Systems Analysis

and Design

Instructor : Huang, Chuen-Min

**Teamwork2 ver.1**

Group 4

|  |  |
| --- | --- |
| ID | Name |
| B10423011 | Lily |
| B10423034 | Ging |
| B10523004 | Rick |
| B10523007 | Bess |
| B10523032 | Xavier |
| B10523034 | Kenny |
| B10523057 | Helen |
| A10523050 | Ian |
| X10698051 | Andy |
| Date 2018/05/28 | |

Context

[**(1) Law of Demeter (LoD)** 2](#_Toc515318699)

[**(2) Coupling** 4](#_Toc515318700)

[**(3) Cohesion** 6](#_Toc515318701)

[**(4) Connascence** 7](#_Toc515318702)

[**(5) CRC card** 8](#_Toc515318703)

[**(6) Contract, Algorithm specification, pre- or post- condition, Structured English, Activity diagram** 8](#_Toc515318704)

[**(7)** **Evaluate project in cohesion, coupling, and connascence** 11](#_Toc515318705)

[**(8)** **Adopt RDBMs, describe referential integrity.** 13](#_Toc515318706)

[**(9)** **Normalization** 15](#_Toc515318707)

[**(10)** **Denormalize, New class diagram** 23](#_Toc515318708)

[**(11)** **Inter-file clusterin, Index strategies** 24](#_Toc515318709)

[**Assignments and participation** 26](#_Toc515318710)

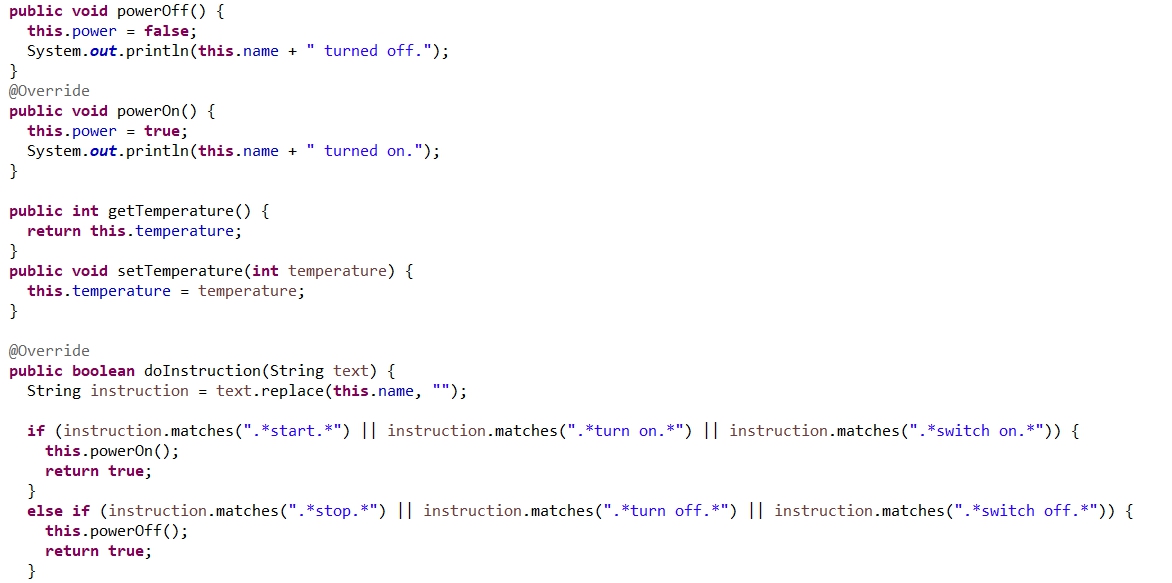
1. **Please explain the Law of Demeter (LoD) by using of your project.**

Law of Demeter has one important rule: Each unit should have only limited knowledge about other unit. That is to say, an object should avoid knowing other methods or attributes too much.

The following are example by using our project.

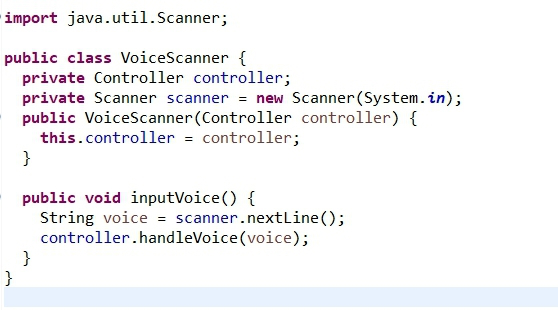
* 1. Law of Demeter1: ”Itself”

In the air conditioner class, doInstruction() calls the method “powerOff()” and “powerOn()” by itself. Also, both of them control their own variable - power. It obeys the Law of Demeter 1—A class use its own member variables and methods.



