CONTROL FLOW GRAPH

SOFTWARE TESTING

PROJECT

BY,

Karthik Kumarasubramanian - 1001549999

Prasanna Venkatesh Sridhar - 1001581554

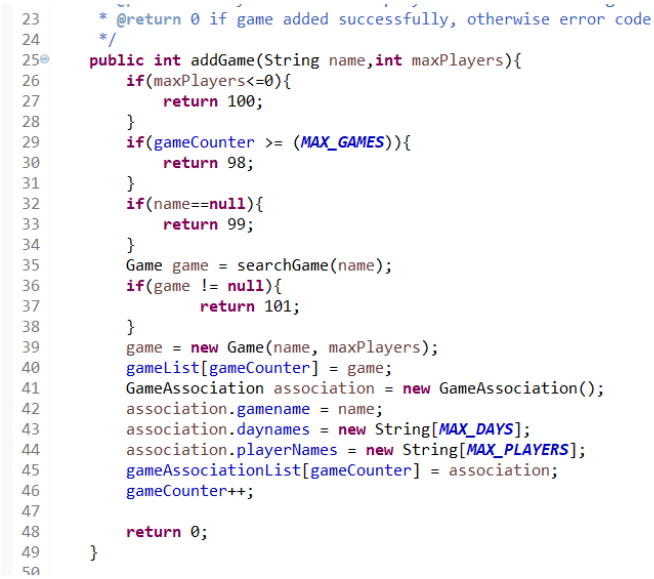
IMPORTANT NOTES

1. The sub-divisions for each method is explained as,
    a. This division attaches the screen shot of the method under test.
    b. This division gives the basic block table for the method under test.
    c. This division gives the CFG for the method under test.

2. The control flow graphs were drawn using LudicChart.com. Hence the end nodes do not have double circles.

1. Method – addGame()

a. The screenshot for the function is given as,

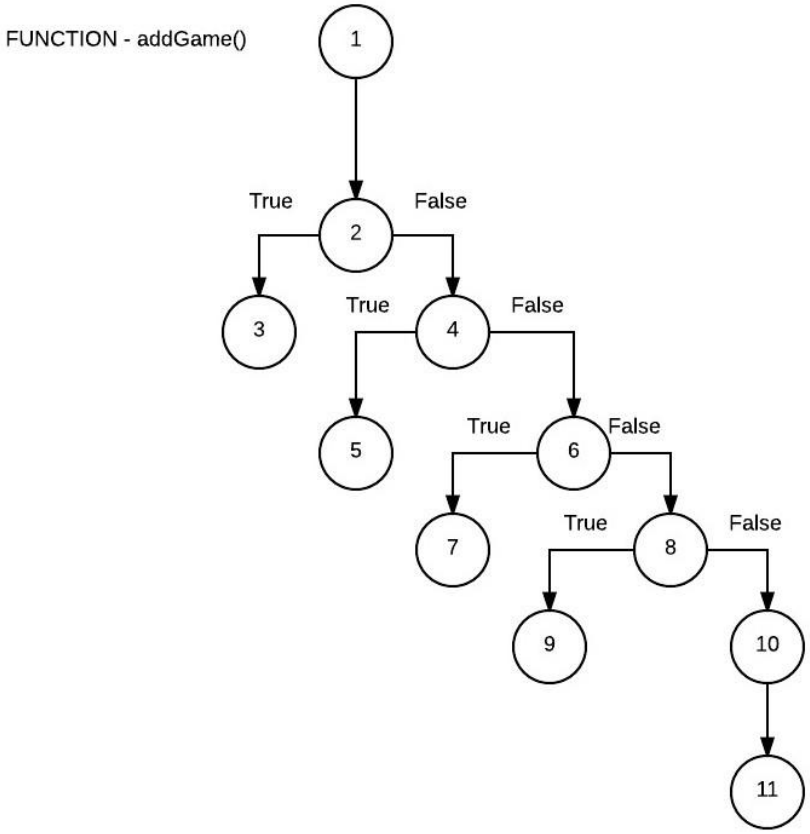


```
23      * @return 0 if game added successfully, otherwise error code
24      */
25     public int addGame(String name,int maxPlayers){
26         if(maxPlayers<=0){
27             return 100;
28         }
29         if(gameCounter >= (MAX_GAMES)){
30             return 98;
31         }
32         if(name==null){
33             return 99;
34         }
35         Game game = searchGame(name);
36         if(game != null){
37                 return 101;
38         }
39         game = new Game(name, maxPlayers);
40         gameList[gameCounter] = game;
41         GameAssociation association = new GameAssociation();
42         association.gamename = name;
43         association.daynames = new String[MAX_DAYS];
44         association.playerNames = new String[MAX_PLAYERS];
45         gameAssociationList[gameCounter] = association;
46         gameCounter++;
47
48         return 0;
49     }
```

