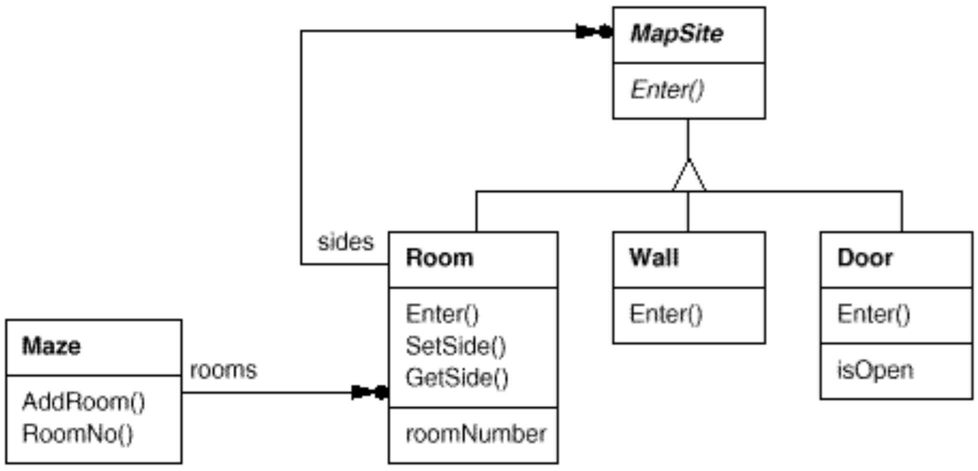
Creational Patterns

Creational design patterns abstract the instantiation process. They help make a system independent of how its objects are created, composed, and represented.
- A class creational pattern uses inheritance to vary the class that's instantiated. (static, specified at compile-time)
- An object creational pattern will delegate instantiation to another object. (dynamic, specified at run-time)

Two recurring themes:
- the all encapsulate knowledge about which concrete classes the system uses.
- the hide how instances of these classes are created and put together.

Example: A Maze game
we'll just focus on how mazes get created.



```
enum Direction { North, South, East, West };

Maze* MazeGame::CreateMaze ()
{
Maze* aMaze = new Maze;
Room* r1 = new Room(1);
Room* r2 = new Room(2);
Door* theDoor = new Door(r1, r2);

aMaze->AddRoom(r1);
aMaze->AddRoom(r2);

r1->SetSide(North, new Wall);
r1->SetSide(East, theDoor);
r1->SetSide(South, new Wall);
r1->SetSide(West, new Wall);

r2->SetSide(North, new Wall);
r2->SetSide(East, new Wall);
r2->SetSide(South, new Wall);
r2->SetSide(West, theDoor);
return aMaze;
}
```

To make it simpler?
the Room constructor could initialize the sides with walls ahead of time. ???

- That just moves the code somewhere else.
- The real problem with this member function isn't its size but its inflexibility.
- It hard-codes the maze layout.

How?

- Factory Method
CreateMaze calls virtual functions instead of constructor calls to create the rooms, walls, and doors it requires, then you can change the classes that get instantiated by making a subclass of MazeGame and redefining those virtual functions.

- Abstract Factory
CreateMaze is passed an object as a parameter to use to create rooms, walls, and doors, then you can change the classes of rooms, walls, and doors by passing a different parameter.

- Builder
CreateMaze is passed an object that can create a new maze in its entirety using operations for adding rooms, doors, and walls to the maze it builds, then you can use inheritance to change parts of the maze or the way the maze is built.

- Prototype
CreateMaze is parameterised by various prototypical room, door, and wall objects, which it then copies and adds to the maze, then you can change the maze's composition by replacing these prototypical objects with different ones.

Discussion

There are two common ways to parameters a system by the classes of objects it creates.
- one way is to subclass the class that creates the objects.(Factory Method)
- the other way to parameters a system relies more on object composition. (Abstract Factory, Builder, Prototype)

Often, designs start out using Factory Method and evolve toward the other creational patterns as the designer discovers where more flexibility is needed.