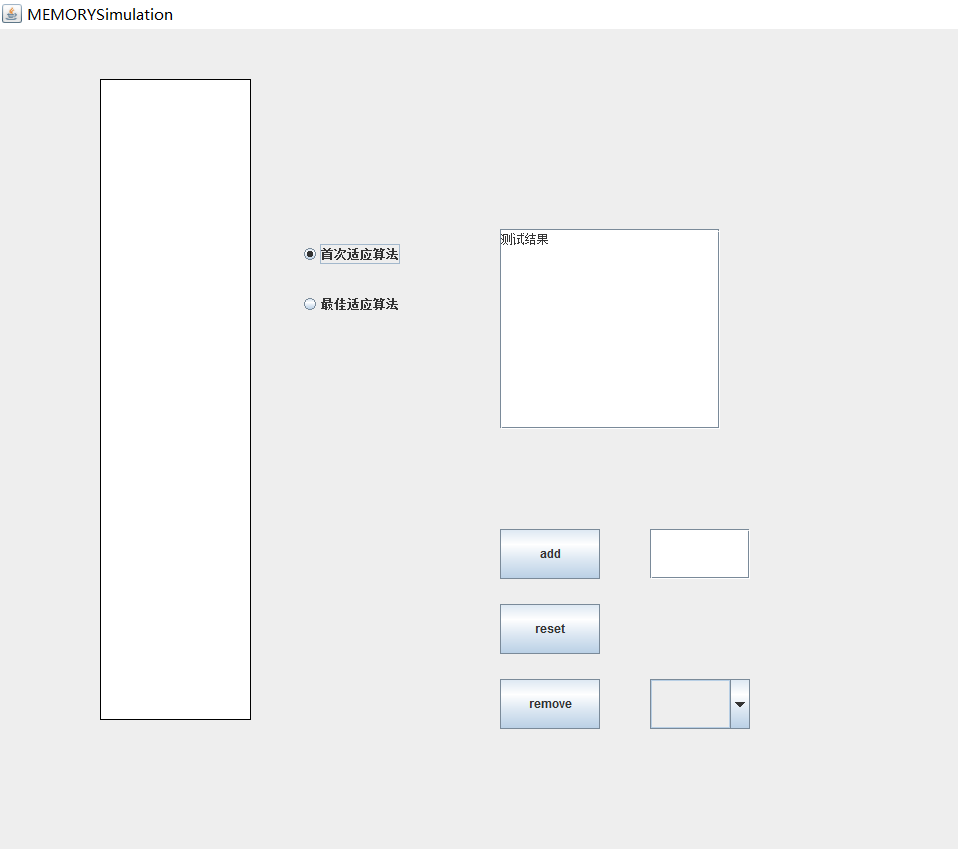
**设计方案的描述报告**

在该项目中，我模拟了一个640K的内存块。

**一．图形界面**



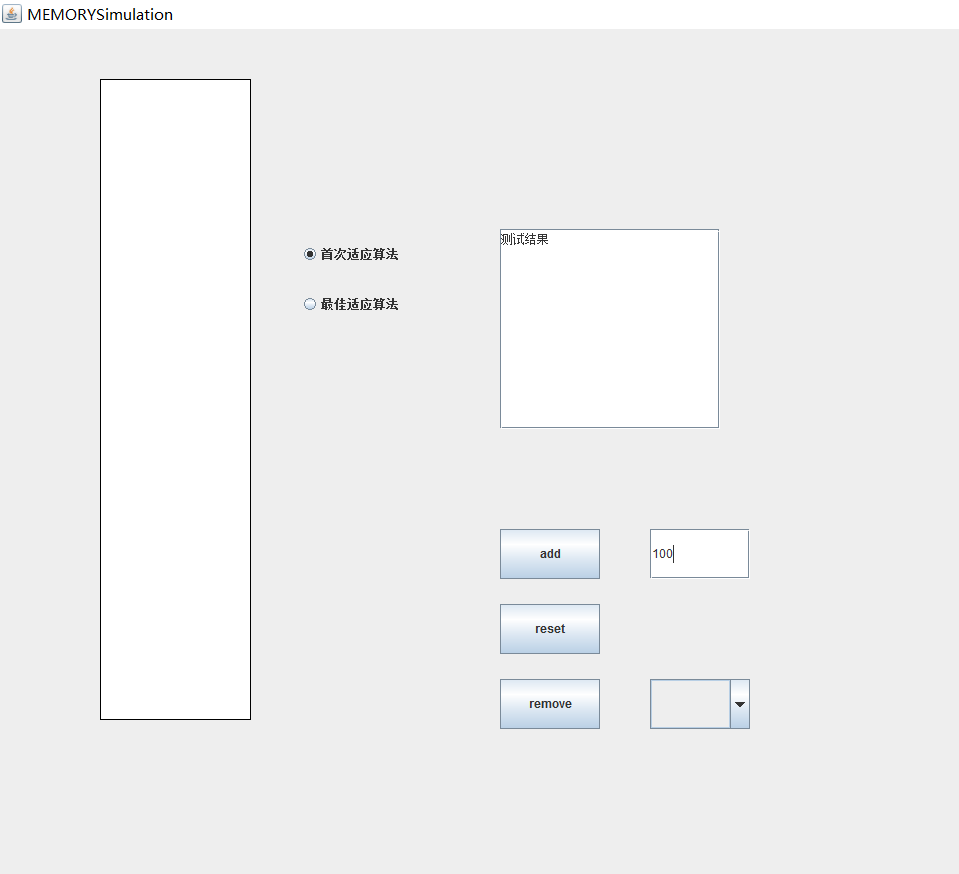
左边白色框为大小为640K的内存块，右边有两个圆形按钮，可选择首次适应算法或者最佳适应算法。 右侧有一个文本框，可以将每次操作的结果显示出来，便于观察和验证。