b. The basic block table for the given function is given as,

BLOCK	LINES	ENTRY	EXIT
2	26,27	26	27
4	29,30	29	30
6	32,33	32	33
8	35,36,37	36	37
10	39,40,41,42,43,44,45,46,48	39	48

c. The Control Flow Graph for the function is given as,



2. Method – searchGame()

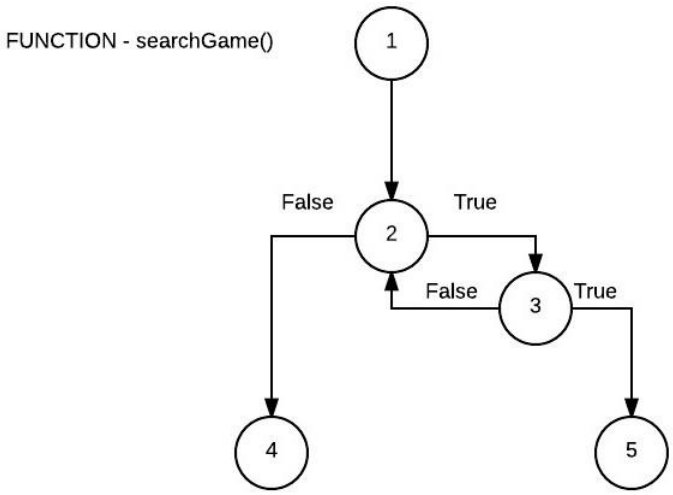
a. The screenshot for the function is given as,

```
52      * This method is used to search game by name
53      * @param name : used to search game by name
54      * @return game object if found else null
55      */
56     public Game searchGame(String name) {
57         for(int i=0; i < gameCounter; i++){
58             Game storedGame = gameList[i];
59             if(storedGame.name.equals(name)){
60                 return storedGame;
61             }
62         }
63         return null;
64     }
```