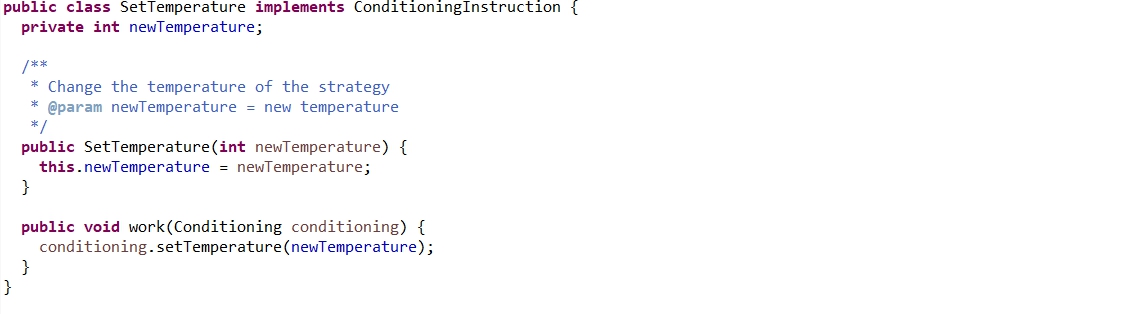
* 1. Law of Demeter 2: An object that is contains in an attribute of the object or one of its superclass.

In the VoiceScanner class, it declares the Controller object, and then passes the voice string to its method. It obeys the Law of Demeter 2.



* 1. Law of Demeter 3: to an object that is passed as a parameter to the method.

In the SetTemperature class, the method “work()” accepts Conditioning object as a parameter and use its method “setTemperature” to change the temperature. It obeys the Law of Demeter 3.



* 1. Law of Demeter 4: to an object that is created by the method.

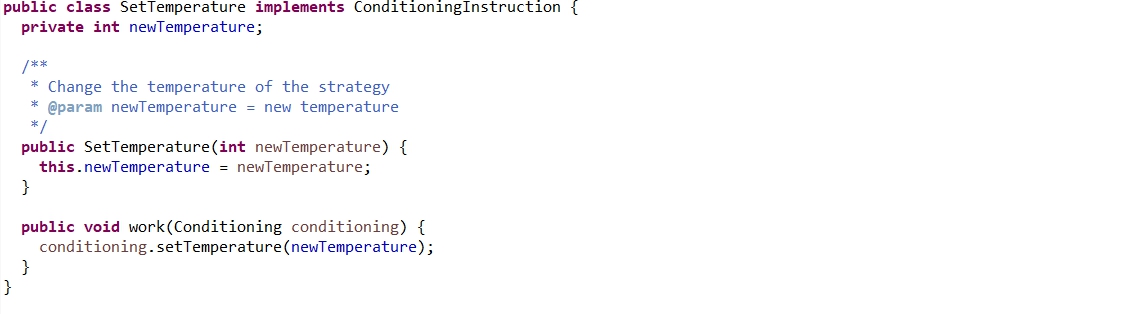
In the GUI class, the method “displaySuccessDoInstruction()” new a Date object and use it to show the success time. It obeys the Law of Demeter 4.





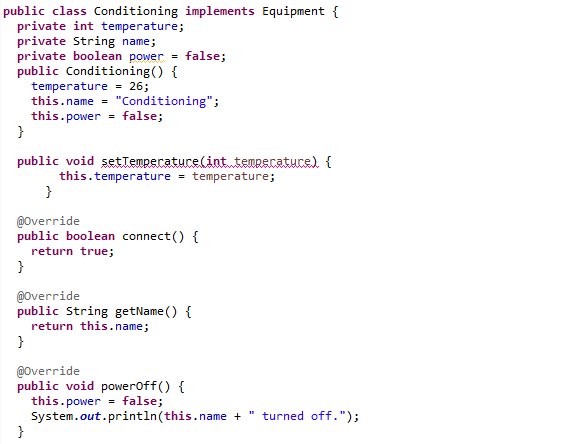
1. **There are six (or seven) types of interaction coupling, each falling on different parts of a good-to-bad continuum. Choose three pieces of your project to describe what types of the coupling they belong to.**
   1. Data

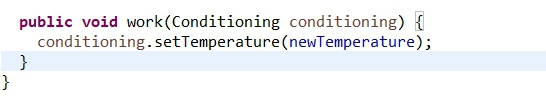
The method “SetTemperature” accept other component passes new temperature and then update it. It belongs to the data coupling.



