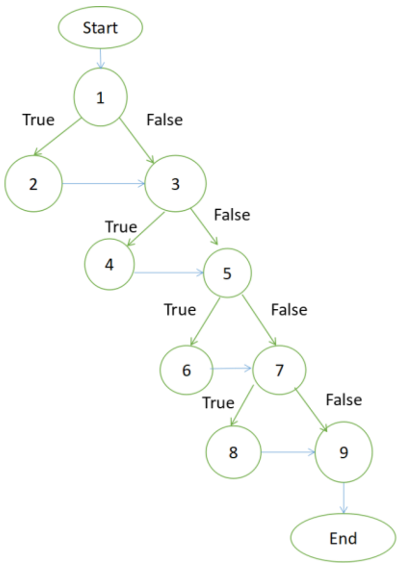
1) addGame

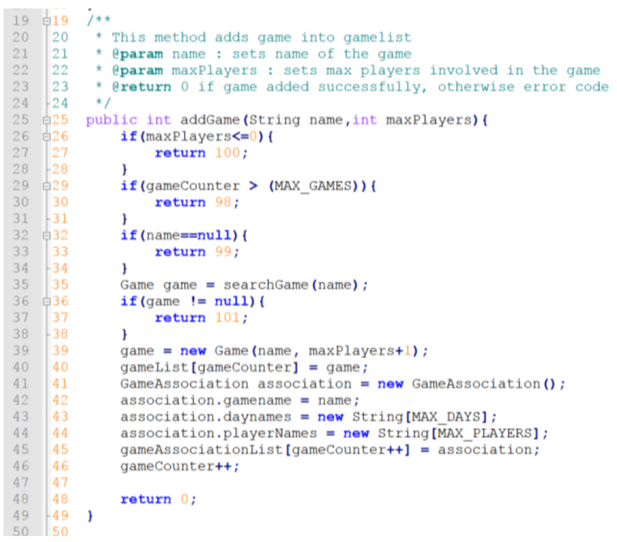
Block	Lines	Entry	Exit
1	26	26	26
2	27	27	27
3	29	29	29
4	30	30	30
5	32	32	32
6	33	33	33
7	35,36	35	36
8	37	37	37
9	39,40,41,42,43,44,45, 46,47,48	39	48



1)addGame
TR for Edge Coverage :(1,2)(1,3)(2,3)(3,4),(3,5)(4,5)(5,6)(5,7)(6,7)(7,8)(7,9)(8,9)
Test Path:
[1,3,5,7,9]: int maxPlayers=0
[1,3,5,7,9]: int gameCounter=25
[1,3,5,7,9]: String name=null
[1,3,5,7,9]: Sting name =Rugby(same game twice)

[1,2,3,4,5,7,9]:String name=Rugby int maxPlayers=70

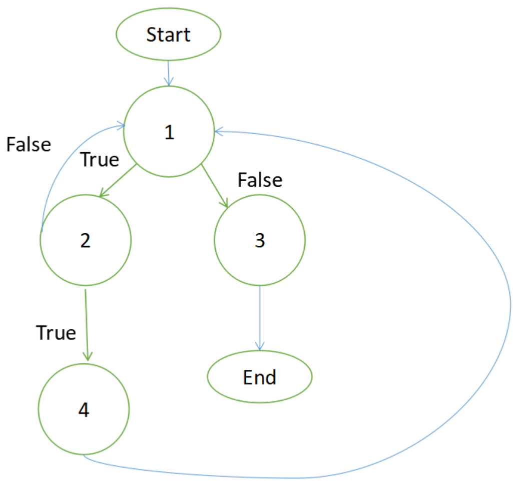
[1,2,3,4,5,6,7,8,9]:String name=Rugby int maxPlayers=70



```
19  /**
20   * This method adds game into gamelist
21   * @param name : sets name of the game
22   * @param maxPlayers : sets max players involved in the game
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+1);
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  }
50
```

