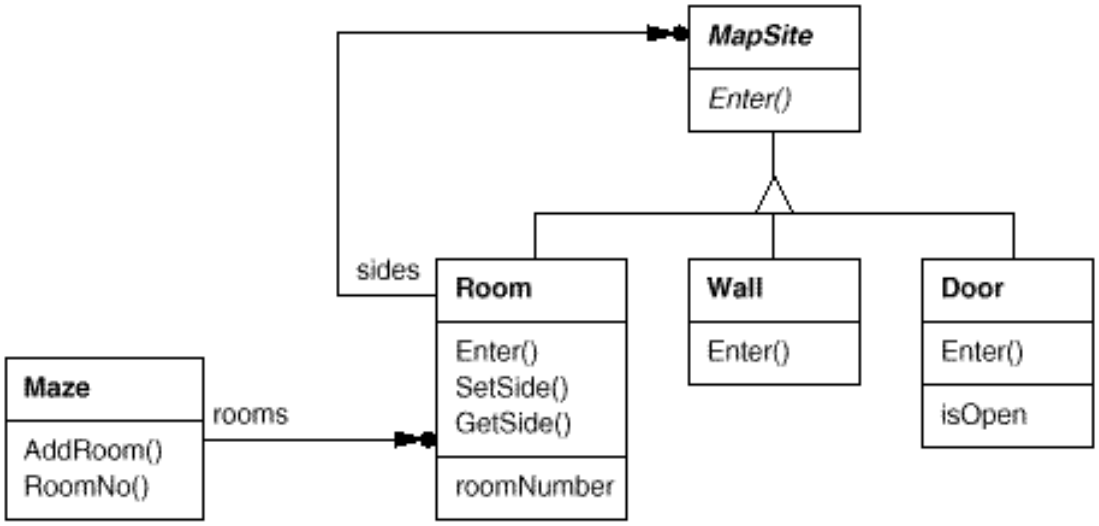
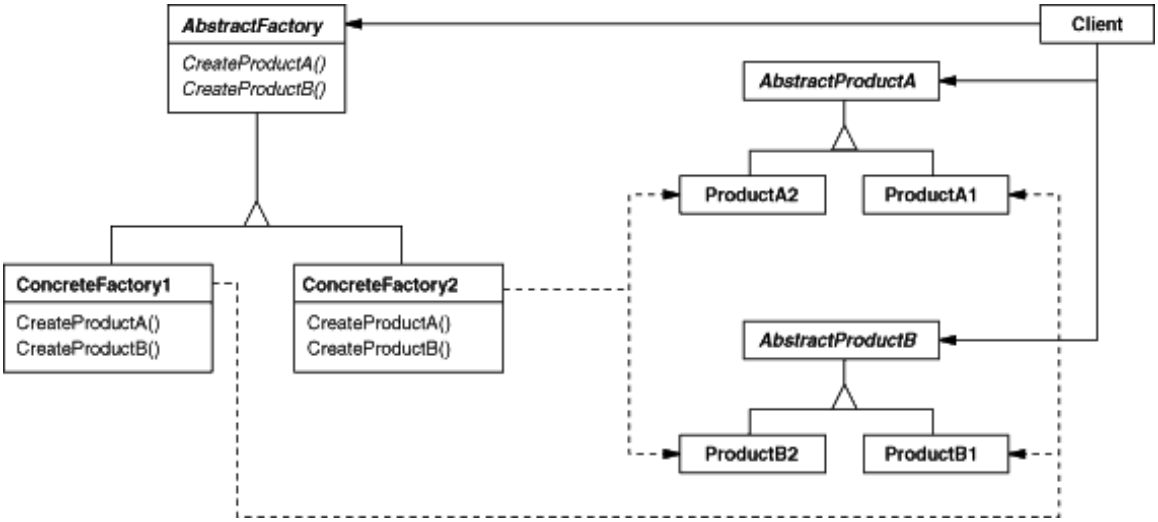
• MazeGame Example: the major classes in the game is described with the following class diagram.




1. Read the class diagram above. Discuss its meaning, come up with as many details as you could. Note that the solid arrows are one-directional association. I.e., the Maze knows about the room, but not the other way around.
   a) In particular, the part on the right. What does the relationship between “Room” and “MapSite” means?
2. Consider the Factory Pattern. In the sample code, how do we define necessary “factory methods”? What’s the difference between the old code and new using factory methods?
3. Now we are adding a “theme” idea to the game. I.e., we can have “EnchantedMazeGame” or “BombedMazeGame”. In the former, the Room, the Door and the Wall would become “Enchanted” versions; in the latter, the Room and the Wall would have “Bombed” versions. How do we do this following the Factory Pattern? Map the classes to the one in the pattern.

4. Let's switch to the Abstract Factory pattern, as shown in the class diagram below.




5. Discuss what's going on with this pattern.
6. Revisit question 3. Following this pattern, what would the implementation look like? Map your classes (and necessary new ones) to the classes in this pattern.

7. How do we create an "EnchantedMazeGame" then? Write some pseudo code.

8. Compare with the Factory pattern, what are the differences? Also, what exactly is "AbstractFactory"? Hint: what does the "factory/creator" classes do in these two patterns?

9. Benefits and drawbacks?