b. The basic block table for the given function is given as,

BLOCK	LINES	ENTRY	EXIT
2	57,58,63	57	63
3	59,60	59	60

c. The Control Flow Graph for the function is given as,



3. Method – addPlayer()

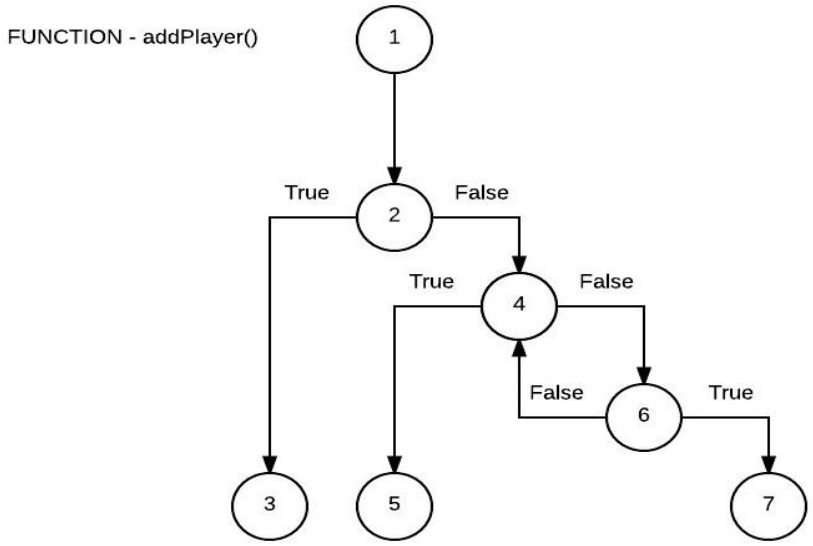
a. The screenshot for the function is given as,

```
69      * @param gameNames : name of games player is playing
70      * @return string indicating successful addition otherwise error
71      */
72     public String addPlayer(String name,String[] gameNames){
73         Player player = searchPlayer(name);
74         if(player != null){
75             return name+" already exists";
76         }
77         //verify every gameName for its validity
78         Game[] gamesPlayed = new Game[gameNames.length];
79         for(int i=0; i < gameNames.length; i++){
80             String gameName = gameNames[i];
81             Game storedGame = searchGame(gameName);
82             if(storedGame==null){
83                 return "Error you cannot be registered for "+gameName;
84             }
85             gamesPlayed[i] = storedGame;
86             GameAssociation association = searchAssociation(storedGame.name);
87             association.playerNames[association.noofPlayers++]=name;
88         }
89         player = new Player(name,gamesPlayed);
90         playerList[playerCounter] = player;
91         playerCounter++;
92         return name+" added successfully";
93     }
```

b. The basic block table for the given function is given as,

BLOCK	LINES	ENTRY	EXIT
2	74,75	74	75
4	78,79,80,81,85,86,87,89,90,91,92	78	92
6	82,83	82	83

c. The Control Flow Graph for the function is given as,



4. Method – addSchedule()

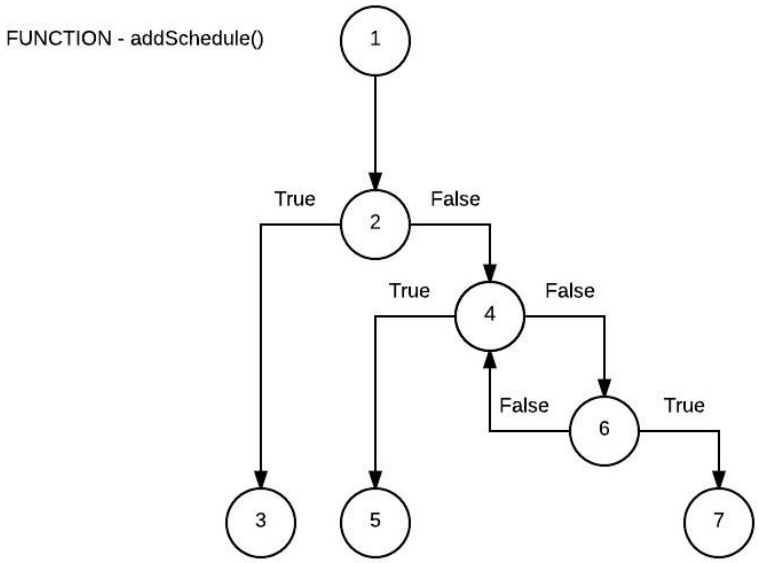
a. The screenshot for the function is given as,

```
98      * @param gameNames : name of games to be played on day
99      * @return string indicating successful addition otherwise error
100     */
101     public String addSchedule(String dayName,String[] gameNames){
102         DaySchedule day = searchDay(dayName);
103         if(day != null){
104             return dayName+" already scheduled";
105         }
106         //verify every gameName for its validity
107         Game[] gamesPlayed = new Game[gameNames.length];
108         for(int i=0; i < gameNames.length; i++){
109             String gameName = gameNames[i];
110             Game storedGame = searchGame(gameName);
111             if(storedGame==null){
112                 return "Error you cannot be registered for "+gameName;
113             }
114             gamesPlayed[i] = storedGame;
115             GameAssociation association = searchAssociation(storedGame.name);
116             association.daynames[association.noofDays++]=dayName;
117         }
118         day = new DaySchedule(dayName,gamesPlayed);
119         scheduleList[scheduleCounter] = day;
120         scheduleCounter++;
121         return dayName+" added successfully";
122     }
```

