Computer Science 20

Final Project

Person of Interest

Zhantong Zhang
(Tony)

Janurary 9, 2013
Western Canada High School

LICENSE

Contents

Part I

Copy of the Project Proposal

The original signed proposal is attached in the binder; the following is a *copy* of the proposal:

Person of Interest
Project Proposal

Tony Zhang

December 9, 2013

1 Description

This project in Java is to build a role-playing game for users over 14 years old based on skills learned in Computer Science 20. In this game, the user is a special agent working for a secret organization to ensure the world security. The user needs to act accordingly toward different targets/missions with his/her knowledge. The mission is determined by the person(s) of interest (POI). POI could be victim(s) or criminal(s). The user should target and monitor POI to prevent the crime(s). The organization has a rank from *Constable* to *Commissioner*, although the user can achieve *Superintendent* at the most. The user would move in the virtual world and solve puzzles and problems. There are missions that may cause the user's death in the virtual world, and, the user should not do things that kill him/her or the game is over. The objective for the user is to become an *Inspector* for a specific region and ensure the security for six months, then, the user becomes a *Superintendent* finishing the game. The game is designed to be over in less than four hours.

1.1 Features

The project shall have following features:

- Let the user choose the base location (Canada for Americas, China for Asia-Pacific and Middle East or France for Europe and Africa).
- Generate POI mission, specifying location, objectives, potential dangers
- Let the user accept POI mission.
- Provide transportation to the POI.
- The user can arrest/protect the POI according to POI's identity.
- The user need to solve puzzles/problems to find the POI.
- There is a point system to measure the user's ability.
- Determine if a mission is success, complete or failed.

- Promotion according to the point system; the rank for the user includes (from lowest to highest) *Constable*, *Corporal*, *Sergeant*, *Staff Sergeant*, *Sergeant Major*, *Inspector*. The higher rank includes *Superintendent*, *Deputy Commissioner* and *Commissioner*.
- The user will follow the order from the higher officers and may choose to follow suggestions of the lower/same officers.
- The user can save his/her character and mission, but the user need to restart his/her mission if come back. Saving the mission progress may be beyond current skills.

The project *may* have following features to enhance user experience:

- The mission progress can be saved.
- The mission can have weight according to the difficulty.
- The user can use firearms and the ability to use firearms is improving in the missions.
- There can be a mode of learning. The rank for the user would be *Cadet*.
- There can be more complex plot like undercover operations.
- The user can change the location after.
- The user can have a partner to help him/her.
- There can be more than one way to finish the game with different outcomes.

2 Timeline

December 8, 2013	Finish the proposal and analysis
December 9, 2013	Start preliminary design
December 10 - 11, 2013	Finish the design, the welcome page and location choice
December 11 - 12, 2013	Finish the loading and saving
December 14 - 15, 2013	Finish the POI generation
December 19, 2013	Finish puzzles/problems
December 22, 2013	Finish the mission procedures
December 25, 2013	Finish determining if the mission is successful
December 27, 2013	Finish the promotion system
December 28, 2013	Finish the POI generation
December 31, 2013	Finish improved user interface and start fixing minor errors
January 1 - 5, 2013	Fix problems and add features if time permitted
January 5, 2014	Finish all things & Start final test
January 8, 2014	Complete

3 Skills Used

This project is written in Java. The basic control structures, variables and some objects are presented through the whole program. The following specific programming skills learned in the course will be used in the project:

- Random number generation is used in:
  - the generation of POI missions
  - the determination of random accidents
  - brute-forcing solving puzzles
- File input & output (including `Serializable` object file) is used in:
  - saving and loading games
- Methods is used extensively in:
  - the user's movement in the virtual world
  - the change of user's status
  - generating, solving and checking puzzles and problems
  - determining the user's outcomes
  - determining the mission's status
  - saving and loading games
  - several decision making like suggestions and orders
- Class is used in:
  - the user's status, name, location, etc.
  - the mission's status, puzzles, etc.
  - POI's properties including name, location, etc.
  - the superiors/subordinates
  - the location's properties
- Array is used extensively in:
  - the ranks of the organization
  - timing for the mission
  - POIs list

4 Success Indicators

When this project is done successfully, it would have a welcome page and a menu that let player to choose to start or load saved game. The user will then be required to create his/her character with name and location. The user will start from a constable. A POI mission is generated having the POI, the location and the dangers. He/she will get a mission and complete it. In the mission, the user need to solve puzzles and problems related to the POI in order to complete the mission and get points. The points are used to promote the user. The user will get another mission and complete it. The mission will have random accidents that can kill/harm the user. The user will get more missions until he/she becomes an inspector. As an inspector, the user will complete a big mission for 6 months. After the successful completion, the user wins and the game over. The user can save his/her character and the mission he/she works on. But, the mission may not be saved as its progress.