2) searchGame

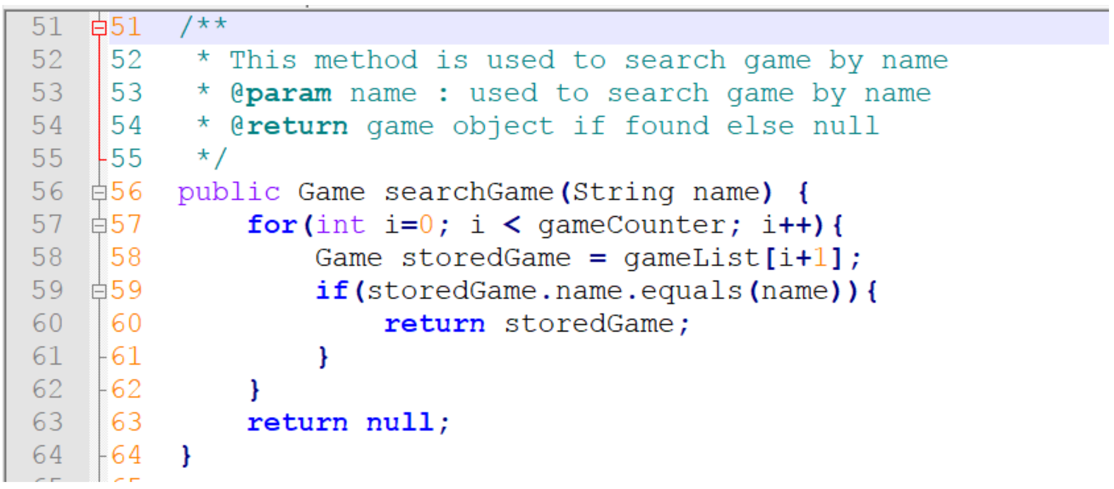
Block	Lines	Entry	Exit
1	57	57	57
2	58,59	58	59
3	63	63	63
4	60	60	60



2)searchGame
TR for Edge Coverage :(1,2)(1,3)(2,4),(2,1)(4,1)
Test Path:
[1,3] : addGame(soccer,22);
searchGame(Football);

[1,2,1,3]: addGame(soccer,22);
searchGame(soccer);

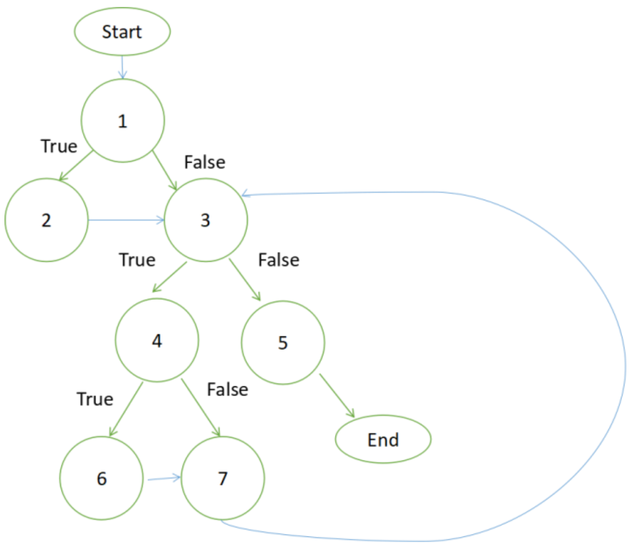
[1,2,4,1,3]: addGame(soccer,22);
searchGame(soccer);



```
51  /**
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+1];
59          if(storedGame.name.equals(name)){
60              return storedGame;
61          }
62      }
63      return null;
64  }
```

3) addPlayer

Block	Lines	Entry	Exit
1	73,74	73	74
2	75	75	75
3	78,79	78	79
4	80,81,82	80	82
5	89,90,91,92	89	92
6	83	83	83
7	85,86,87	85	87