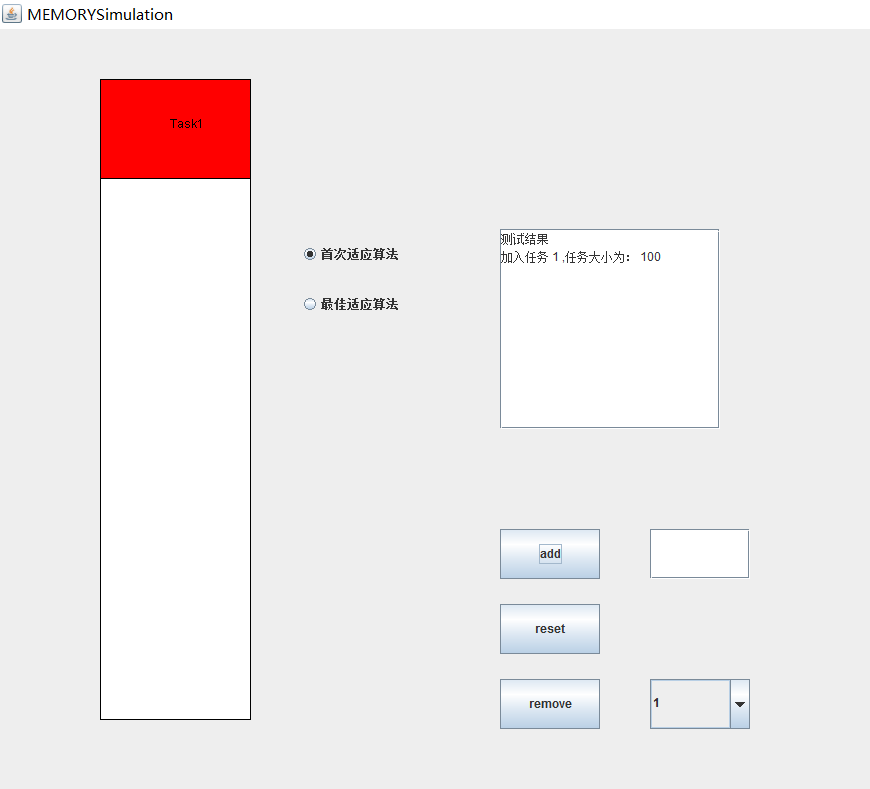
右下方有三个按钮，分别为添加，重置，释放。添加任务时，请在add后方的文本框中输入1~640范围的数值，代表任务的大小。要移除某个任务时，可以在remove按钮后方的下拉选项中，选择相应的任务标号，再点击remove即可删除。当更换算法时，请先点击reset按钮，再选择要使用的算法，之后进行操作即可。