Teacher's Approval

Part II

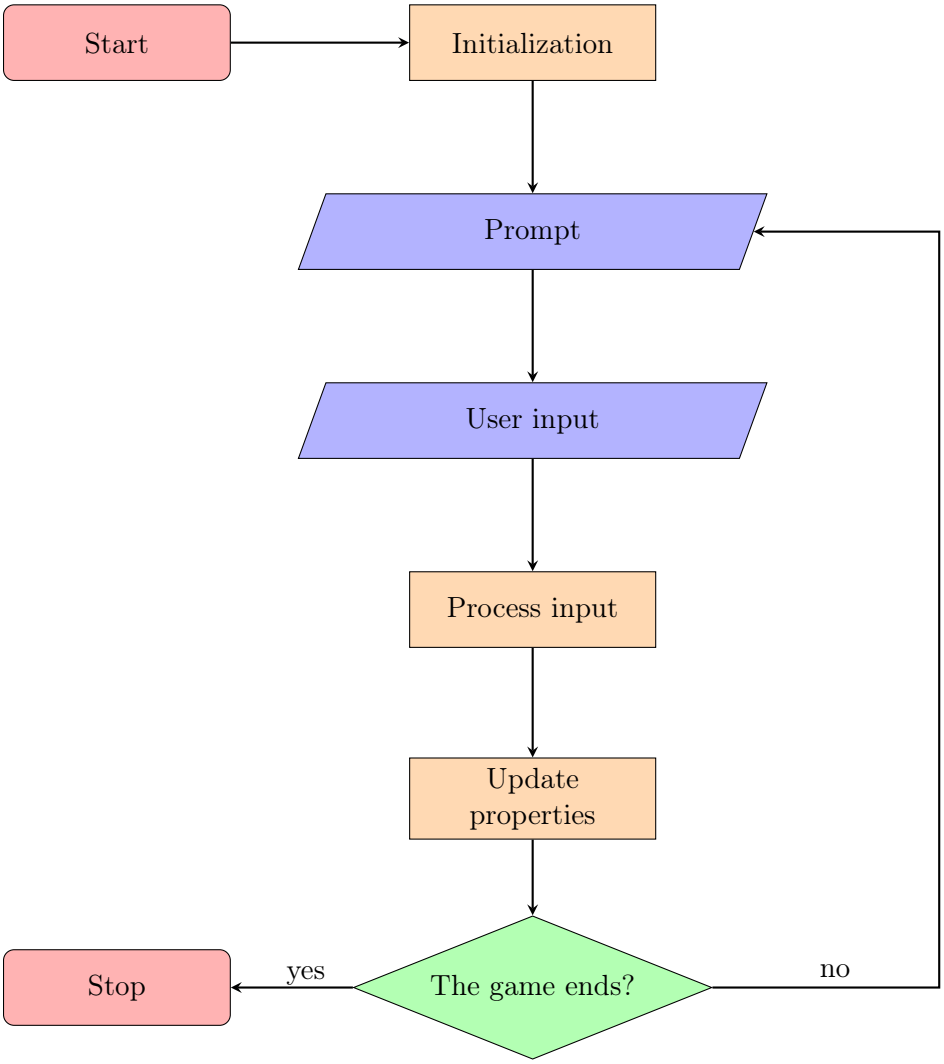
Preliminary Design

Chapter 1

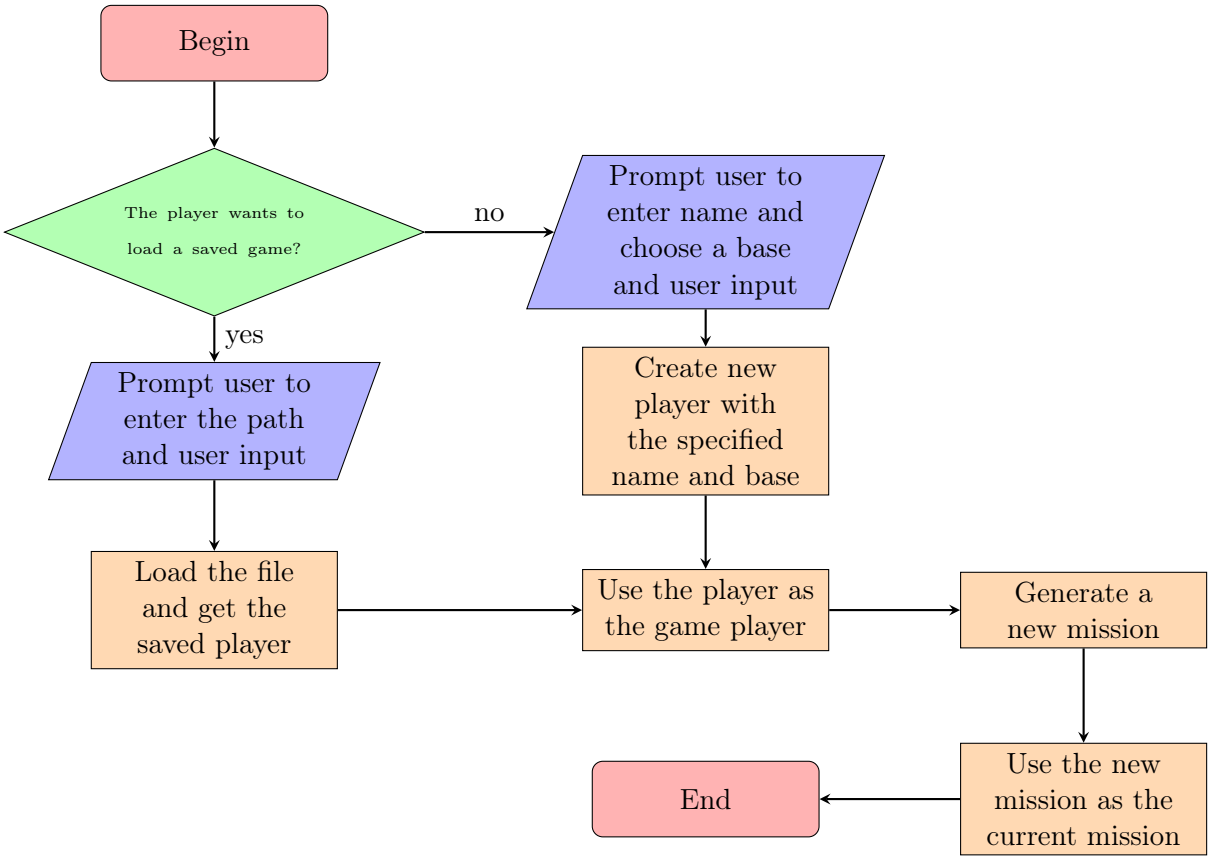
Diagrams

1.1 Program Process

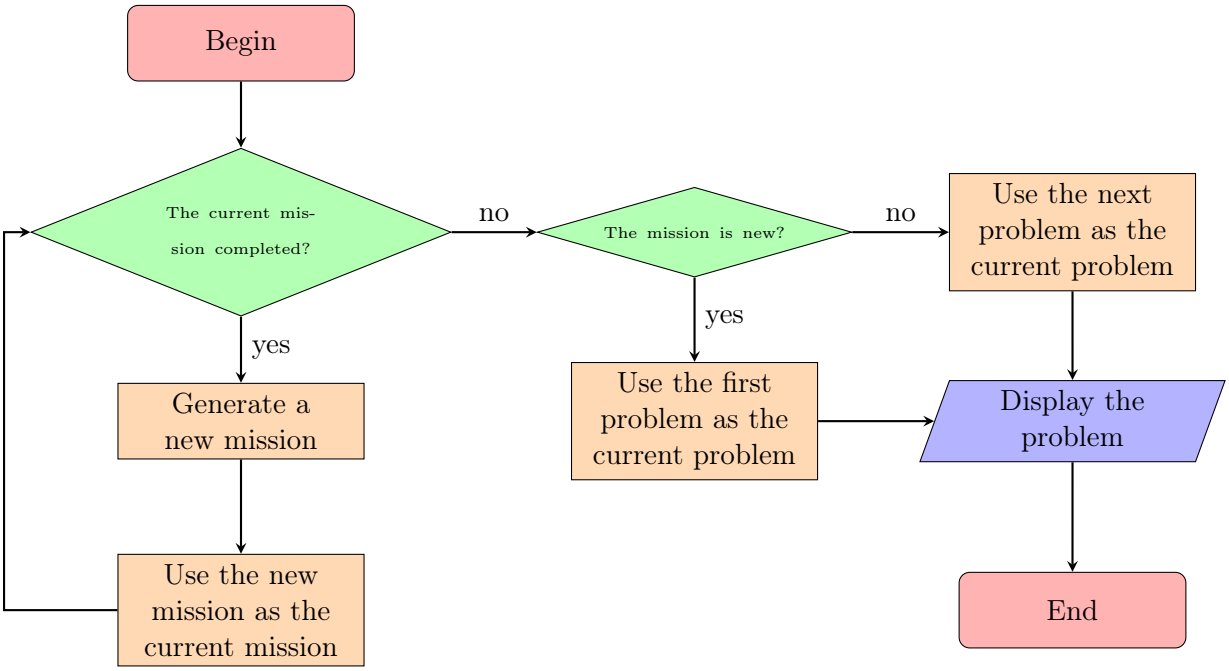
1.1.1 Overall Process



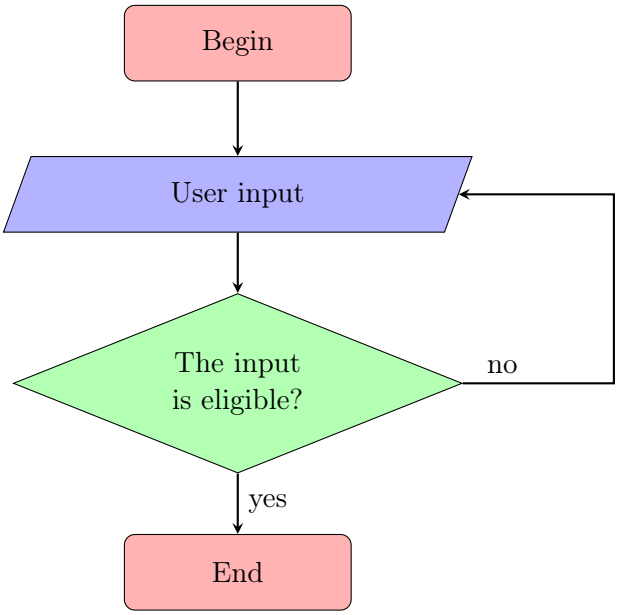
1.1.2 Initialization



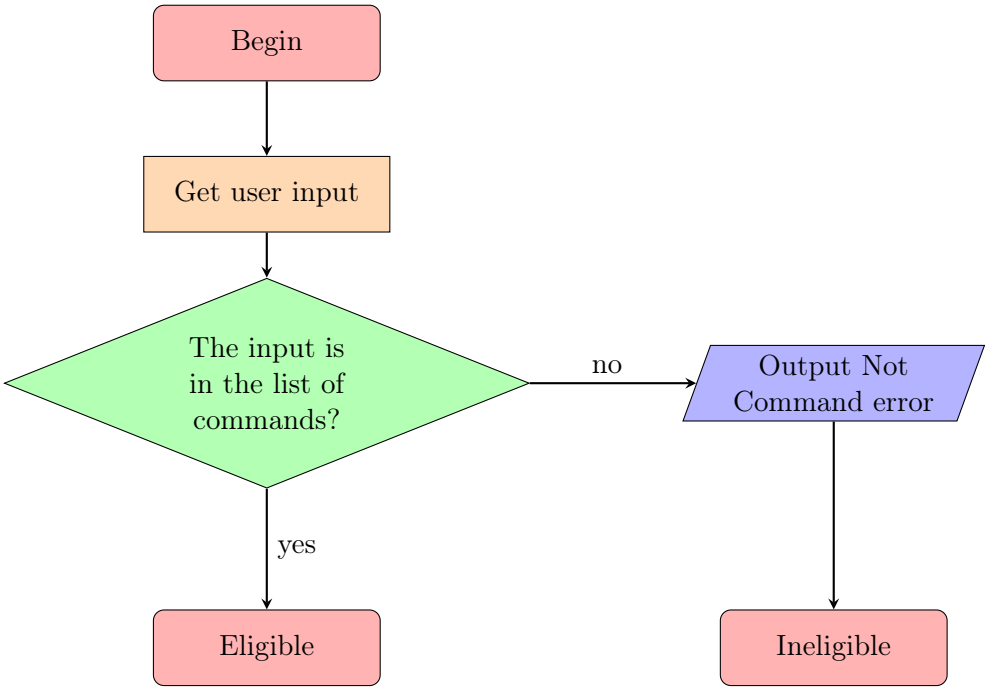
1.1.3 Prompt



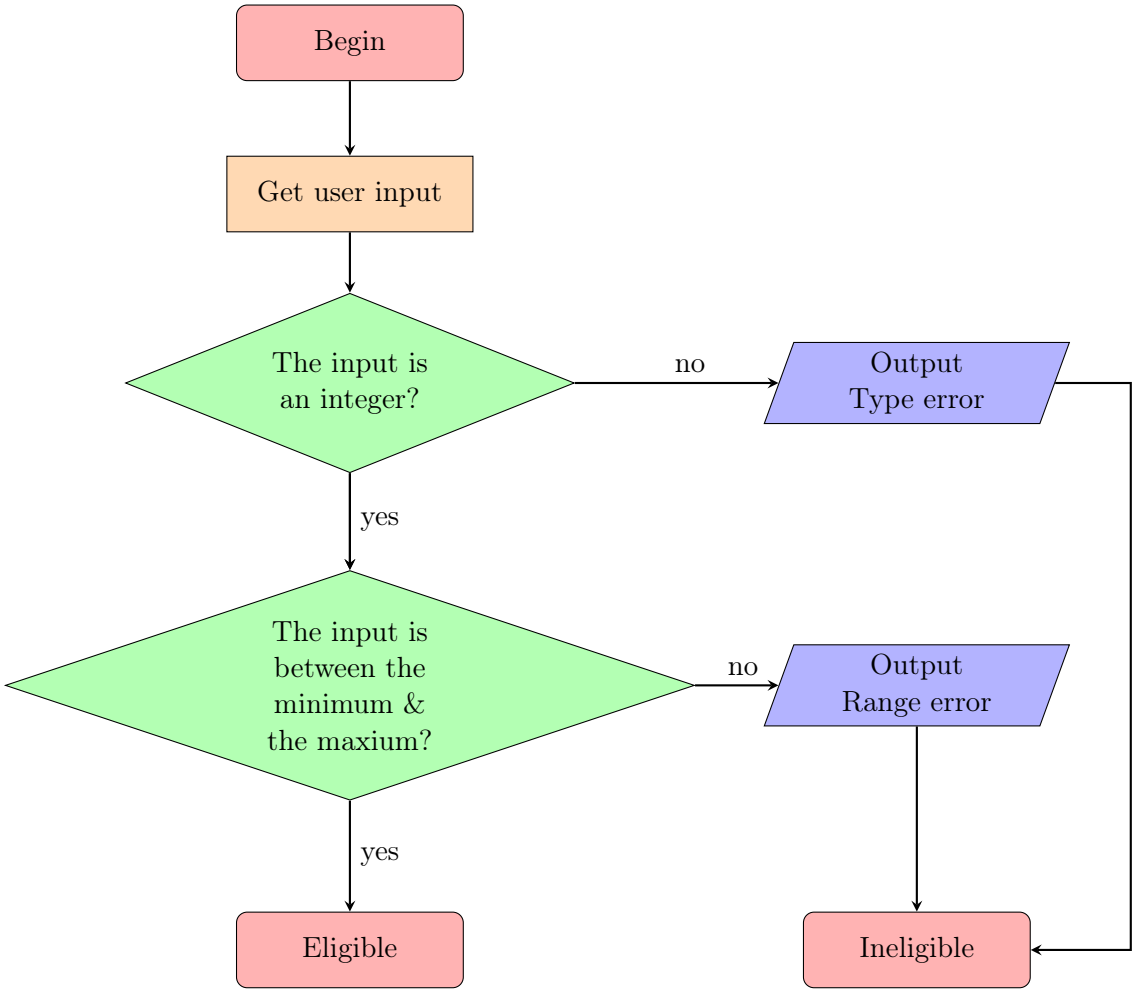
1.1.4 User Input



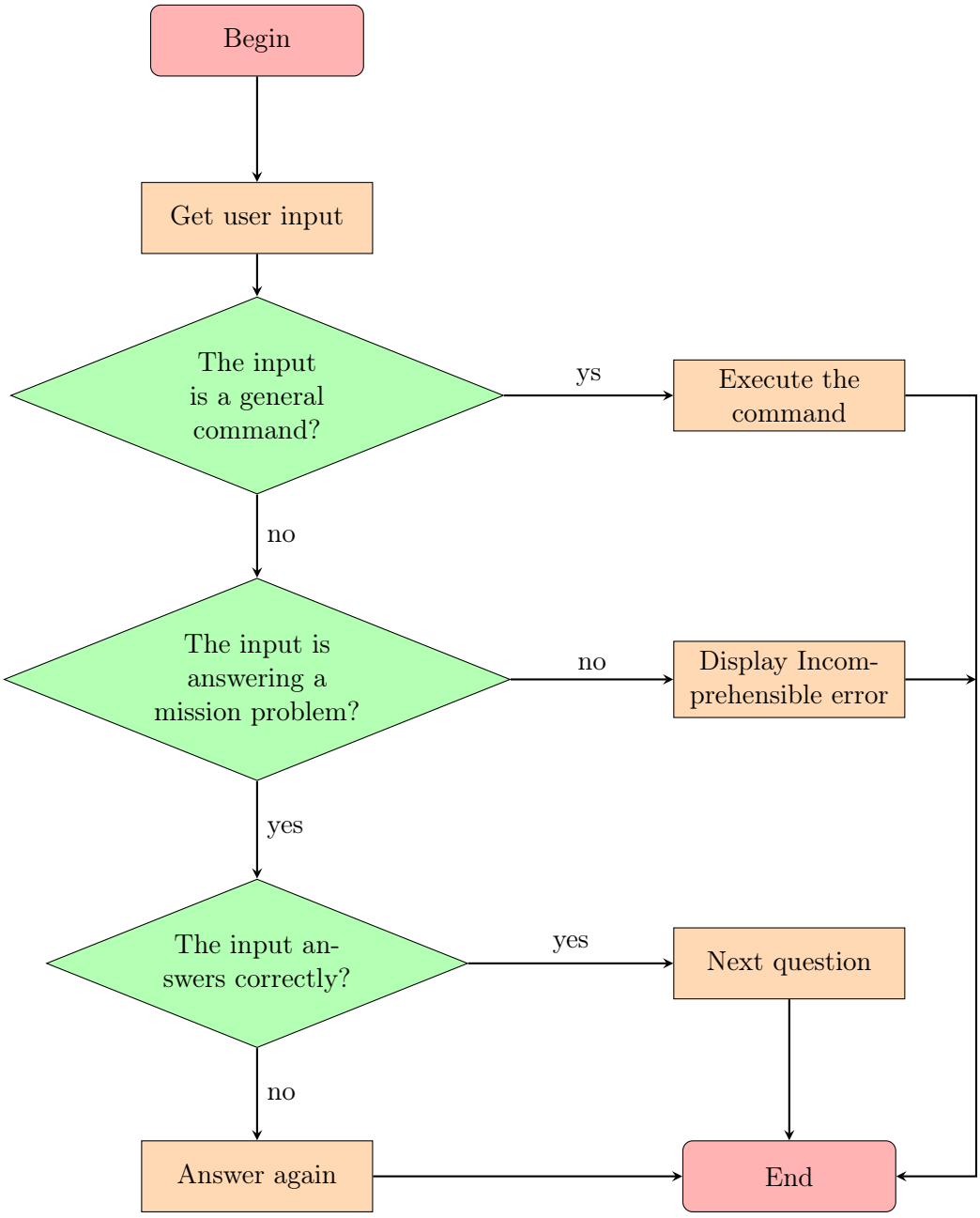
Determine if the input is eligible for commands



Determine if the input is eligible for integers in a specific range



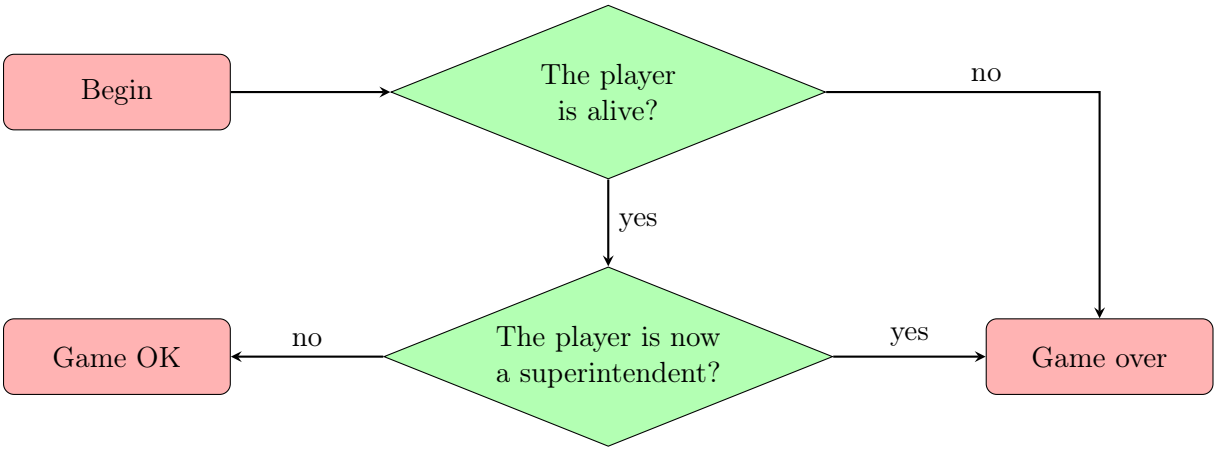
1.1.5 Process Input & Update Properties



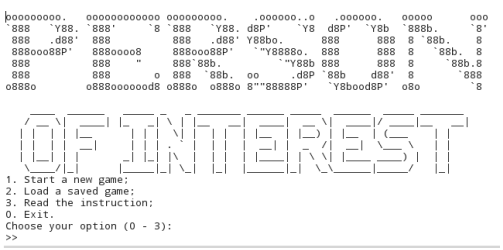
Commands

The following commands are planned as a general commands for the whole game instead of a particular mission, those in bracket are shortcut: *save* (*s*), *quit* (*q*),*help* (*h*), *load*, *myinfo*, and *feedback*.

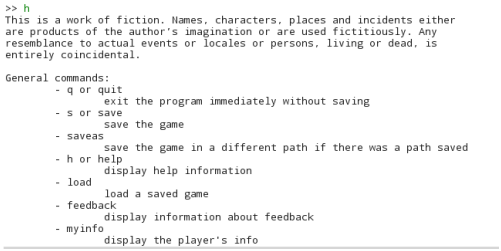
1.1.6 Game-ending Decision



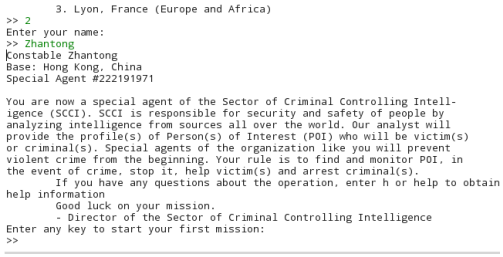
1.2 User Interface



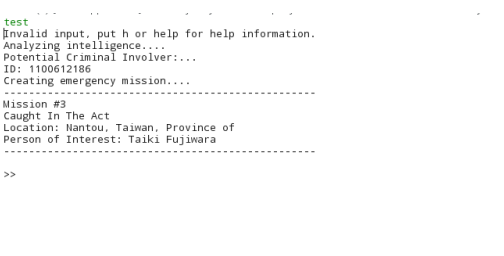
(a) start page



(b) help document



(c) game beginning



(d) mission generation

Figure 1.1: UI examples

Part III

Program Source

Chapter 2

Source

The source code of the program can be also found at `https://github.com/zhantongz/person-of-interest` with other program resources, and LaTeX sources and generated PDFs of the program documents.

The Java source code and program resources are placed in different directory. `src` is for program source that define how the program work. And `res` is for resources that provide meaningful resources to be processed by the program.

2.1 Agent.java

```
1  package personOfInterest;
2
3  /**
4   * Agent of the organization
5   *
6   * ref: http://docs.oracle.com/javase/tutorial/java/javaOO/index.html
7   *
8   * @author ZHANG, Zhantong <zhantongz@gmail.com>
9   * @version 1.0
10  * @since 2013-12-13 UTC
11  */
12 public class Agent extends Person {
13    /** serialization id */
14    private static final long serialVersionUID = -3828196555631548910L;
15    /** base location of an agent */
16    Location base;
17    /** an agent's id number */
18    int id;
19    /** rank of an agent */
20    int rank;
21    /**
22     * ranks' list
23     *
24     * ref:
25     * http://java.about.com/od/understandingdatatypes/a/Using-Constants.htm
26     */
27    public final static String[] RANKS = {"", "Constable", "Corporal", "Sergeant", "
          Staff Sergeant", "Sergeant Major",
28        "Inspector", "Superintendent", "Deputy Commissioner", "Commissioner"};
29    /** calendar of an agent */
30    int[] calendar;
```

```
31    /** current day in the calendar */
32    int currentDay;
33
34    /**
35     * Construct a new agent with specified name, rank and base location and a
36     * random id
37     *
38     * @param inName
39     *            specified name for the new agent
40     * @param inRank
41     *            specified rank for the new agent
42     * @param inBase
43     *            specified base location for the new agent
44     */
45    public Agent(String inName, int inRank, Location inBase) {
46      this(inName, inRank, inBase, Game.randomID());
47    }
48
49    /**
50     * Construct a new agent with specified name, rank and base location and a
51     * random id
52     *
53     * @param inName
54     *            specified name for the new agent
55     * @param inRank
56     *            specified rank for the new agent
57     * @param inBase
58     *            specified base location for the new agent
59     * @param inId
60     *            specified id number for the new agent
61     */
62    public Agent(String inName, int inRank, Location inBase, int inId) {
63      name = inName;
64      if (inRank <= 9 && inRank >= 1)
65        rank = inRank;
66      else
67        System.out.println("Rank out of range.");
68      base = inBase;
69      ifAlive = true;
70
71      this.id = inId;
72    }
73
74    /**
75     * Display an agent's info including rank, name, base and ID
76     */
77    public void displayInfo() {
78      System.out.println(RANKS[this.rank] + " " + this.name);
79      System.out.println("Base: " + this.base.name + ", " + this.base.country);
80      System.out.println("Special Agent #" + this.id);
81    }
82
83    /**
84     * Reset calendar of an agent
85     */
86    public void resetCal() {
87      this.calendar = null;
88      this.currentDay = 0;
89    }
```

```
90
91    /**
92     * Reset calendar of an agent to a certain period
93     */
94    public void resetCal(int days) {
95      this.calendar = new int[days];
96      this.currentDay = 0;
97    }
98
99    /**
100    * Pass a certain number of days
101    *
102    * @param days
103    *             the number of days passed
104    */
105   public void passDays(int days) {
106     for (int i = 0; i < days; i++)
107       this.passDay();
108   }
109
110   /**
111    * Pass current day
112    */
113   public void passDay() {
114     this.calendar[this.currentDay] = 1;
115     this.currentDay++;
116   }
117 }
```