* 1. Stamp

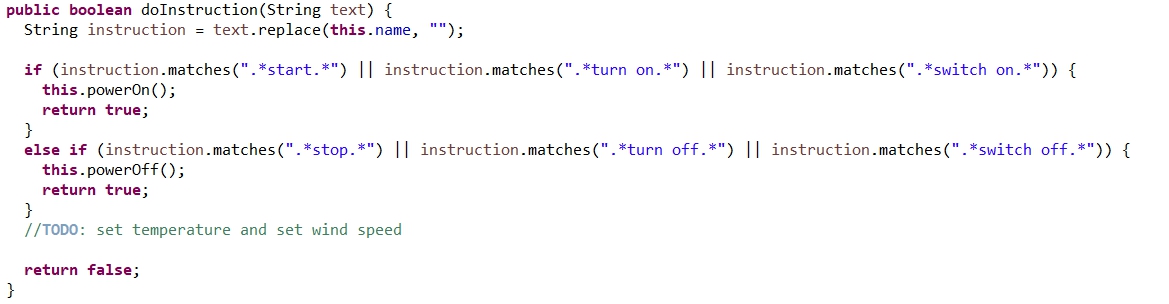
The method “work()” accept the Conditioning object, which has several attribute like temperature, name, and power, using its method to change the value. It belongs to stamp coupling.





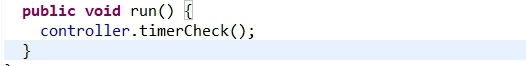
* 1. Control

The method “doInstruction()” accepts text data from the other component and use it to determine which function will be executed. It belongs to the control coupling.



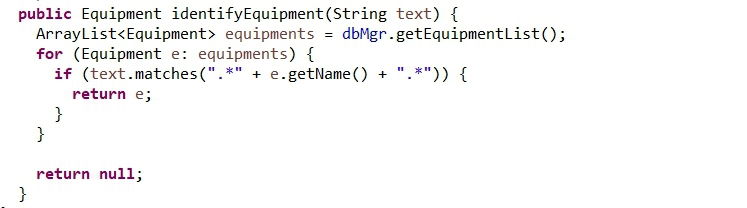
1. **There are seven types of method cohesion, choose three pieces of your project to describe what types of the cohesion they belong to.** 
   1. Functional

In the method “run()”, it only handle one function – controller.timeCheck(). It belongs to Functional cohesion – every essential element to a single computation is contained in the component.



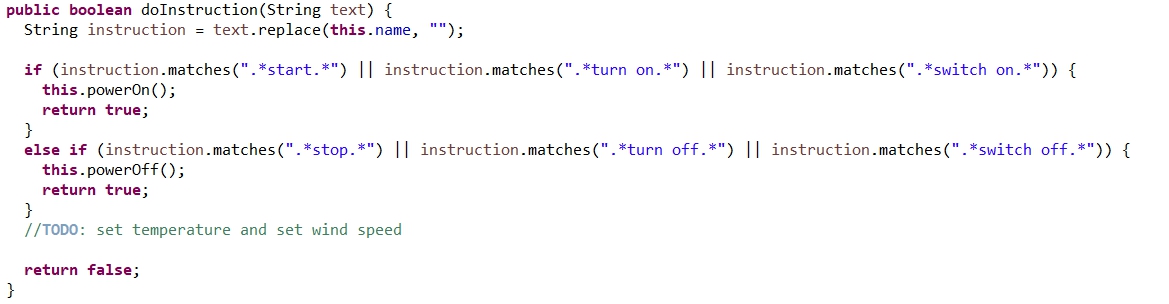
* 1. Sequential

In the method “identifyEquipemnt()”, it retrieve equipment list from database, and then use it to check the corresponding equipment. It belongs to Sequential cohesion – the output of one part is the input to another.



