COM1003 Java Programming
Autumn Semester 2022-3
Programming Assignment 3

Dr Siobhán North
Department of Computer Science, The University of Sheffield

Learning outcomes

This assignment will assess your ability to:

- Understand an unfamiliar program involving interacting classes;
- Write a program from a specification;
- Write clear, good quality program code that meets its specification;
- Manipulate arrays in Java
- Use methods in Java

This assignment is worth 20% of your mark for the first semester of the module and must be submitted by 18 January 2023. You will find information about the exact deadline, the marking scheme and how you must submit your work at the end of this document.

The specification below is quite precise in telling you what you must output but usually does not tell you how to achieve it. This is deliberate and part of the test. However it is not meant to be ambiguous so if there is something you don't understand you can ask questions, ideally by email. I will put all the email questions and their answers on an FAQ page as they come in. Even if you think you understand everything it is a good idea to check this page before you submit and again near the deadline if you submit very early.

You do not need to do everything described below to hand in your work and get marks for it. The marks will be awarded depending on how much you have achieved (see below for details) but if you hand in a program it must compile and run to get any marks. So a well written program that does something is always better than a program which would do more if it had worked.

For this assignment you have been provided with a a set of Java classes with no documentation but you should know that the `main` method is in the class provided called `Assignment3.java`. Each of your tasks involves changing one or more methods of one or more classes by adding clear well structured code which meets the specification. You are only allowed to change the method bodies as described in the task below, not anything else in any of the classes. This is an essential part of the specification.

The scenario is very loosely adapted from a board game called Cluedo see Wikipedia where a group of characters move around a series of rooms and are accused of murder using one of a set of weapons. The point is to identify who attacked the victim with what weapon in what room. Your version is going to be seriously simplified because there will be no dice and no moving around a board. The user of the program just has to guess the solution but once they get one aspect right they are only asked about the remaining aspects.

A possible run of the program is shown below.

```
....>javac *.java

....>java Assignment3
The weapons are the candlestick, the dagger, the lead
   pipe, the revolver, the rope and the wrench
The suspects are Miss Scarlett, Reverend Green,
   Colonel Mustard, Professor Plum, Mrs Peacock and
   Mrs White
The rooms are the kitchen, the ballroom, the
   conservatory, the billiard room, the library, the
   study, the hall, the lounge, the dining room
Where are you? the hall
Who do you accuse? Reverend Green
With what weapon? the rope
The victim was not attacked in the hall by Reverend
   Green with the rope
Where are you? library
Who do you accuse? miss scarlett
With what weapon? wrench
The victim was not attacked in the library by Miss
   Scarlett
Where are you? KITCHEN
Who do you accuse? Mrs White
The victim was not attacked in the kitchen by Mrs
   White
Where are you? ballroom
Who do you accuse? professor plum
The victim was not attacked by Professor Plum
Who do you accuse? Colonel mustard
Well done!
The victim was attacked in the ballroom by Colonel
   Mustard with the wrench
Play again? n
```

You have been provided with a zipped folder called `Assignment3` which you should download and unzip. It contains all the class files and two data files. You should put all of them in the directory you plan to run the program from. It will compile and run from the start, it just won't do much.

As usual it is a good idea to upload your work to Blackboard as you go on (see the end of the document for what to upload) but only upload something that works.

Task 1

The first task is to read in the weapons and characters. I have provided you with two files, `weapons.txt` and `suspects.txt` based on the original board game but you are welcome to substitute your own data. The weapons file must contain the names of six weapons, one to a line, but if you would prefer a non lethal version feel free to substitute alternatives (feather duster, pillow...) but there must be six, the file must be called `weapons.txt` and must be in the same directory as all the class files.

For the suspects file you can also substitute different names (your flat mates perhaps or those of your more irritating relatives). These must also be one to a line but, unlike in the game, there can be any number between three and ten. This file must called `suspects.txt` and must also be in the same directory as all the class files.

To read these data files in you must modify both the `Weapon.java` class and the `Suspect.java` class. I suggest you tackle them in that order because the weapons are easier and you can still get marks for getting that right even if `Suspect.java` doesn't work.

The `Weapon.java` class contains this method

```
public static void initializeThem() {
  //Fill in this method body with your code for Task 1
}
```

and you should replace its body with your own code to read in the names of the weapons from `weapons.txt` using `EasyReader` and turn each of them into a `Weapon` object stored in the array called `allTheWeapons`. Having done so you must print them out by calling the method `listTheWeapons()` which has already been created for you. In doing this you can only change the body of the method `initializeThem()`, nothing else.

Reading in `suspects.txt` is similar except that the file may not contain six suspects. The method whose body you have to change is called `initializeEveryone()` and the array you need to populate with `Suspect` objects is called `allTheSuspects`. For this task you also need to make sure that your method sets the value of the variable `numberOfSuspects`. Your method body must end with a call to the method `listTheSuspects()` which already exists.

You can test your code by running the program. The output should now inclued the first two lines of the sample output above - the ones listing the weapons and suspects although possibly with different suspects and weapons.

If you do this with two perfectly written methods that demonstrate the techniques identified in the marking scheme you can expect to get 22%.