2.2 Game.java

```
1  package personOfInterest;
2
3  import java.io.BufferedReader;
4  import java.io.File;
5  import java.io.FileInputStream;
6  import java.io.FileNotFoundException;
7  import java.io.FileOutputStream;
8  import java.io.FileReader;
9  import java.io.IOException;
10 import java.io.ObjectInputStream;
11 import java.io.ObjectOutputStream;
12 import java.io.StreamCorruptedException;
13 import java.text.SimpleDateFormat;
14 import java.util.Calendar;
15 import java.util.Scanner;
16 import java.util.StringTokenizer;
17
18 /**
19  * Main game program
20  *
21  * @author ZHANG, Zhantong <zhantongz@gmail.com>
22  * @version 1.0
23  * @since 2013-12-10 UTC
24  */
25 public class Game {
26   /* general tools and variables */
27   /** scanner that reads from keyboard */
```

```
28    static Scanner input = new Scanner(System.in);
29    /** saved path for saving game */
30    static String savedPath = null;
31    /** saved name for saving game */
32    static String savedName = null;
33    /** game calendar  */
34    static Calendar gameCal = Calendar.getInstance();
35    static int[] locArea = new int[8];
36    static int[] poiArea = new int[8];
37    /** if the player wants to start a new mission */
38    static boolean newMission = false;
39    static boolean processed = false;
40
41    /*
42     * methods for updating the game - start
43     *
44     * Modified from:
45     * http://www.mkyong.com/java/how-to-read-and-parse-csv-file-in-java/
46     */
47    /**
48     * Update location information from the csv file
49     */
50    public static void locs() throws NumberFormatException, IOException,
          InterruptedException {
51      String file = "res/locs/locs.csv";
52      BufferedReader br = null;
53      String line = "";
54      String delimiter = "=";
55      br = new BufferedReader(new FileReader(file));
56      int index = 1;
57      int currentArea = 0;
58      while ((line = br.readLine()) != null) {
59        String[] locInfo = line.split(delimiter);
60        ObjectOutputStream output = new ObjectOutputStream(new FileOutputStream("res/
              locs/" + index + ".loc"));
61        Location loc = new Location(locInfo[0],
62            locInfo[1],
63            locInfo[2],
64            Integer.parseInt(locInfo[3]),
65            addLinebreaks(locInfo[4]));
66        output.writeObject(loc);
67        output.close();
68
69        if (loc.area != currentArea) {
70          currentArea = loc.area;
71          locArea[currentArea] = index;
72        }
73
74        index++;
75      }
76      locArea[7] = index;
77      br.close();
78
79      System.out.println("Locations Updated");
80    }
81
82    /**
83     * Update location information from the csv file
84     */
```

```
85    public static void pois() throws NumberFormatException, IOException,
          InterruptedException {
86      String file = "res/pois/pois.csv";
87      BufferedReader br = null;
88      String line = "";
89      String delimiter = "=";
90      br = new BufferedReader(new FileReader(file));
91      int index = 1;
92      int currentArea = 0;
93      while ((line = br.readLine()) != null) {
94        String[] poiInfo = line.split(delimiter);
95        ObjectOutputStream output = new ObjectOutputStream(new FileOutputStream("res/
              pois/" + index + ".poi"));
96        POI poi = new POI(poiInfo[0], toBoolean(Integer.parseInt(poiInfo[2])));
97        poi.area = Integer.parseInt(poiInfo[1]);
98        output.writeObject(poi);
99        output.close();
100
101       if (poi.area != currentArea) {
102         currentArea = poi.area;
103         poiArea[currentArea] = index;
104       }
105
106       index++;
107     }
108     poiArea[7] = index;
109     br.close();
110
111     file = "res/pois/final/finalPOI.csv";
112     br = null;
113     line = "";
114     br = new BufferedReader(new FileReader(file));
115     index = 1;
116     while ((line = br.readLine()) != null) {
117       String[] poiInfo = line.split(delimiter);
118       ObjectOutputStream output = new ObjectOutputStream(new FileOutputStream("res/
              pois/final/" + index + ".poi"));
119       POI poi = new POI(poiInfo[0], toBoolean(Integer.parseInt(poiInfo[2])));
120       poi.area = Integer.parseInt(poiInfo[1]);
121       output.writeObject(poi);
122       output.close();
123
124       index++;
125     }
126     br.close();
127
128     System.out.println("POIs Updated");
129   }
130
131   /**
132    * Update location information from the csv file
133    */
134   public static void msts() throws NumberFormatException, IOException,
          InterruptedException {
135     String file = "res/msts/msts.csv";
136     BufferedReader br = null;
137     String line = "";
138     String delimiter = "=";
139     br = new BufferedReader(new FileReader(file));
```

```
140       int index = 0;
141
142       while ((line = br.readLine()) != null)
143         index++;
144       br.close();
145
146       br = new BufferedReader(new FileReader(file));
147       String[] msts = new String[index];
148       index = 0;
149       while ((line = br.readLine()) != null) {
150         String[] mstInfo = line.split(delimiter);
151         // System.out.println(mstInfo[0]);
152         msts[index] = mstInfo[0];
153         index++;
154       }
155       ObjectOutputStream output = new ObjectOutputStream(new FileOutputStream("res/msts/
              msts.mst"));
156       output.writeObject(msts);
157       output.close();
158
159       br.close();
160
161       System.out.println("Missions Updated");
162     }
163
164     /**
165      * Update problems
166      */
167     public static void prbs() throws IOException {
168       for (int rank = 1; rank < 7; rank++) {
169         String file = "res/prbs/" + rank + ".csv";
170         BufferedReader br = null;
171         String line = "";
172         String delimiter = "=";
173         br = new BufferedReader(new FileReader(file));
174         int index = 1;
175         while ((line = br.readLine()) != null) {
176           String[] prbInfo = line.split(delimiter);
177           ObjectOutputStream output = new ObjectOutputStream(new FileOutputStream("res/
                  prbs/" + rank + "/"
178                   + index + ".prb"));
179           // System.out.println(poiInfo[0]);
180           Problem prb = null;
181           switch (Integer.parseInt(prbInfo[2])) {
182           case 0: // int
183             prb = new Problem(prbInfo[0],
184                 prbInfo[1],
185                 Integer.parseInt(prbInfo[2]),
186                 Integer.parseInt(prbInfo[3]),
187                 Integer.parseInt(prbInfo[4]),
188                 Integer.parseInt(prbInfo[5]));
189             break;
190           case 1: // double
191             prb = new Problem(prbInfo[0],
192                 prbInfo[1],
193                 Integer.parseInt(prbInfo[2]),
194                 Double.parseDouble(prbInfo[3]),
195                 Integer.parseInt(prbInfo[4]),
196                 Integer.parseInt(prbInfo[5]));
```

```
197           break;
198         case 2: // boolean
199           if (prbInfo[3].equals("true"))
200             prb = new Problem(prbInfo[0],
201                 prbInfo[1],
202                 Integer.parseInt(prbInfo[2]),
203                 true,
204                 Integer.parseInt(prbInfo[4]),
205                 Integer.parseInt(prbInfo[5]));
206           else if (prbInfo[3].equals("false"))
207             prb = new Problem(prbInfo[0],
208                 prbInfo[1],
209                 Integer.parseInt(prbInfo[2]),
210                 false,
211                 Integer.parseInt(prbInfo[4]),
212                 Integer.parseInt(prbInfo[5]));
213           break;
214         case 3: // string
215           prb = new Problem(prbInfo[0],
216               prbInfo[1],
217               Integer.parseInt(prbInfo[2]),
218               prbInfo[3],
219               Integer.parseInt(prbInfo[4]),
220               Integer.parseInt(prbInfo[5]));
221           break;
222         }
223         prb.goodMessage = prbInfo[6];
224         prb.punishment = prbInfo[7];
225         prb.days = Integer.parseInt(prbInfo[8]);
226         output.writeObject(prb);
227         output.close();
228         index++;
229       }
230       br.close();
231     }
232
233     System.out.println("Problems Updated");
234   }
235
236   /* methods for updating the game - end */
237
238   /**
239    * Cast int to boolean
240    *
241    * @param num
242    *            the int ready to be casted
243    * @return false if the int is 0; true otherwise
244    */
245   public static boolean toBoolean(int num) {
246     if (num == 0)
247       return false;
248     return true;
249   }
250
251   /**
252    * Determine if the game is over
253    *
254    * @return true if the player is alive and not completed the game; false
255    *         otherwise
```

```
256     */
257    public static boolean isGameOver(Player player) {
258      if (!player.ifAlive)
259        return true;
260      if (player.rank >= 7)
261        return true;
262
263      return false;
264    }
265
266    /**
267     * Generate a 9-digits id number for agent id, mission id, etc.
268     *
269     * @return the generated 9-digits number
270     */
271    public static int randomID() {
272      return Math.abs((int) (Math.random() * 1000000000));
273    }
274
275    /**
276     * Generate a random number in a certain range
277     *
278     * @param min
279     *            the minimum for the number, inclusive
280     * @param max
281     *            the maximum for the number, inclusive
282     * @return the generated number
283     */
284    public static int randomNum(int min, int max) {
285      // ensure the max and min are correct
286      if (max < min) {
287        int temp = min;
288        min = max;
289        max = temp;
290      }
291
292      return (int) (Math.random() * (max - min + 1) + min);
293    }
294
295    /**
296     * Save the game
297     *
298     * @param player
299     *            the game's player
300     * @return true if the saving is a success; false otherwise
301     */
302    public static boolean saveGame(Player player) throws FileNotFoundException,
         IOException {
303      boolean success = true;
304      String path = null; // file directory path
305      String name = null; // file name
306      do {
307        success = true;
308        try {
309          Saved saved = new Saved(player); // saved file as an object
310          saved.prbNum = Mission.prbNum;
311          // get file info
312          path = inputLn("Enter the path that you want to save your game to (press enter
               if the same): ");
```

```
313         if (path.equals("")) {
314           path = savedPath;
315           name = savedName;
316         } else
317           name = inputLn("Enter the name that you want to save your game to (
                overwirtten if exists): ");
318
319         // create the directory if not exists
320         // ref
321         // http://www.roseindia.net/java/beginners/java-create-directory.shtml
322         new File(path).mkdir();
323
324         // save the path
325         if (savedPath == null || !savedPath.equalsIgnoreCase(path)) {
326           savedPath = path;
327           savedName = name;
328         }
329
330         // output stream that writes the object into a file
331         ObjectOutputStream output = new ObjectOutputStream(new FileOutputStream(
                savedPath + "/" + savedName
332              + ".game"));
333         output.writeObject(saved); // write the file
334         output.close(); // close the output
335       } catch (Exception e) {
336         System.err.println("Error. Permission may be denied; path may not be saved
                before.");
337         success = false;
338       }
339     } while (!success);
340
341     System.out.println("The game is saved successfully to\n"
342         + new File(path + "/" + name + ".game").getCanonicalPath());
343
344     return true;
345   }
346
347   /**
348    *
349    * Process the user's input
350    *
351    * @param uinput
352    *            the user's input
353    * @param player
354    *            the player
355    * @return true if the input is processed; false otherwise
356    */
357   public static boolean processUserInput(String uInput, Player player) throws
          FileNotFoundException,
358   IOException,
359   ClassNotFoundException,
360   InterruptedException {
361     processed = false;
362     uInput = uInput.toLowerCase();
363     if (uInput.equals("s") || uInput.equals("save"))
364       saveGame(player);
365     else if (uInput.equals("saveas")) {
366       // reset saved info
367       savedName = null;
```

```
368         savedPath = null;
369         saveGame(player);
370       } else if (uInput.equals("q") || uInput.equals("quit"))
371         System.exit(0);
372       else if (uInput.equals("h") || uInput.equals("help"))
373         help();
374       else if (uInput.equals("load")) {
375         Saved saved = (Saved) loadFile("Enter the path for your saved game: ");
376         player = saved.savedPlayer;
377       } else if (uInput.equals("myinfo")) {
378         System.out.println("Today is " + displayDate() + ".");
379         player.displayInfo();
380       } else if (uInput.equals("date"))
381         displayDate();
382       else if (uInput.equals("feedback"))
383         System.out.println(addLinebreaks("Go to https://github.com/zhantongz/person-of-
              interest/"
384             + "issues to submit feedback, thank you."));
385       else if (uInput.equals("newmission") || uInput.equals("n"))
386         newMission = true;
387       else if (uInput.equals("newgame"))
388         main(null);
389       else {
390         System.out.println("Invalid Input or Wrong Answer, enter h or help for help
              information.");
391         return false;
392       }
393       processed = true;
394       return true;
395     }
396
397     /**
398      * Process the user's input that intends for a problem
399      *
400      * @param problem
401      *            the problem
402      * @param player
403      *            the player
404      * @return true if the user answers the question correctly; false if answers
405      *         incorrectly or doesn't answer
406      */
407     public static boolean processUserInput(Problem problem, Player player) throws
          IOException,
408     ClassNotFoundException,
409     InterruptedException {
410       processed = false;
411       String uInput = "";
412       boolean correct = false;
413       try {
414         switch (problem.type) {
415         case 0: // int
416           uInput = input("Solve the problem:");
417           correct = problem.ifCorrect(Integer.parseInt(uInput));
418           break;
419         case 1: // double
420           uInput = input("Solve the problem:");
421           correct = problem.ifCorrect(Double.parseDouble(uInput));
422           break;
423         case 2: // boolean
```

```
424         uInput = input("Solve the problem:").toLowerCase();
425         if (uInput.equals("true"))
426           correct = problem.ifCorrect(true);
427         else if (uInput.equals("false"))
428           correct = problem.ifCorrect(false);
429         break;
430       case 3: // string
431         uInput = inputLn("Solve the problem:").toLowerCase();
432         correct = problem.ifCorrect(uInput);
433         break;
434       }
435     } catch (Exception e) {
436       correct = false;
437     }
438
439     if (!correct)
440       processed = processUserInput(uInput, player);
441
442     return correct;
443   }
444
445   /**
446    * Display help information
447    */
448   public static void help() throws FileNotFoundException {
449     File helpFile = new File("res/help.txt");
450     Scanner infile = new Scanner(helpFile);
451     while (infile.hasNext())
452       System.out.println(infile.nextLine());
453     infile.close();
454   }
455
456   /**
457    * Pause the program for a period of time while outputing three dots ref:
458    * http://docs.oracle.com/javase/tutorial/essential/concurrency/sleep.html
459    *
460    * @param timeInMs
461    *            the period of time to be paused for
462    */
463   public static void sleep(int timeInMs) throws InterruptedException {
464     timeInMs /= 3;
465     Thread.sleep(timeInMs);
466     System.out.print(".");
467     Thread.sleep(timeInMs);
468     System.out.print(".");
469     Thread.sleep(timeInMs);
470     System.out.println(".");
471   }
472
473   /**
474    * Print SCCI logo
475    */
476   public static void scciSeal() {
477     System.out.println(" __           _                        __");
478     System.out.println("/ _\\ ___  ___| |_ ___  _ __    ___  / _|");
479     System.out.println("\\ \\ / _ \\/ __| __/ _ \\| \'__|  / _ \\| |_");
480     System.out.println("_\\ \\  __/ (__| || (_) | |    | (_) |  _|");
481     System.out.println("\\__/\\___|\\___|\\__\\___/|_|     \\___/|_|");
482     System.out.println("   ___     _           _             _     ___            _
```

```
                    _ _ _");
483     System.out.println("  / __\\ __(_)_ __ ___  (_)_ __   __ _| |   / __\\___  _ __ | |
            _ _ __ ___ | | (_)_ __   __ _");
484     System.out.println(" / / | \'__| | \'_ ` _ \\| | \'_ \\ / _` | |  / /  / _ \\| \'_
             \\| __| \'__/ _ \\| | | | \'_ \\ / _` |");
485     System.out.println("/ /__| |  | | | | | | | | | | | (_| | | / /__| (_) | | | | |_|
             | | (_) | | | | | | | (_| |");
486     System.out.println("\\____/_|  |_|_| |_| |_|_|_| |_|\\__,_|_| \\____/\\___/|_| |_
            |\\__|_|  \\___/|_|_|_|_|_| |_|\\__, |");
487     System.out.println("  ______       _       _ _ _
                                                    |___/");
488     System.out.println("  \\_   \\_ __ | |_ ___| | (_) __ _  ___ _ __   ___ ___");
489     System.out.println("   / /\\/ \'_ \\| __/ _ \\ | | |/ _` |/ _ \\ \'_ \\ / __/ _ \\
            ");
490     System.out.println("/\\/ /_ | | | | ||  __/ | | | (_| |  __/ | | | (_|  __/");
491     System.out.println("\\____/ |_| |_|\\__\\___|_|_|_|\\__, |\\___|_| |_|\\___\\___|"
            );
492     System.out.println("                               |___/");
493
494   }
495
496   /**
497    * Show the start page with ASCII art
498    */
499   public static void showStartPage() {
500     System.out.println("ooooooooo.   oooooooooooo ooooooooo.    .oooooo..o   .oooooo.
            ooooo      ooo");
501     System.out.println("`888   `Y88. `888\'     `8 `888   `Y88. d8P\'    `Y8  d8P\'  `
            Y8b  `888b.     `8\'");
502     System.out.println(" 888   .d88\'  888          888   .d88\' Y88bo.      888
            888  8 `88b.    8");
503     System.out.println(" 888ooo88P\'   888oooo8     888ooo88P\'   `\"Y8888o.  888
                 888  8   `88b.  8");
504     System.out.println(" 888          888    \"     888`88b.          `\"Y88b 888
            888  8     `88b.8");
505     System.out.println(" 888          888       o  888  `88b.  oo     .d8P `88b    d88
            \'  8       `888");
506     System.out.println("o888o        o888ooooood8 o888o  o888o 8\"\"\"88888P\'   `
            Y8bood8P\'  o8o        `8");
507     System.out.println();
508     System.out.println("    ____  ______   _____ _   _ _______ ______ _____  ______
            _____ ________");
509     System.out.println("   / __ \\|  ____| |_   _| \\ | |__   __|  ____|  __ \\|  ____
            |/ ____|__   __|");
510     System.out.println("  | |  | | |__      | | |  \\| |  | |  | |__  | |__) | |__  |
            (___    | |");
511     System.out.println("  | |  | | __|     | | | . ` |  | |  |  __| |  _  /|  __|  \\
            ___ \\   | |");
512     System.out.println("  | |__| | |       _| |_| |\\  |  | |  | |____| | \\ \\| |____
             ____) |  | |");
513     System.out.println("   \\____/|_|      |_____|_| \\_|  |_|  |______|_|  \\_\\
            ______|_____/   |_|");
514   }
515
516   /**
517    * Get the user's input with space as the delimiter
518    *
519    * @param prompt
520    *            prompt for the user
```

```
521     * @return the string user entered
522     */
523    public static String input(String prompt) throws IOException {
524      System.out.println(prompt);
525      System.out.print(">> ");
526      String data = input.next();
527      input.nextLine();
528      return data;
529    }
530
531    /**
532     * Get the user's input with line break as the delimiter
533     *
534     * @param prompt
535     *            prompt for the user
536     * @return the string user entered
537     */
538    public static String inputLn(String prompt) {
539      System.out.println(prompt);
540      System.out.print(">> ");
541      return input.nextLine();
542    }
543
544    /**
545     * Check if the user's input is valid and return a valid data
546     *
547     * @param prompt
548     *            prompt for the user
549     * @param min
550     *            the minimum value for a valid data
551     * @param max
552     *            the maximum value for a valid data
553     * @return a valid integer data between min and max
554     */
555    public static int input(String prompt, int min, int max) {
556      // ensure the max and min are correct
557      if (max < min) {
558        int temp = min;
559        min = max;
560        max = temp;
561      }
562
563      boolean passed;
564      int data = min - 1;
565
566      do {
567        System.out.println(prompt);
568        System.out.print(">> ");
569        passed = true;
570        try {
571          data = input.nextInt();
572          if (data < min || data > max) {
573            passed = false;
574            System.out.println("Invalid range, enter a number between " + min + " and "
                   + max + ".");
575          }
576        } catch (Exception e) {
577          passed = false;
578          System.out.println("Invalid type, enter a number between " + min + " and " +
```

```
            max + ".");
579         input.next();
580       }
581     } while (!passed);
582     input.nextLine();
583     return data;
584   }
585
586   /**
587    * Load a serialized file
588    *
589    * @param prompt
590    *            prompt for the user
591    * @return the object included in the file
592    */
593   public static Object loadFile(String prompt) throws IOException {
594     boolean errorExists = false;
595     Object object = null;
596     do {
597       errorExists = false;
598       try {
599         ObjectInputStream inobj = new ObjectInputStream(new FileInputStream(input(
              prompt)));
600         object = inobj.readObject();
601         inobj.close();
602       } catch (FileNotFoundException e) {
603         System.out.println("Error. The file doesn't exist.");
604         errorExists = true;
605       } catch (ClassNotFoundException e) {
606         System.out.println("Error. File type may be wrong.");
607         errorExists = true;
608       } catch (StreamCorruptedException e) {
609         System.out.println("Error. File type may be wrong.");
610         errorExists = true;
611       } catch (NullPointerException e) {
612         errorExists = true;
613       }
614     } while (errorExists);
615
616     return object;
617   }
618
619   /**
620    * Transport an agent from his/her current location to a location requested
621    * through a certain way
622    *
623    * @param agent
624    *            the agent that need to be transported
625    * @param method
626    *            the way of transportation; 1 for train, 2 for plane, 3 for
627    *            normal vehicle and 4 for high-speed rail
628    * @param from
629    *            the current location
630    * @param to
631    *            the location requested
632    */
633   public static void transport(Agent agent, int method, Location from, Location to) {
634     double accidentRate = 0;
635     double speed = 0;
```

```
636       int distance = 0;
637
638       switch (method) {
639       case 1:
640         accidentRate = 0.00001;
641         speed = 105;
642         break;
643       case 2:
644         accidentRate = 0.0001;
645         speed = 885;
646         distance += 90;
647         break;
648       case 3:
649         accidentRate = 0.005;
650         speed = 100;
651         break;
652       case 4:
653         accidentRate = 0.0005;
654         speed = 300;
655         break;
656       }
657
658       if (randomNum(1, 100000) < 100000 * accidentRate)
659         if (randomNum(1, 10) == 5) {
660           System.out.println("Accident happened, you are killed.");
661           agent.kill();
662         } else {
663           int delay = randomNum(0, 3);
664           System.out.println("Accident happened, you are injured and your agenda is
                  delayed for " + delay
665                + " day(s).");
666           for (int i = 0; i < delay; i++) {
667             agent.passDay();
668             passDay();
669           }
670           distance -= 200;
671         }
672
673       if (!from.country.equals(to))
674         distance += randomNum(1500, 8000);
675       else
676         distance += randomNum(500, 2000);
677
678       for (int i = 0; i < (int) Math.ceil((distance / speed) / 20); i++) {
679         agent.passDay();
680         passDay();
681       }
682
683       System.out.println("You are now arrived in " + to.name + ".");
684       System.out.println(to.description);
685     }
686
687     /**
688      * Add a certain number of days to the game calendar
689      *
690      * @param days
691      *            the certain number of days
692      */
693     public static void passDays(int days) {
```

```
694       gameCal.add(Calendar.DATE, days);
695     }
696
697     /**
698      * Add 1 day to the calendar
699      */
700     public static void passDay() {
701       gameCal.add(Calendar.DATE, 1);
702     }
703
704     /**
705      * Display the date today (as in the game)
706      *
707      * @return the current date in the game
708      */
709     public static String displayDate() {
710       return new SimpleDateFormat("MMMMM d, yyyy").format(gameCal.getTime());
711     }
712
713     /**
714      * Determine if a player complete his/her mission in a certain time
715      *
716      * @param calendar
717      *            the calendar of the player/mission
718      * @return true if the player doesn't fail to comply the calendar; false
719      *         otherwise
720      */
721     public static boolean ifGoodCalendar(int[] calendar) {
722       for (int i = 0; i < calendar.length; i++)
723         if (calendar[i] != 1)
724           return true;
725       return false;
726     }
727
728     /**
729      * Break lines in a long string to make display nicer
730      *
731      * ref: http://stackoverflow.com/questions/7528045/large-string-split-into-
732      * lines-with-maximum-length-in-java
733      *
734      * @param input
735      *            the string need to be broken
736      * @param maxLineLength
737      *            the maxium line length to insert break line
738      * @return the string with proper breaks
739      */
740     public static String addLinebreaks(String input, int maxLineLength) {
741       StringTokenizer tok = new StringTokenizer(input, " ");
742       StringBuilder output = new StringBuilder(input.length());
743       int lineLen = 0;
744       while (tok.hasMoreTokens()) {
745         String word = tok.nextToken() + " ";
746
747         if (lineLen + word.length() > maxLineLength) {
748           output.append("\n");
749           lineLen = 0;
750         }
751         output.append(word);
752         lineLen += word.length();
```

```
753       }
754       return output.toString();
755     }
756
757     public static boolean takeChance(int divisor) {
758       return randomNum(1, divisor) == divisor;
759     }
760
761     public static boolean takeChance(double chance) {
762       return 100 * chance >= randomNum(1, 100);
763     }
764
765     /**
766      * Show a string in typewriter style
767      *
768      * @param message
769      *            the string need to be displayed
770      */
771     public static void typeString(String message) throws InterruptedException {
772       char[] mes = message.toCharArray();
773       for (int i = 0; i < mes.length; i++) {
774         System.out.print(mes[i]);
775         if (!(mes[i] == ' ') || (i < mes.length - 1 && mes[i + 1] == ' '))
776           Thread.sleep(50);
777       }
778       System.out.println();
779     }
780
781     /**
782      * Break lines in a long string with maximum length of 80
783      *
784      * @param input
785      *            the string need to be broken
786      * @return the string with proper breaks
787      */
788     public static String addLinebreaks(String input) {
789       return addLinebreaks(input, 80);
790     }
791
792     /**
793      * Main program
794      */
795     public static void main(String[] args) throws IOException, ClassNotFoundException,
          InterruptedException {
796       // update the game from resource files
797       locs();
798       pois();
799       msts();
800       prbs();
801
802       // start the game
803       showStartPage(); // show the start
804       Player player = null; // game's player
805       int prbNum = 0; // default prbNum
806
807       boolean started = false; // if the player has started
808       do {
809         started = false; // reset to false
810         System.out.println("1. Start a new game;");
```

```
811         System.out.println("2. Load a saved game;");
812         System.out.println("3. Read the instruction;");
813         System.out.println("0. Exit.");
814 
815         // initialization
816         gameCal.add(Calendar.DATE, randomNum(-256, 256));
817         gameCal.add(Calendar.YEAR, randomNum(25, 100));
818 
819         switch (input("Choose your option (0 - 3):", -1, 3)) {
820         case -1: // update problems mode
821           main(null);
822           break;
823         case 0:
824           System.exit(0);
825           break;
826         case 1: // new game
827           // set base location
828           Location base = null;
829           int baseIndex = 0;
830           switch (input("Choose your base location (1 - 3):\n"
831               + "\t1. Harbin, China (Asia-Pacific and Middle East)\n" + "\t2. Calgary,
                      Canada (Americas)\n"
832               + "\t3. Lyon, France (Europe and Africa)", 1, 3)) {
833               case 1:
834                 baseIndex = 5;
835                 break;
836               case 2:
837                 baseIndex = 24;
838                 break;
839               case 3:
840                 baseIndex = 49;
841           }
842           ObjectInputStream inloc = new ObjectInputStream(new FileInputStream("res/locs/
                  " + baseIndex + ".loc"));
843           base = (Location) inloc.readObject();
844           System.out.println(base.description);
845 
846           // set player's name
847           player = new Player(inputLn("Enter your name:"), 1, base);
848           started = true; // the player has started the game
849           break;
850         case 2: // load saved game
851           Saved saved = (Saved) loadFile("Enter the path for your saved game: ");
852           player = saved.savedPlayer;
853           prbNum = saved.prbNum;
854           started = true; // the player has started the game
855           break;
856         case 3:
857           help();
858         }
859       } while (!started);
860       player.displayInfo();
861       System.out.println();
862 
863       typeString(addLinebreaks(" " +
864           "SCCI is responsible for security and safety of people by analyzing
                  intelligence from sources all over the world. "
865           +
866           "Our analysts will provide the profile(s) of Person(s) of Interest (POI) who
```

```
            will be victim(s) or criminal(s). "
867         +
868         "Special agents of the organization like you will prevent violent crime from
                the beginning. "
869         +
870         "Your role is to find and monitor POI, in the event of crime, stop it, help
                victim(s) and arrest criminal(s)."
871         ) +
872         "\n\tIf you have any questions about the operation, enter h or help to obtain\
                n"
873         + "help information.\n" + "\tGood luck on your mission.\n" +
874         "\t- Director of the Sector of Criminal Controlling Intelligence");
875       System.out.print("Enter any key to start your first mission:\n>>");
876       player.location = player.base;
877       input.nextLine();
878       scciSeal();
879       sleep(3000);
880
881       // start the game loop
882       do {
883         newMission = false;
884         Mission mission = Mission.generateMission(player.base.area, player.rank);
885         if (player.rank == 6) {
886           mission = Mission.generateFinalMission(player.base.area);
887         }
888         mission.player = player;
889         System.out.print("Analyzing intelligence.");
890         sleep(2400);
891         System.out.print("Potential Criminal Involver:");
892         sleep(1500);
893         System.out.println("ID: " + mission.poi.id);
894         System.out.print("Creating emergency mission.");
895         sleep(2000);
896         System.out.println("-------------------------------------------------");
897         System.out.println("Mission #" + (player.missionCompleted + 1));
898         System.out.println(mission.name);
899         System.out.println("Location: " + mission.location.name + ", " + mission.
                location.country);
900         System.out.println("Person of Interest: " + mission.poi.name);
901         System.out.println("Time Permitted: " + mission.calendar.length + " days");
902         System.out.println("Potential Difficulties and Dangers: ");
903         System.out.println(mission.dangers);
904         System.out.println("-------------------------------------------------");
905         sleep(900);
906         typeString(mission.description);
907
908         player.calendar = mission.calendar;
909
910         if (inputLn("Type decline to decline this mission, type any other keys to start"
                ).equalsIgnoreCase("decline")) {
911           System.out.println("Due to the importance of emergency missions, you are now
                  discharged.\n"
912               + "SCCI acknowledges your service and thanks you.\n" + "GAME OVER");
913           System.exit(0);
914         }
915
916         System.out.println("Today is " + displayDate() + ".");
917
918         transport(player,
```

```
919             input(addLinebreaks("Choose your transportation, your current location is "
920                 + player.location.name
921                 + ", you should consider carefully about your transportation so that you
                        can complete mission.")
922                 + "\n\t1. Train\t2. Airplane\n" + "\t3. Automobile\t4. High-speed rail",
923                 1,
924                 4),
925                 player.location,
926                 mission.location);
927         System.out.println();
928         typeString(addLinebreaks(mission.arriveMessage));
929
930         if (!mission.completing(prbNum))
931           if (randomNum(1, 10) == 5) {
932             System.out.println(addLinebreaks("You failed to complete mission in required
                    time, "
933                 + "the mission failed as a result. As SCCI need agents who are "
934                 + "with great strength and intelligence in order to keep world "
935                 + "safe and secure, you are now discharged.\n")
936                 + "SCCI acknowledges your service and thanks you.\n"
937                 + "SCCI welcomes you to re-apply for agents after training.");
938             System.exit(0);
939           } else {
940             System.out.println(addLinebreaks("You failed to complete mission in required
                    time, "
941                 + "however, you still complete the mission. As SCCI need agents "
942                 + "with great strength and intelligence in order to keep world "
943                 + "safe and secure, you are now demoted in order to get training."));
944             if (player.rank > 1)
945               player.demote();
946             else
947               player.points = 5;
948           }
949
950         processUserInput(input("Enter n or newmission for a new mission"), player);
951         prbNum = 0;
952       } while (!isGameOver(player) && newMission);
953     }
954 }
```

