一、需求分析

（1）通过请求页式管理方式中页面置换算法的模拟设计，实现 OPT（最佳置换）、FIFO 算法和 LRU 算法，并计算缺页率。

（2）本程序的运行实际上不用输入测试数据，它能利用随机函数生成测试数据，对于 用户来讲，可以直接观察运行结果，方便使用。

（3）输出的形式是先在窗口中显示生成的随机数，接着转为对应的页号，最后计算出三种算法在不同模拟页框数下的命中率，方便查看和对比，最后统计时通过另一个窗口以表格和折线图的形式给用户，表格中是生成的随机地址序列，和地址对应的页号，命中率和页框数的关系用折线图表示出来。

二、概要设计

1. 用随机数生成函数产生一个“指令将要访问的地址序列”，然后将地址序列变换成相应的页地址流（即页访问序列）。
2. 本程序一共四个模块，分别如下：
3. 随机数产生转换模块

RandomNumber类

ConvertToPageNumber类

1. 算法计算模块

Culculation类

1. 图表统计模块

AccuracyChart类

1. 主程序模块

Main类

1. 主程序流程图

按下开始模拟键，生成随机数

按下产生页号键，转换随机数为页号

按下计算命中率键

FIFO算法

OPT算法

LRU算法

按下统计键，生成表格和折线图

1. 各程序模块之间的层次关系

主程序入口start( )

RandomNumber

ConvertToPageNumber

Calculation

AccuracyChart

（1）Main类是应用程序的入口类，它继承自Application类，并重写了start方法。调用了其他自定义类和方法，如RandomNumber.produce()、ConvertToPageNumber.convert()、Culsulation.calculateOPT()、Culsulation.calculateFIFO()、Culsulation.calculateLRU()等。

（2）AccuracyChart类在Main类中的countButton的事件处理程序中被创建和调用，用于创建并显示一个图表。

5、抽象数据类型的定义

public static double[][] accuracy = new double[4][37];

public static int[][] pageCapacity;

public static int[] addresses;

public static int[] pageAddresses;

三、详细设计

1、程序中定义的抽象数据类型

（1） /\* accuracy[0]、accuracy[1]、accuracy[2]分别存储不同页框数下三种页面置换算法的命中率；accuracy[3]存储FIFO与LRU算法的命中率大小对比结果 \*/

public static double[][] accuracy = new double[4][37];

/\* pageCapacity[ ][ ]存储页框里的页地址信息，每行表示一个页框，第一列存储页地址，第二列表示该页地址在未来是否出现, "是"用1表示，否则置0 \*/

public static int[][] pageCapacity;

（2）public static int[] addresses;//存储产生的随机数

public static int[] pageAddresses;//存储addresses经过转换后产生的页号

2、 随机数产生转换模块

1. RandomNumber类中的方法和功能：

方法：produce（）

输入值：无

返回值：insAddresses //记录模拟指令地址

操作变量：insAdresses[400]

功能：生成400条随机的指令地址，并将其存储在一个整数数组中。

方法实现：首先创建一个大小为400的整型数组instructionAddresses，用于存储随机生成的指令地址。用Math.random( )方法使用非顺序与顺序方法轮流生成随机指令地址。最后返回随机生成的指令地址数组。

伪码：

while(k < 200){

对数组的第一个元素赋值一个 0~199 的随机数

数组的第二个元素等于第一个元素加 1

对数组的第三个元素赋值一个 200~399 的随机数

数组的第四个元素等于第三个元素加 1

K 自身加 2

}

1. ConvertToPageNumber类中的方法和功能 ：

方法：convert( )

输入值：来自RandomNumber类的insAddresses

返回值: pageAddresses

操作变量：pageAddresses[400]

功能：接受一个包含指令地址的整数数组作为参数，将其转换为页号，并返回转换后的页号数组。

方法实现：首先创建一个大小为400的整型数组pageAddresses，用于存储转换生成的页地址（页号）。遍历指令地址数组，将每个指令地址除以10得到对应的页地址（页号），并存储到pageAddresses数组中。如果指令地址小于0或大于399，输出错误信息。