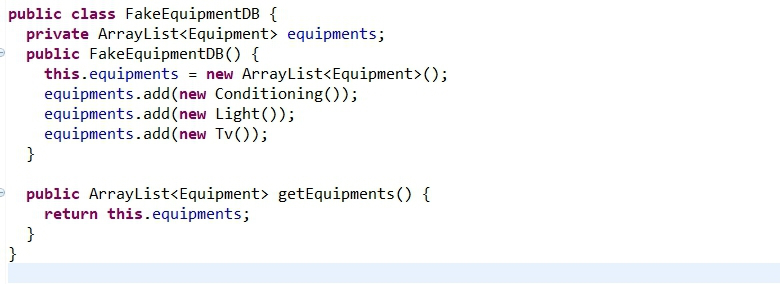
* 1. Logical

In the method “doInstruction()”, it totally do two things. The first is accept the text data and split its words. The second takes the data to match corresponding instruction. They are related logically, not functionally. It belongs to logical cohesion.



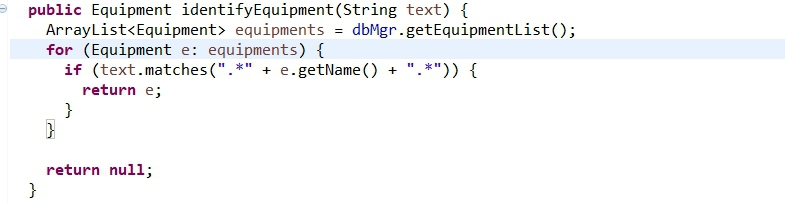
1. **Connascence generalized the ideas of cohesion and coupling, use three pieces of your project to describe what types of the connascence they belong to.** 
   1. Name

In the class FakeEquipmentDB, if equipment name changes, other objects which used it need to change at the same time. It belongs to name connascence.



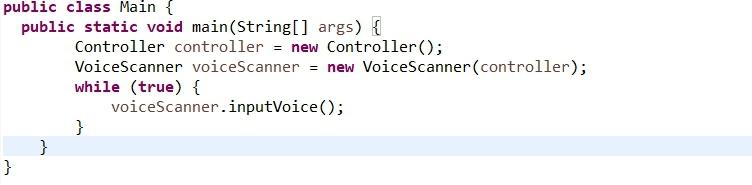
* 1. Type

In the method “identifyEquipemnt()”, if equipment data type changes, the for loop which uses the data to do function will also need to change its data type. It belongs to type connascence.



* 1. Position

In the method “main()”, if the order of “new Controller()” and “new voiceScanner()” changes reversed, it will affect whether it can be executed correctly. It belongs to position connascence.



1. **Use one class from your project that can create a set of invariants and add them to the CRC card or the class diagram.**

|  |  |  |  |
| --- | --- | --- | --- |
| Front | | | |
| Class Name: VoiceScanner | ID: 4 | | Type: Concrete, Domain |
| Description: It is responsible for accepting voice data and sending it to Controller. | | | Association Use Case: 1 |
| Responsibilities: | | Collaborators: | |
| inputVoice | | Controller | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Back | | | | |
| Attributes: | | | | |
| scanner | (0..\*) | (Scanner) | | {scanner = new Scanner(System.in)} |
| Relationships: | | | | |
| Genelization(a kind of): | | | | |
| Aggregation(has-parts): | | | | |
| Other Association: | | | Controller{0..1} | |

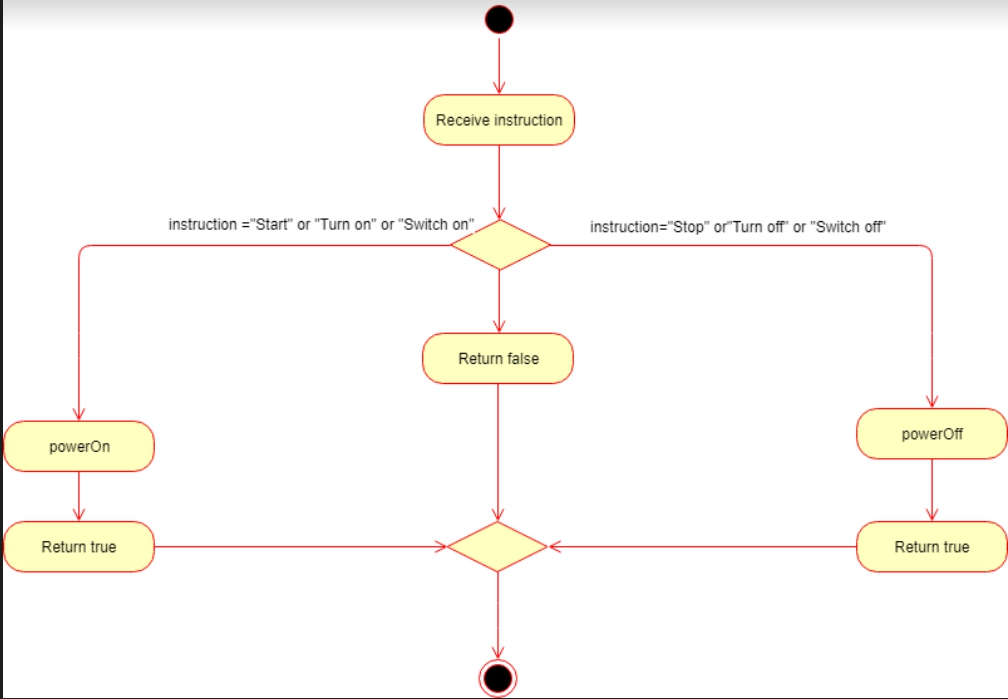
1. **Use a method of a class from your project that can create a contract and describe its algorithm specification. Specify the pre- or post- condition and use both Structured English and an activity diagram to specify the algorithm.**