2.3 Location.java

```
1  package personOfInterest;
2
3  import java.io.Serializable;
4
5  /**
6   * Location
7   *
8   * @author ZHANG, Zhantong <zhantongz@gmail.com>
9   * @version 1.0
10  * @since 2013-12-10 UTC
11  */
12 public class Location implements Serializable {
13   /** serialization id */
14   private static final long serialVersionUID = -2479125786089738935L;
15   /** a location's name */
16   String name;
```

```
17    /** a location's area as the index in AREAS */
18    int area;
19    /** city that a location situated */
20    String city;
21    /** country that a location situated */
22    String country;
23    /** locations that are in a location */
24    Location[] sublocations;
25    /** the number of sublocations */
26    int subNum;
27    /** description of a location */
28    String description;
29    /** areas that a location can be included in */
30    static final String[] AREAS = {"Antarctica", "Asia-Pacific", "Middle East", "North
          America", "South America",
31        "Europe", "Africa"};
32
33    /**
34     * Construct a location with name.
35     *
36     * @param inName
37     *            the location's name
38     */
39    public Location(String inName, String inCity, String inCountry, int inArea, String
          inDescription) {
40      name = inName;
41      city = inCity;
42      country = inCountry;
43      area = inArea;
44      description = inDescription;
45      subNum = 0;
46    }
47
48    /**
49     * Add a sublocation
50     *
51     * @param subloc
52     *            the sublocation that needs to be added
53     */
54    public void addSublocation(Location subloc) {
55      this.subNum++;
56      Location[] temp = this.sublocations;
57      this.sublocations = new Location[subNum];
58      for (int i = 0; i < this.subNum - 1; i++)
59        sublocations[i] = temp[i];
60      this.sublocations[subNum - 1] = subloc;
61    }
62
63    /**
64     * Add a series of sublocation
65     *
66     * @param sublocs
67     *            the sublocations that needs to be added
68     */
69    public void addSublocations(Location[] sublocs) {
70      for (int i = 0; i < sublocs.length; i++)
71        this.addSublocation(sublocs[i]);
72    }
73  }
```

2.4 Mission.java

```
1  package personOfInterest;
2
3  import java.io.File;
4  import java.io.FileInputStream;
5  import java.io.FileNotFoundException;
6  import java.io.IOException;
7  import java.io.ObjectInputStream;
8  import java.util.Scanner;
9
10 /**
11  * Mission of the player
12  *
13  * @author ZHANG, Zhantong <zhantongz@gmail.com>
14  * @version 1.0
15  * @since 2013-12-15 UTC
16  */
17 public class Mission {
18   /** location that a mission takes place */
19   Location location;
20   /** name of a mission */
21   String name;
22   /** person of interest in a mission */
23   POI poi;
24   /** if a mission is completed */
25   boolean completed;
26   /** rank that have abilities to complete a mission */
27   int rank;
28   /** money that can be used to complete a mission */
29   int budget;
30   /** calendar of a mission; time permitted */
31   int[] calendar;
32   /** points that awarded after completion */
33   int points;
34   /** player who is completing a mission */
35   Player player;
36   /** potential dangers of a mission */
37   String dangers;
38   /** description of a mission */
39   String description;
40   /** situations in a mission */
41   Problem[] problems;
42   /** message displayed after arrival */
43   String arriveMessage;
44   /** indication of where minor situations happens */
45   int[] minorNum;
46   /** default number for problem */
47   static int prbNum = 0;
48
49   /**
50    *
51    * Construct a mission
52    *
53    * @param person
54    *            the POI in a mission
55    * @param inname
56    *            the name of a mission
```

```
57      * @param inloc
58      *            the location of a mission
59      * @param days
60      *            the time limit in days for the mission
61      * @param inPoints
62      *            the points required to complete the mission
63      */
64     public Mission(POI person, String inname, Location inloc, int days, int inPoints) {
65       name = inname;
66       poi = person;
67       location = inloc;
68       completed = false;
69       points = inPoints;
70       calendar = new int[days];
71     }
72
73     /**
74      * Generate a random mission
75      *
76      * @param area
77      *            the area of the mission
78      * @param rank
79      *            the rank that the mission is designed for
80      * @return generated mission object
81      */
82     public static Mission generateMission(int area, int rank) throws
           FileNotFoundException,
83     IOException,
84     ClassNotFoundException {
85       ObjectInputStream inloc = new ObjectInputStream(new FileInputStream("res/locs/"
86           + (Game.randomNum(Game.locArea[area], Game.locArea[area + 2] - 1)) + ".loc"));
87       Location misLoc = (Location) inloc.readObject();
88       inloc.close();
89
90       ObjectInputStream inpoi = new ObjectInputStream(new FileInputStream("res/pois/"
91           + (Game.randomNum(Game.poiArea[area], Game.poiArea[area + 2] - 1)) + ".poi"));
92       POI poi = (POI) inpoi.readObject();
93       inpoi.close();
94       return new Mission(poi, "", misLoc, Game.randomNum(10, 15), 40, rank);
95     }
96
97     /**
98      * Complete a mission
99      *
100     * @param player
101     *            the player who completed the mission
102     */
103    public void complete() {
104      this.completed = true;
105      this.player.missionCompleted++;
106    }
107
108    /**
109     * Generate a final Mission
110     *
111     * @param area
112     *            the area of the mission
113     * @return the generated final mission
114     */
```

```
115   public static Mission generateFinalMission(int area) throws FileNotFoundException,
116   IOException,
117   ClassNotFoundException {
118     ObjectInputStream inloc = new ObjectInputStream(new FileInputStream("res/locs/"
119         + (Game.randomNum(Game.locArea[area], Game.locArea[area + 2] - 1)) + ".loc"));
120     Location misLoc = (Location) inloc.readObject();
121     inloc.close();
122
123     File[] finalPOIs = new File("res/pois/final/").listFiles();
124     ObjectInputStream inpoi = new ObjectInputStream(new FileInputStream(finalPOIs[Game
          .randomNum(0,
125         finalPOIs.length)]));
126     POI poi = (POI) inpoi.readObject();
127     inpoi.close();
128
129     return new Mission(poi, "Grande Finale", misLoc, 180, 500);
130   }
131
132   /**
133    * Process a mission
134    *
135    * @param prbsNum the start number of problems in a mission
136    * @return true if complete the mission; false otherwise
137    */
138   public boolean completing(int prbsNum) throws InterruptedException,
          ClassNotFoundException, IOException {
139     prbNum = prbsNum;
140     while (!this.completed) {
141       System.out.println("\nToday is " + Game.displayDate() + ".");
142
143       // mission problems
144       Problem currentProblem = this.problems[prbNum];
145       currentProblem.mission = this;
146       System.out.println("You have a situation now: ");
147       System.out.println("Situation #" + (prbNum + 1) + " " + currentProblem.name);
148
149       int wrongCount = 0;
150       boolean correct = false;
151       Game.typeString(Game.addLinebreaks(replaceName(currentProblem.description, this)
              ));
152       do {
153         if (wrongCount % 3 == 0 && wrongCount != 0)
154           System.out.println(replaceName(currentProblem.description, this));
155
156         if (!Game.processed)
157           wrongCount++;
158         if (wrongCount > currentProblem.chances)
159           currentProblem.punishment();
160         else {
161           System.out.println("The answer is [a(n)] " + currentProblem.getType() + ".")
                ;
162           correct = Game.processUserInput(currentProblem, player);
163         }
164       } while (!correct && !Game.processed && wrongCount <= currentProblem.chances);
165
166       if (correct) {
167         player.points += currentProblem.points;
168         Game.typeString(Game.addLinebreaks(replaceName(currentProblem.goodMessage,
              this)));
```

```
169           for (int i = 0; i < minorNum.length; i++)
170             if (prbNum + 1 == this.minorNum[i])
171               this.minorSituation(this.player.rank, i + 1);
172         }
173         Game.passDays(currentProblem.days);
174         player.passDays(currentProblem.days);
175
176         player.promote();
177         if (!Game.ifGoodCalendar(this.calendar))
178           return false;
179
180         prbNum++;
181         if (prbNum == this.problems.length) {
182           this.complete();
183           this.player.points += this.points;
184           return true;
185         }
186       }
187       return completed;
188     }
189
190     /**
191      * Replace certain tags in a string with certain variables
192      *
193      * @param s
194      *            the string need to be replaced
195      * @param mission
196      *            the mission that indicate variables
197      * @return the string that the certain tags are replaced
198      */
199     public static String replaceName(String s, Mission mission) {
200       String[] divide = s.split("\\|");
201       if (divide.length != 0) {
202         for (int i = 0; i < divide.length; i++) {
203           if (divide[i].equals("POI"))
204             divide[i] = mission.poi.name;
205           if (divide[i].equals("LocMst"))
206             divide[i] = mission.location.name;
207         }
208         String result = "";
209         for (int i = 0; i < divide.length; i++)
210           result += divide[i];
211         return result;
212       }
213       return s;
214     }
215
216     public Mission(POI person, String inname, Location inloc, int days, int inPoints,
          int rank)
217         throws FileNotFoundException, IOException, ClassNotFoundException {
218       this(person, inname, inloc, days, inPoints);
219       this.rank = rank;
220       File[] caughtPrbs = new File("res/prbs/1/").listFiles();
221       this.problems = new Problem[caughtPrbs.length];
222
223       for (int i = 0; i < caughtPrbs.length; i++) {
224         ObjectInputStream inprb = new ObjectInputStream(new FileInputStream("res/prbs/"
              + rank + "/" + (i + 1)
225             + ".prb"));
```

```
226         this.problems[i] = (Problem) inprb.readObject();
227         inprb.close();
228       }
229
230       File file = new File("res/msts/" + rank + ".csv");
231       Scanner inmst = new Scanner(file);
232       String[] infos = inmst.next().split("=");
233       inmst.close();
234
235       this.name = infos[0];
236
237       this.dangers = Game.addLinebreaks(replaceName(infos[1], this));
238       this.description = Game.addLinebreaks(replaceName(infos[2], this));
239       this.arriveMessage = Game.addLinebreaks(replaceName(infos[3], this));
240       this.minorNum = new int[] {Integer.parseInt(infos[4])};
241     }
242
243     /**
244      * Execute a customized minor situation
245      *
246      * @param index
247      *            the index of the minor situation
248      */
249     public void minorSituation(int rank, int index) throws InterruptedException {
250       switch (rank) {
251       case 1:
252         switch (index) {
253         case 1:
254           Game.typeString(Game.addLinebreaks("Your movement in a private residence
                causes"
255               + " a neighbour to call the police. A police officer is asking you to show
                    your ID"
256               + " and explain your purpose. What should you do?"));
257           switch (Game.input("1: Show your SCCI badge and told him you are on duty;\n"
258               + "2: Tell him you are here by accident, you thought this is your grandma'
                    s house;\n"
259               + "3: Get him inside and sedate the officer.", 1, 3)) {
260               case 1:
261                 if (Game.takeChance(8)) {
262                   System.out.println("A dirty police informs " + this.poi.name + ".");
263                   if (Game.takeChance(2)) {
264                     System.out.println("The mission continues with a delay of 2 days.");
265                     Game.passDays(2);
266                     player.passDays(2);
267                   } else {
268                     System.out.println("The mission failed.");
269                     for (int i = 0; i < this.calendar.length; i++)
270                       this.calendar[i] = 1;
271                   }
272                 } else if (Game.takeChance(2)) {
273                   System.out.println(Game.addLinebreaks("You are arrested for
                        impersonation of an officer of law as the police don't know this
                        operation."
274                       + "You are detained for 1 day until your ID is verified. Your rank
                            credit is cleared for your misdoing."));
275                   Game.passDay();
276                   player.passDays(1);
277                   player.points = 0;
278                 } else
```

```
279                 System.out.println("The officer accepts your badge and leaves.");
280               break;
281             case 2:
282               if (Game.takeChance(2)) {
283                 System.out.println("The officer arrests you for invasion of private
                        property.");
284                 if (Game.takeChance(7)) {
285                   System.out.println("You are charged and prosecuted, GAME OVER.");
286                   System.exit(0);
287                 } else
288                   System.out.println("You are released after paying $200 of fine.");
289               } else
290                 System.out.println("The officer reminds you of local laws and asks you
                        to leave.");
291               break;
292             case 3:
293               if (Game.takeChance(2))
294                 System.out.println("The officer forgets everything.");
295               else {
296                 System.out.println("The officer shoots you for self-defence, GAME OVER
                        .");
297                 System.exit(0);
298               }
299               break;
300         }
301       }
302     }
303   }
304 }
```

