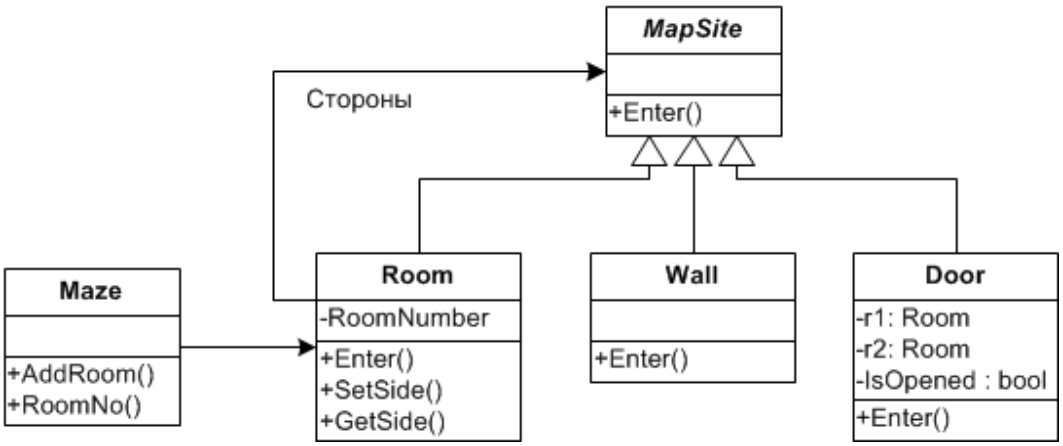
Описание паттернов проектирования

- Название и классификация паттерна
- Назначение
- Известен также под именем
- Мотивация (описание)
- Использование
- Структура
- Участники
- Отношения
- Результаты (достоинства и недостатки)
- Реализация
- Пример кода
- Известные применения
- Родственные паттерны

MazeGame

Диаграмма классов




```
using System;
using System.Collections.Generic;

namespace MazeCommon
{
    public enum Direction { North, South, East, West };

    public abstract class MapSite {
        public abstract void Enter();
        public object Clone()
          { return this.MemberwiseClone(); }
    }

    public class Room: MapSite {
        private MapSite[] sides = new MapSite[4];
        public Room(int no)
          { RoomNumber = no; }
        public MapSite GetSide(Direction d)
          { return sides[(int)d]; }
        public void SetSide(Direction d, MapSite m)
        { sides[(int)d] = m; }
        public override void Enter()
        { }
        public int RoomNumber {get; set; }
    }
```