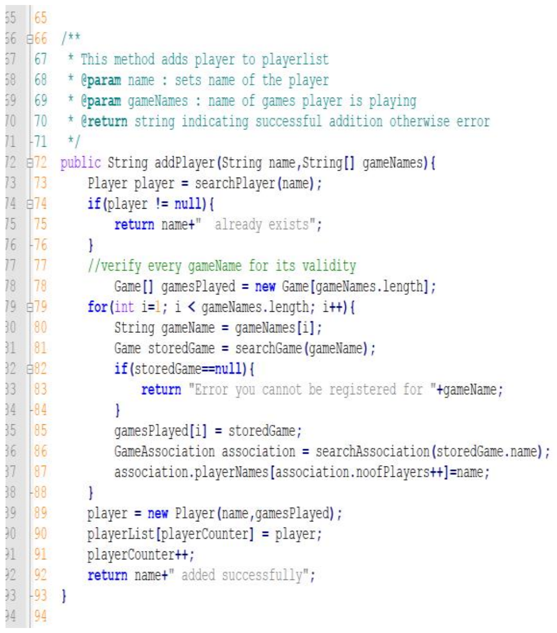


3)addPlayer
TR for Edge Coverage :(1,2)(1,3)(2,3)(3,4),(3,5)(4,6)(4,7)(6,7)(7,3)
Test Path:
[1,3,5]: addPlayer(Prakash)
addPlayer(Prakash)
Return(“Prakash already exists)

[1,2,3,5]: addPlayer(Prakash)
Return(“Prakash added successfully)

[1,3,4,7,3,5] : addPlayer(Prakash)
addPlayer(Prakash)
Return(“Prakash already exists)

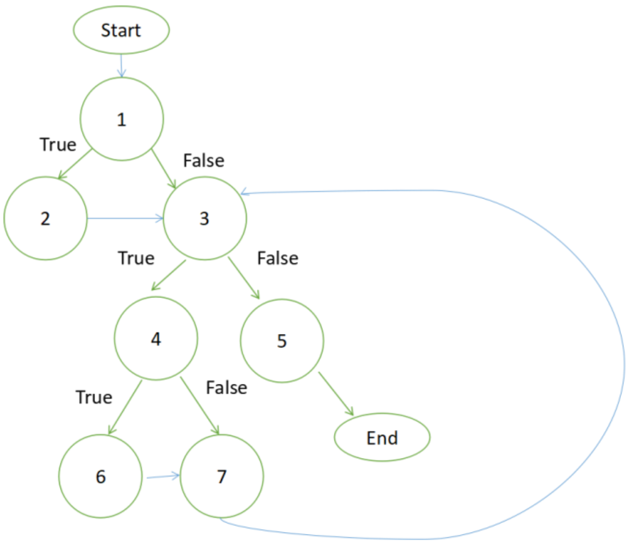
[1,3,4,6,7,3,5]: [1,2,3,5]: addPlayer(Prakash)
Return(“Prakash added successfully)



```
65
66  /**
67   * This method adds player to playerlist
68   * @param name : sets name of the player
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=1; 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  }
94
```

4) addSchedule

Block	Lines	Entry	Exit
1	102,103	102	103
2	104	104	104
3	107,108	107	108
4	109,110,111	109	111
5	112	112	112
6	114,115,116	114	116
7	118,119,120,121	118	121

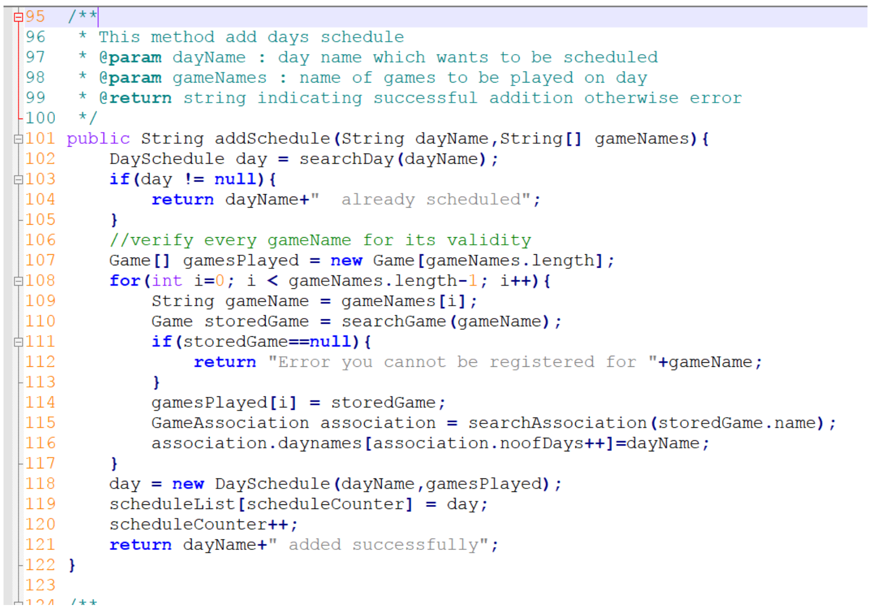


4)addSchedule
TR for Edge Coverage :(1,2)(1,3)(2,3)(3,4),(3,5)(4,6)(4,7)(6,7)(7,3)
Test Path:
[1,3,5]