b. The basic block table for the given function is given as,

BLOCK	LINES	ENTRY	EXIT
2	103,104	103	104
4	107,108,109,110,114,115,116,118,119,120,121	107	121
6	111,112	82	83

c. The Control Flow Graph for the function is given as,



5. Method – searchPlayer()

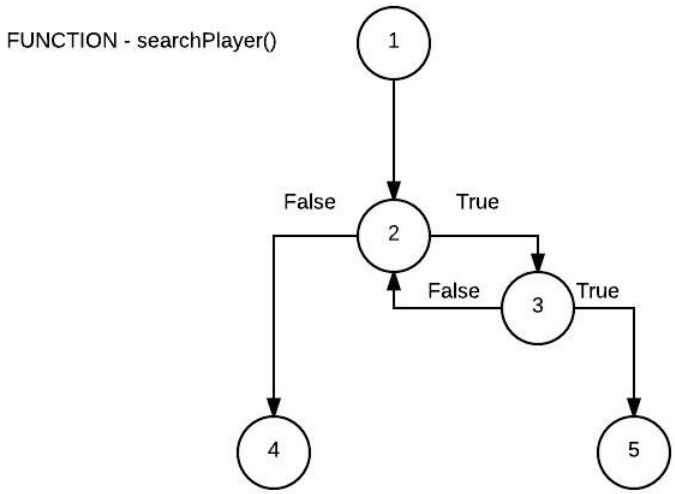
a. The screenshot for the function is given as,

```
137     /**
138      * This method finds player by player name
139      * @param name : name of the player
140      * @return player details in Player object if found
141      */
142     public Player searchPlayer(String name) {
143         for(int i=0; i < playerCounter; i++){
144             Player storedPlayer = playerList[i];
145             if(storedPlayer.name.equals(name)){
146                 return storedPlayer;
147             }
148         }
149         return null;
150     }
151
```

b. The basic block table for the given function is given as,

BLOCK	LINES	ENTRY	EXIT
2	143,144,149	143	149
3	145,146	145	146

c. The Control Flow Graph for the function is given as,



6. Method – searchDay()

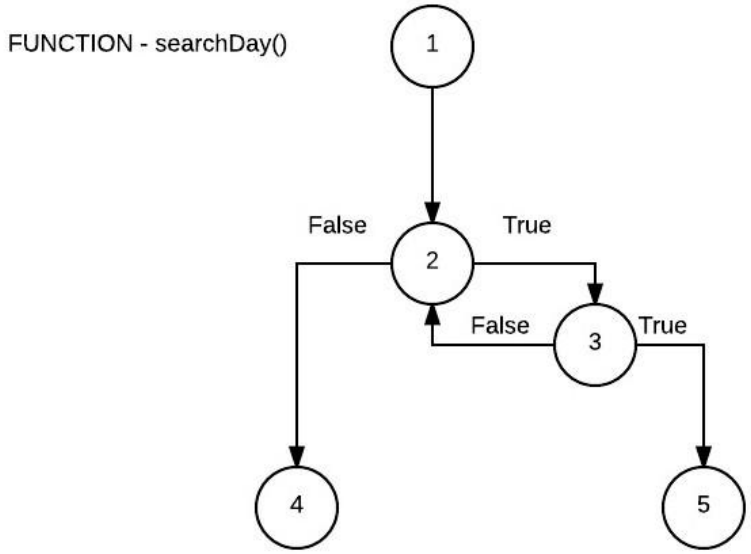
a. The screenshot for the function is given as,

```
151
152     /**
153      * This method finds day schedule by day name
154      * @param name: name of the day
155      * @return details of days schedule if found, else null
156      */
157     public DaySchedule searchDay(String name) {
158         for(int i=1; i <=scheduleCounter; i++){
159             DaySchedule storedDay = scheduleList[i-1];
160             if(storedDay.dayName.equals(name)){
161                 return storedDay;
162             }
163         }
164         return null;
165     }
166
167
```