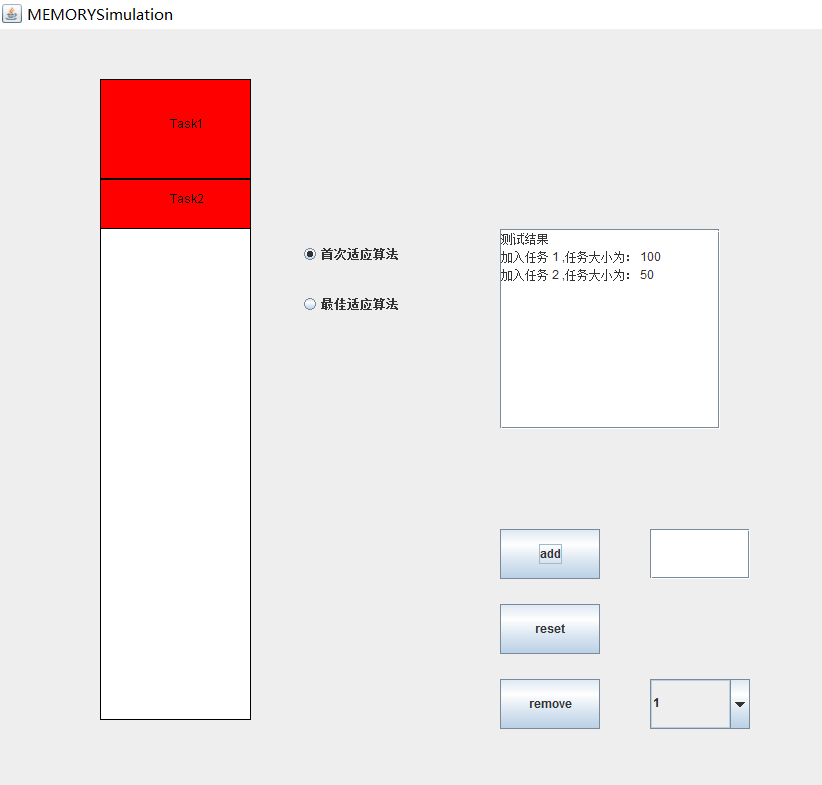
操作演示如下：

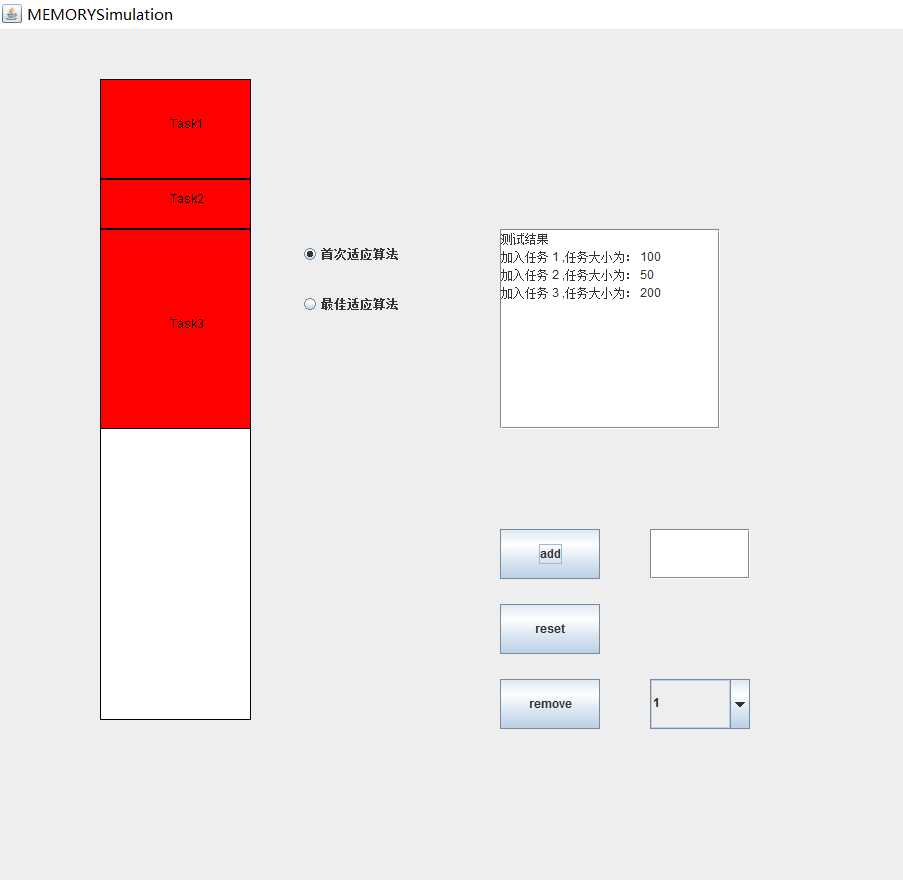
1. 输入新任务100.
2. 输入新任务50.
3. 输入新任务200
4. 移除任务2
5. 输入新任务20
6. 输入新任务200
7. 释放任务3
8. 输入新任务50