[1,2,3,5]

[1,3,4,7,3,5]

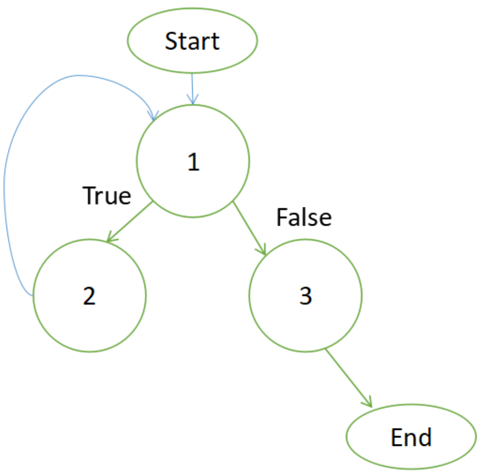
[1,3,4,6,7,3,5]



```
95   /**
96    * This method add days schedule
97    * @param dayName : day name which wants to be scheduled
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-1; 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  }
123
124  /**
```

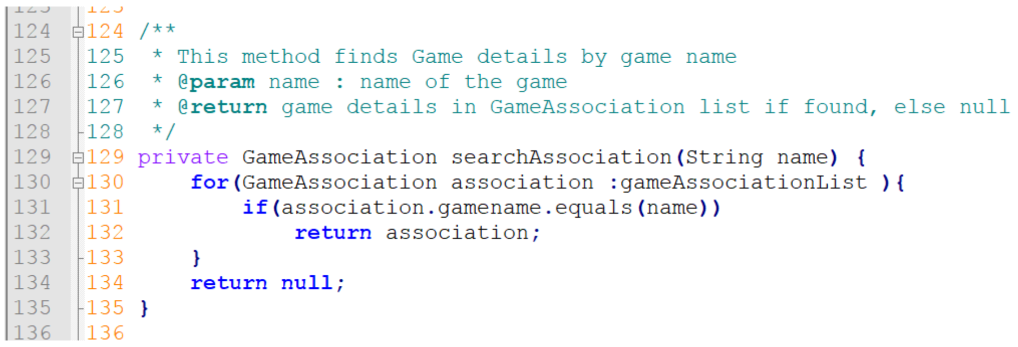
5) searchAssociation

Block	Lines	Entry	Exit
1	130,131	130	131
2	132	132	132
3	134	134	134



5)searchAssociation
TR for Edge Coverage :(1,2)(1,3)(2,1)
Test Path:
[1,3]

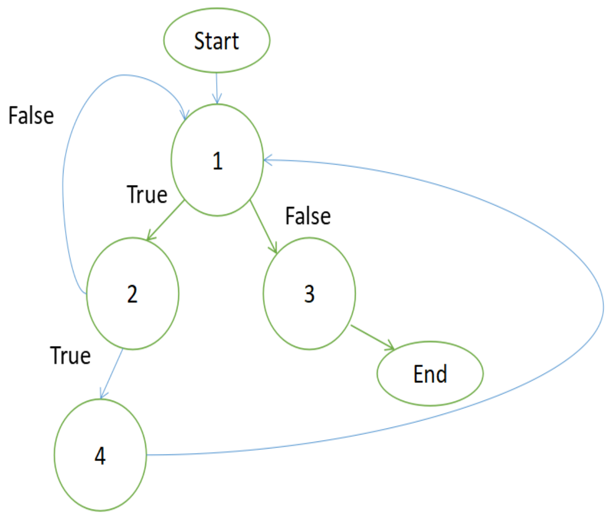
[1,2,1,3]



```
124 /**
125  * This method finds Game details by game name
126  * @param name : name of the game
127  * @return game details in GameAssociation list if found, else null
128  */
129 private GameAssociation searchAssociation(String name) {
130     for(GameAssociation association :gameAssociationList ){
131         if(association.gamename.equals(name))
132             return association;
133     }
134     return null;
135 }
136
```