Task 2

The rooms in which the murder might take place are already identified by an `enum` called `Room.java`. It contains this method

```
public static void listThem() {
        //Replace this for Task 2
}
```

This is the method that should produce the list of rooms that follows the list of suspects in the initial example above. Modify this method so it works, ideally

even if another room is added to the list. In order to do this you will find the static method `values()` that all enums, including `Room`, have a built in. This method returns an array of all the `enum` objects in the order they were declared. `Room` already has a `to String()` method you should use to print them out. The list should start "The rooms are " as in the example.

If you do this with a well written method that demonstrate the techniques identified in the marking scheme you can expect to get another 10% so 32% in all and your output, when you run the program `Assignment3.java` will now include the line about the rooms.

Task 3

The program will not be able to do anything until it has decided who the murderer was, where the murder happened and what the weapon was. The class called `Scenario.java` deals with this using the following method.

```
public void setAtRandom() {
        attackedWith = Weapon.pickedAtRandom();
        attacker = Suspect.pickedAtRandom();
        attackedIn = Room.pickedAtRandom();
}
```

This tasks consists of writing the three `pickedAtRandom()` methods for the three classes `Weapon`, `Suspect` and `Room`.

If you do this with well written methods that demonstrate the techniques identified in the marking scheme you can expect to get another 27% (9% for each method) so 59% in all and your output but when you run `Assignment3.java` the output will not have chaged. You need to ask the user to guess the answer. That is the next task.

Task 4

The class called `Scenario.java` deals with the user's initial guess using the following method.

```
public void setByAsking(EasyReader keyboard) {
        attackedIn = Room.askWhichOne(keyboard);
        attacker = Suspect.askWho(keyboard);
        attackedWith = Weapon.askWhichOne(keyboard);
}
```

which calls three more methods from `Room`, `Suspect` and `Weapon`.

`Room.askWhichOne(keyboard)` asks the user "Where are you? " using `EasyReader`. This and the wording of the prompt is part of the specification. The method should turn the name of the room into a Room object and return it. Ideally it should work whatever case the user uses, whether they preface their answer by "the " or not and whether they use underscore or space for word breaks.

For perfect marks this method should return `null` if the answer is not the name of a room but you can obtain most of the marks if you ignore this. You cannot obtain the extra marks by using an `Exception` because I haven't taught you about them and that would give experts an unfair advantage.

The other two method calls which ask for the suspect and weapon follow the same rules. The questions are "Who do you accuse? " and "With what weapon? " as shown in the sample execution at the start.

If you do this with well written methods that demonstrate the techniques identified in the marking scheme you can expect to get another 33% (11% for each method) so 92% in all and your output but when you run `Assignment3.java` the output will now tell you that you have got the right answer first time. That will not necessarily be true because you need to do the last task to make it work properly.

Task 5

The class called `Scenario.java` contains the following method

```
public boolean hasBeenDiscovered(Scenario guess) {
  //Replace this method body by your code for Task 5
  return true;
}
```

As well as returning true if the guess is correct and false if it isn't it should say what is wrong with the answer. In the example at the start it was this method that produced "The victim was not attacked in the hall by Reverend Green with the rope" but, after the next guess, "The victim was not attacked in the library by Miss Scarlett" because that guess included the correct weapon.

Again the wording is part of the specification. If you do this with a well written method that demonstrate the techniques identified in the marking scheme you can expect to get the remaining 8% so 100% in all and when you run `Assignment3.java` program will finally work properly.

Submission and deadline

You must submit, at most, the files called `Weapon.java`, `Suspect.java`, `Room.java` and `Scenario.java`. But if you didn't get far enough to change all of them there is no need to submit unchanged files. These files must be uploaded as individual `.java` files *not zipped up and not in any other format.*

Blackboard is very bad at displaying Java programs so you may think that Blackboard has messed up the layout of your program but I will download the program text and the layout will be preserved. You can upload your `.java` files as many times as you like before the deadline and the last versions you uploaded will be marked but but you must upload *all* the files you have modified each time. Blackboard will not remember your earlier submissions and include them. If you don't upload anything before the deadline the first version you upload afterwards will be marked.

Late work will be penalised using the standard University scale (a penalty of 5% per working day late; work will be awarded a mark of zero if it is more than 5 working days late). This is an individual assignment. You must work on it alone and hand in your own work. If you work collaboratively and then pretend you did the work alone we will find out (we have a very good plagiarism checker and all submitted work will go through it) and, as you have already been told, we take the use of unfair means in the assignment process very seriously. Don't even think about handing in work you didn't do yourself.

The Marking Scheme

The mark for this assignment is worth 20% of the first semester mark and so 10% of the overall mark for COM1003.

The marking scheme for the various versions is as follows:

Final Version	Task 1	Task 2	Task 3	Task 4	Task 5
Use of EasyReader	2	2	2	3	3
Use of Objects	4	4	4	6	7
Use of Arrays	2	4	10	13	13
Number and String Manipulation		1	4	13	14
Use of Methods	4	6	9	12	13
Meets Specification	8	12	24	36	40
Style	2	3	6	9	10
Total	22	32	59	92	100

The style marks for this assignment are low because you have to write short pieces of program code but the style marks will depend on readability which includes layout, indention and meaningful variable names.