main

```
37  public static void main(String[] args) throws IOException {
38      // TODO Auto-generated method stub
39      String name = null;
40      int permission_level;
41      int type_id;
42
43      if (args.length == 0) {
44          BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
45          String input = reader.readLine();
46          String[] parsed = input.split(" ");
47          name = parsed[0];
48          permission_level = Integer.parseInt(parsed[1]);
49          type_id = Integer.parseInt(parsed[2]);
50      } else {
51          name = args[0];
52          permission_level = Integer.parseInt(args[1]);
53          type_id = Integer.parseInt(args[2]);
54      }
55
56      System.out.printf("Adding user \"%s\".\n", name);
57
58      SampleProgram instance = new SampleProgram();
59      String[] output = instance.add_user(name, permission_level, type_id);
60      if (output != null) {
61          System.out.printf("%s user %s added.\n", output[1], output[0]);
62      } else {
63          System.out.println("adding user failed. ");
64      }
65
66  }
```

No need to include function signature, comments or blank lines in or as a basic block.

No need to consider "*else*" statement as a basic block.

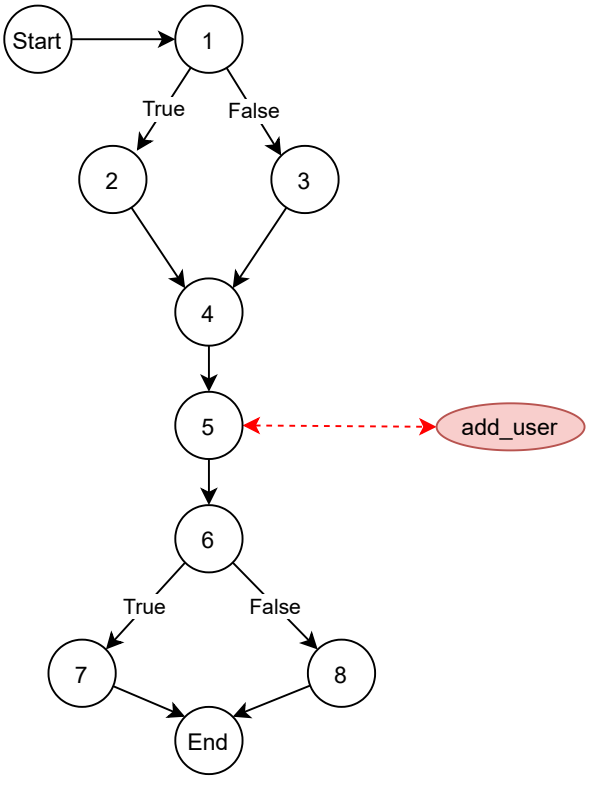
A line with user implemented function call is consider a basic block by itself.

When asserting console output, note that "*System.out.println()*" prints strings end with **\r\n** in Windows.

No need to consider lines with only brackets or curly brackets as basic blocks.

Block Number	Lines	Entry	Exit	Function Calls
1	39, 40, 41, 43	39	43	
2	44, 45, 46, 47, 48, 49	44	49	
3	51, 52, 53	51	53	
4	56, 58	56	58	
5	59	59	59	add_user
6	60	60	60	
7	61	61	61	
8	63	63	63	

This part is only specific to this project, you **don't** need to worry about this in home works or exams.



Drawing a CFG

1. Nodes represent basic blocks
2. Edge represent control flow from one block to another
3. When there are branches, remember to annotate the branching conditions
3. Always remember to include the start and end node
4. Always remember to annotate nodes with user implemented function calls.

add_user

```
18  String[] add_user(String name, int permission_level, int type) {
19      try {
20          // if first letter of name is not capitalized, capitalize it.
21          boolean need_capitalization = false;
22          if (Character.isLowerCase(name.charAt(0))) need_capitalization = true;
23          name = capitalize_first_ch(need_capitalization, name);
24
25          users.add(name);
26
27          if (type == 0 || type == 2 || permission_level == 3) {
28              super_users.add(name);
29          }
30
31          return new String[] {name, id_to_name[type]};
32      } catch(ArrayIndexOutOfBoundsException e) {
33          return null;
34      }
35  }
```

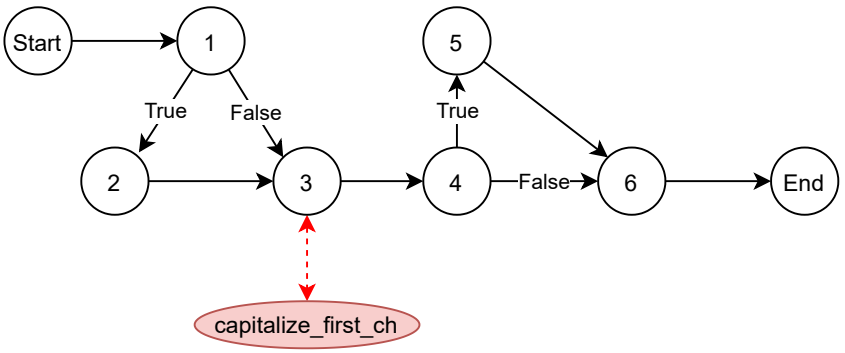
Ignore "*try*" statements like they don't even exist.

This if statement actually contains three basic blocks and six branches on the instruction level, each predicate is a basic block by itself. However, in this project, you can **treat such cases as one single block** for simplicity purpose.

Ignore "*catch*" statements and everything within them.

When if statements are in-lined, split the line into a and b.

Block Number	Lines	Entry	Exit	Function Calls
1	21, 22a	21	22a	
2	22b	22b	22b	
3	23	23	23	capitalize_first_ch
4	25, 27	25	27	
5	28	28	28	
6	31	31	31	



capitalize_first_ch

```java
10    String capitalize_first_ch(boolean flag, String name) {
11        if (!flag) {
12            return name;
13        }else {
14            return name.substring(0, 1).toUpperCase() + name.substring(1);
15        }
16    }
```

Block Number	Lines	Entry	Exit	Function Calls
1	11	11	11	
2	12	12	12	
3	14	14	14