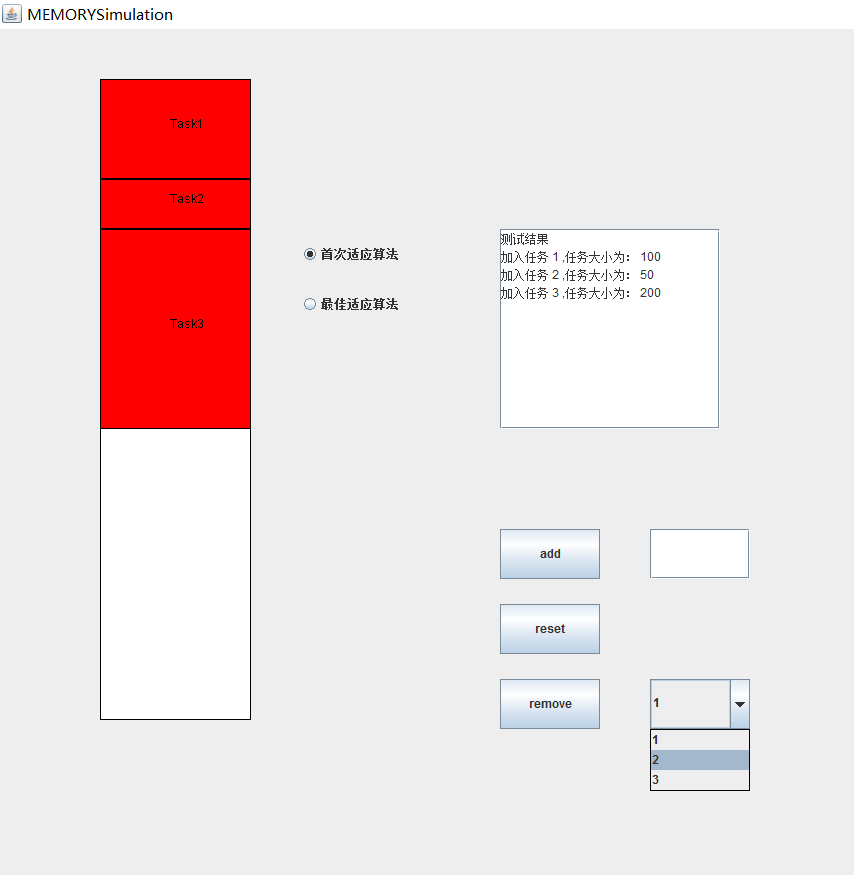
算法一：

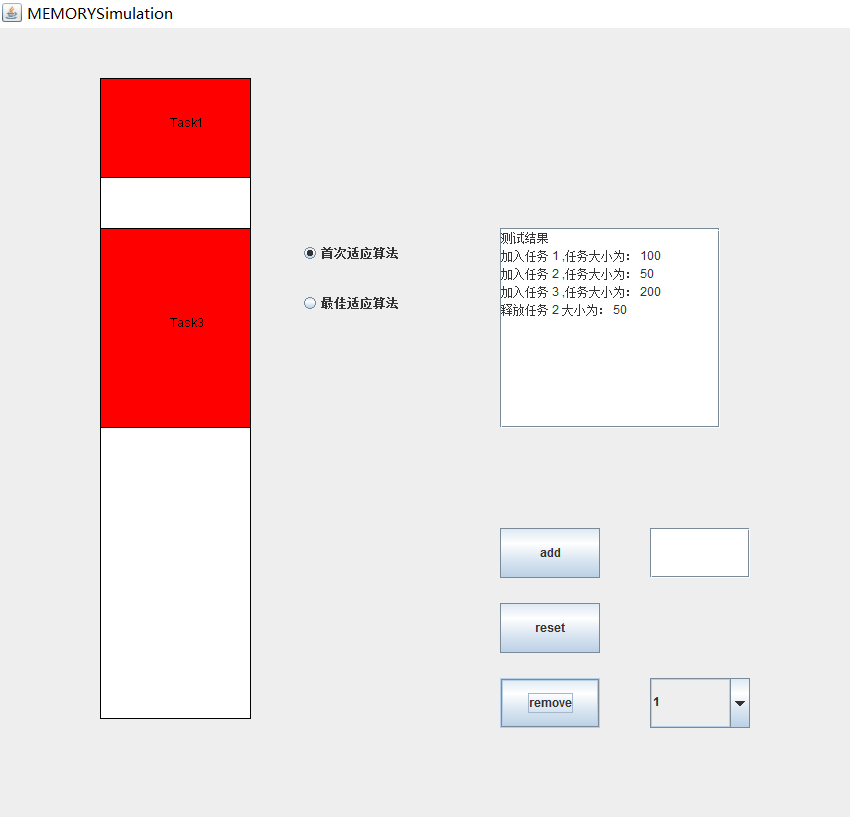


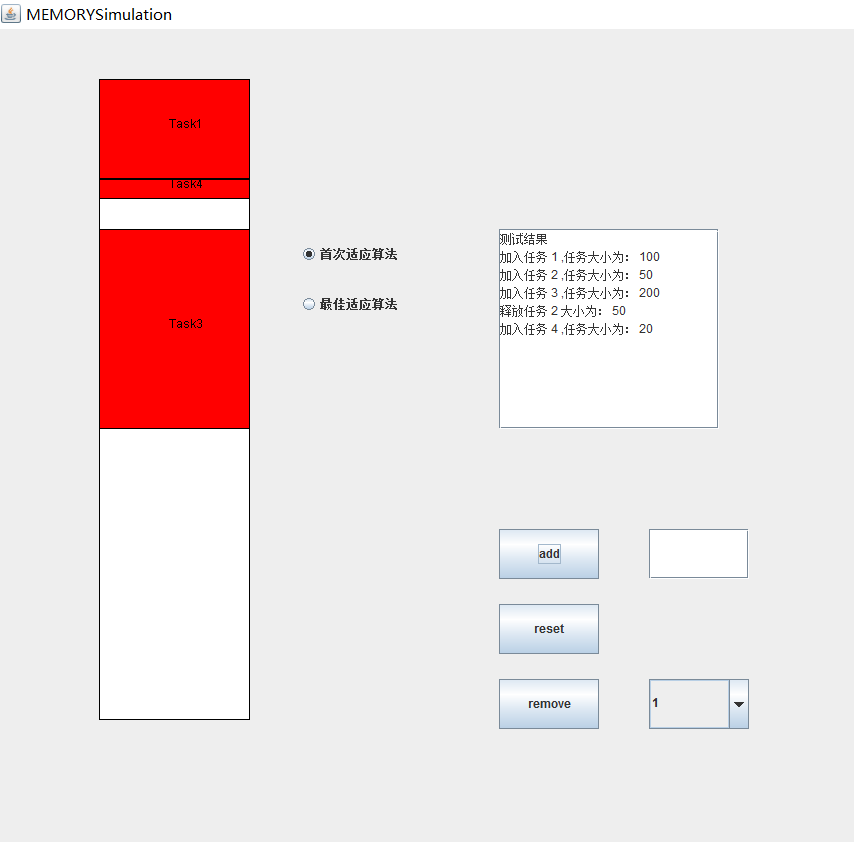


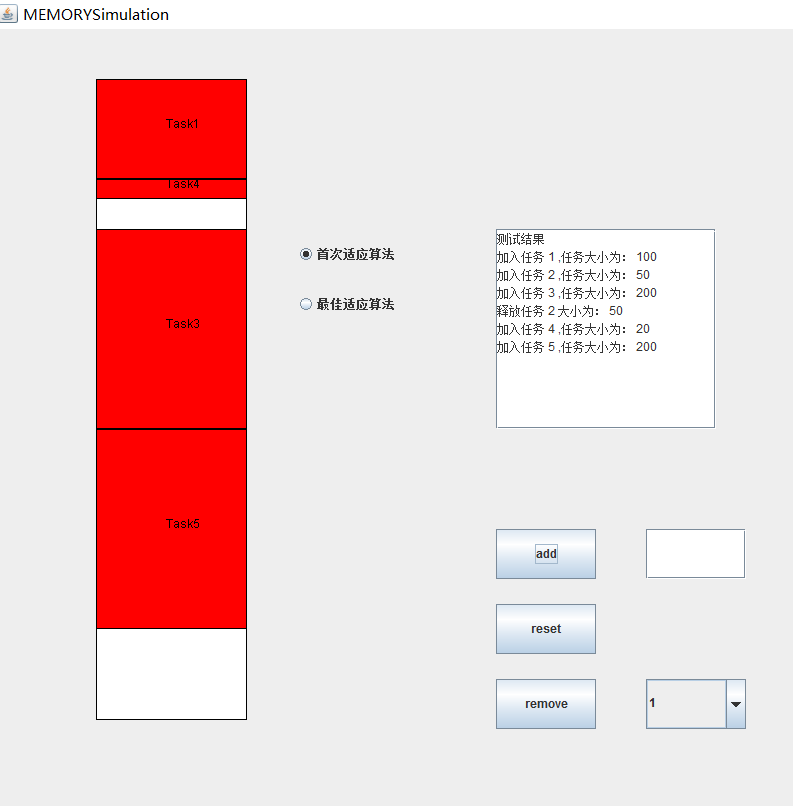


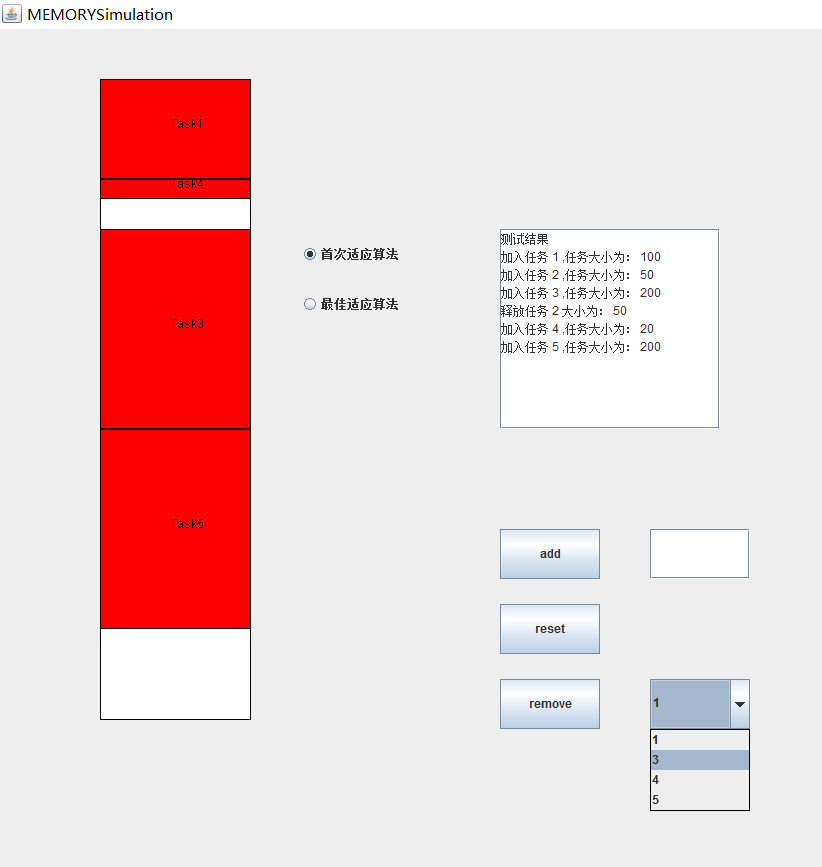


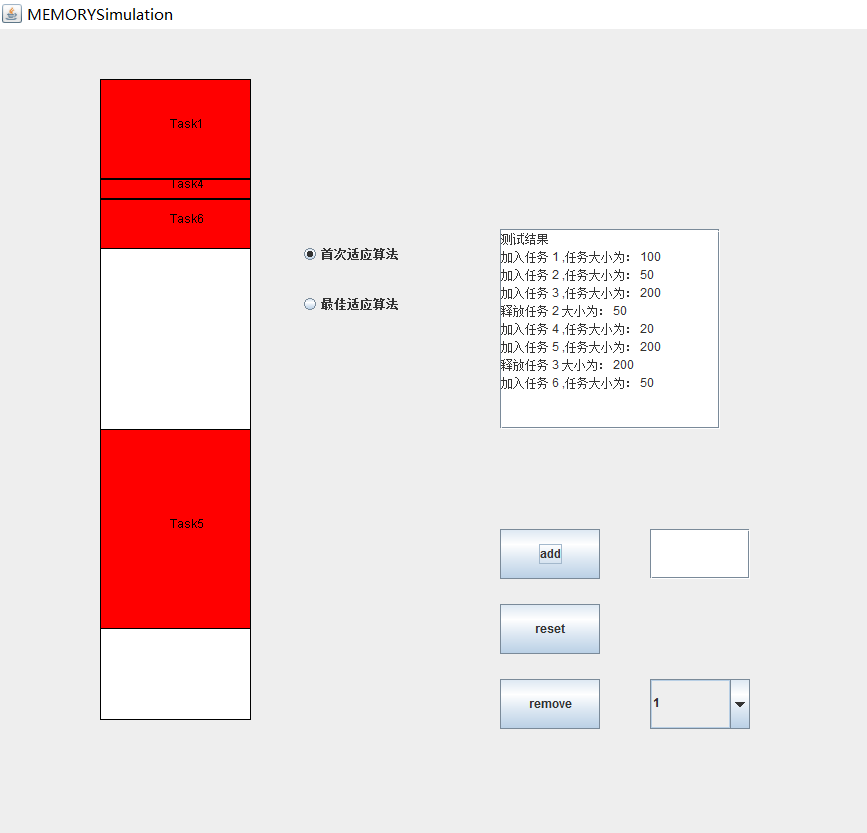




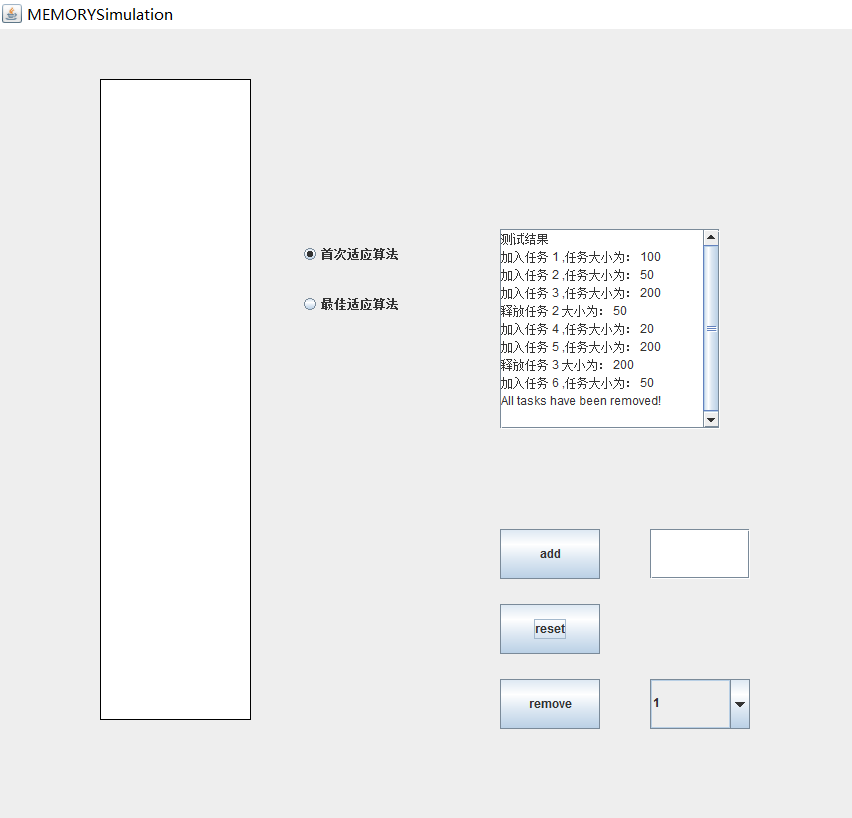




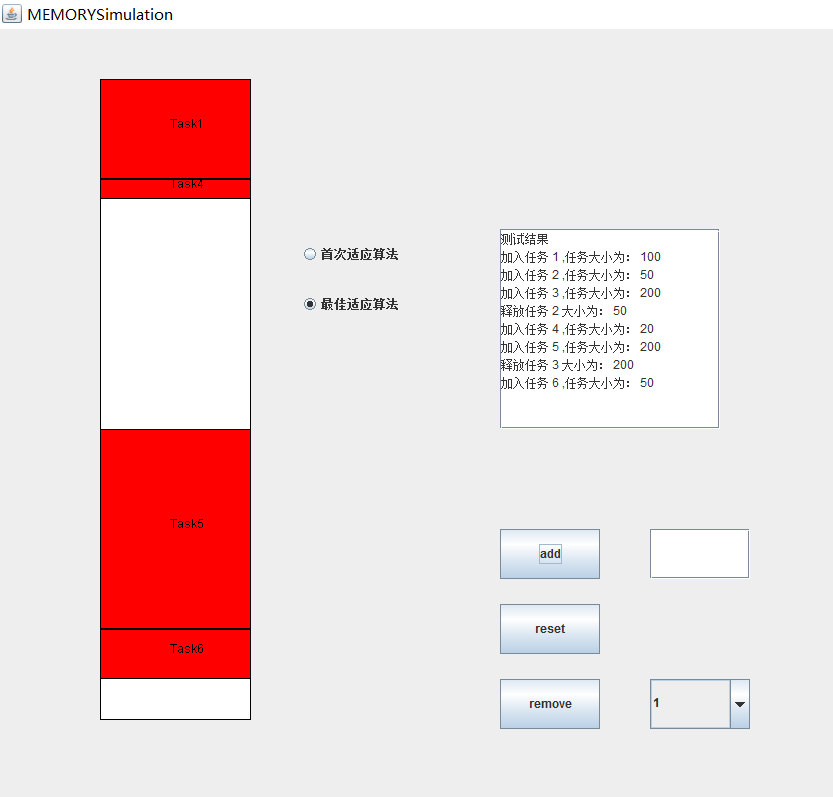