|  |  |  |
| --- | --- | --- |
| Method Name: doInstruction | Class Name: Conditioning | ID: 5 |
| Client(Consumers): DBMgr | | |
| Associated Use Case: Operate Air Condition | | |
| Description of Responsibilities: Perform the action with the received instruction | | |
| Arguments Received:  text: String | | |
| Type of Value Returned:  boolean | | |
| Pre-Conditions:  1)The instruction is "Start", "Turn on" or "Switch on".  2)The instruction is "Stop", "Turn off" or "Switch off". | | |
| Post-Conditions:  1) Confirm that the air conditioning is on.  2) Confirm that the air conditioning is off. | | |

Contract

|  |  |  |
| --- | --- | --- |
| Method Name: doInstruction | Class Name: Conditioning | ID: 4 |
| Contract ID: 1 | Programmer: group 4 | Date Due: 5/29 |
| Programming Language:  □Visual Basic □Smalltalk □C++ 🗹Java | | |
| Triggers/Events: Controller send instruction to the air conditioner | | |
| Arguments Recevied :  　　　　 Data Type : | Notes : | |
| text: String | Follow the instructions to control the air conditioner. | |
| Messages & Arguments Passed:  ClassName.MethodName: | Data Type : | Notes : |
|  |  |  |
| Arguments Returned :  　　 　 Data Type : | Notes : |  |
| result: boolean | Return the result to control the air conditioner. | |
| Algorithm Specification:  Receive instruction from DBMgr  If instruction ="Start" or "Turn on" or "Switch on"  powerOn()  Return true  Else If instruction="Stop" or "Turn off" or "Switch off"  powerOff()  Return true  Else  Return false  End If |  | |
| Misc. Notes : None. | | |

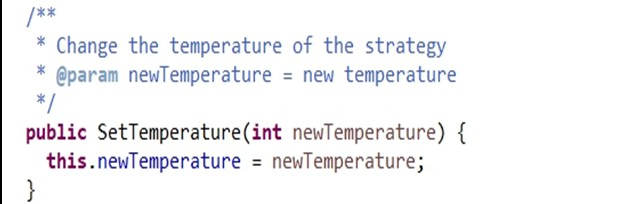
Method Specification Form



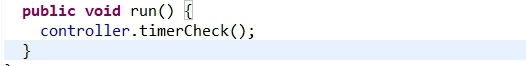
Activity Diagram-base Algorithm Specification

1. **Please evaluate any piece of your project in terms of cohesion, coupling, and connascence perspective.**
2. Interaction Coupling type: Data (Low-Coupling)

For system, it is the best way to use data coupling because the method will only accept simple data. Like the following java code, it only accept int data from other component in the method “SetTemperature()” and change it to the new temperature. It can ensure the stability of the data.