6) searchPlayer

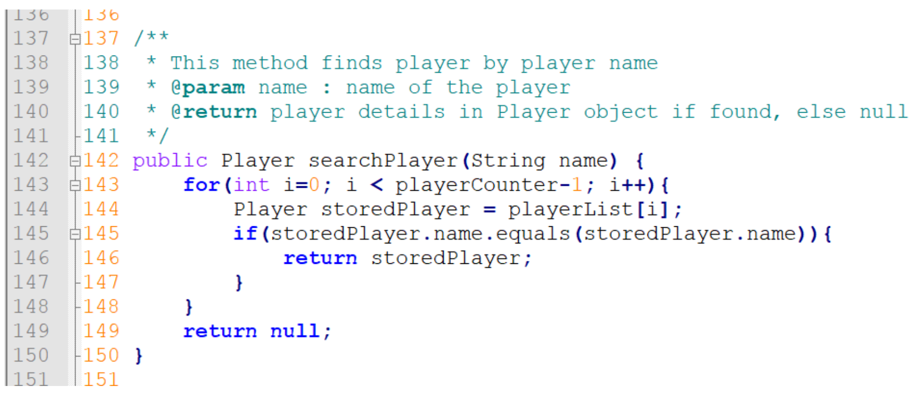
Block	Lines	Entry	Exit
1	143	143	143
2	144,145	144	145
3	149	149	149
4	146	146	146



6)searchPlayer
TR for Edge Coverage :(1,2)(1,3)(2,1)(2,4),(4,1)
Test Path:
[1,3]

[1,2,1,3]

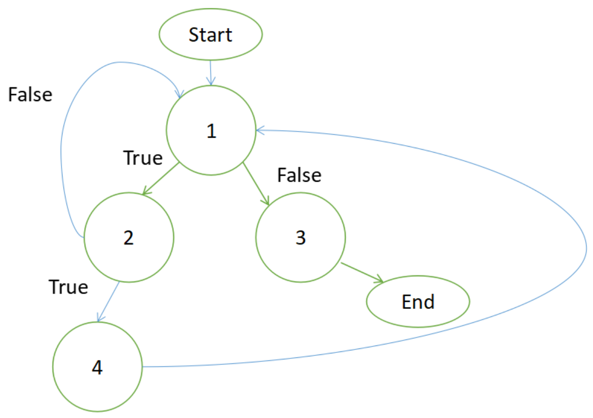
[1,2,4,1,3]



```
136  136
137  137 /**
138  138  * This method finds player by player name
139  139  * @param name : name of the player
140  140  * @return player details in Player object if found, else null
141  141  */
142  142 public Player searchPlayer(String name) {
143  143     for(int i=0; i < playerCounter-1; i++){
144  144         Player storedPlayer = playerList[i];
145  145         if(storedPlayer.name.equals(storedPlayer.name)){
146  146             return storedPlayer;
147  147         }
148  148     }
149  149     return null;
150  150 }
151  151
```

7) searchDay

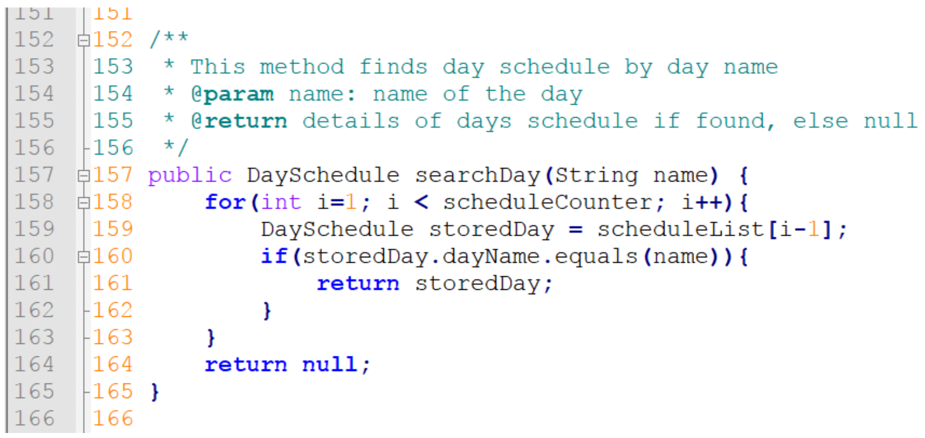
Block	Lines	Entry	Exit
1	158	158	158
2	159,160	159,160	159,160
3	164	164	164
4	161	161	161



