Sunday, June 15, 2014

Title

Subject

- a) My personal information
- b) Expatiation
- c) Some cases
- d) The development detail and personal idea(A brief introduction.)

Part A:

Something about the developer:

name	:	谢志杰
student number	:	1352975
mobile phone number	:	15221356551
e-mail address	:	xie510894496@gmail.com/1352975@tongji.edu.cn

Part B:

使用动规划解决背包问题，算法细节：

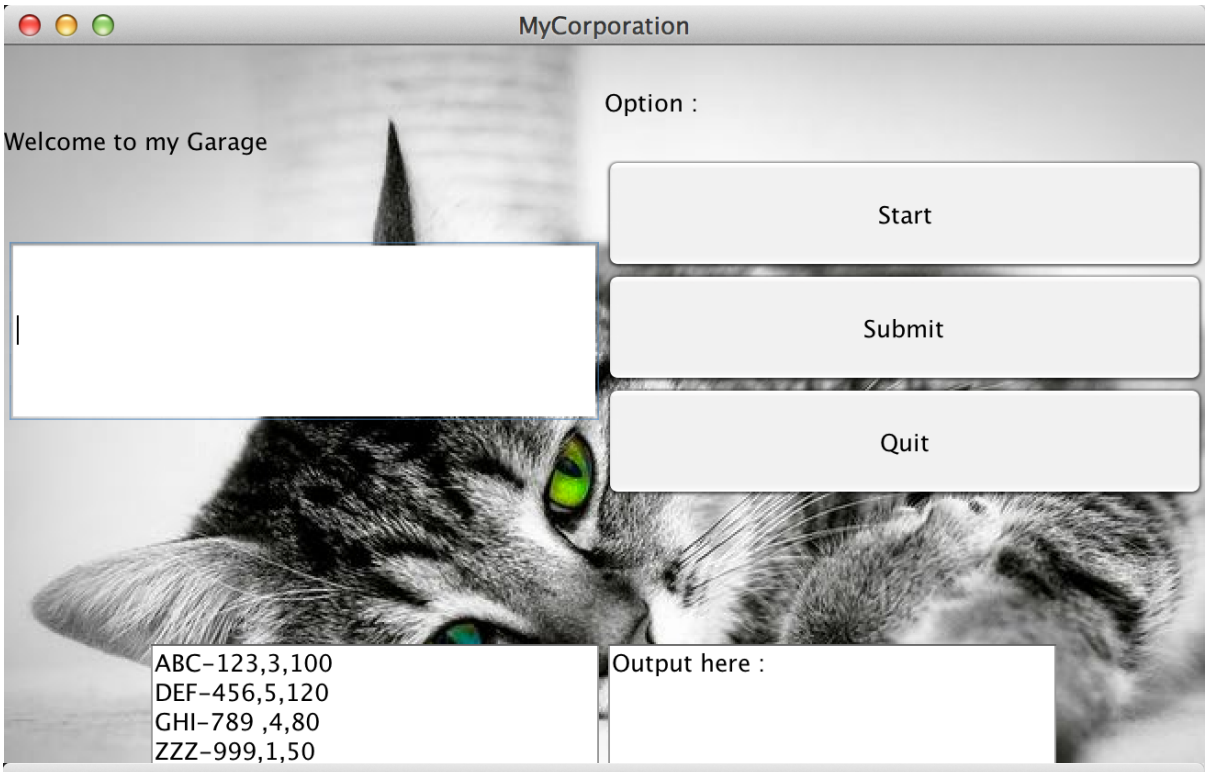
使用递归计算出DP矩阵，然后得出最大值，同时得出具体是哪些车辆。

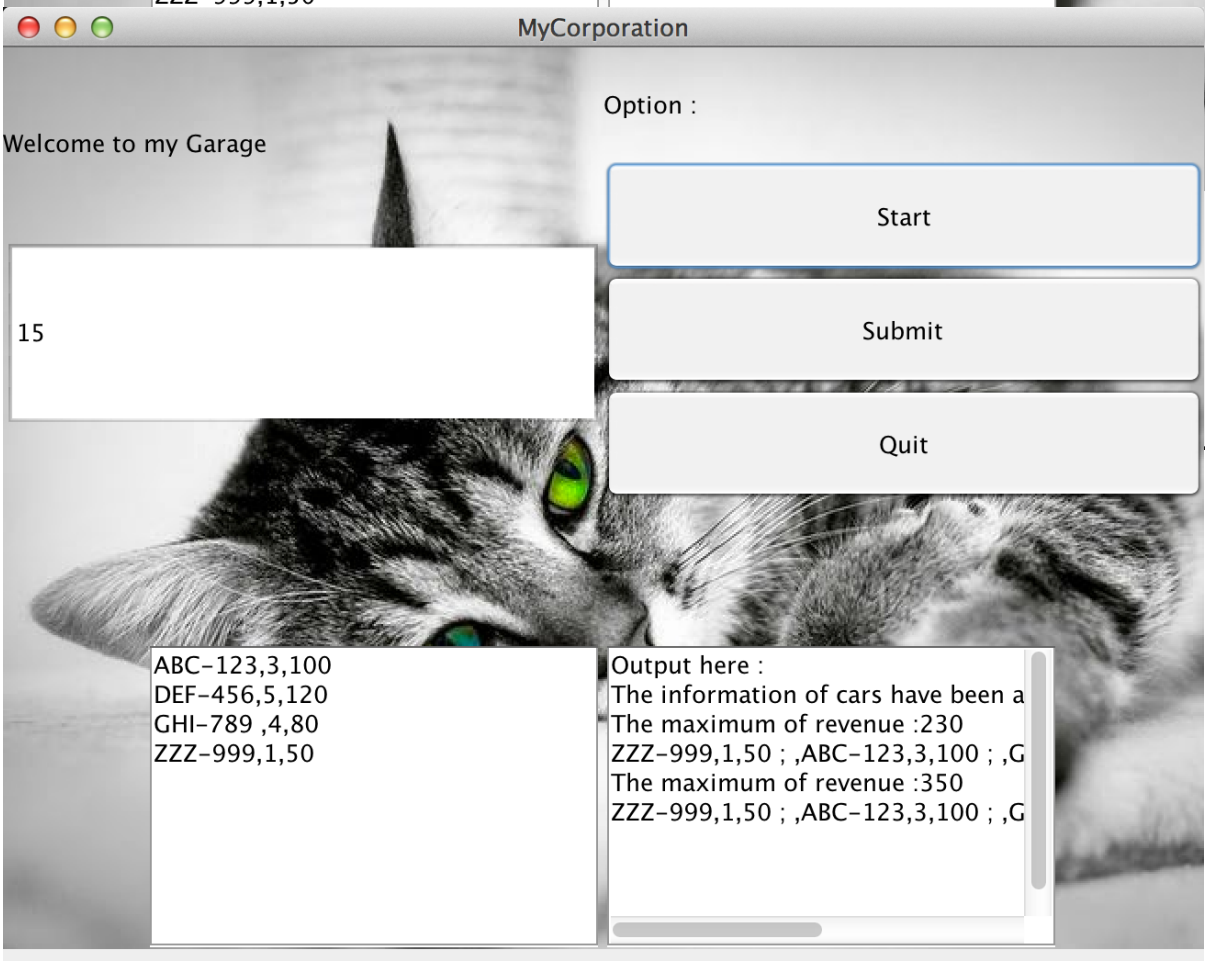
```
141     private ArrayList<Integer> find(int number, int availableHours, int[] allCars, int[] time, int[][] c) {
142         int remainTime = availableHours;
143         ArrayList<Integer> result = new ArrayList<Integer>();
144 
145         for(int i = number; i > 0; --i) {
146             for (int j = number - 1; j >= 0; --j) {
147                 if (c[i][remainTime] - allCars[j] >= 0 && remainTime - time[j] >= 0) {
148                     if ((c[i][remainTime] - allCars[j]) == c[i - 1][remainTime - time[j]]){
149                         result.add(new java.lang.Integer(j));
150                         remainTime = remainTime - time[j];
151                         break;
152                     }
153                 }
154             }
155         }
156 
157         return result;
158 
159     }
160 
161     /**
162      * The algerithm of knapsack problem.
163      * @param  n       [description]
164      * @param  w       [description]
165      * @param  allCars [description]
166      * @param  time    [description]
167      * @param  c       [description]
168      * @return         [description]
169      */
170     private int knapsack(int n, int w, int[] allCars, int[] time, int[][] c) {
171         if (w <  0) return -100000;
172         if (n == 0) return 0;
173 
174         if (c[n][w] != 0) return c[n][w];
175 
176         return c[n][w] = max(
177             knapsack(n - 1, w - time[n - 1], allCars, time, c) + allCars[n - 1],
178             knapsack(n - 1, w, allCars, time, c),
179             allCars,
180             n - 1
181         );
182     }
```

