35/35

Milestone

At the moment, I have written most of the code I will need for the final product. I intend to write one more class that makes use of all of the others, allowing the user to simulate an actual building, and demonstrating the functionality of the simulation. I currently haven't tested anything, so that also needs to be done. I think it works though. All of my research is finished at this point. The written section needs a lot of work. I will need to revise the report, do a bibliography, and finish the code as described above, and I should be ready to turn in. I think 2 weeks is plenty of time for these things.

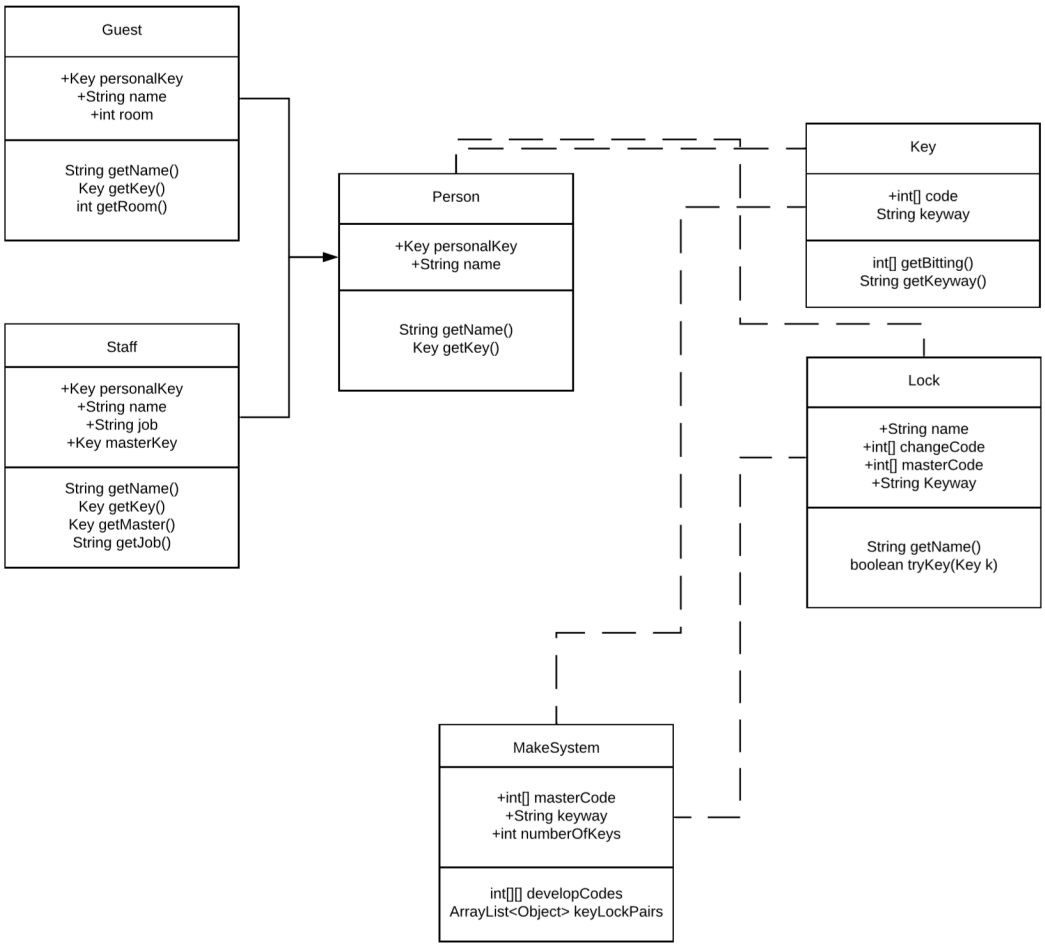
Abstract

This program was created with the intention of creating and simulating a mastered system of pin-tumbler locks as it would be implemented by a locksmith. This paper will explain the specific execution and use of said program.

Introduction

Prior to this project, I had taken up an interest in locksmithing, and the mechanics of pin-tumbler locks. When this was introduced, it seemed natural for me to base my project off of this interest. When doing some initial research, I was looking for used methods of developing bitting codes for mastered key systems, and found that there are no common ways of doing this. When implementing such a system, most locksmiths will simply rely on lock manufacturers to provide codes (and keys) from bitting arrays that are calculated and created by said manufacturers. My program allows for the creation of a simple mastered system without the use of existing bitting array, which can be used to simulate a system in greater detail rather than using placeholders as is commonly done.

Detailed System Description



This UML will probably be updated before the final writeup. I also don't have a class for user interaction written yet. This section will reflect those changes later on.

Requirements

The most unique part of this program is the development of a bitting array from a master key. This type of resource is generally not freely available, which is a problem I hope to solve with this software.

Literature Review

KeySoft has released a program previously that has similar functionality to mine, however: Their software is not free, and it does not allow for simulation of the use of the system. The main use of their software is to create bitting arrays.

Beyond this program, I couldn't find any software that attempted to address this problem.

User Manual

(This will additionally be updated as a class is added for user interaction.)

Conclusion

(to be finished later)

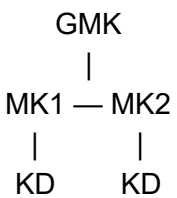
References and Research

https://extranet.assaabloydss.com/library/keysystems/pdf/mathematics_of_mk_webinar.pdf
Basic Mathematics of Master Keying

https://unitedlocksmith.net/blog/how-master-key-systems-work
Introduction to mechanics of physical system

https://www.assaabloyacademy.com/Local/assaabloyacademyCOM/Americas/Lesson%20Resources/2018KeySystemAssessment/1%20General%20Documents%20and%20Interactive%20Video%20-%20Designing%20Master%20Key%20Systems/Keying%20Systems%20and%20Nomenclature_2008.pdf
Longer and more clear document explaining the planning and execution of a key system

So plan:



KD KD
KD KD
KD KD
KD KD

This simulates a building with 2 floors, and 5 rooms on each floor, each of which are keyed differently. Each floor has a master key, and there is a grandmaster key that operates all doors in the building.

https://www.locksmithledger.com/keys-tools/article/10290193/the-concepts-and-mechanics-of-master-keying-developing-master-keys-using-generic-types
To help develop bitting codes, used in main method?

https://www.locksmithledger.com/keys-tools/article/10323366/masterkeying-by-the-numbers
This is better, guide on developing a key bitting array

Bitting generally goes X-Y-Z-A-B for master; change keys will originate along the line of (X±2)-(Y±2)... and so on. Subsequent change keys generally follow AA1 = HIJKL; AA2 = HIJK(L+2); AA3 = HIJK(L+4); and so on.

http://lockwiki.com/index.php/Differs
This is an interesting piece of info for real world locksmithing, but for my purposes I have no reason to get this complex. I will approximate MACS requirements by the method outlined above.