7)searchDay
TR for Edge Coverage :(1,2)(1,3)(2,1)(2,4),(4,1)
Test Path:
[1,3]

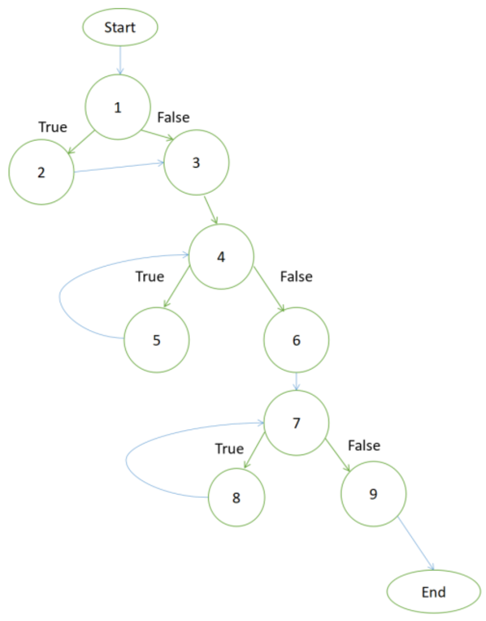
[1,2,1,3]

[1,2,4,1,3]


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

8) displayGameWiseSchedule

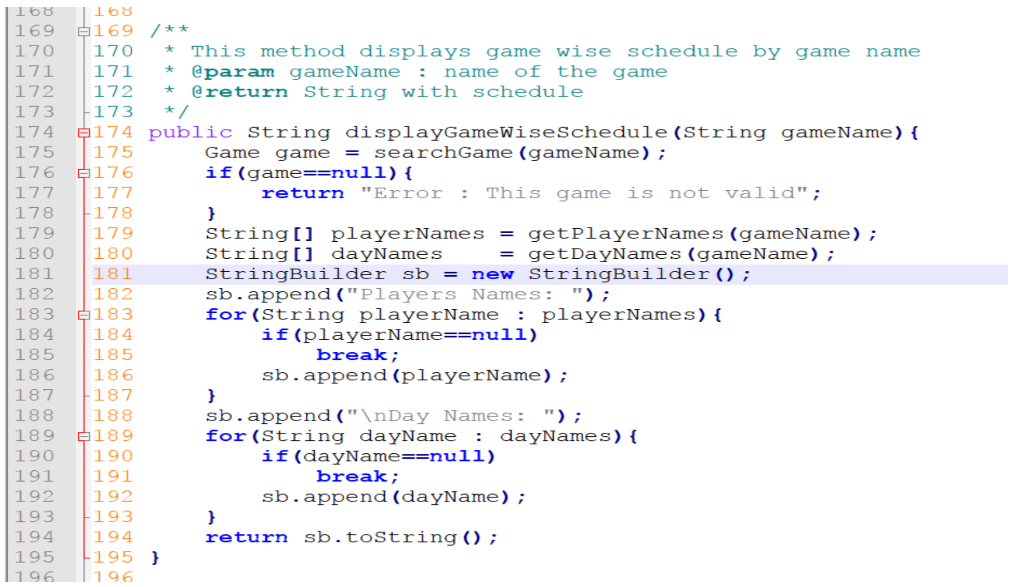
Block	Lines	Entry	Exit
1	175,176	175	176
2	177	177	177
3	179,180,181,182	179	182
4	183,184	183	184
5	185,186	185	186
6	188	188	188
7	189,190	189	190
8	191,192	191	192
9	194	194	194



8)displayGameWiseSchedule

TR for Edge Coverage :(1,2)(1,3)(2,3)(3,4),(4,5)(4,6)(5,4)(6,7)(7,8)(7,9)(8,7)
Test Path:
[1,3,4,6,7,9]

