

# 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	:	<u><a href="mailto:xie510894496@gmail.com/1352975@tongji.edu.cn">xie510894496@gmail.com/1352975@tongji.edu.cn</a></u>

## 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 algorithm 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 50 0 0 50 50 50 50 50 50
    0 0 0 0 0 0 150 0 0 150 150
    0 0 0 0 0 0 0 230 0 0 0 230
    0 0 0 0 0 0 0 0 0 0 0 350

```

```

141 private ArrayList<
142     int remanTime
143     ArrayList<int>
144
145     for(int i = 0;
146         if (i < 1)
147             if
148
149
150
151
152     }
153     }
154     }
155     }
156
157     return result;
158
159 }
160
161
162
163
164
165
166
167
168
169
170
171 private int knapsack
172     if (w == 0) ret
173     if (n == 0) ret
174
175     if (c[n][w] != 0)
176         return c[n][w];
177     if (w < 0) return 0;
178     if (n < 0) return 0;
179
180     int max = 0;
181     for (int i = 0; i < cars.length; i++) {
182         int w1 = w - cars[i].weight;
183         int v1 = cars[i].value;
184         int w2 = w;
185         int v2 = 0;
186         if (w1 < 0) w1 = 0;
187         if (w2 < 0) w2 = 0;
188         int v3 = knapsack(n-1, w1);
189         int v4 = knapsack(n-1, w2);
190         int v5 = knapsack(n-1, w3);
191         int v6 = knapsack(n-1, w4);
192         int v7 = knapsack(n-1, w5);
193         int v8 = knapsack(n-1, w6);
194         int v9 = knapsack(n-1, w7);
195         int v10 = knapsack(n-1, w8);
196         int v11 = knapsack(n-1, w9);
197         int v12 = knapsack(n-1, w10);
198         int v13 = knapsack(n-1, w11);
199         int v14 = knapsack(n-1, w12);
200         int v15 = knapsack(n-1, w13);
201         int v16 = knapsack(n-1, w14);
202         int v17 = knapsack(n-1, w15);
203         int v18 = knapsack(n-1, w16);
204         int v19 = knapsack(n-1, w17);
205         int v20 = knapsack(n-1, w18);
206         int v21 = knapsack(n-1, w19);
207         int v22 = knapsack(n-1, w20);
208         int v23 = knapsack(n-1, w21);
209         int v24 = knapsack(n-1, w22);
210         int v25 = knapsack(n-1, w23);
211         int v26 = knapsack(n-1, w24);
212         int v27 = knapsack(n-1, w25);
213         int v28 = knapsack(n-1, w26);
214         int v29 = knapsack(n-1, w27);
215         int v30 = knapsack(n-1, w28);
216         int v31 = knapsack(n-1, w29);
217         int v32 = knapsack(n-1, w30);
218         int v33 = knapsack(n-1, w31);
219         int v34 = knapsack(n-1, w32);
220         int v35 = knapsack(n-1, w33);
221         int v36 = knapsack(n-1, w34);
222         int v37 = knapsack(n-1, w35);
223         int v38 = knapsack(n-1, w36);
224         int v39 = knapsack(n-1, w37);
225         int v40 = knapsack(n-1, w38);
226         int v41 = knapsack(n-1, w39);
227         int v42 = knapsack(n-1, w40);
228         int v43 = knapsack(n-1, w41);
229         int v44 = knapsack(n-1, w42);
230         int v45 = knapsack(n-1, w43);
231         int v46 = knapsack(n-1, w44);
232         int v47 = knapsack(n-1, w45);
233         int v48 = knapsack(n-1, w46);
234         int v49 = knapsack(n-1, w47);
235         int v50 = knapsack(n-1, w48);
236         int v51 = knapsack(n-1, w49);
237         int v52 = knapsack(n-1, w50);
238         int v53 = knapsack(n-1, w51);
239         int v54 = knapsack(n-1, w52);
240         int v55 = knapsack(n-1, w53);
241         int v56 = knapsack(n-1, w54);
242         int v57 = knapsack(n-1, w55);
243         int v58 = knapsack(n-1, w56);
244         int v59 = knapsack(n-1, w57);
245         int v60 = knapsack(n-1, w58);
246         int v61 = knapsack(n-1, w59);
247         int v62 = knapsack(n-1, w60);
248         int v63 = knapsack(n-1, w61);
249         int v64 = knapsack(n-1, w62);
250         int v65 = knapsack(n-1, w63);
251         int v66 = knapsack(n-1, w64);
252         int v67 = knapsack(n-1, w65);
253         int v68 = knapsack(n-1, w66);
254         int v69 = knapsack(n-1, w67);
255         int v70 = knapsack(n-1, w68);
256         int v71 = knapsack(n-1, w69);
257         int v72 = knapsack(n-1, w70);
258         int v73 = knapsack(n-1, w71);
259         int v74 = knapsack(n-1, w72);
260         int v75 = knapsack(n-1, w73);
261         int v76 = knapsack(n-1, w74);
