## 推箱子控制台的实现

**思路：**

通过对图形界面，展示出整个游戏界面，通过按键实现人物的上下左右移动，遇到墙就无法通行，遇到箱子，则判断箱子的下一个位置是否是空地，从而实现箱子的移动。

**准备工作：**

找到6张图片分别代表箱子、人物、地、墙、目标位置和箱子被推到目标位置的显示，大小一致。

**程序书写：**

**Config.java**

表示游戏的配置，定义图形界面的宽高以及图片的间距，用一个二维数组**int** [][] ***map***表示这个界面的初始绘制，0表示地，1表示墙；一个二维数组**int**[][] *targersBox*表示目标位置，以及箱子的初始位置**int**[][] *boxFirst*。

以及四个常量上下左右，代表人物的移动方向

**Game.java**

游戏的界面的绘制，首先初始化界面，绘制不同的图形，首先绘制地图，其次是箱子和目标位置，箱子会覆盖地面。

使用paint(Graphics g)

public void **paint**([Graphics](mk:@MSITStore:F:\JDK_API_1_6_zh_CN.CHM::/javax/swing/../../java/awt/Graphics.html) g)

由 Swing 调用，以绘制组件。应用程序不应直接调用 paint，而是应该使用 repaint 方法来安排重绘组件。

此方法实际上将绘制工作委托给三个受保护的方法：paintComponent、paintBorder 和 paintChildren。按列出的顺序调用这些方法，以确保子组件出现在组件本身的顶部。一般来说，不应在分配给边框的 insets 区域绘制组件及其子组件。子类可以始终重写此方法。只想特殊化 UI（外观）委托的 paint 方法的子类只需重写 paintComponent。

### 类 Graphics

public abstract class **Graphics**

extends [Object](mk:@MSITStore:F:\JDK_API_1_6_zh_CN.CHM::/java/awt/../../java/lang/Object.html)

Graphics 类是所有图形上下文的抽象基类，允许应用程序在组件（已经在各种设备上实现）以及闭屏图像上进行绘制。

Graphics 对象封装了 Java 支持的基本呈现操作所需的状态信息。此状态信息包括以下属性：

* 要在其上绘制的 Component 对象。
* 呈现和剪贴坐标的转换原点。
* 当前剪贴区。
* 当前颜色。
* 当前字体。
* 当前逻辑像素操作函数（XOR 或 Paint）。
* 当前 XOR 交替颜色（参见 [setXORMode(java.awt.Color)](mk:@MSITStore:F:\JDK_API_1_6_zh_CN.CHM::/java/awt/../../java/awt/Graphics.html#setXORMode(java.awt.Color))）。

坐标是无限细分的，并且位于设备的像素之间。绘制图形轮廓的操作是通过使用像素大小的画笔遍历像素间无限细分路径的操作，画笔从路径上的锚点向下和向右绘制。填充图形的操作是填充图形内部区域无限细分路径操作。呈现水平文本的操作是呈现字符字形完全位于基线坐标之上的上升部分。

[**Graphics**](mk:@MSITStore:F:\JDK_API_1_6_zh_CN.CHM::/java/awt/../../java/awt/Graphics.html#Graphics())()   
          构造一个新的 Graphics 对象。

g.setColor(Color.***white***);

g.fillRect(0, 0, Config.*WIDTH*, Config.*HEIGHT*);

g.drawImage(target, Config.*targersBox*[i][1] \* Config.*CELL*, Config.*targersBox*[i][0] \* Config.*CELL*,**null**);

使用这三个方法实现矩形的图形填充

判断：

通过键盘上按键的不同执行不同的程序

首先判断下一个位置是否是空地，改变人物坐标

如果下一个是墙则不能前进

最主要的是如果下一个箱子，则需要判断下一个位置的点上是什么，然后进行坐标交换。