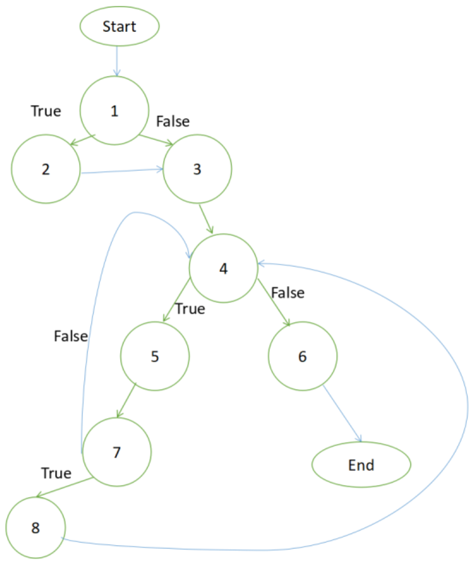
[1,2,3,4,5,4,6,7,8,7,9]



```
168
169 /**
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
```

9)displayDayWiseSchedule

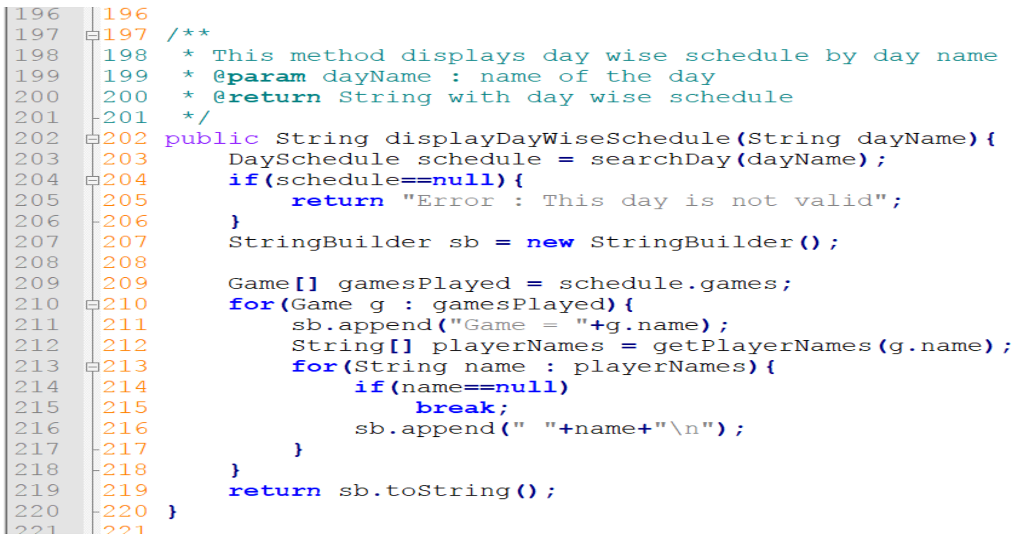
Block	Lines	Entry	Exit
1	203,204	203	204
2	205	205	205
3	207,208,209	207	209
4	210	210	210
5	211,212	211	212
6	219	219	219
7	213,214	213	214
8	216	216	216



9)displayDayWiseSchedule
TR for Edge Coverage :(1,2)(1,3)(2,3)(3,4),(4,5)(4,6)(5,7)(7,4)(7,8)(8,4)
Test Path:
[1,3,4,6]

[1,2,3,4,5,7,4,6]

