**面向对象程序设计**

**魔塔游戏软件重构说明书**

第十组 曾凯、颜俊

目录

[1 Introduction 简介 3](#_Toc29245)

[1.1 Purpose 目的 3](#_Toc27196)

[1.2 Scope 范围 3](#_Toc23239)

[1.2.1 Name 软件名称 3](#_Toc13437)

[1.2.2 Function 软件功能 3](#_Toc18056)

[2 Design 系统总体设计 4](#_Toc3922)

[2.1 总体类图设计 4](#_Toc25316)

[2.2 模块分解描述 5](#_Toc7761)

[2.2.1 block抽象类 5](#_Toc18898)

[2.2.2 block具体类 5](#_Toc28003)

[2.2.3 一系列指令类 5](#_Toc4642)

[2.2.4 驱动类 5](#_Toc12414)

[2.3 游戏流程说明 5](#_Toc16077)

[2.3.1 简要说明 5](#_Toc27180)

[2.3.2 顺序图 6](#_Toc28127)

# 1 Introduction 简介

## 1.1 Purpose 目的

本文档是魔塔游戏系统设计的概要描述。

## 1.2 Scope 范围

### 1.2.1 Name 软件名称

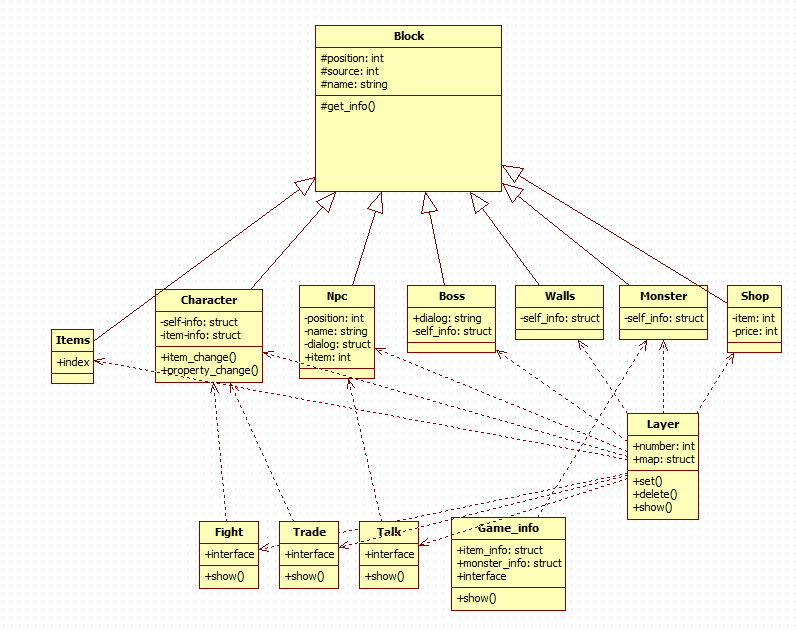
魔塔。

### 1.2.2 Function 软件功能

魔塔游戏的面向对象实现方式。

# 2 Design 系统总体设计

## 2.1 总体类图设计



## 2.2 模块分解描述

### 2.2.1 block抽象类

block类：抽象类，小方块的抽象，有一个抽象方法get\_info，用来得到自身的某些值。

### 2.2.2 block具体类

character类、npc类、boss类、walls类、monster类、shop类：分别是不同种的消方块，继承自block类，自身有私有的方法和成员变量。character类含有方法item\_change和property\_change，用于改变拥有的物品和自身状态。

### 2.2.3 一系列指令类

fight类、trade类、talk类：分别是控制玩家动作的方法，包括战斗指令、交易指令、谈话指令等等。它们要实现一个共同的接口show，用于显示触发相应指令时的游戏界面。

### 2.2.4 驱动类

layer类：驱动类，魔塔的每一层即是一个layer类，它通过set方法来为地图的某位置放置方块，实际上是通过实例化一个新的block来实现的，通过delete方法来删除地图上某位置的方块，show用来显示游戏界面。