b. The basic block table for the given function is given as,

BLOCK	LINES	ENTRY	EXIT
2	158,159,164	158	164
4	160,161	160	161

c. The Control Flow Graph for the function is given as,



7. Method – displayGameWiseSchedule()

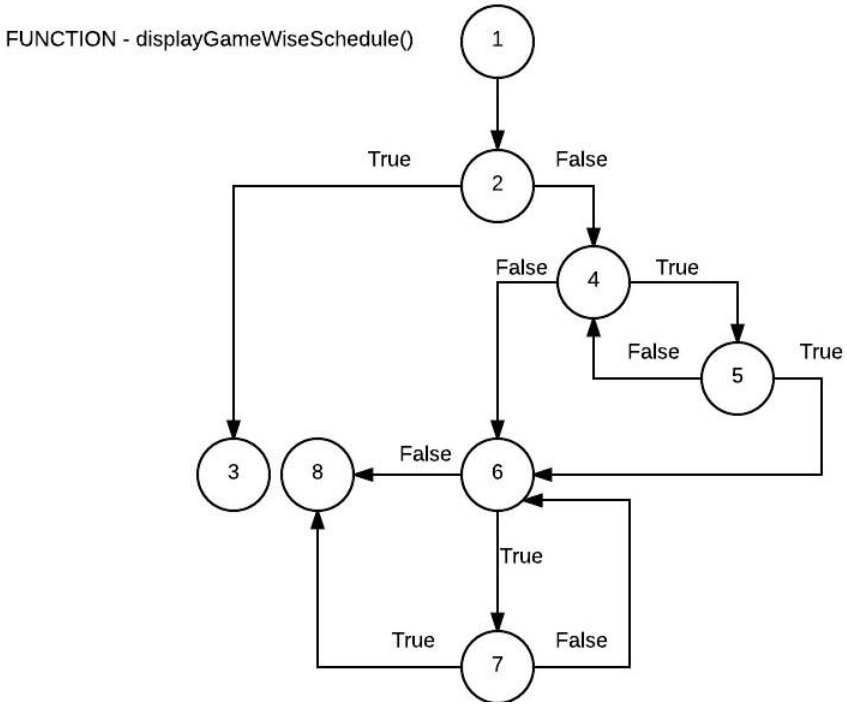
a. The screenshot for the function is given as,

```
170     * This method displays game wise schedule by game name
171     * @param gameName : name of the game
172     * @return String with schedule
173     */
174    public String displayGameWiseSchedule(String gameName){
175        Game game = searchGame(gameName);
176        if(game==null){
177            return "Error : This game is not valid";
178        }
179        String[] playerNames = getPlayerNames(gameName);
180        String[] dayNames    = getDayNames(gameName);
181        StringBuilder sb = new StringBuilder();
182        sb.append("Players Names: ");
183        for(String playerName : playerNames){
184            if(playerName==null)
185                break;
186            sb.append(playerName);
187        }
188        sb.append("\nDay Names: ");
189        for(String dayName : dayNames){
190            if(dayName==null)
191                break;
192            sb.append(dayName);
193        }
194        return sb.toString();
195    }
196
```

b. The basic block table for the given function is given as,

BLOCK	LINES	ENTRY	EXIT
2	175,176,177	176	177
4	179,180,181,182,183,186,188	179	188
5	184,185	184	185
6	189,192,194	189	194
7	190,191	190	191