点击reset重置程序



重复上述的步骤，前几步均一样，仅最后添加任务6时，因为是最佳适应算法，所有放置的位置发生改变



**二．项目实现**

**1.界面布局**

本程序是居于java swing库实现的。源文件有五个，Main.java ,Task.java, MemoryRectangle.java, ContentPane.java 和ShowFrame.java 。其中ContentPane.java 是创建一个可视化的界面，并实现整个动态内存操作的核心。改写JPanel ,在其中用三个表taskList, usedList, freeList 来记录任务，占用的内存空间和空余的内存空间。布局上的基本操作便不赘述。基本上有三个button 两个radiobutton 一个rectangle图形 一个scrollpane容器， 一个combox下拉条，一个文本输入框和一个文本显示框。

**2监听事件**

首先对选择算法的两个按钮进行绑定监听事件，我通过设置了一个boolean型的值fristAdapt来标记选择了哪个算法，true为首次适应，false为最佳适应。 之后对addButton和removeButton 进行设置。将addText 文本框中的内容提取出，并以其中的值创建新的任务。将combobox runningTask下拉表与usedList中的每一项任务的ID绑定，可以直接通过runningTask 选择任务的序号进行删除。对reset按钮设置监听事件，功能为将三个表全部清空，并将所有图形界面恢复至初始界面（仅有一个640的内存块），文本框信息清空。