2.5 Person.java

```
1  package personOfInterest;
2
3  import java.io.Serializable;
4
5  /**
6   * Person including POI, agents of the organization, and others
7   *
8   * @author ZHANG, Zhantong <zhantongz@gmail.com>
9   * @version 1.0
10  * @since 2013-12-12 UTC
11  */
12 public class Person implements Serializable {
13   /** serialization id */
14   private static final long serialVersionUID = -2117345102201788590L;
15   /** person's name */
16   String name;
17   /** person's living status */
18   boolean ifAlive;
19   /** person's location */
20   Location location;
21
22   /**
23    * Display a person's information
24    */
25   public void displayInfo() {
26     System.out.println(this.name);
```

```
27      System.out.println("Location: " + this.location.name + ", " + this.location.
            country);
28    }
29
30    /**
31     * Kill a person
32     */
33    public void kill() {
34      this.ifAlive = false;
35    }
36  }
```

2.6 Player.java

```
1   package personOfInterest;
2
3   /**
4    * Player who play the game in the virtual world
5    *
6    * ref: http://docs.oracle.com/javase/tutorial/java/javaOO/index.html
7    *
8    * @author ZHANG, Zhantong <zhantongz@gmail.com>
9    * @version 1.0
10   * @since 2013-12-10 UTC
11   */
12  public class Player extends Agent {
13    /** serialization id */
14    private static final long serialVersionUID = -3247273499544130537L;
15    /** number of completed missions */
16    int missionCompleted;
17    /** a player's points used in promotion system */
18    int points;
19
20    /**
21     * Construct a player with a name, a rank and a base
22     *
23     * @param inName
24     *            specified name
25     * @param inRank
26     *            specified rank
27     * @param inBase
28     *            specified base
29     */
30    public Player(String inName, int inRank, Location inBase) {
31      super(inName, inRank, inBase);
32      this.missionCompleted = 0;
33    }
34
35    /**
36     * Promote a player's rank according to his/her points
37     */
38    public void promote() {
39      if (this.points >= 100) { // corporal
40        this.rank = 2;
41        if (this.points >= 150) { // sergeant
42          this.rank++;
43          if (this.points >= 220) { // staff sergeant
44            this.rank++;
```

```
45            if (this.points >= 320) { // sergeant major
46              this.rank++;
47              if (this.points >= 500) { // inspector
48                this.rank++;
49                if (this.points > 1000) // superintendent
50                  this.rank++;
51              }
52            }
53          }
54        }
55      }
56    }
57
58    /**
59     * Demote a player
60     */
61    public void demote() {
62      this.rank--;
63      switch (rank) {
64      case 1:
65        this.points = 20;
66        break;
67      case 2:
68        this.points = 100;
69        break;
70      case 3:
71        this.points = 150;
72        break;
73      case 4:
74        this.points = 220;
75        break;
76      case 5:
77        this.points = 320;
78        break;
79      case 6:
80        this.points = 500;
81        break;
82      }
83    }
84  }
```

2.7 POI.java

```
1  package personOfInterest;
2
3  /**
4   * Persons of Interest
5   *
6   * ref: http://docs.oracle.com/javase/tutorial/java/javaOO/index.html
7   *
8   * @author ZHANG, Zhantong <zhantongz@gmail.com>
9   * @version 1.0
10  * @since 2013-12-15 UTC
11  */
12 public class POI extends Person {
13   /** serialization id */
14   private static final long serialVersionUID = 9179990849010679531L;
15   /** true is the POI is criminal; false otherwise */
```

```
16    boolean ifCriminal;
17    /** POI's id number */
18    int id;
19    /** area of a POI */
20    int area;
21
22    /**
23     * Construct a POI with a name specifying if he/she is a criminal
24     *
25     * @param inName
26     *            POI's name
27     * @param criminality
28     *            true if the POI is a criminal; false otherwise
29     */
30    public POI(String inName, boolean criminality) {
31      name = inName;
32      ifAlive = true;
33      id = Game.randomID();
34    }
35
36    /**
37     * Get POI's ID
38     */
39    public int getID() {
40      return this.id;
41    }
42  }
```

2.8 Problem.java

```
1   package personOfInterest;
2
3   import java.io.Serializable;
4
5   /**
6    * Problem that will occur in a mission
7    *
8    * ref: http://docs.oracle.com/javase/tutorial/java/javaOO/index.html
9    *
10   * @author ZHANG, Zhantong <zhantongz@gmail.com>
11   * @version 1.0
12   * @since 2013-12-17 UTC
13   */
14  public class Problem implements Serializable {
15    /** serialization id */
16    private static final long serialVersionUID = 4395782311889272031L;
17    /** mission that contains a problem */
18    Mission mission;
19    /** a problem's name */
20    String name;
21    /** details of a problem */
22    String description;
23    /** a problem's answer; might be integer, string, boolean, etc. */
24    Object answer;
25    /**
26     * type of a problem's answer; 0 for int, 1 for double, 2 for boolean, 3 for
27     * String
28     */
```

```
29    int type;
30    /** location where a problem occurred */
31    Location location;
32    /** value of a problem in points (promotion system) */
33    int points;
34    /** maximum chances to answer a problem */
35    int chances;
36    /** message after good answer */
37    String goodMessage;
38    /** days that a mission used */
39    int days;
40    /** punishment */
41    String punishment;
42
43    /**
44     * Construct a problem
45     *
46     * @param inName
47     *            problem's name
48     * @param inDesc
49     *            problem's description
50     * @param type
51     *            type of problem answer
52     * @param answer
53     *            problem's answer
54     */
55    public Problem(String inName, String inDesc, int inType, Object inAnswer, int
          inPoints, int inChances) {
56      name = inName;
57      description = inDesc;
58      answer = inAnswer;
59      type = inType;
60      points = inPoints;
61      chances = inChances;
62    }
63
64    /**
65     * Check if a response from a source is correct
66     *
67     * @param response
68     *            the response from the user
69     * @return if the response is consistent with the answer
70     */
71    public boolean ifCorrect(Object response) {
72      return response.equals(answer);
73    }
74
75    /**
76     * Get a problem's type
77     *
78     * @return the type of the problem
79     */
80    public String getType() {
81      switch (this.type) {
82      case 0: // int
83        return "INTEGER";
84      case 1: // double
85        return "DECIMAL NUMBER";
86      case 2: // boolean
```

```
87        return "TRUE OR FALSE";
88      case 3: // string
89        return "WORD(S)";
90      default:
91        return "";
92      }
93    }
94
95    /**
96     * Punishment for wrong answers
97     */
98    public boolean punishment() {
99      String[] puns = this.punishment.split("]");
100     for (int i = 0; i < puns.length; i++) {
101       String pun = puns[i];
102       String[] detail = pun.split("<");
103       if (Game.takeChance(Double.parseDouble(detail[0]))) {
104         System.out.println(Game.addLinebreaks(Mission.replaceName(detail[1], this.
              mission)));
105         int delay = Game.randomNum(Integer.parseInt(detail[2]), Integer.parseInt(
              detail[3]));
106         if (delay < 100) {
107           try {
108             Game.passDays(delay);
109             this.mission.player.passDays(delay);
110           } catch (ArrayIndexOutOfBoundsException e) {
111             for (int j = 0; j < this.mission.calendar.length; j++)
112               this.mission.calendar[j] = 1;
113           }
114         } else if (delay == 110) {// code for game over
115           System.out.println("GAME OVER");
116           System.exit(0);
117         } else if (delay == 911)
118           for (int j = 0; j < this.mission.calendar.length; j++)
119             this.mission.calendar[j] = 1;
120         return true;
121       }
122     }
123     return false;
124   }
125 }
```

2.9 Saved.java

```
1  package personOfInterest;
2
3  import java.io.Serializable;
4
5  /**
6   * Saved file for the game
7   *
8   * @author ZHANG, Zhantong <zhantongz@gmail.com>
9   * @version 1.0
10  * @since 2013-12-12 UTC
11  */
12 public class Saved implements Serializable {
13   /** serialization id */
14   private static final long serialVersionUID = -364063165363614855L;
```

```
15    /** player saved */
16    public Player savedPlayer;
17    /** location saved */
18    public Location savedLocation;
19    /** mission number */
20    public int missionNum;
21    /** problem number */
22    public int prbNum;
23
24    /**
25     * Construct a Saved class containing player and location
26     *
27     * @param player
28     *            the player intending to be saved
29     * @param location
30     *            the location intending to be saved
31     */
32    public Saved(Player player, Location location) {
33      savedPlayer = player;
34      savedLocation = location;
35    }
36
37    /**
38     * Construct a Saved class containing player only
39     *
40     * @param player
41     *            the player intending to be saved
42     */
43    public Saved(Player player) {
44      savedPlayer = player;
45      savedLocation = null;
46    }
47  }
```

Program resources are not included because of their large sizes and the inability to show source (like binary files). Those resources will be updated and can be obtained from the git repository at `https://github.com/zhantongz/person-of-interest`. The following are examples of text resources.

2.10 locs.csv

This file contains information of locations in the game.

```
1  Brisbane=Brisbane=Australia=1=Brisbane is the capital and most populous city in the
       Australian state of Queensland, and the third most populous city in Australia. The
       metropolitan area extends in all directions along the floodplain of the Brisbane
       River valley between the bay and the Great Dividing Range. Brisbane was chosen as
       the capital when Queensland was proclaimed a separate colony from New South Wales
       in 1859. The city played a central role in the Allied campaign during World War II
       as the South West Pacific headquarters for General Douglas MacArthur.
2  Busan=Busan=South Korea=1=Busan is South Korea's second largest city. The population
       is approximately 3.6 million. It is the largest port city in South Korea. As well,
       it ranks fifth in the busiest seaport by cargo tonnage. The city is located on the
       southeastern-most of the Korean peninsula.
3  East Timor==Asia=1=East Timor or Timor-Leste, officially the Democratic Republic of
       Timor-Leste, is a country in Southeast Asia. It comprises the eastern half of the
       island of Timor, the nearby islands of Atauro and Jaco, and Oecusse, an exclave on
       the northwestern side of the island, within Indonesian West Timor.
```

4 Hanoi=Hanoi=Vietnam=1=Hanoi is the capital of Vietnam and the country's second largest city. It was the most important political centre of Vietnam. It was eclipsed by Hu , the imperial capital of Vietnam during the Nguyn Dynasty, but Hanoi served as the capital of French Indochina. The city lies on the right bank of the Red River.

5 Harbin=Harbin=China=1=Harbin is the capital of Heilongjiang located in the northeastern-most part of China. Serving as a key political, economic, scientific, cultural and communications hub in Northeast China, Harbin is notable **for** its beautiful ice sculptures in winter known as the "Ice City" and its Russian legacy, and it still plays an important role in Sino-Russian trade today.

6 Hoh Xil==China=1=Hoh Xil or Kekexili, (Mongolian **for** "Blue ridge", also Aqênganggyai **for** "Lord of ten thousand mountains"), is an isolated region in the northwestern part of the Qinghai-Tibet Plateau in China. It is China's least and the world's third-least populated area. The Qingzang railway and China National Highway 109 run along the eastern boundary of the reserve. Hoh Xil is home to more than 230 species of wild animals, 20 of which are under state protection.

7 Hong Kong=Hong Kong=China=1=Hong Kong is one of the two Special Administrative Regions of the People's Republic of China. It is situated on China's south coast and, enclosed by the Pearl River Delta and South China Sea, it is known **for** its expansive skyline and deep natural harbour. As one of the world's leading international financial centres, Hong Kong has a major capitalist service economy characterised by low taxation and free trade.

8 Manila=Mayanila=Philippines=1=Manila is the capital and second largest city of the Philippines. It is one of the sixteen cities which, along with the municipality of Pateros, make up Metro Manila, the National Capital Region, whose overall population is around 12 million.

9 Medog=Medog=China=1=Mêdog, Metok, or Motuo County is a county of the Nyingtri Prefecture in the Tibet Autonomous Region of People's Republic of China. Chinese claims include parts of Arunachal Pradesh, south of the McMahon Line, what was casus belli **for** the 1962 Sino-Indian War.

10 Nantou=Nantou=Taiwan, Province of=1=Nantou County is the second largest county of Taiwan. It is also the only landlocked county in Taiwan. Its name derives from the Hoanya Taiwanese aboriginal word Ramtau. Nantou County is officially administered as a county of Taiwan Province.

11 Osaka=Osaka=Japan=1=Osaka is a city in the Kansai region of Japan's main island of Honshu, a designated city under the Local Autonomy Law, the capital city of Osaka Prefecture and also the largest part of the Keihanshin metropolis. Located at the mouth of the Yodo River on Osaka Bay, Keihanshin is one of the largest metropolitan areas highly ranked in the world, with nearly 19 million people, and by GDP the second largest area in Japan and the seventh largest area in the world. It's also a home to some of the most well known electronic companies such as Panasonic, Sharp, and Sanyo.

12 Pyongyang=Pyongyang=North Korea=1=Pyongyang is the capital of the Democratic People's Republic of Korea, commonly known as North Korea, and the largest city in the country. Pyongyang is located on the Taedong River. The city was split from the South P'yngan province in 1946. It is administered as a directly governed city (chikhalsi), on the same level as provincial governments.

13 Wellington=Wellington=New Zealand=1=Wellington is the capital city and second most populous urban area of New Zealand. It is at the southwestern tip of the North Island, between Cook Strait and the Rimutaka Range. The Wellington urban area is the major population centre of the southern North Island, and is the seat of the Wellington Region which in addition to the urban area covers the Kapiti Coast and Wairarapa. Wellington also holds the distinction of being the world's southernmost capital city.