返回转换后的页地址（页号）数组。

伪码：

For(i不超过400，每次循环加1）{

如果产生的数有小于0或者大于399的，判断为错误数据

如果是正确数据，对每个输入值做除以10操作，结果放在返回值数组

}

3 、算法计算模块

这段代码是一个名为Culsulation的Java类，位于calculation包中。它包含了计算页面置换算法OPT、FIFO和LRU的方法。代码还包括一个名为accuracy的数组，用于存储算法的命中率，以及一个名为pageCapacity的数组，用于存储页面地址在页面框中的信息。

输入值：pageAddresses[ ]，pageFrame

返回值：无

操作变量: pageCapacity[ ] [ ] , accuracy[ ][ ]

（1）方法：calculateOPT（）

方法实现：根据OPT算法计算页面置换的命中率。该方法接受页地址数组pageAddresses和页框数pageFrame作为参数。它首先创建pageCapacity数组来存储页框中的页地址信息，并初始化缺页次数pageFault为0。然后，对于每个页地址，它检查是否已存在于页框中，如果存在则继续下一个地址；如果不存在，则检查是否有剩余的用户内存容量，如果有，则将该页地址插入页框中；如果没有剩余内存容量，则根据OPT算法选择要置换的页地址，并将新页地址插入页框中。最后，计算并记录命中率。

伪码：

当前页框数=0

for(在输入的页框数范围内){

对当前页框内的数据置位-1，意为当前页框为空

}

for(不超过 400){

for(在输入的页框数范围内){

如果当前页地址数等于当前页框内的值，则退出循环

}

if(当前页地址数不等于当前页框内的值或者有页框为空){

如果有页框为空，则把当前页地址数放入页框

如果所有页框都有数据，则对当前所有页框内的数与之后的页地址数进行

比较，当匹配到当前的页地址数与某个页框的值相等，则当前的页框会在

未来某个数时会再出现，当匹配到最后一个页框时，意为当前页框的值永

不再使用或下次访问距当前时间最长，该页框的数值被淘汰将被当前的页

地址数代替

}

}

（2）方法：calculateFIFO（）

方法实现：

根据FIFO算法计算页面置换的命中率。与calculateOPT方法类似，该方法也接受页地址数组pageAddresses和页框数pageFrame作为参数，并按照FIFO算法的规则进行页置换。最后，计算并记录命中率。

伪码：

pageCapacity = new int[pageFrame][2]; //用于记录页框内的数据以及滞留时长

for(在输入的页框数范围内){

对当前页框内的数据置位-1，意为当前页框为空

}

for(不超过 400){

for(在输入的页框数范围内){

如果当前页地址数等于当前页框内的值，则所有不为空的页框的滞留时长

加 1，并退出循环

}

if(当前页地址数等于pageFrame) { //需要插入新页地址时

/\*对于在输入的页框范围内

如果当前页框不为空

增加当前页框中有效也地址的滞留时长，当前页地址的滞留时长置为1 \*/

If(当前页地址数等于当前页框内的值){ //当需要置换页地址时

/\*比较选出当前滞留时长最大的页地址，出现相同情况则按顺序选取\*/

/\*所有页框数的滞留时长加 1，被代替的页框数的滞留时长置为 1\*/

增加缺页中断次数

}

}

/\* 计算缺页率=1-缺页次数/400 \*/

}

（3）方法：calculateLRU（）

方法实现：

根据LRU算法计算页面置换的命中率。与前两个方法类似，该方法接受页地址数组pageAddresses和页框数pageFrame作为参数，并按照LRU算法的规则进行页置换。最后，计算并记录命中率。

伪码：

//用于记录页框内的数据以及最新被访问时间

pageCapacity= new int[pageFrame][2];

for(在输入的页框数范围内){

对当前页框内的数据置位-1，意为当前页框为空

}

for(400次循环){

//判断是否已存在该页地址

for(在输入的页框数范围内){

如果当前页地址数等于当前页框内的值，则当前页框的最新被访问时间

置为 0，并退出循环

}

//处理需插入新页地址的情况

if(当前页地址数等于当前页框内的值){

/ \* 对于在输入页框范围内时，如果有页框为空，则把当前页地址数放入页框，对其他所有不为空的页框增加有效页地址的最新被访问时间值，当前页地址的最新被访问时间值置为0 \*/