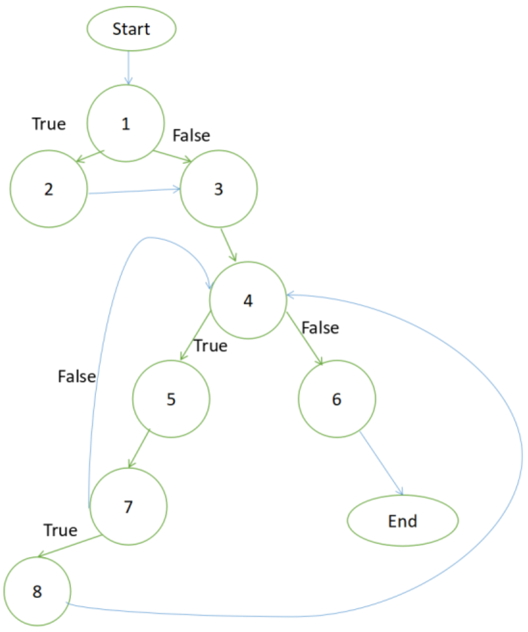
[1,2,3,4,5,7,8,4,6]



```
196 196
197 197 /**
198 198  * This method displays day wise schedule by day name
199 199  * @param dayName : name of the day
200 200  * @return String with day wise schedule
201 201  */
202 202 public String displayDayWiseSchedule(String dayName){
203 203     DaySchedule schedule = searchDay(dayName);
204 204     if(schedule==null){
205 205         return "Error : This day is not valid";
206 206     }
207 207     StringBuilder sb = new StringBuilder();
208 208
209 209     Game[] gamesPlayed = schedule.games;
210 210     for(Game g : gamesPlayed){
211 211         sb.append("Game = "+g.name);
212 212         String[] playerNames = getPlayerNames(g.name);
213 213         for(String name : playerNames){
214 214             if(name==null)
215 215                 break;
216 216             sb.append("  "+name+"\n");
217 217         }
218 218     }
219 219     return sb.toString();
220 220 }
```

10)displayPlayerWiseSchedule

Block	Lines	Entry	Exit
1	248,249	248	249
2	250	250	250
3	252,253	252	253
4	254	254	254
5	255,256	255	256
6	263	263	263
7	257,258	257	258
8	259,260	259	260



10)displayPlayerWiseSchedule

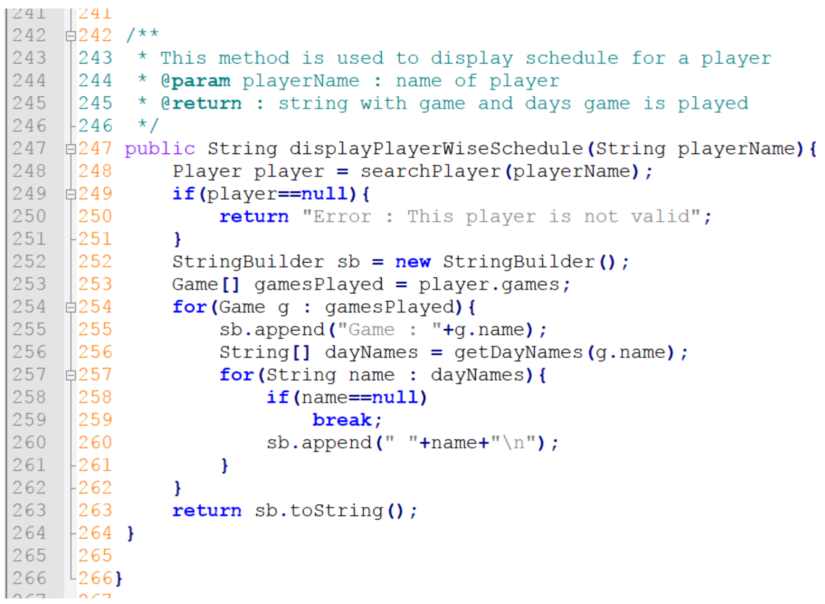
TR for Edge Coverage :(1,2)(1,3)(2,3)(3,4),(4,5)(4,6)(5,7)(7,4)(7,8)(8,4)

Test Path:

[1,3,4,6]

[1,2,3,4,5,7,4,6]

[1,2,3,4,5,7,8,4,6]



```
242 242 /**
243 243  * This method is used to display schedule for a player
244 244  * @param playerName : name of player
245 245  * @return : string with game and days game is played
246 246  */
247 247 public String displayPlayerWiseSchedule(String playerName){
248 248     Player player = searchPlayer(playerName);
249 249     if(player==null){
250 250         return "Error : This player is not valid";
251 251     }
252 252     StringBuilder sb = new StringBuilder();
253 253     Game[] gamesPlayed = player.games;
254 254     for(Game g : gamesPlayed){
255 255         sb.append("Game : "+g.name);
256 256         String[] dayNames = getDayNames(g.name);
257 257         for(String name : dayNames){
258 258             if(name==null)
259 259                 break;
260 260             sb.append(" "+name+"\n");
261 261         }
262 262     }
263 263     return sb.toString();
264 264 }
265 265
266 266}
```