**3.内部逻辑**

对于新添加的任务，首先判断任务大小是否能在空余内存空间中放下，如果可以，便将该任务添加至taskList表中，之后持续访问taskList表，每次都找出表中第一个任务，判断该任务的分配状态，如果是true, 即可以执行setNewTask方法，即将任务放置进内存中去。在放置过程中，我们将遍历整个空闲空间，找到能够存放该任务的空余内存块后，对该内存块的边界进行重置，并将该任务添加进usedList中。之后调用repaint()方法，对矩形块的图形进行重画。（通过改写paintComponent方法,实现绘制一个图形，并用之前存放于usedList中的矩形去填充该图形，通过矩形的上下底的数据，绘制矩形边界，实现图形显示内存的添加。同理，释放内存的方法和上述添加任务的方法大同小异，在释放内存时，我们首先找到任务的Id，并在usedList 中找到该任务，通过比对该任务所占用的内存块的上下边界与剩余的空闲内存块的位置关系，找到相邻或者不相邻的内存块，重置内存块的大小。并在最后从usedList中删掉该任务，并向freeList中添加新的内存块（刚释放的）。对于释放后，我们还有一个额外操作，即对内存块的最适空间寻找，通过找到最小的且适合的空间，这样即可对最佳适应算法进行操作。

**4. 其余文件说明**

ShowFrame.java Frame类， 新建一个contentPane, 并使之显示。

Task.java 任务类，记录每个任务的ID，大小，分配状态

MemoryRectangle.java 内存矩形类，记录内存块的上下底位置，以及内存块的ID

Main.java 主函数，运行操作。