}

//如果所有页框都有数据

If(当前页地址数等于当前页框内的值）{

/ \* 比较所有页框的最新被访问时间，出现相同情况则按顺序选取,最新被访问

时间最大的页框数将被当前当前页地址数替代 \*/

/ \* 把当前页框的最新被访问时间置为 0，其他页框的最新被访问时间加 1 \*/

}

/\* 计算缺页率=1-缺页次数/400 存入accuracy[2][pageframe-4] \*/

}

1. 图表统计模块

（1） AccuracyChart类

AccuracyChart类是一个JavaFX应用程序，用JavaFX的LineChart创建一个准确率图表来显示不同页面置换算法（OPT、FIFO、LRU）在不同页框数下的命中率和用ListView组件来显示一个包含地址和页号的列表。

程序的主要部分是重写的start()方法，它创建了一个Stage（舞台）和一个Scene（场景）。场景中，创建了一个HBox作为根节点，其中包含了一个VBox作为图表容器和一个VBox作为ListView容器。LineChart、ListView以及一些按钮和滚动面板被添加到对应的容器中。最后，将场景设置到舞台并显示出来。

LineChart部分包括创建X轴和Y轴，创建不同算法的数据系列，将数据系列添加到LineChart中，并添加了一些鼠标事件监听器，用于实现拖拽和缩放功能。

ListView部分创建了一个包含地址和页号的ObservableList，并将其设置到ListView中。为ListView设置了自定义的单元格工厂，用于显示每个单元格的地址和页号。

详细设计：

创建了一个带有X轴和Y轴的折线图。

创建了三个数据系列：OPT、FIFO和LRU，并将它们添加到折线图中。

创建了一个ListView组件，并使用ObservableList填充了ListView的项。每个项显示了页号和随机地址值序列。将折线图和ListView添加到场景中，并显示出来。实现了鼠标事件的监听器，包括拖动和缩放功能，以便用户可以对图表进行交互操作。添加了隐藏OPT、FIFO和LRU数据系列的按钮，并实现了按钮的点击事件处理程序。

1. 主程序模块

Main类

Main类是一个JavaFX应用程序的主类，用于模拟页面置换算法的运行并展示结果。

主要功能如下：

创建包含文本区域和四个按钮的窗口，实现与用户之间的交互、按下某个步骤的按钮即可以展示该步骤的结果（如随机生成的指令地址、转换出来的页号、三种算法在不同页框下的命中率）

调用模块：

RandomNumber的produce方法、ConvertToPageNumber的convert方法、

Calculation的calculateOPT、calculateFIFO、calculateLRU方法、

详细设计：

在JavaFX应用程序中创建了一个窗口，包含一个文本区域和几个按钮。点击"开始模拟"按钮会生成400个随机地址，并将其显示在文本区域中。点击"产生页号"按钮会将随机地址转换为相应的页号，并将其显示在文本区域中。点击"计算命中率"按钮会计算不同页框数下的OPT、FIFO和LRU算法的命中率，并将结果显示在文本区域中。点击"统计"按钮会打开一个新窗口，显示不同页框数下的命中率统计图表。这段代码的功能主要集中在start()方法中。它创建了一个包含按钮和文本区域的垂直布局容器，并为每个按钮设置了相应的事件处理程序。整个应用程序的界面布局使用了JavaFX的布局容器，包括VBox和HBox，以及滚动面板（ScrollPane）来处理文本区域的滚动。

四、调试分析

（1）调试过程中遇到的问题是如何解决的以及对设计与实现的讨论和分析：

刚开始在编写三种页面置换算法时存在一些逻辑错误，后来通过查阅他人的计算方法，找到了漏洞并进行了修改。本来javafx使用的不是很熟悉，比如滚动面板、ListView的自定义单元格工厂，在这次实验之前都对此不了解，都是这次实验给了我学习的机会。比如尝试使用滚动面板时，忘记将滚动面板添加到父容器中，导致一直没有显示。

（2）算法的时间复杂性(包括基本操作和其他算法的时间复杂性的分析)和改进设想

对于本次opt，lru，fifo的三个算法时间复杂度均为O(N^2)，对于算法改进的设想有：比如优化页地址查找目前的算法在判断页地址是否存在时需要进行多次遍历，可以考虑使用更高效的数据结构来优化查找过程，例如使用哈希表或二叉查找树来存储页地址，以减少查找的时间复杂度。