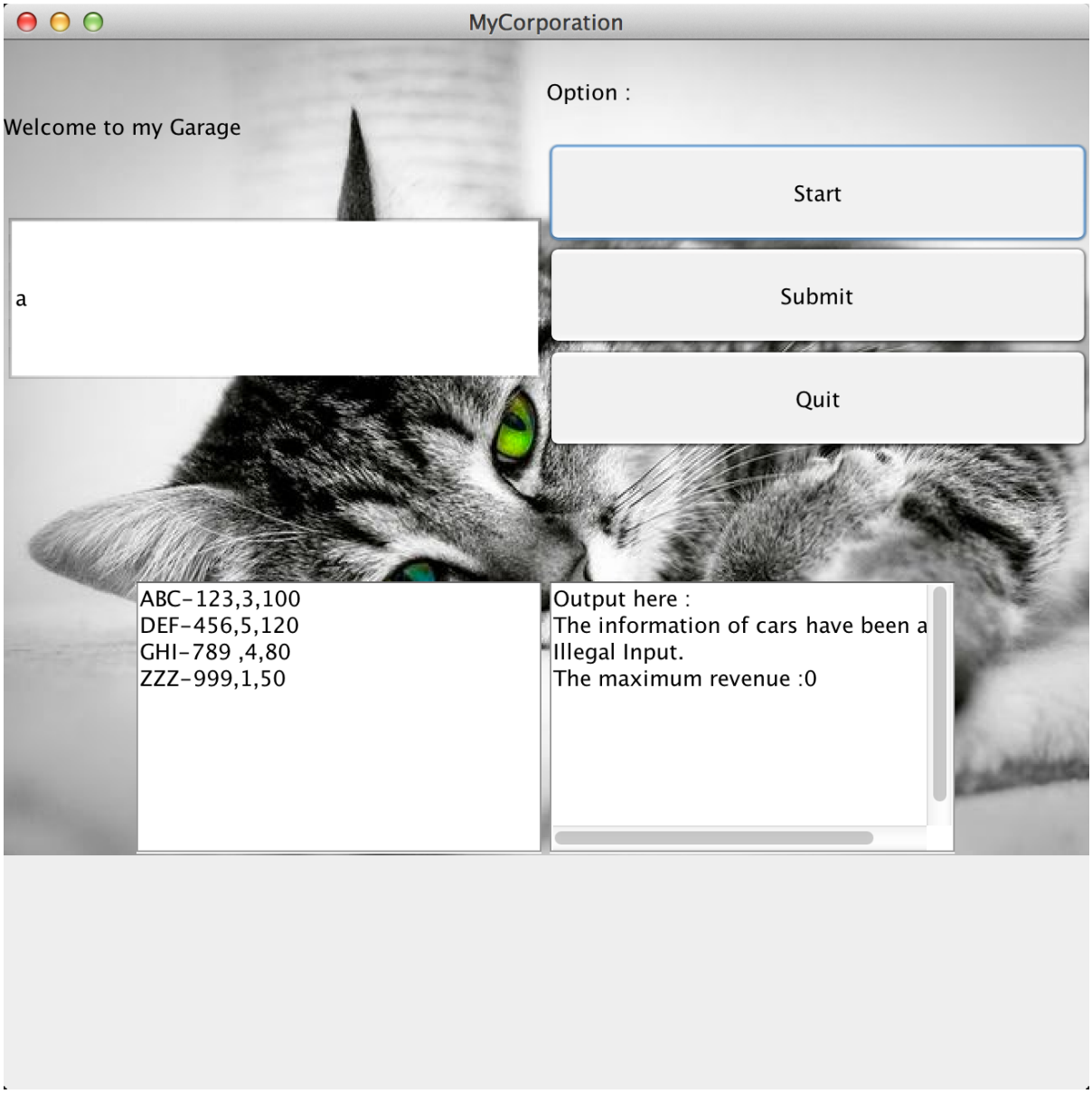
```
230
 0  0  0  0  0  0  0  0  0

 0  50  0  50  50  50  0  0  50

 0  0  0  100  150  0  0  0  150

 0  0  0  100  0  0  0  0  230

 0  0  0  0  0  0  0  0  230

150
 0  0  0  0  0  0

 0  50  50  0  0  50

 0  50  0  0  0  150

 0  0  0  0  0  150

 0  0  0  0  0  150

350
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0

 0  0  0  50  0  0  50  50  50  0  50  50  50  0  0  50

 0  0  0  0  0  0  150  0  0  0  150  150  0  0  0  150

 0  0  0  0  0  0  0  0  0  0  230  0  0  0  0  230

 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  350
```

PS：在命令行下使用“java -jar MyCorporation”可以获得上述命令行输出。

Part C: some cases are as follow:







示例数据和格式已经默认给出，用户可以根据自己的需求修改。

Sunday, June 15, 2014

Part D:

开发历程：

1、第一天中午：算法部分完成，算法未进行空间复杂度的优化，实际上，可以进一步缩小DP数组的大小。

2、第一天晚上：GUI与底层实现完成，软件打包。

实现细节：

1、Swing的组件使用GridLayout布局。

2、添加了一个背景图片，美化界面。jpg也一起打包上交了。

3、对不合法输入进行了异常处理。

4、由于界面最下方的两个文本域使用FlowLayout布局所以当用户变更界面大小的时候可能会有不合理的布局。

另外，由于这次的项目规模较小。项目文档可能略显单薄。