can either push it or pull it to
open.
1. Push
2. Pull
1
You push the door.
Congratulations, you opened the
door.
```

©2025 Cleary

```
You encounter a door with a
combination lock, you can spin
the dial to a number 1-10.
Enter a number 1-10: 5
You turn the dial to... 5
Try a lower value.
Enter a number (1-10): 2
You turn the dial to... 2
Try a lower number.
Enter a number (1-10): 1
You turn the dial to... 1
You found the correct value and
opened the door.

You encounter a double deadbolt
door, both deadbolts must be
unlocked to open it, but you
can't tell from looking at them
whether they're locked or
unlocked.
1. Toggle Bolt 1
2. Toggle Bolt 2
1
You toggle the first bolt.
You jiggle the door...it's
completely locked.
1. Toggle Bolt 1
2. Toggle Bolt 2
1
You toggle the first bolt.
You jiggle the door... it seems
like one of the bolts is unlocked.
1. Toggle Bolt 1
2. Toggle Bolt 2
2
You toggle the second bolt.
You unlocked both deadbolts and
opened the door

Congratulations! You
escaped...this time.
```

Notes:

1. You should have 5 different files: main.py, door.py, basic_door.py, plus two (or more) other subclasses of door.
2. Check all user input using the get_int_range function in the check_input module. Use the value returned from get_menu_max as the upper bound.
3. Do not create any extra methods, attributes, functions, parameters, etc.
4. Please do not create any global variables or use the attributes globally (ie. do not access any of the attributes using the underscores).
5. Use docstrings to document each of the classes, their attributes, and each of their methods. See the lecture notes for examples.
6. Place your names, date, and a brief description of the program in a comment block at the top of your main file. Place brief comments throughout your code.
7. Thoroughly test your program before submitting:
    a. Make sure that the program randomly chooses three doors for the user to open (repeats are fine).
    b. If your program allows repeats, make sure that the door is reconstructed. Otherwise, the door might still have values from the previous time it was opened, and may either still be in the unlocked state or have the same solution.
    c. Make sure that each of your doors can be created and opened.
    d. It is fine if a door randomizes to the unlocked state, such as the Deadbolt Door, or Code Door. However, make sure that the user is still prompted to unlock the door. It should not skip the menu and give them the congratulatory message until they have interacted with the door themselves.
    e. Make sure that each of the options performs the correct operation.
    f. Make sure the clues are displayed when the user is unsuccessful and that they display the correct information.

©2025 Cleary

Escape Room Rubric – Time estimate: 4 hours

Escape Room **10 points**	Correct. 2 points	A minor mistake. 1.5 points	A few mistakes. 1 point	Several mistakes. 0.5 points	No attempt. 0 points
Door interface (in a separate file)**:** 1. It is abstract. 2. Has all methods for: examine_door, menu_options, get_menu_max, attempt, is_unlocked, clue, and success. 3. Uses decorator to make all methods abstract.					
Basic Door class (separate file)**:** 1. Inherits from Door class. 2. Has correct attributes. 3. init method randomizes the initial state of the door (push or pull). 4. Correctly overrides all Door methods and the attempt method sets the value for the input attribute.					
Two other door classes (sep. files)**:** 1. Inherit from Door class. 2. Has correct attributes. 3. init method randomizes initial state of the door. 4. Correctly overrides all Door methods.					
Main file (in separate file)**:** 1. Constructs 3 random doors in a loop. 2. Open_door method prompts the user for their menu selection and correctly calls each of the door methods to help the user open the door. 3. Error checks all user input. 4. Moves on to next door when user solves it. 5. Ends game when user solves 3 doors.					
Code Formatting: 1. Correct spacing. 2. Meaningful variable names. 3. No exceptions thrown. 4. No global variables or accessing attributes directly. 5. Correct documentation.					

©2025 Cleary