（3）设计过程的经验和体会

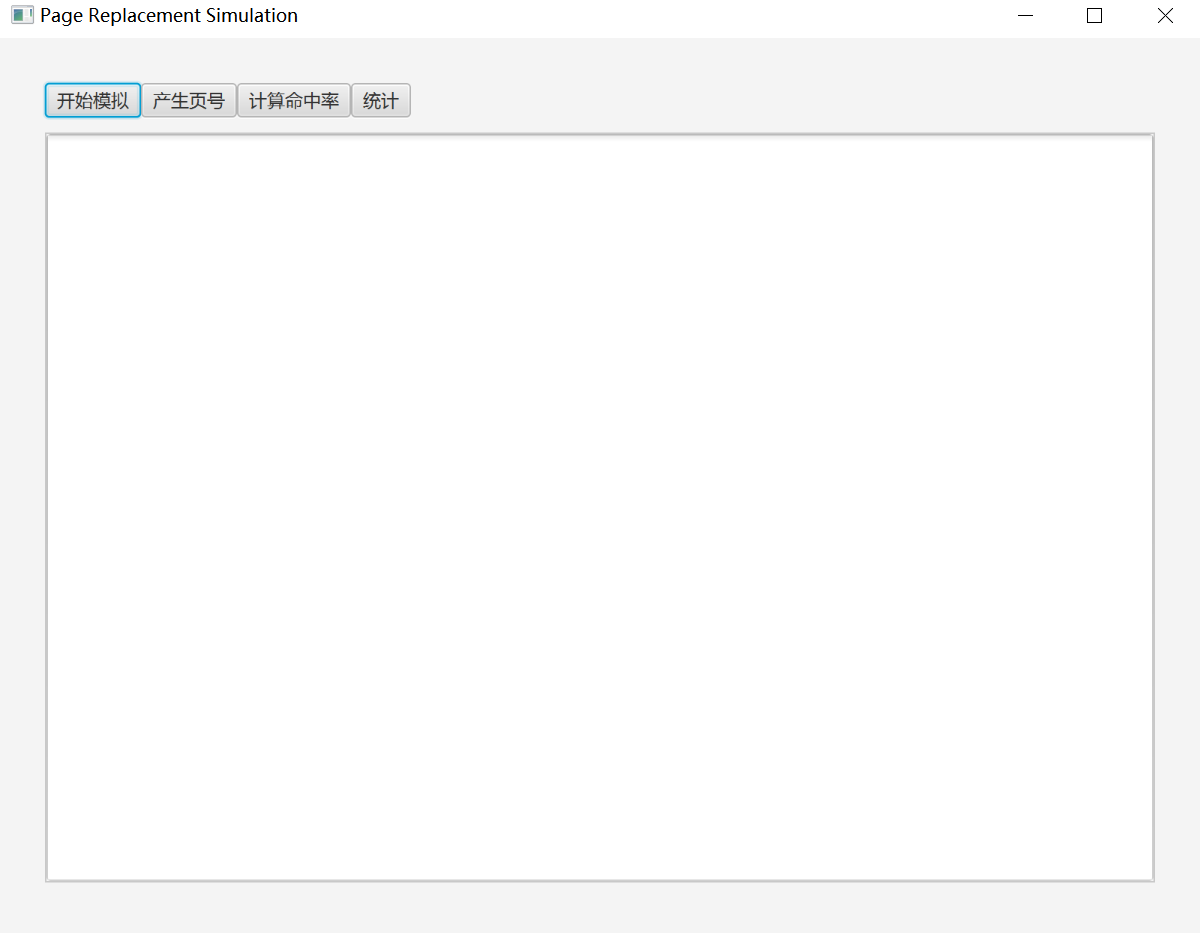
当我拿到这个题目时，发现难度并不是特别的高，老师在课堂上有把这些算法详细地介绍给我们，课后作业也有做过，就我个人而言，由于对 javafx 的不熟悉，所以打算拿简单的题目练练手。一开始设计时，按题目要求一步步完成，先生成随机的模拟页地址数，再计算三种方法的命中率，最初设计时只把计算的结果在执行的黑窗口中显示，过于单调，也不能完成老师要求的折线图显示对比，无法直观地看到各算法之间命中率的比较。后来经过查阅资料学习，决定尝试使用 javafx 的表格组件，最后找到了合适的方法把数据放进去。通过表格和折线图的方式，用户可以更加直观地看到随机生成的地址序列和其对应的页地址数、三种算法的命中率以及命中率之间比较，并且为三种命中率折线提供了显示与隐藏按钮，能更好地实现横向对比和纵向对比。但是我觉得刚开始main函数的窗口按钮实现的不够好，必须按顺序按下四个按钮，并且没有设置操作提醒，这个我会在以后的课程设计当中加以改进。通过本次课程设计，提高了我的编程能力，对页面置换算法也更加地熟悉，但是在 javafx当中还有很多我没有使用过的组件，所以在今后的课设实验当中，我会尝试使用更多的组件，努力让程序设计更加完美。