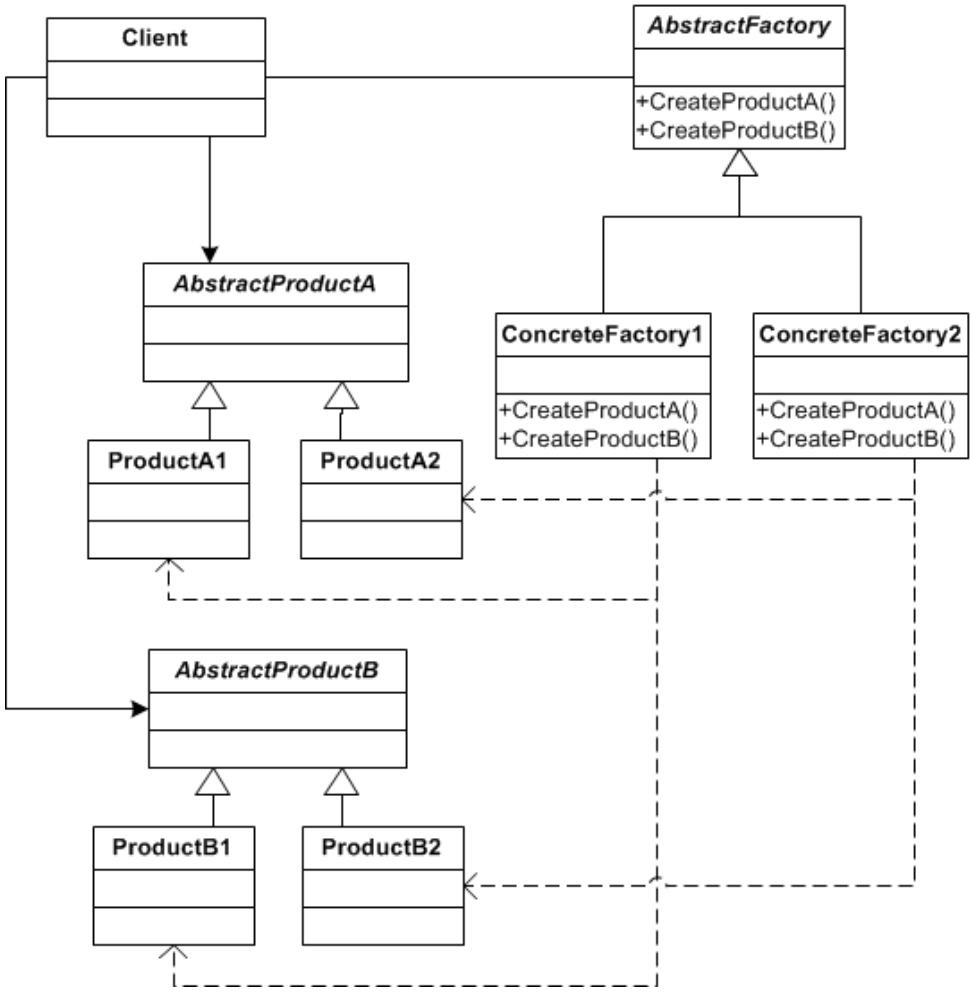
```
    public class Wall : MapSite {
        public override void Enter()
        { }
    };

    public class Door : MapSite {
        private Room r1, r2;
        private bool isOpened = true;
        public Door(Room r1 = null, Room r2 = null)
          { this.r1 = r1; this.r2 = r2; }
        public void Initialize(Room r1 = null, Room r2 = null)
          { this.r1 = r1; this.r2 = r2; }
        public override void Enter()
          { }
        public Room OtherSideFrom(Room r) {
            if (r == r1)
                return r2;
            else return r1;
        }
    }

    public class Maze {
        private List<Room> rlist = new List<Room>();
        public void AddRoom(Room r)
          { rlist.Add(r); }
        public Room RoomNo(int n) {
            if (n > rlist.Count)
                return null;
            return rlist[n-1];
        }
    }

    public class MazeGame {
        public Maze CreateMaze()
        {
            Maze aMaze = new Maze();
            Room r1 = new Room(1);
            Room r2 = new Room(2);
            Door d = new Door(r1,r2);

            aMaze.AddRoom(r1);
            aMaze.AddRoom(r2);

            r1.SetSide(Direction.North, new Wall());
            r1.SetSide(Direction.East,  d);
            r1.SetSide(Direction.South, new Wall());
            r1.SetSide(Direction.West,  new Wall());
            r2.SetSide(Direction.North, new Wall());
            r2.SetSide(Direction.East,  new Wall());
            r2.SetSide(Direction.South, new Wall());
            r2.SetSide(Direction.West,  d);

            return aMaze;
        }
    }

    class ProgramBase {
        static void Main() {
            var game = new MazeGame();
            Maze m = game.CreateMaze();
        }
    }
}
```

Порождающие паттерны

- Абстрактная фабрика (Abstract Factory)
- Строитель (Builder)
- Фабричный метод (Factory Method)
- Прототип (Prototype)
- Одиночка (Singleton)

Абстрактная фабрика (Abstract Factory)

Диаграмма классов



```
using System;
using System.Collections.Generic;
using MazeCommon;

namespace MazeGameAbstractFactory
{
    public class MazeFactory {
        public virtual Maze MakeMaze()
          { return new Maze(); }
        public virtual Wall MakeWall()
          { return new Wall(); }
        public virtual Room MakeRoom(int n)
          { return new Room(n); }
        public virtual Door MakeDoor(Room r1, Room r2)
          { return new Door(r1,r2); }
    };

    public class MazeGame {
        public Maze CreateMaze(MazeFactory f)
        {
            Maze aMaze = f.MakeMaze();
            Room r1 = f.MakeRoom(1);
            Room r2 = f.MakeRoom(2);
            Door d = f.MakeDoor(r1,r2);

            aMaze.AddRoom(r1);
            aMaze.AddRoom(r2);

            r1.SetSide(Direction.North, f.MakeWall());
            r1.SetSide(Direction.East, d);
            r1.SetSide(Direction.South, f.MakeWall());
            r1.SetSide(Direction.West, f.MakeWall());
            r2.SetSide(Direction.North, f.MakeWall());
            r2.SetSide(Direction.East, f.MakeWall());
            r2.SetSide(Direction.South, f.MakeWall());
            r2.SetSide(Direction.West, d);

            return aMaze;
        }
    }

    public class BombedWall: Wall { }

    public class RoomWithABomb: Room {
        public RoomWithABomb(int n): base(n)
        { }
    }

    public class BombedMazeFactory : MazeFactory {
        public override Room MakeRoom(int n)
          { return new RoomWithABomb(n); }
        public override Wall MakeWall()
          { return new BombedWall(); }
    };

    public class ProgramAbstractFactory {
        static void Main() {
            Console.WriteLine("AbstractFactory");
            MazeGame game = new MazeGame();
            BombedMazeFactory f = new BombedMazeFactory();
            game.CreateMaze(f);
        }
    }
}
```