14 Sipsongpanna=Sipsongpanna=China=1=Xishuangbanna, Sipsongpanna, or Sibsongbanna, abbreviated to Xidai, is an autonomous prefecture in the south of Yunnan province, People's Republic of China. The prefectural seat is Jinghong, the largest settlement in the area and one that straddles the Mekong River, called the Lancang River in Chinese. Xishuangbanna harbors much of the biodiversity of Yunnan Province , which harbors much of the biodiversity of China.

```
15 Yichun=Yichun=China=1=Yichun is a prefecture-level city on the Songhua river in
       Heilongjiang province, People's Republic of China. The city is separated from
       Russia by the Amur River and has an international border of 246 kilometres. At the
       2010 census, Yichun has a total population of 1,148,126 while 729,202 people live
       in 15 districts separated by forests. The greening rate of Yichun is up to 83%. The
        nickname of Yichun is Lindu (literally "forest capital"). Yichun has a monsoon-
       influenced, hemiboreal climate, with long, bitterly cold, but very dry winters, and
        very warm, humid summers.
16 Ankara=Ankara=Turkey=2=Ankara is the capital of Turkey since the empire's fall in 1923
        and the country's second largest city, Istanbul being the largest. The city has a
       mean elevation of 938 metres (3,077 ft). Centrally located in Anatolia, Ankara is
       an important commercial and industrial city. It is the center of the Turkish
       Government. It is an important crossroads of trade, strategically located at the
       centre of Turkey's highway and railway networks, and serves as the marketing centre
        for the surrounding agricultural area.
17 Damascus=Damascus=Syria=2=Damascus is the capital and the second largest city of Syria
        after Aleppo. Commonly known in Syria as ash-Sham and nicknamed as the City of
       Jasmine, it borders Quneitra, Daraa and As-Suwayda to the south, Jordan to the east
       , Homs to the north, and Lebanon to the west. It is also the capital city of one of
        the country's 14 governorates. In addition to being one of the oldest continuously
        inhabited cities in the world, Damascus is a major cultural and religious center
       of the Levant.
18 Isfahan=Isfahan=Iran=2=Isfahan, historically also rendered in English as Ispahan,
       Sepahan or Hispahan, is the capital of Isfahan Province in Iran, located about 340
       kilometres south of Tehran. Isfahan is located on the main north-south and east-
       west routes crossing Iran, and was once one of the largest cities in the world.
       Even today, the city retains much of its past glory. It is famous for its Islamic
       architecture, with many beautiful boulevards, palaces, mosques, and minarets. This
       led to the Persian proverb "Isfahan is half of the world".
19 Jerusalem=Jerusalem=United Nations=2=Jerusalem located on a plateau in the Judean
       Mountains between the Mediterranean and the Dead Sea, is one of the oldest cities
       in the world. It is considered holy to the three major Abrahamic religions -
       Judaism, Christianity and Islam. Israelis and Palestinians both claim Jerusalem as
       their capital, as Israel maintains its primary governmental institutions there and
       the State of Palestine ultimately foresees it as its seat of power; however,
       neither claim is widely recognized internationally. It is a "international" city
       considered by UNGA Resolution 194.
20 Riyadh=Riyadh=Saudi Arabia=2=Riyadh is the capital and largest city of Saudi Arabia.
       It is also the capital of Riyadh Province, and belongs to the historical regions of
        Najd and Al-Yamama. It is situated in the center of the Arabian Peninsula on a
       large plateau, and is home to 5,254,560 people, and the urban center of a region
       with a population of close to 7 million people. It has been designated as a Beta
       World City.
21 Tehran=Tehran=Iran=2=Tehran is the capital of Iran and Tehran Province. With a
       population of around 8.3 million and surpassing 14 million in the wider
       metropolitan area, Tehran is Iran's largest city and urban area, and the largest
       city in Western Asia. In the 20th and 21st centuries, Tehran has been the subject
       of mass migration of people from all around Iran. The city is home to many historic
        mosques, churches, synagogues and Zoroastrian fire temples. However, modern
       structures, notably Azadi (Freedom) Tower and the Milad Tower, have come to
       symbolise the city.
22 Akron=Akron=United States=3=Akron is the fifth largest city in the U.S. state of Ohio
       and is the county seat of Summit County. It is located in the Great Lakes region
       approximately 63 km south of Lake Erie along the Little Cuyahoga River. As of the
       2010 census, the city had a population of 199,110. The Akron Metropolitan
       Statistical Area covers Summit and Portage counties, and in 2010 had a population
       of 703,200. Akron is also part of the larger Cleveland-Akron-Elyria Combined
       Statistical Area, which in 2010 had a population of 2,780,440.
```

23 Austin=Austin=United States=3=Austin is the capital of Texas and the seat of Travis County. Located in Central Texas and the American Southwest, it is the 11th-largest city in the United States of America and the fourth-largest city in the state of Texas. It was the third-fastest-growing large city in the nation from 2000 to 2006. Austin is also the second largest state capital in the United States. The city is the cultural and economic center of the five-county AustinRound Rock metropolitan area.

24 Calgary=Calgary=Canada=3=Calgary is a city in the province of Alberta, Canada. It is situated on the Bow River in the south of the province, in an area of foothills and prairie, approximately 80 km east of the front ranges of the Canadian Rockies. The city is located in the grassland and parkland natural regions of Alberta. The City of Calgary is the largest city in Alberta, and the third-largest municipality and fifth-largest metropolitan area in Canada. The city is located 294 km south of Edmonton. Economic activity in Calgary is mostly centred on the petroleum industry and agriculture. In 1988, Calgary became the first Canadian city to host the Olympic Winter Games.

25 Cape Coral=Cape Coral=United States=3=Cape Coral is a municipality located in Lee County, Florida, United States, on the Gulf of Mexico. Founded in 1957 and developed as a master-planned, pre-platted community. With an area of 310 suqare km , Cape Coral is the largest city between Tampa and Miami. It is a principal city in the Cape Coral Fort Myers, Florida Metropolitan Statistical Area. The population estimate **for** the statistical area was 645,899 **for** 2009. The city is known as a " Waterfront Wonderland", since with over 640 km of navigable waterways, Cape Coral has more miles of canals than any other city in the world.

26 Gatineau=Gatineau=Canada=3=Gatineau is a city in western Quebec, Canada. It is the fourth largest city in the province. Located on the northern banks of the Ottawa River, immediately across from Ottawa, together they form Canada's National Capital Region. As of 2011 Gatineau had a population of 265,349, and a metropolitan population of 314,501. The OttawaGatineau census metropolitan area had a population of 1,236,324. Gatineau is coextensive with a territory equivalent to a regional county municipality (TE) and census division (CD) of the same name.

27 La Habana=La Habana=Cuba=3=Havana, or La Habana, is the capital city, province, major port, and leading commercial centre of Cuba. The city proper has a population of 2.1 million inhabitants, and it spans a total of 728.26 square km making it the largest city by area, the most populous city, and the third largest metropolitan area in the Caribbean region. The sluggish Almendares River traverses the city from south to north, entering the Straits of Florida a few miles west of the bay.

28 Hayes=Hayes=Jamaica=3=Hayes is a settlement in Jamaica. It has a population of 9,798 as of 2009.

29 Iowa City=Iowa City=United States=3=Iowa City is a city in Johnson County, Iowa. Iowa City is the county seat of Johnson County and home to the University of Iowa. Iowa City is located adjacent to the town of Coralville, and it surrounds the town of University Heights, with which it forms a contiguous urban area. Iowa City is the principal city of the Iowa City Metropolitan Statistical Area, which encompasses Johnson County and Washington County. Iowa City was the second capital of the Iowa Territory, and it was also the first capital city of the State of Iowa. Forbes Magazine named Iowa City the second-best small metropolitan area for doing business in the United States.

30 Las Vegas=Las Vegas=United States=3=Las Vegas is the most populous city in the U.S. state of Nevada and the county seat of Clark County. Las Vegas is an internationally renowned major resort city known primarily for gambling, shopping, fine dining, and nightlife and is the leading financial and cultural center for Southern Nevada. The city bills itself as The Entertainment Capital of the World, and is famous for its consolidated casinohotels and associated entertainment. Today , Las Vegas is one of the top tourist destinations in the world.

31 Mexico City=Mexico City=Mexico=3=Mexico City is the Federal District, capital of Mexico and seat of the federal powers of Mexico. Mexico City is the country's largest city as well as its most important political, cultural, educational and financial center. As an "alpha" global city, Mexico City is one of the most

important financial centers in North America. It is located in the Valley of Mexico , a large valley in the high plateaus at the center of Mexico, at an altitude of 2,240 m.

32 Mississauga=Mississauga=Canada=3=Mississauga Listeni is a city in Southern Ontario, Canada. It lies on the shores of Lake Ontario, located in the Regional Municipality of Peel, in the western part of the Greater Toronto Area. The city has a population of 713,443 as of the Canada 2011 Census, and is Canada's sixth-most populous municipality. Toronto Pearson International Airport, Canada's busiest airport is located in the city, and it is the location of several major corporate headquarters **for** Canada, such as Walmart Canada, Target Canada, Microsoft and GE.

33 Ottawa=Ottawa=Canada=3=Ottawa is the capital of Canada, and the fourth largest city in the country. The city stands on the south bank of the Ottawa River in the eastern portion of Southern Ontario. Ottawa borders Gatineau, Quebec, and together they form the National Capital Region (NCR). Founded in 1826 as Bytown and incorporated as "Ottawa" in 1855, the city has evolved into a political and technological centre of Canada. The name "Ottawa" is derived from the Algonquin word adawe, meaning "to trade".

34 Port-au-Prince=Port-au-Prince=Haiti=3=Port-au-Prince is the capital and largest city of the Caribbean country of Haiti. The city of Port-au-Prince is on the Gulf of Gonâve: the bay on which the city lies, which acts as a natural harbor, has sustained economic activity since the civilizations of the Arawaks. Port-au-Prince was catastrophically affected by an earthquake on January 12, 2010, with large numbers of structures damaged or destroyed.

35 San Jose=San Jose=Costa Rica=3=San Jose is the capital of Costa Rica, head of the province of San José, and the nation's largest city. Located in the Central Valley, San José is the seat of national government, the focal point of political and economic activity, and the major transportation hub of this Central American nation . San José is the sixth most important destination in Latin America, according to The MasterCard Global Destinations Cities Index 2012. San José ranked 15th in the worlds fastest growing destination cities by visitor cross-border spending.

36 St. John's=St. John's=Canada=3=St. John's is the capital and largest city in Newfoundland and Labrador, and is considered by some to be the oldest English-founded city in North America. It is located on the eastern tip of the Avalon Peninsula on the island of Newfoundland. With a population of 200,600 as of 2012, the St. John's Metropolitan Area is the second largest Census Metropolitan Area (CMA) in Atlantic Canada after Halifax and the 20th largest metropolitan area in Canada.

37 St. Pierre et Miquelon==France=3=St. Pierre et Miquelon is a self-governing territorial overseas collectivity of France. It is the only remnant of the former colonial empire of New France that remains under French control. The islands are situated at the entrance of Fortune Bay. They are 3,819km from Brest, the nearest point in Metropolitan France, but just 20km off the Burin Peninsula of Newfoundland , Canada.

38 North Pole=North Pole=Canada=3=The North Pole, also known as the Geographic North Pole or Terrestrial North Pole, is defined as the point in the Northern Hemisphere where the Earth's axis of rotation meets its surface. The North Pole is the northernmost point on the Earth, lying diametrically opposite the South Pole. It defines geodetic latitude 90 degree North, as well as the direction of **true** north. At the North Pole all directions point south; all lines of longitude converge there , so its longitude can be defined as any degree value.

39 Yellowknife=Yellowknife=Canada=3=Yellowknife is the capital and largest city of the Northwest Territories (NT), Canada. It is located on the northern shore of Great Slave Lake, approximately 400 km south of the Arctic Circle, on the west side of Yellowknife Bay near the outlet of the Yellowknife River. Yellowknife and its surrounding water bodies were named after a local Dene tribe once known as the ' Copper Indians' or 'Yellowknife Indians' (now referred to locally as the Yellowknives Dene (First Nation)) who traded tools made from copper deposits near the Arctic Coast.

40 Yorkton=Yorkton=Canada=3=Yorkton is a city located in southeastern Saskatchewan, Canada, near the Manitoba border. Founded and incorporated in 1882 by a group of settlers from Ontario, it has grown to 15,038 residents as of the 2006 census. The city is bordered by the Rural Municipality of Orkney No. 244 and the Rural Municipality of Wallace No. 243. The Yorkton Film Festival has been held there every year since 1947.

41 Caracas=Caracas=Venezuela=4=Caracas, officially Santiago de León de Caracas, is the capital and largest city of Venezuela. Caracas is located in the northern part of the country, following the contours of the narrow Caracas Valley on the Venezuelan coastal mountain range. Terrain suitable **for** building lies between 760 and 910 m above sea level. The valley is close to the Caribbean Sea, separated from the coast by a steep 2,200 m high mountain range, Cerro El Ávila; to the south there are more hills and mountains.

42 Santiago=Santiago=Chile=4=Santiago, also Santiago de Chile, is the capital of Chile and the center of its largest conurbation. It is located in the country's central valley, at an elevation of 520 m above mean sea level. Founded in 1541, Santiago has been the capital city since colonial times. The city has a downtown core of 19 th century neoclassical architecture and winding side-streets, dotted by art deco, neo-gothic, and other styles. Santiago's cityscape is shaped by several stand-alone hills and the fast-flowing Mapocho River, lined by parks such as Parque Forestal.

43 Sao Paulo=Sao Paulo=Brazil=4=Sao Paulo is the largest city in Brazil, the largest city proper in the southern hemisphere, and the world's seventh largest city by population. The metropolis is anchor to the Sao Paulo metropolitan area, ranked as the second most populous metropolitan area in the Americas and among the ten largest metropolitan areas in the world. Sao Paulo is the capital of the state of Sao Paulo, Brazil's most populous state. It exerts strong regional influence in commerce, finance, arts and entertainment and a strong international influence. The name of the city honors Saint Paul of Tarsus.

44 Buenos Aires=Buenos Aires=Argentina=4=uenos Aires is the capital and largest city of Argentina, and the second-largest metropolitan area in South America, after Greater São Paulo. It is located on the western shore of the estuary of the Río de la Plata, on the southeastern coast of the South American continent. The Greater Buenos Aires conurbation, which also includes several Buenos Aires Province districts, constitutes the third-largest conurbation in Latin America, with a population of around thirteen million. Buenos Aires is a top tourist destination, and is known **for** its European-style architecture and rich cultural life, with the highest concentration of theatres in the world.

45 Chernobyl=Chernobyl=Ukraine=5=Chernobyl or Chornobyl is a city in the restricted Chernobyl Exclusion Zone situated in northern Kiev Oblast, Ukraine near the border with Belarus. The city had been the administrative centre of the Chernobyl Raion since 1932. The city was evacuated in 1986 due to the Chernobyl disaster at the Chernobyl Nuclear Power Plant, located 14.5 km north-northwest. The power plant was within Chernobyl Raion (District). After the accident the Chernobyl Raion administration was transferred to the neighboring Ivankiv Raion. Though the city today is mostly uninhabited, a small number of people reside in houses marked with signs stating that the "Owner of this house lives here".

46 Chechnya==Russia=5=The Chechen Republic, commonly referred to as Chechnya, also spelled Chechnia or Chechenia, is a federal subject (a republic) of the Russian Federation. It is located in the North Caucasus, situated in the southernmost part of Eastern Europe, and within 100 kilometers of the Caspian Sea. After the dissolution of the Soviet Union in 1991, the splitted Chechen Republic proclaimed the Chechen Republic of Ichkeria, which sought independence. Following the First Chechen War with Russia, Chechnya gained de facto independence as the Chechen Republic of Ichkeria. Russian federal control was restored during the Second Chechen War.

47 Dublin=Dublin=Ireland=5=Dublin is the capital and most populous city of Ireland.[3][4] The English name **for** the city is derived from the Irish name Dubhlinn, meaning " black pool". Dublin is situated in the province of Leinster near the midpoint of Ireland's east coast, at the mouth of the River Liffey and the centre of the Dublin

Region.

48 London=London=United Kingdom=5=London is the capital city of England and of the United Kingdom. It is the most populous region, urban zone and metropolitan area in the UK. Standing on the River Thames, London has been a major settlement for two millennia, its history going back to its founding by the Romans, who named it Londinium. London is a leading global city, with strengths in the arts, commerce, education, entertainment, fashion, finance, media, professional services, research and development, tourism and transport all contributing to its prominence. It is one of the world's leading financial centres and has the fifth- or sixth-largest metropolitan area GDP in the world depending on measurement.

49 Lyon=Lyon=France=5=Lyon is a city in east-central France in the Rhône-Alpes region, situated between Paris and Marseille. Etymologically it relates to the Celtic God Lugoves, Lugh as **do** Laon and Leiden. Lyon is located approximately 470 km from Paris, 320 km from Marseille, 420 km from Strasbourg, 160 km from Geneva, 280 km from Turin. Together with its suburbs and satellite towns, Lyon forms the largest conurbation in France outside Paris. The city is known **for** its historical and architectural landmarks and is a UNESCO World Heritage Site. Lyon has developed a reputation as the capital of gastronomy in France.

50 Reykjavik=Reykjavik=Iceland=5=Reykjavík is the capital and largest city of Iceland. Its latitude, at 64 degree N, makes it the world's northernmost capital of a sovereign state. It is located in southwestern Iceland, on the southern shore of the Faxaflói Bay. With a population of around 120,000, it is the heart of Iceland's cultural, economic and governmental activity. Reykjavík is believed to be the location of the first permanent settlement in Iceland, which Ingólfur Arnarson is said to have established around AD 870.

51 The Hague=The Hague=Netherlands=5=The Hague is the seat of government in the Netherlands, and the capital city of the province of South Holland. Located in the west of the Netherlands, The Hague is in the centre of the Haaglanden conurbation and lies at the southwest corner of the larger Randstad conurbation. The Hague is the seat of the Dutch government and parliament, the Supreme Court, and the Council of State, but the city is not the capital of the Netherlands which constitutionally is Amsterdam. Most foreign embassies in the Netherlands and 150 international organisations are located in the city, including the International Court of Justice and the International Criminal Court, which makes The Hague one of the major cities hosting the United Nations, along with New York, Vienna, Geneva, Tokyo and Nairobi.

52 Torshavn=Torshavn=Faroe Islands=5=Tórshavn is the capital and largest town of the Faroe Islands. It is located in the southern part on the east coast of Streymoy. To the north west of the town lies the 347-m high mountain Húsareyn, and to the southwest, the 350-m high Kirkjuboreyn. The town has grown rapidly ever since the turn of the 20th century into the undisputed administrative, economic and cultural centre of the Faroes.

53 Nairobi=Nairobi=Kenya=6=Nairobi is the capital and largest city of Kenya. The city and its surrounding area also form the Nairobi County. Nairobi is now one of the most prominent cities in Africa both politically and financially. Home to thousands of Kenyan businesses and over 100 major international companies and organisations, including the UNEP and the main co-ordinating and headquarters **for** the UN in Africa & Middle East, the United Nations Office in Nairobi (UNON), Nairobi is an established hub **for** business and culture. The Nairobi Stock Exchange (NSE) is one of the largest in Africa and the second oldest exchange on the continent.

54 Sierra Leone==Africa=6=Sierra Leone, officially the Republic of Sierra Leone, is a country in West Africa. Sierra Leone has relied on mining, especially diamonds, **for** its economic base. It also has rich deposits of rutile, titanium, bauxite and gold . Despite **this** natural wealth, 70% of its people live in poverty. Between 1991 and 2002, the Sierra Leone Civil War resulted in more than 50,000 people dead, much of the country's infrastructure destroyed, and over two million people displaced as refugees.