五、用户使用说明

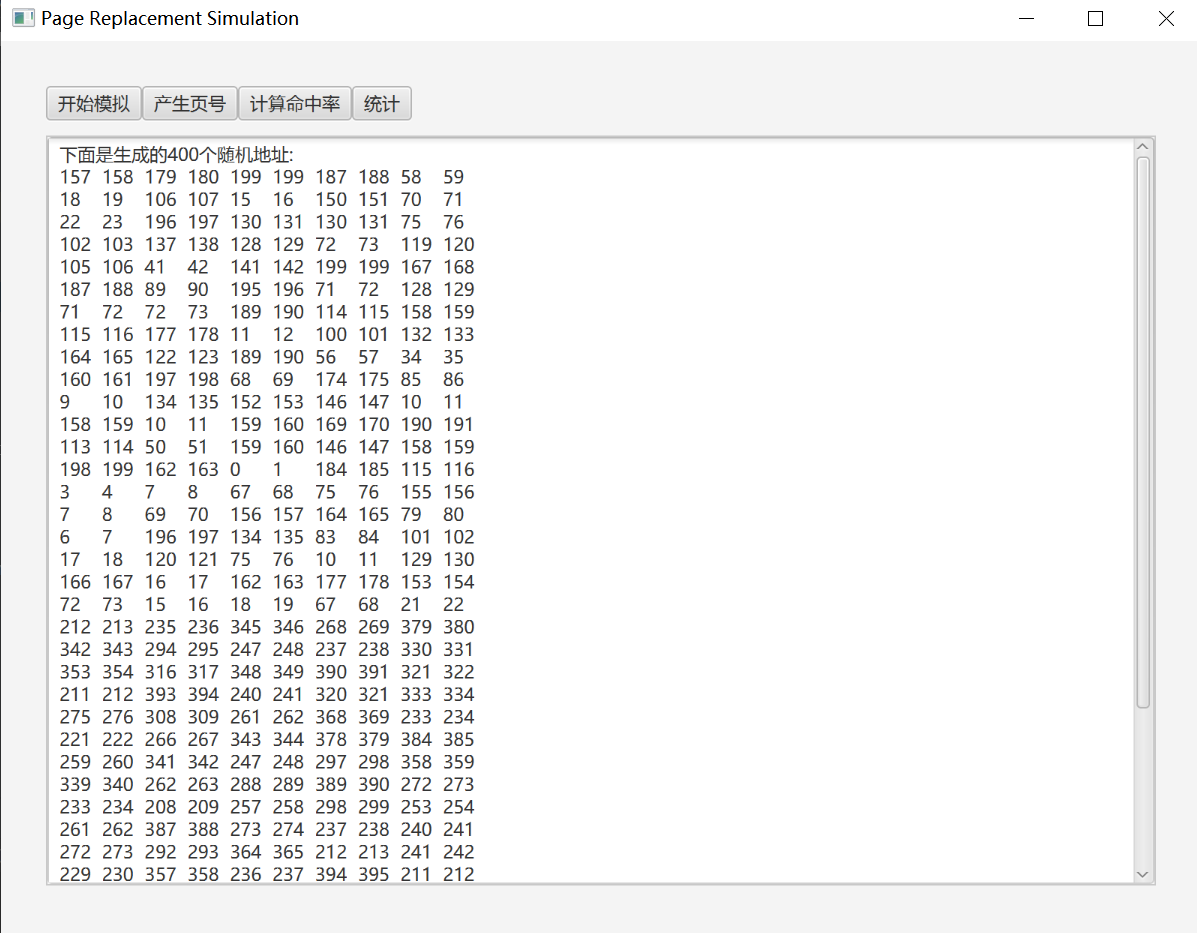
本次课程设计的题目为模拟实现 OPT（最佳置换）、FIFO 和 LRU 算法，并计算命中率。程序开始运行时，并没有产生对应的数据，而是打开一个含有四个步骤的按钮的窗口，用户需要按下开始模拟按钮，程序才会产生随机地址序列，按下产生页号按钮后计算页号、按下计算命中率按钮后得到三种算法的命中率，均可以使用滚动条滚动查看，按下统计按钮后打开新窗口生成统计折线图和表格，表格中的数据可以通过滚动条滚动查看，折线图中可以按下hide按钮进行数据类型的隐藏和显示，可以拖动滚轮放大折线图便于观察具体数据。当用户再次按下开始模拟按钮时，随机的页地址流会重新生成，重复按步骤顺序按下按钮对应生成的三种算法的命中率也会发生改变。按步骤顺序按下按钮可以方便对比和检查每一步所得到的数据。