262         int v77 = knapsack(n-1, w75);
263         int v78 = knapsack(n-1, w76);
264         int v79 = knapsack(n-1, w77);
265         int v80 = knapsack(n-1, w78);
266         int v81 = knapsack(n-1, w79);
267         int v82 = knapsack(n-1, w80);
268         int v83 = knapsack(n-1, w81);
269         int v84 = knapsack(n-1, w82);
270         int v85 = knapsack(n-1, w83);
271         int v86 = knapsack(n-1, w84);
272         int v87 = knapsack(n-1, w85);
273         int v88 = knapsack(n-1, w86);
274         int v89 = knapsack(n-1, w87);
275         int v90 = knapsack(n-1, w88);
276         int v91 = knapsack(n-1, w89);
277         int v92 = knapsack(n-1, w90);
278         int v93 = knapsack(n-1, w91);
279         int v94 = knapsack(n-1, w92);
280         int v95 = knapsack(n-1, w93);
281         int v96 = knapsack(n-1, w94);
282         int v97 = knapsack(n-1, w95);
283         int v98 = knapsack(n-1, w96);
284         int v99 = knapsack(n-1, w97);
285         int v100 = knapsack(n-1, w98);
286         int v101 = knapsack(n-1, w99);
287         int v102 = knapsack(n-1, w100);
288         int v103 = knapsack(n-1, w101);
289         int v104 = knapsack(n-1, w102);
290         int v105 = knapsack(n-1, w103);
291         int v106 = knapsack(n-1, w104);
292         int v107 = knapsack(n-1, w105);
293         int v108 = knapsack(n-1, w106);
294         int v109 = knapsack(n-1, w107);
295         int v110 = knapsack(n-1, w108);
296         int v111 = knapsack(n-1, w109);
297         int v112 = knapsack(n-1, w110);
298         int v113 = knapsack(n-1, w111);
299         int v114 = knapsack(n-1, w112);
300         int v115 = knapsack(n-1, w113);
301         int v116 = knapsack(n-1, w114);
302         int v117 = knapsack(n-1, w115);
303         int v118 = knapsack(n-1, w116);
304         int v119 = knapsack(n-1, w117);
305         int v120 = knapsack(n-1, w118);
306         int v121 = knapsack(n-1, w119);
307         int v122 = knapsack(n-1, w120);
308         int v123 = knapsack(n-1, w121);
309         int v124 = knapsack(n-1, w122);
310         int v125 = knapsack(n-1, w123);
311         int v126 = knapsack(n-1, w124);
312         int v127 = knapsack(n-1, w125);
313         int v128 = knapsack(n-1, w126);
314         int v129 = knapsack(n-1, w127);
315         int v130 = knapsack(n-1, w128);
316         int v131 = knapsack(n-1, w129);
317         int v132 = knapsack(n-1, w130);
318         int v133 = knapsack(n-1, w131);
319         int v134 = knapsack(n-1, w132);
320         int v135 = knapsack(n-1, w133);
321         int v136 = knapsack(n-1, w134);
322         int v137 = knapsack(n-1, w135);
323         int v138 = knapsack(n-1, w136);
324         int v139 = knapsack(n-1, w137);
325         int v140 = knapsack(n-1, w138);
326         int v141 = knapsack(n-1, w139);
327         int v142 = knapsack(n-1, w140);
328         int v143 = knapsack(n-1, w141);
329         int v144 = knapsack(n-1, w142);
330         int v145 = knapsack(n-1, w143);
331         int v146 = knapsack(n-1, w144);
332         int v147 = knapsack(n-1, w145);
333         int v148 = knapsack(n-1, w146);
334         int v149 = knapsack(n-1, w147);
335         int v150 = knapsack(n-1, w148);
336         int v151 = knapsack(n-1, w149);
337         int v152 = knapsack(n-1, w150);
338         int v153 = knapsack(n-1, w151);
339         int v154 = knapsack(n-1, w152);
340         int v155 = knapsack(n-1,
```

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:

MyCorporation

Welcome to my Garage

Option :

Start

Submit

Quit

ABC-123,3,100  
DEF-456,5,120  
GHI-789 ,4,80  
ZZZ-999,1,50

Output here :

MyCorporation

Welcome to my Garage

Option :

Start

Submit

Quit

15

ABC-123,3,100  
DEF-456,5,120  
GHI-789 ,4,80  
ZZZ-999,1,50

Output here :  
The information of cars have been a  
The maximum of revenue :230  
ZZZ-999,1,50 ; ,ABC-123,3,100 ; ,G  
The maximum of revenue :350  
ZZZ-999,1,50 ; ,ABC-123,3,100 ; ,G

MyCorporation

Welcome to my Garage

Option :

Start

Submit

Quit

a

ABC-123,3,100  
DEF-456,5,120  
GHI-789,4,80  
ZZZ-999,1,50

Output here :  
The information of cars have been a  
Illegal Input.  
The maximum revenue :0

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。

## 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。