55 Antananarivo=Antananarivo=Madagascar=6=Antananarivo, formerly Tananarive, is the capital and largest city in Madagascar. It is also known by its French colonial

shorthand form Tana. The larger urban area surrounding the city, known as Antananarivo-Renivohitra, is the capital of Analamanga Region. The name roughly translates to "City of the Thousand".

56 Dar es Salaam=Dar es Salaam=Tanzania=6=The City of Dar es Salaam, formerly Mzizima, is Tanzania's largest and richest city, serving as a regionally important economic centre. The city is located within the Dar es Salaam Region, an administrative province within Tanzania, and consists of three local government areas or administrative districts: northern Kinondoni, central Ilala, and southern Temeke. Though Dar es Salaam lost its official status as capital city to Dodoma in 1974 (a move not completed until 1996), it remains the locus of the permanent central government bureaucracy, continuing to serve as the capital of the surrounding eponymous region.

57 Garowe=Garowe=Somalia=6=Garowe is the administrative capital of the autonomous Puntland region in northeastern Somalia. The area was ruled in the early modern period by the Majeerteen Sultanate. It would later be incorporated into Italian Somaliland. Garowe is located in the Nugal province, and is the seat of the regional parliament, the presidential palace and government ministries. A fast-growing city, it has also evolved into a local media and cultural hub.

2.11 mst/1.csv

This file is description for the first mission.

1 Caught In The Act=1. you may be killed by the POI;\n2. the operation is exposed and you are injured;\n3. local corrupt authorities refuse to co-operate and may expose the operation to the POI;\n4. the Sector may choose to give the POI immunity in exchange of information, however, the POI may affect your future missions.=You will travel to |LocMst| to follow the POI. The POI's name is |POI|. This POI is related to a semi-underground group called the Blake Association. This group is unkown to the authorities (or ignored by the authorities) as their powerful connections. As your first mission, you need to be extremely cautious. The Sector may provide some help as the importance and danger of this mission. Although the local authorities are important for the Sector to operate, you shouldn't trust them as you trust the Sector as connection of |POI| to the police in **this** particular mission. You must bring |POI| into SCCI's custody to make sure he/she can be prosecuted fairly internationally.\nSCCI thanks for your service."=After you arrived, you got a GPS tracking device from the Sector. The source shows |POI| is on the move. He/she has moved to a location where most of intel sources ignored. You know his/her residence and you come down there.=1

2.12 prbs/1.csv

This file contains problems in the first mission.

1 Addition=In |POI|'s residence, you find a locked box, and you want to know what's inside. The box can be unlocked with a very simple problem, WHAT IS RESULT OF 1 + 1?. The box also has a logo of IBM.=0=10=5=4=You successfully opened the box containing the number and passwords of |POI| and get the GPS data from the POI's cell.=1<The box destroyed itself to protect the POI's info. You need several days more to find the POI.<1<2=0

2 Patient Emergency=The POI goes to a hospital. You checked his/her record. As the POI thinks someone is following him/her, the patient's body temperature is described with an inequality to increase the difficulty. You know the temperature T satisfied the inequality abs(T 38.6) is less than or equal to 2. Find the lowest temperature the patient could have had during the illness in order to know his/her disease.=1=36.6=10=2=You now know the POI is actually faking his/her illness. You

continue to monitor him/her.=1<The POI finds you checking his/her record, you are sedated and moved to a mountain. Your mission is delayed.<2<3=2

3 Fast & Furious=Your source indicate the POI's car traveled 400 kilometres in 4 hours 41 minutes. Now, you need its average speed (to the tenth in km/h) to calculate what can the POI go and **do** under your surveillance.=1=85.4=15=2=The POI's moving radius is known. In the following days, the POI's whereabouts is essential to the mission.=0.5<You cannot find the POI in hours. Fortunately, SCCI got **new** information saving the mission.<0<0]0.5<You loses |POI| and his/her whereabouts is confusing.<1<1=1

4 All Covered=There is an area that the POI would likely to meet someone. You need to cover the area with surveillance equipment. The perimeter of the area is 50 metres, and you need 1 camera **for** every 10 square metres and 1 mic **for** every 5 square metres. Without knowing the actual shape of the area, you need know the maximum amount to ask SCCI to provide equipment. You need the total number of mics and cameras obtaining APPLICATION FORM OF RESTRICTED EQUIPMENT.=0=60=15=2=The area is covered and **if** |POI| enters, the surveillance will start automatically.=0.75<The area is not covered correctly, the POI's action and his/her meeting is not complete. You use several hours to figure out what |POI| is really doing.<1<1]0.25<The area is not covered correctly, the POI's action and his/her meeting is not complete. You use several hours to figure out what the POI is really doing. The evidence obtained is illegible/incomplete to be used against |POI|. You need more days to obtain enough evidence.<2<3=1

5 Je t'attrape=Finally, you get a footage and recording that says |POI| will carry out the first human trafficking deal. You and other SCCI cadet agents are ready to make arrests and end the ring of heinous human trafficking. Now, you are required to make an appropriate ambush plan. You now know there are 200 square metres of a small port to cover and at most 7 criminals. The most important guys are |POI| and the head of another party. You review the old file of an interpol detective who is gone missing in the investigation. There is a note left in the binder cover: 5508 51 6170, the suspects at that time are Bill Craig, Karl Meier, Ellie Stig and Olig Ok. You need the name to determine your strategy and you don't have any time. Who is the real criminal (only need last name)?=3=ok=25=1=You finish your mission successfully, you are now able to operate independently protecting people.=0.25<The name is wrong and you and the cadets are at risk. The mission failed. Suspects at large.<911<911]0.5<The name is wrong and you and the cadets are at risk. You are killed in the shooting.<110<110]0.25<You and the cadets complete the mission .<0<0;=0

Part IV

Evaluation

Over all, this project contains 1056 lines of Java codes.

Chapter 3

Program Structure

The detailed description of classes, constants, methods and variables are listed in this chapter, following the alphabetical order of class names. The purpose and the parameter and return value for methods are described in order to provide a development guide for others. This chapter is automatically generated with `javadoc` using TeXDoclet at `http://doclet.github.io/` from the source code.

3.1 Class Agent

Agent of the organization
ref: `http://docs.oracle.com/javase/tutorial/java/javaOO/index.html`

3.1.1 Declaration

public class Agent
extends personOfInterest.Person (in 3.5, page 77)

3.1.2 All known subclasses

Player (in 3.6, page 79)

3.1.3 Field summary

base base location of an agent
calendar calendar of an agent
currentDay current day in the calendar
id an agent's id number
rank rank of an agent
RANKS ranks' list
ref: `http://java.about.com/od/understandingdatatypes/a/Using-Constants.htm`
serialVersionUID serialization id

3.1.4 Constructor summary

Agent(String, int, Location) Construct a new agent with specified name, rank and base location and a random id

Agent(String, int, Location, int) Construct a new agent with specified name, rank and base location and a random id

3.1.5 Method summary

displayInfo() Display an agent's info including rank, name, base and ID
passDay() Pass current day
passDays(int) Pass a certain number of days
resetCal() Reset calendar of an agent
resetCal(int) Reset calendar of an agent to a certain period

3.1.6 Fields

- private static final long **serialVersionUID**
  - serialization id
- Location **base**
  - base location of an agent
- int **id**
  - an agent's id number
- int **rank**
  - rank of an agent
- public static final java.lang.String **RANKS**
  - ranks' list
    ref: `http://java.about.com/od/understandingdatatypes/a/Using-Constants.htm`
- int **calendar**
  - calendar of an agent
- int **currentDay**
  - current day in the calendar

3.1.7 Constructors

- **Agent**
  `public` **Agent**`(java.lang.String` **inName,** `int` **inRank,** `Location` **inBase**`)`
  - **Description**
    Construct a new agent with specified name, rank and base location and a random id
  - **Parameters**
    - `inName` – specified name for the new agent
    - `inRank` – specified rank for the new agent
    - `inBase` – specified base location for the new agent

- **Agent**
  `public` **Agent**`(java.lang.String` **inName,** `int` **inRank,** `Location` **inBase,** `int` **inId**`)`
  - **Description**
    Construct a new agent with specified name, rank and base location and a random id
  - **Parameters**
    * `inName` – specified name for the new agent
    * `inRank` – specified rank for the new agent
    * `inBase` – specified base location for the new agent
    * `inId` – specified id number for the new agent

3.1.8 Methods

- **displayInfo**
  `public void` **displayInfo**`()`
  - **Description**
    Display an agent's info including rank, name, base and ID
- **passDay**
  `public void` **passDay**`()`
  - **Description**
    Pass current day
- **passDays**
  `public void` **passDays**`(int` **days**`)`
  - **Description**
    Pass a certain number of days
  - **Parameters**
    * `days` – the number of days passed
- **resetCal**
  `public void` **resetCal**`()`
  - **Description**
    Reset calendar of an agent
- **resetCal**
  `public void` **resetCal**`(int` **days**`)`
  - **Description**
    Reset calendar of an agent to a certain period

3.1.9 Members inherited from class Person

`personOfInterest.Person` (in 3.5, page 77)

- `public void` **displayInfo**`()`
- **ifAlive**
- `public void` **kill**`()`
- **location**
- **name**
- `private static final` **serialVersionUID**

3.2 Class Game

Main game program

3.2.1 Declaration

public class Game
extends java.lang.Object

3.2.2 Field summary

gameCal game calendar
input scanner that reads from keyboard
locArea
newMission if the player wants to start a new mission
poiArea
processed
savedName saved name for saving game
savedPath saved path for saving game

3.2.3 Method summary

addLinebreaks(String) Break lines in a long string with maximum length of 80
addLinebreaks(String, int) Break lines in a long string to make display nicer
ref: `http://stackoverflow.com/questions/7528045/large-string-split-into-lines-with-maximum-length-in-java`
displayDate() Display the date today (as in the game)
help() Display help information
ifGoodCalendar(int[]) Determine if a player complete his/her mission in a certain time
input(String) Get the user's input with space as the delimiter
input(String, int, int) Check if the user's input is valid and return a valid data
inputLn(String) Get the user's input with line break as the delimiter
isGameOver(Player) Determine if the game is over
loadFile(String) Load a serialized file
locs() Update location information from the csv file
main(String[]) Main program
msts() Update location information from the csv file
passDay() Add 1 day to the calendar

passDays(int) Add a certain number of days to the game calendar
pois() Update location information from the csv file
prbs() Update problems
processUserInput(Problem, Player) Process the user's input that intends for a problem
processUserInput(String, Player) Process the user's input
randomID() Generate a 9-digits id number for agent id, mission id, etc.
randomNum(int, int) Generate a random number in a certain range
saveGame(Player) Save the game
scciSeal() Print SCCI logo
showStartPage() Show the start page with ASCII art
sleep(int) Pause the program for a period of time while outputing three dots
ref: `http://docs.oracle.com/javase/tutorial/essential/concurrency/sleep.html`
takeChance(double)
takeChance(int)
toBoolean(int) Cast int to boolean
transport(Agent, int, Location, Location) Transport an agent from his/her current location to a location requested through a certain way
typeString(String) Show a string in typewriter style

3.2.4 Fields

- static java.util.Scanner **input**
  - scanner that reads from keyboard
- static java.lang.String **savedPath**
  - saved path for saving game
- static java.lang.String **savedName**
  - saved name for saving game
- static java.util.Calendar **gameCal**
  - game calendar
- static int **locArea**
- static int **poiArea**
- static boolean **newMission**
  - if the player wants to start a new mission
- static boolean **processed**

3.2.5 Methods

- **addLinebreaks**
  `public static java.lang.String` **addLinebreaks**`(java.lang.String` **input**`)`

- **Description**
  Break lines in a long string with maximum length of 80
- **Parameters**
  * `input` – the string need to be broken
- **Returns** – the string with proper breaks

- **addLinebreaks**
  `public static java.lang.String` **addLinebreaks**`(java.lang.String` **input,** `int` **maxLineLength**`)`
  - **Description**
    Break lines in a long string to make display nicer
    ref: `http://stackoverflow.com/questions/7528045/large-string-split-into-lines-with-maximum-length-in-java`
  - **Parameters**
    * `input` – the string need to be broken
    * `maxLineLength` – the maxium line length to insert break line
  - **Returns** – the string with proper breaks

- **displayDate**
  `public static java.lang.String` **displayDate**`()`
  - **Description**
    Display the date today (as in the game)
  - **Returns** – the current date in the game

- **help**
  `public static void` **help**`() throws java.io.FileNotFoundException`
  - **Description**
    Display help information

- **ifGoodCalendar**
  `public static boolean` **ifGoodCalendar**`(int[]` **calendar**`)`
  - **Description**
    Determine if a player complete his/her mission in a certain time
  - **Parameters**
    * `calendar` – the calendar of the player/mission
  - **Returns** – true if the player doesn't fail to comply the calendar; false otherwise

- **input**
  `public static java.lang.String` **input**`(java.lang.String` **prompt**`) throws java.io.IOException`
  - **Description**
    Get the user's input with space as the delimiter

- **Parameters**
  - `prompt` – prompt for the user
- **Returns** – the string user entered

- **input**
  `public static int` **input**`(java.lang.String` **prompt,** `int` **min,** `int` **max**`)`
  - **Description**
    Check if the user's input is valid and return a valid data
  - **Parameters**
    - `prompt` – prompt for the user
    - `min` – the minimum value for a valid data
    - `max` – the maximum value for a valid data
  - **Returns** – a valid integer data between min and max

- **inputLn**
  `public static java.lang.String` **inputLn**`(java.lang.String` **prompt**`)`
  - **Description**
    Get the user's input with line break as the delimiter
  - **Parameters**
    - `prompt` – prompt for the user
  - **Returns** – the string user entered

- **isGameOver**
  `public static boolean` **isGameOver**`(Player` **player**`)`
  - **Description**
    Determine if the game is over
  - **Returns** – true if the player is alive and not completed the game; false otherwise

- **loadFile**
  `public static java.lang.Object` **loadFile**`(java.lang.String` **prompt**`)`
  `throws java.io.IOException`
  - **Description**
    Load a serialized file
  - **Parameters**
    - `prompt` – prompt for the user
  - **Returns** – the object included in the file

- **locs**
  `public static void` **locs**`() throws java.lang.NumberFormatException,`
  `java.io.IOException, java.lang.InterruptedException`
  - **Description**
    Update location information from the csv file

- **main**
  `public static void` **main**`(java.lang.String[]` **args**`) throws java.io.IOException, java.lang.ClassNotFoundException, java.lang.InterruptedException`
  - **Description**
    Main program
- **msts**
  `public static void` **msts**`() throws java.lang.NumberFormatException, java.io.IOException, java.lang.InterruptedException`
  - **Description**
    Update location information from the csv file
- **passDay**
  `public static void` **passDay**`()`
  - **Description**
    Add 1 day to the calendar
- **passDays**
  `public static void` **passDays**`(int` **days**`)`
  - **Description**
    Add a certain number of days to the game calendar
  - **Parameters**
    * `days` – the certain number of days
- **pois**
  `public static void` **pois**`() throws java.lang.NumberFormatException, java.io.IOException, java.lang.InterruptedException`
  - **Description**
    Update location information from the csv file
- **prbs**
  `public static void` **prbs**`() throws java.io.IOException`
  - **Description**
    Update problems
- **processUserInput**
  `public static boolean` **processUserInput**`(Problem` **problem**`, Player` **player**`) throws java.io.IOException, java.lang.ClassNotFoundException, java.lang.InterruptedException`
  - **Description**
    Process the user's input that intends for a problem
  - **Parameters**

    * `problem` – the problem
    * `player` – the player
  - **Returns** – true if the user answers the question correctly; false if answers incorrectly or doesn't answer

- **processUserInput**
  `public static boolean` **processUserInput**`(java.lang.String` **uInput,** `Player` **player**`) throws java.io.FileNotFoundException, java.io.IOException, java.lang.ClassNotFoundException, java.lang.InterruptedException`
  - **Description**
    Process the user's input
  - **Parameters**
    * `uinput` – the user's input
    * `player` – the player
  - **Returns** – true if the input is processed; false otherwise

- **randomID**
  `public static int` **randomID**`()`
  - **Description**
    Generate a 9-digits id number for agent id, mission id, etc.
  - **Returns** – the generated 9-digits number

- **randomNum**
  `public static int` **randomNum**`(int` **min,** `int` **max**`)`
  - **Description**
    Generate a random number in a certain range
  - **Parameters**
    * `min` – the minimum for the number, inclusive
    * `max` – the maximum for the number, inclusive
  - **Returns** – the generated number

- **saveGame**
  `public static boolean` **saveGame**`(Player` **player**`) throws java.io.FileNotFoundException, java.io.IOException`
  - **Description**
    Save the game
  - **Parameters**
    * `player` – the game's player
  - **Returns** – true if the saving is a success; false otherwise

- **scciSeal**
  `public static void` **scciSeal**`()`

- **Description**
  Print SCCI logo

- **showStartPage**
  `public static void` **showStartPage**`()`
  - **Description**
    Show the start page with ASCII art

- **sleep**
  `public static void` **sleep**`(int` **timeInMs**`) throws java.lang.InterruptedException`
  - **Description**
    Pause the program for a period of time while outputing three dots
    ref: `http://docs.oracle.com/javase/tutorial/essential/concurrency/sleep.html`
  - **Parameters**
    * `timeInMs` – the period of time to be paused for

- **takeChance**
  `public static boolean` **takeChance**`(double` **chance**`)`

- **takeChance**
  `public static boolean` **takeChance**`(int` **divisor**`)`

- **toBoolean**
  `public static boolean` **toBoolean**`(int` **num**`)`
  - **Description**
    Cast int to boolean
  - **Parameters**
    * `num` – the int ready to be casted
  - **Returns** – false if the int is 0; true otherwise

- **transport**
  `public static void` **transport**`(Agent` **agent,** `int` **method,** `Location` **from,** `Location` **to**`)`
  - **Description**
    Transport an agent from his/her current location to a location requested through a certain way
  - **Parameters**
    * `agent` – the agent that need to be transported
    * `method` – the way of transportation; 1 for train, 2 for plane, 3 for normal vehicle and 4 for high-speed rail
    * `from` – the current location
    * `to` – the location requested

- **typeString**
  `public static void` **typeString**`(java.lang.String` **message**`) throws java.lang.InterruptedException`
  - **Description**
    Show a string in typewriter style
  - **Parameters**
    * `message` – the string need to be displayed

3.3 Class Location

Location

3.3.1 Declaration

public class Location
extends java.lang.Object
implements java.io.Serializable

3.3.2 Field summary

area a location's area as the index in AREAS
AREAS areas that a location can be included in
city city that a location situated
country country that a location situated
description description of a location
name a location's name
serialVersionUID serialization id
sublocations locations that are in a location
subNum the number of sublocations

3.3.3 Constructor summary

Location(String, String, String, int, String) Construct a location with name.