六、测试与运行结果

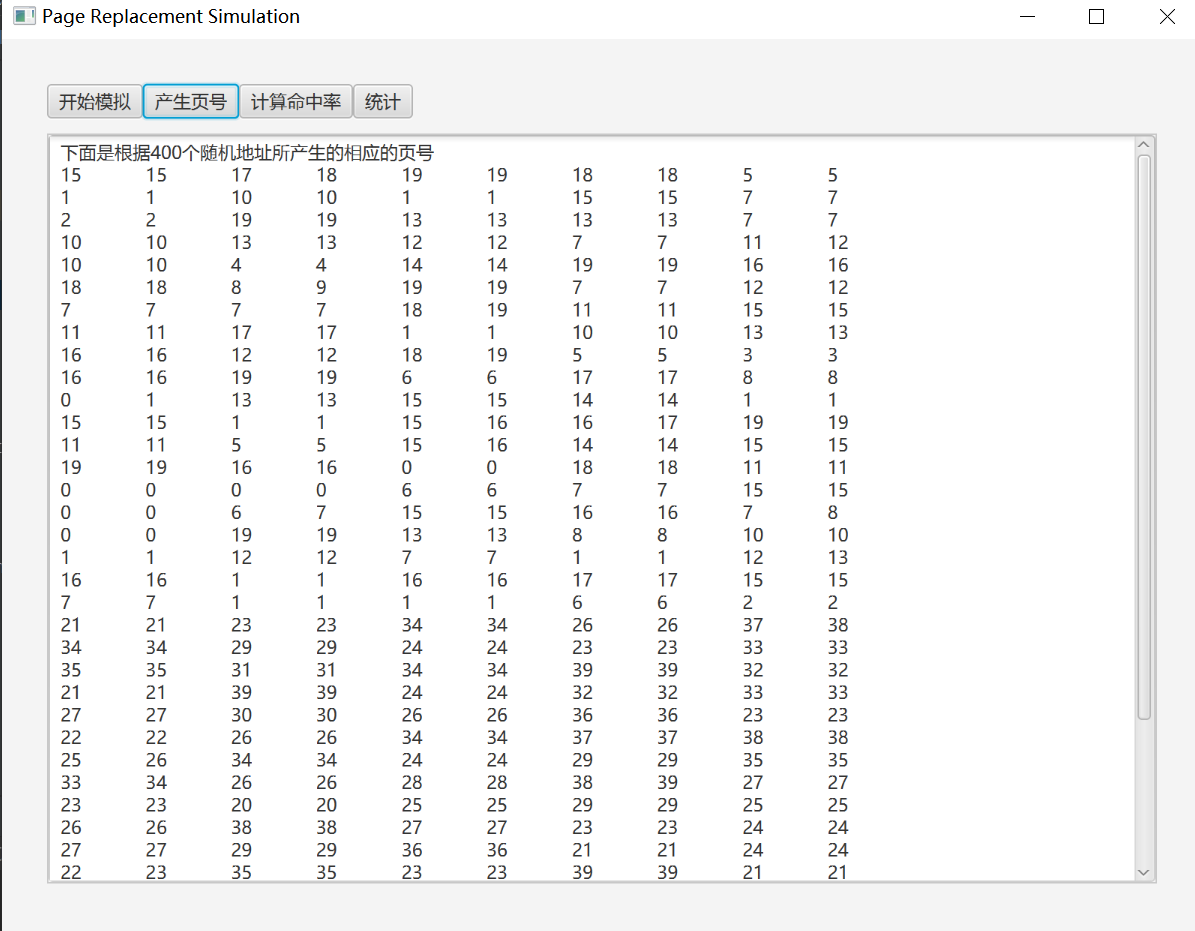
初始化界面