## 2.3 游戏流程说明

### 2.3.1 简要说明

整个游戏由layer类驱动，它调用各种block类，将它们显示出来构成了整个地图。

玩家控制人物时，游戏通过判断玩家的操作，以及人物上下左右的方块情况来决定触发什么样的动作，比如，当玩家发出向左的指令时，游戏检测到人物左边是一个商店，那么就调用trade类来进行一次交易的操作，同理，如果遇到一个怪物，就进入战斗模式。

character类根据动作结果实时更新人物状态，主要实现是通过调用property\_change和item\_change完成的。完成动作后，layer类根据结果来更新地图的显示，具体体现为调用set或者delete方法来增加或者删除block块。

有一个单独的game\_info类，来为玩家提供资料查询服务。

### 2.3.2 顺序图

地图

人物

战斗

交易

谈话

信息

结束

结果

查询

结果

结果

结果

结果

遇npc

遇商店

遇怪

更新位置

显示地图

# Refactoring 系统重构

## 3.1 工厂方法模式

### 3.1.1 适用性

当一个类希望由它的子类来指定它所创建的对象时。

### 3.1.2 使用原因

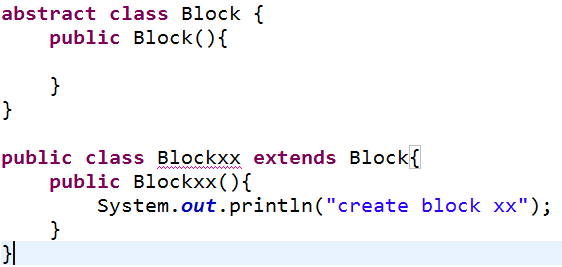
显然魔塔地图中的每一个小方块是一个抽象的概念，而我们构建地图时需要根据需求实例化不同的方块，可以是一块墙，可以是一个怪，也可以是一扇门。

### 3.1.3 实现

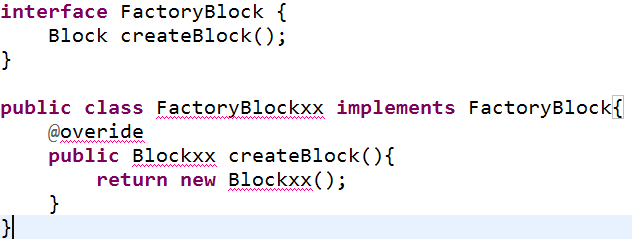
小方块的概念可以看作抽象产品，我们真正使用的方块则是具体产品。

### 3.1.4 实例

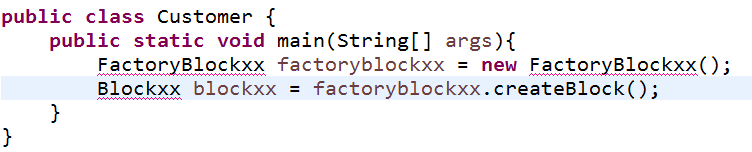
抽象产品类和具体产品类



工厂类



客户类



### 3.1.5 优势

客户直接使用抽象产品提供的接口就能根据需求创建不同的小方块了。在添加新类型的小方块时（比如新的怪），不需要改动已存在的代码，只需要实现相应的具体产品类和具体工厂类就行了。

## 3.2 享元模式

### 3.2.1 适用性

一个程序使用了大量的对象，不可共享的状态都可变为外部状态。

### 3.2.2 使用原因

魔塔里面由很多相同的怪，它们的血量、攻击力等属性都是相同的，唯一不同的是位置信息，因此排除位置这个外部状态，每一个怪都是可以共享的一个对象。

### 3.2.3 实现

把怪的攻击、血量等属性共享出来，位置不共享。

### 3.2.4 优势

极大减少内存中对象的数量，外部状态独立不影响内部状态。对对象的维护变得非常简单。