c. The Control Flow Graph for the function is given as,



8. Method – displayDayWiseSchedule()

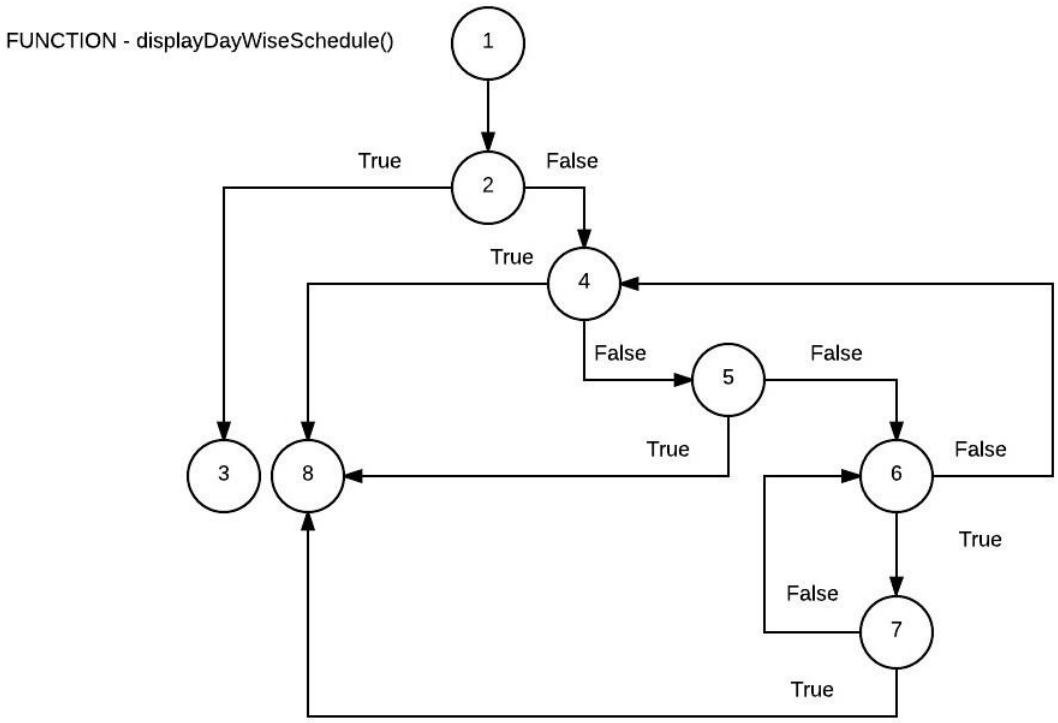
a. The screenshot for the function is given as,

```
199      * @param dayName : name of the day
200      * @return String with day wise schedule
201      */
202     public String displayDayWiseSchedule(String dayName){
203         DaySchedule schedule = searchDay(dayName);
204         if(schedule==null){
205             return "Error : This day is not valid";
206         }
207         StringBuilder sb = new StringBuilder();
208 
209         Game[] gamesPlayed = schedule.games;
210         for(Game g : gamesPlayed){
211             if(g == null)
212                 break;
213             sb.append("Game = "+g.name);
214             String[] playerNames = getPlayerNames(g.name);
215             for(String name : playerNames){
216                 if(name==null)
217                     break;
218                 sb.append(" "+name+"\n");
219             }
220         }
221         return sb.toString();
222     }
```

b. The basic block table for the given function is given as,

BLOCK	LINES	ENTRY	EXIT
2	203,204,205	204	205
4	207,209,210,221	207	221
5	211,212	211	212
6	213,214,215,218,221	213	221
7	216,217	216	217

c. The Control Flow Graph for the function is given as,



9. Method – displayDayWiseSchedule()

a. The screenshot for the function is given as,

```
243
244     /**
245      * This method is used to display schedule for a player
246      * @param playerName : name of player
247      * @return : string with game and days game is played
248      */
249     public String displayPlayerWiseSchedule(String playerName){
250         Player player = searchPlayer(playerName);
251         if(player==null){
252             return "Error : This player is not valid";
253         }
254         StringBuilder sb = new StringBuilder();
255         Game[] gamesPlayed = player.games;
256         for(Game g : gamesPlayed){
257             if(g == null)
258                 break;
259             sb.append("Game : "+g.name);
260             String[] dayNames = getDayNames(g.name);
261             for(String name : dayNames){
262                 if(name==null)
263                     break;
264                 sb.append(" "+name+"\n");
265             }
266         }
267         return sb.toString();
268     }
269
```

b. The basic block table for the given function is given as,

BLOCK	LINES	ENTRY	EXIT
2	250,251,252	251	252
4	254,255,256,267	254	267
5	257,258	257	258
6	259,260,261,267	259	267
7	262,263	262	263

c. The Control Flow Graph for the function is given as,