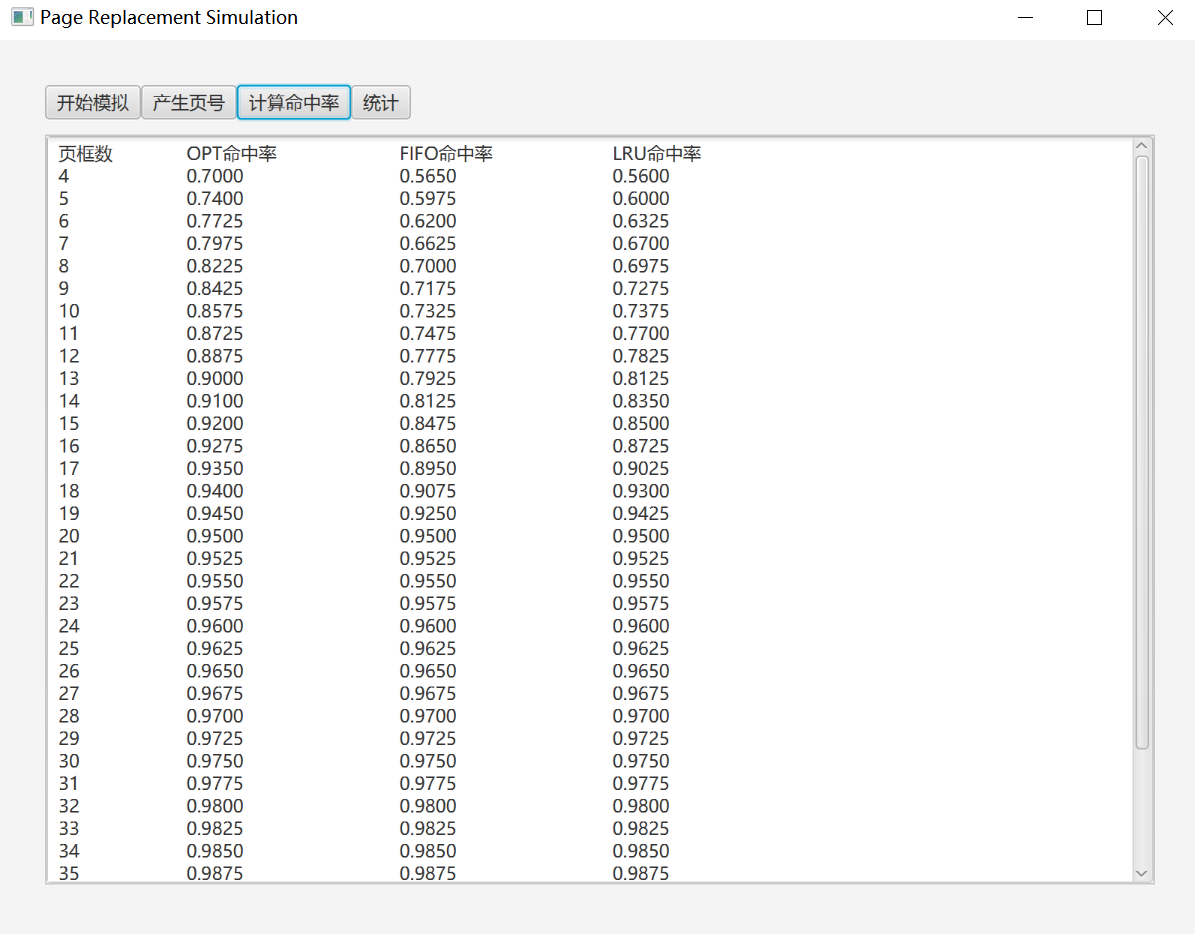
开始模拟：



产生页号：



计算命中率



统计：