3.3.4 Method summary

addSublocation(Location) Add a sublocation
addSublocations(Location[]) Add a series of sublocation

3.3.5 Fields

- private static final long **serialVersionUID**
  - serialization id
- java.lang.String **name**
  - a location's name

- int **area**
  - a location's area as the index in AREAS
- java.lang.String **city**
  - city that a location situated
- java.lang.String **country**
  - country that a location situated
- Location **sublocations**
  - locations that are in a location
- int **subNum**
  - the number of sublocations
- java.lang.String **description**
  - description of a location
- static final java.lang.String **AREAS**
  - areas that a location can be included in

3.3.6 Constructors

- **Location**
  `public` **Location**`(java.lang.String` **inName,** `java.lang.String` **inCity,** `java.lang.String` **inCountry,** `int` **inArea,** `java.lang.String` **inDescription**`)`
  - **Description**
    Construct a location with name.
  - **Parameters**
    - `inName` – the location's name

3.3.7 Methods

- **addSublocation**
  `public void` **addSublocation**`(Location` **subloc**`)`
  - **Description**
    Add a sublocation
  - **Parameters**
    - `subloc` – the sublocation that needs to be added
- **addSublocations**
  `public void` **addSublocations**`(Location[]` **sublocs**`)`
  - **Description**
    Add a series of sublocation
  - **Parameters**
    - `sublocs` – the sublocations that needs to be added

3.4 Class Mission

Mission of the player

3.4.1 Declaration

public class Mission
extends java.lang.Object

3.4.2 Field summary

arriveMessage message displayed after arrival
budget money that can be used to complete a mission
calendar calendar of a mission; time permitted
completed if a mission is completed
dangers potential dangers of a mission
description description of a mission
location location that a mission takes place
minorNum indication of where minor situations happens
name name of a mission
player player who is completing a mission
poi person of interest in a mission
points points that awarded after completion
prbNum default number for problem
problems situations in a mission
rank rank that have abilities to complete a mission

3.4.3 Constructor summary

Mission(POI, String, Location, int, int) Construct a mission
Mission(POI, String, Location, int, int, int)

3.4.4 Method summary

complete() Complete a mission
completing(int) Process a mission
generateFinalMission(int) Generate a final Mission
generateMission(int, int) Generate a random mission
minorSituation(int, int) Execute a customized minor situation
replaceName(String, Mission) Replace certain tags in a string with certain variables

3.4.5 Fields

- Location **location**
  - location that a mission takes place
- java.lang.String **name**
  - name of a mission

- POI **poi**
  - person of interest in a mission
- boolean **completed**
  - if a mission is completed
- int **rank**
  - rank that have abilities to complete a mission
- int **budget**
  - money that can be used to complete a mission
- int **calendar**
  - calendar of a mission; time permitted
- int **points**
  - points that awarded after completion
- Player **player**
  - player who is completing a mission
- java.lang.String **dangers**
  - potential dangers of a mission
- java.lang.String **description**
  - description of a mission
- Problem **problems**
  - situations in a mission
- java.lang.String **arriveMessage**
  - message displayed after arrival
- int **minorNum**
  - indication of where minor situations happens
- static int **prbNum**
  - default number for problem

3.4.6 Constructors

- **Mission**
  `public` **Mission**`(POI` **person**`, java.lang.String` **inname**`, Location` **inloc**`, int` **days**`, int` **inPoints**`)`
  - **Description**
    Construct a mission
  - **Parameters**
    * `person` – the POI in a mission
    * `inname` – the name of a mission
    * `inloc` – the location of a mission
    * `days` – the time limit in days for the mission
    * `inPoints` – the points required to complete the mission
- **Mission**
  `public` **Mission**`(POI` **person**`, java.lang.String` **inname**`, Location` **inloc**`, int` **days**`, int` **inPoints**`, int` **rank**`) throws java.io.FileNotFoundException, java.io.IOException, java.lang.ClassNotFoundException`

3.4.7 Methods

- **complete**
  `public void` **complete**`()`
  - **Description**
    Complete a mission
  - **Parameters**
    * `player` – the player who completed the mission
- **completing**
  `public boolean` **completing**`(int` **prbsNum**`) throws java.lang.InterruptedException, java.lang.ClassNotFoundException, java.io.IOException`
  - **Description**
    Process a mission
  - **Parameters**
    * `prbsNum` – the start number of problems in a mission
  - **Returns** – true if complete the mission; false otherwise
- **generateFinalMission**
  `public static Mission` **generateFinalMission**`(int` **area**`) throws java.io.FileNotFoundException, java.io.IOException, java.lang.ClassNotFoundException`
  - **Description**
    Generate a final Mission

- **Parameters**
  - `area` – the area of the mission
- **Returns** – the generated final mission

- **generateMission**
  `public static Mission` **generateMission**`(int` **area,** `int` **rank**`)`
  `throws java.io.FileNotFoundException, java.io.IOException,`
  `java.lang.ClassNotFoundException`
  - **Description**
    Generate a random mission
  - **Parameters**
    - `area` – the area of the mission
    - `rank` – the rank that the mission is designed for
  - **Returns** – generated mission object

- **minorSituation**
  `public void` **minorSituation**`(int` **rank,** `int` **index**`) throws`
  `java.lang.InterruptedException`
  - **Description**
    Execute a customized minor situation
  - **Parameters**
    - `index` – the index of the minor situation

- **replaceName**
  `public static java.lang.String` **replaceName**`(java.lang.String` **s,**
  `Mission` **mission**`)`
  - **Description**
    Replace certain tags in a string with certain variables
  - **Parameters**
    - `s` – the string need to be replaced
    - `mission` – the mission that indicate variables
  - **Returns** – the string that the certain tags are replaced

3.5 Class Person

Person including POI, agents of the organization, and others

3.5.1 Declaration

public class Person
extends java.lang.Object
implements java.io.Serializable

3.5.2 All known subclasses

Player (in 3.6, page 79), Agent (in 3.1, page 62), POI (in 3.7, page 80)

3.5.3 Field summary

ifAlive person's living status
location person's location
name person's name
serialVersionUID serialization id

3.5.4 Constructor summary

Person()

3.5.5 Method summary

displayInfo() Display a person's information
kill() Kill a person

3.5.6 Fields

- private static final long **serialVersionUID**
  - serialization id
- java.lang.String **name**
  - person's name
- boolean **ifAlive**
  - person's living status
- Location **location**
  - person's location

3.5.7 Constructors

- **Person**
  `public` **Person**`()`

3.5.8 Methods

- **displayInfo**
  `public void` **displayInfo**`()`
  - **Description**
    Display a person's information
- **kill**
  `public void` **kill**`()`
  - **Description**
    Kill a person

3.6 Class Player

Player who play the game in the virtual world
ref: `http://docs.oracle.com/javase/tutorial/java/javaOO/index.html`

3.6.1 Declaration

public class Player
extends personOfInterest.Agent (in 3.1, page 62)

3.6.2 Field summary

missionCompleted number of completed missions
points a player's points used in promotion system
serialVersionUID serialization id

3.6.3 Constructor summary

Player(String, int, Location) Construct a player with a name, a rank and a base

3.6.4 Method summary

demote() Demote a player
promote() Promote a player's rank according to his/her points

3.6.5 Fields

- private static final long **serialVersionUID**
  - serialization id
- int **missionCompleted**
  - number of completed missions
- int **points**
  - a player's points used in promotion system

3.6.6 Constructors

- **Player**
  `public` **Player**`(java.lang.String` **inName,** `int` **inRank,** `Location` **inBase**`)`
  - **Description**
    Construct a player with a name, a rank and a base
  - **Parameters**
    * `inName` – specified name
    * `inRank` – specified rank
    * `inBase` – specified base

3.6.7 Methods

- **demote**
  `public void` **demote**`()`
  - **Description**
    Demote a player
- **promote**
  `public void` **promote**`()`
  - **Description**
    Promote a player's rank according to his/her points

3.6.8 Members inherited from class Agent

`personOfInterest.Agent` (in 3.1, page 62)

- **base**
- **calendar**
- **currentDay**
- `public void` **displayInfo**`()`
- **id**
- `public void` **passDay**`()`
- `public void` **passDays**`(int` **days**`)`
- **rank**
- `public static final` **RANKS**
- `public void` **resetCal**`()`
- `public void` **resetCal**`(int` **days**`)`
- `private static final` **serialVersionUID**

3.6.9 Members inherited from class Person

`personOfInterest.Person` (in 3.5, page 77)

- `public void` **displayInfo**`()`
- **ifAlive**
- `public void` **kill**`()`
- **location**
- **name**
- `private static final` **serialVersionUID**

3.7 Class POI

Persons of Interest
ref: `http://docs.oracle.com/javase/tutorial/java/javaOO/index.html`

3.7.1 Declaration

public class POI
extends personOfInterest.Person (in 3.5, page 77)

3.7.2 Field summary

area area of a POI
id POI's id number
ifCriminal true is the POI is criminal; false otherwise
serialVersionUID serialization id

3.7.3 Constructor summary

POI(String, boolean) Construct a POI with a name specifying if he/she is a criminal

3.7.4 Method summary

getID() Get POI's ID

3.7.5 Fields

- private static final long **serialVersionUID**
  - serialization id
- boolean **ifCriminal**
  - true is the POI is criminal; false otherwise
- int **id**
  - POI's id number
- int **area**
  - area of a POI

3.7.6 Constructors

- **POI**
  `public` **POI**`(java.lang.String` **inName,** `boolean` **criminality**`)`
  - **Description**
    Construct a POI with a name specifying if he/she is a criminal
  - **Parameters**
    - `inName` – POI's name
    - `criminality` – true if the POI is a criminal; false otherwise

3.7.7 Methods

- **getID**
  `public int` **getID**`()`
  - **Description**
    Get POI's ID

3.7.8 Members inherited from class Person

`personOfInterest.Person` (in 3.5, page 77)

- `public void` **displayInfo**`()`
- **ifAlive**
- `public void` **kill**`()`
- **location**
- **name**
- `private static final` **serialVersionUID**

3.8 Class Problem

Problem that will occur in a mission
ref: `http://docs.oracle.com/javase/tutorial/java/javaOO/index.html`

3.8.1 Declaration

public class Problem
extends java.lang.Object
implements java.io.Serializable

3.8.2 Field summary

answer a problem's answer; might be integer, string, boolean, etc.
chances maximum chances to answer a problem
days days that a mission used
description details of a problem
goodMessage message after good answer
location location where a problem occurred
mission mission that contains a problem
name a problem's name
points value of a problem in points (promotion system)
punishment punishment
serialVersionUID serialization id
type type of a problem's answer; 0 for int, 1 for double, 2 for boolean, 3 for String

3.8.3 Constructor summary

Problem(String, String, int, Object, int, int) Construct a problem

3.8.4 Method summary

getType() Get a problem's type
ifCorrect(Object) Check if a response from a source is correct
punishment() Punishment for wrong answers

3.8.5 Fields

- private static final long **serialVersionUID**
  - serialization id
- Mission **mission**
  - mission that contains a problem
- java.lang.String **name**
  - a problem's name
- java.lang.String **description**
  - details of a problem
- java.lang.Object **answer**
  - a problem's answer; might be integer, string, boolean, etc.
- int **type**
  - type of a problem's answer; 0 for int, 1 for double, 2 for boolean, 3 for String
- Location **location**
  - location where a problem occurred
- int **points**
  - value of a problem in points (promotion system)
- int **chances**
  - maximum chances to answer a problem
- java.lang.String **goodMessage**
  - message after good answer
- int **days**
  - days that a mission used
- java.lang.String **punishment**
  - punishment

3.8.6 Constructors

- **Problem**
  `public` **Problem**`(java.lang.String` **inName,** `java.lang.String` **inDesc,** `int` **inType,** `java.lang.Object` **inAnswer,** `int` **inPoints,** `int` **inChances**`)`
  - **Description**
    Construct a problem
  - **Parameters**
    - `inName` – problem's name
    - `inDesc` – problem's description
    - `type` – type of problem answer
    - `answer` – problem's answer

3.8.7 Methods

- **getType**
  `public java.lang.String` **getType**`()`
  - **Description**
    Get a problem's type
  - **Returns** – the type of the problem
- **ifCorrect**
  `public boolean` **ifCorrect**`(java.lang.Object` **response**`)`
  - **Description**
    Check if a response from a source is correct
  - **Parameters**
    * `response` – the response from the user
  - **Returns** – if the response is consistent with the answer
- **punishment**
  `public boolean` **punishment**`()`
  - **Description**
    Punishment for wrong answers

3.9 Class Saved

Saved file for the game

3.9.1 Declaration

public class Saved
extends java.lang.Object
implements java.io.Serializable

3.9.2 Field summary

missionNum mission number
prbNum problem number
savedLocation location saved
savedPlayer player saved
serialVersionUID serialization id

3.9.3 Constructor summary

Saved(Player) Construct a Saved class containing player only
Saved(Player, Location) Construct a Saved class containing player and location

3.9.4 Fields

- private static final long **serialVersionUID**
  - serialization id
- public Player **savedPlayer**
  - player saved
- public Location **savedLocation**
  - location saved
- public int **missionNum**
  - mission number
- public int **prbNum**
  - problem number

3.9.5 Constructors

- **Saved**
  `public` **Saved**`(Player` **player**`)`
  - **Description**
    Construct a Saved class containing player only
  - **Parameters**
    * `player` – the player intending to be saved
- **Saved**
  `public` **Saved**`(Player` **player**`,` `Location` **location**`)`
  - **Description**
    Construct a Saved class containing player and location
  - **Parameters**
    * `player` – the player intending to be saved
    * `location` – the location intending to be saved

Chapter 4

Program Playability

The playability of the game could be assessed from different aspects: playing, story, effects, mechanics and usability.

The game has different missions with different plots. The plots are most adopt from varies of TV dramas, movies, novels and documentaries. The plots are logic and related to mission. The attraction to the gamers is the feeling of being a 'hero' saving people and use mind to solve problems. Varies of applied maths, physics and chemistry are presented in the game. The continuous storyline keeps player to complete missions and stay on the game. However, the plots and situations in them are not with much variety due to the limitation of programming skills, game experiences and time.

For a command line game, the effects are not so exciting like in those video-games (e.g. League of Legends, Civilization). But, the game has effects such as typewriter-style output and time-delay/waiting in order to enhance user experience. The user interface are designed so that the player would clear see the description, the task and the expected input. Help command is available for first-timers.

For mechanics of the game, the response to user's input are reasonable and complying with the storyline. There are several accidents happening so the user will not be too boring. The storylines are not out of life context and the player has control over them. `myinfo` command is available to make player have clear goal and identity his/her status.

The playability of the game can be improved. The improvements are listed in later chapter.

Chapter 5

Program Expansibility

In the process of designing and programming this game, the benefits of letting users to modify and expand the game plot are noticed. The game is programmed that most resources, such as the names of POIs, the descriptions, dangers and arrival messages of missions, the names, cities, countries and descriptions of locations and the names, descriptions, answers and their types, chances, points, punishments and time used of problems, are obtained from CSV files which are easy to modify with spreadsheet softwares. The program portion of the CSV reading is referred to and modified from `http://www.mkyong.com/java/how-to-read-and-parse-csv-file-in-java/`, whose author is mkyong. However, the expansibility can be improved if the details of minor situations could be saved in CSV files, while my limited experience and ability stopped me.

The program codes are with comments explaining the purpose of the classes, constants, variables and methods. One with basic knowledge of Java could modify the program easily. There are some variables created but not really used as they are designed because of time and ability limitation for this project. Those variable are useful for future development when the game has more complex plot. The minor situations can be coded for each rank following the example. The source code of this program is released, so there could be more people adding location informations (see LICENSE on the back of the cover).

Chapter 6

Initial Design and Final Product and Improvement when with more Knowledge and time

There are several differences between the initial design and the final product. As well, the game can be improved various ways with more time and knowledge. There are also problems with the program existing that could not be fixed with current knowledge. The basic indicators of success are met.

6.1 Problems' Generation and Plots

In the initial design, the problems/situations in missions are supposed to be generated randomly, however, the current knowledge and experience of java limits the logic of mission plot. Both variety and storyline are important for a game. But, in the current state of the project, the problems are written in separate mission files in chronological order sacrificing variety. In the future development, the generation should be not only random but also relevant to the missions, and also logic.

Plots of the game are important to players as they provide exciting and challenging feeling for them. The plots' addition is dependent on the creativity and the ability to find appropriate dramas, literature, etc. to adapt. The source code of this program is released, so there could be more people adding location information (see LICENSE on the back of the cover).

6.2 Detailed Locations

There is an idea that a map of a country, a city, or even a building in the city should be outlined. This feature is possible by the unused variable `Locations[] sublocations`. This array stores sublocation data. However, in the process of programming, it is difficult and extremely time-consuming to make and coordinate so many locations and their sublocations. The time limits the program to the current state. But with the improvement of skills, this problem should be less time-consuming as there should be a way to generate them randomly and fast. The source code of this program is released, so there could be more people adding location information (see LICENSE on the back of the cover).

6.3 "Answer or Command"

In the program, when there is a new situation, the player is asked to solve the situation. Sometimes, the answer could also be a command. At that time, the case is ambiguous. For now, the answer is the priority. However, a better solution should be faced by the players ensuring the player will not be punished because of the program.

6.4 Problems' Difficulties

The problems are more difficult when the rank increases. The feedback from test players shows that the problems' difficulties are not increased very evenly. Although an effort is made to select problems from different areas, the ability of different people are very diverse and a better selection of problems should be made. Another thought is let the player choose the difficulties, even that the program can generate different players with different attributes (i.e. some are better at algebra, some are better at physics and others are better at chemistry) and select different problems for them.

6.5 Multi-player and More Characters

The feature of multi-player/more characters is made possible by class `Person` and its subclasses like `Agent`. As well, the ranks in `Agent` make a more complex operation structure of the organization possible. Those classes can be extended to add methods for interactions. The intelligence of these characters is a very difficult problem. Multi-player is hard to operate in a command-line environment. But these features can increase the excitement of the game dramaticly.