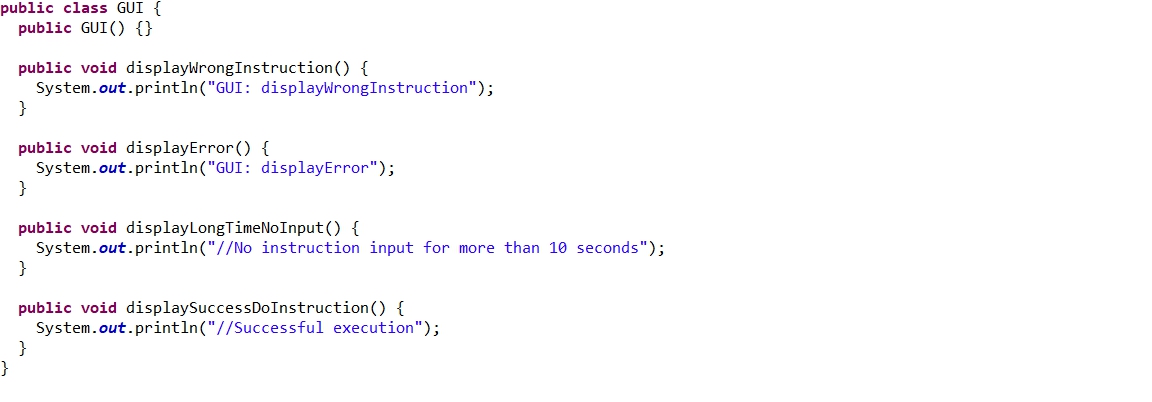


1. Method Cohesion type: Functional (High-Cohesion)

In the method "run()", it only do one thing - timer.Check(). Doing it caused system high maintainability and readability. It is also easy to code.

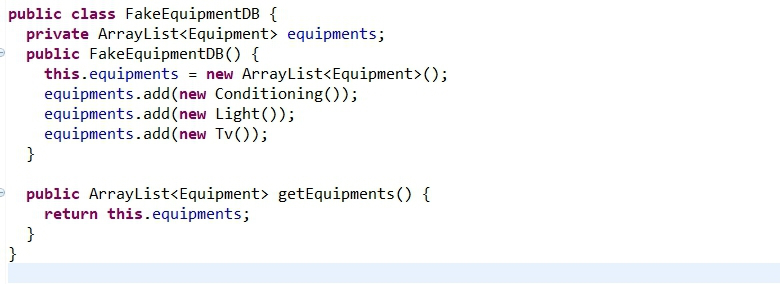
1. Class Cohesion type: Ideal

In the GUI class, it is suitable to has class cohesion because it has several public methods which outside classes can use. Also, every methods only do one function and don’t have any control coupling between them.

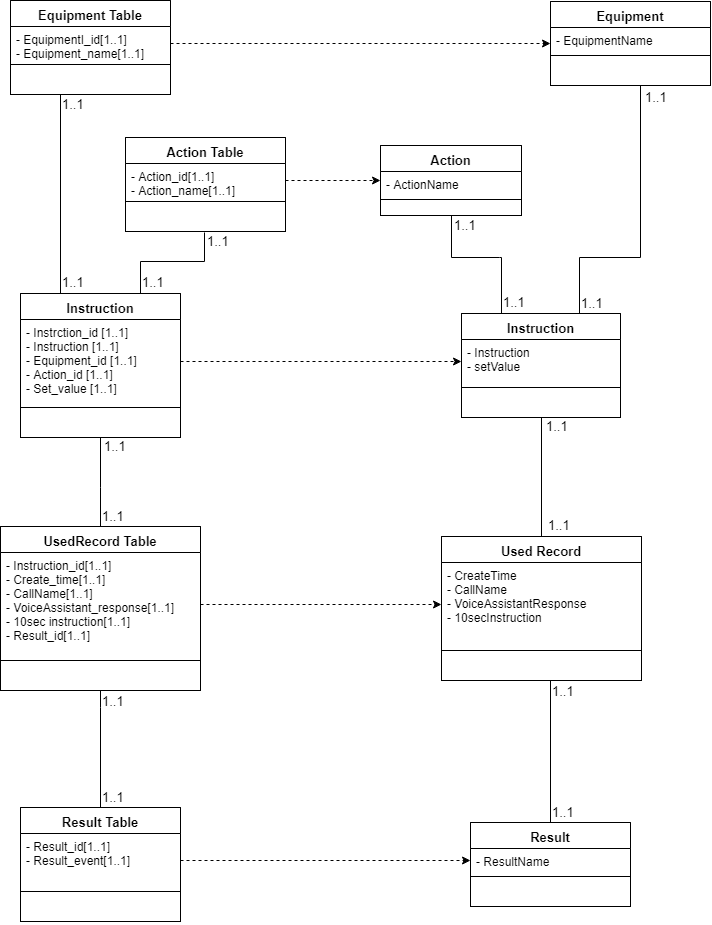


1. Connascence: Name

In the FakeEquipmentDB class, the method “getEquipments()” will connect to the equipment array which is the attribute in this class. If equipment name changes, the content of the method will also need to change. It belongs to name connascence and doing this can ensure the data consistency.

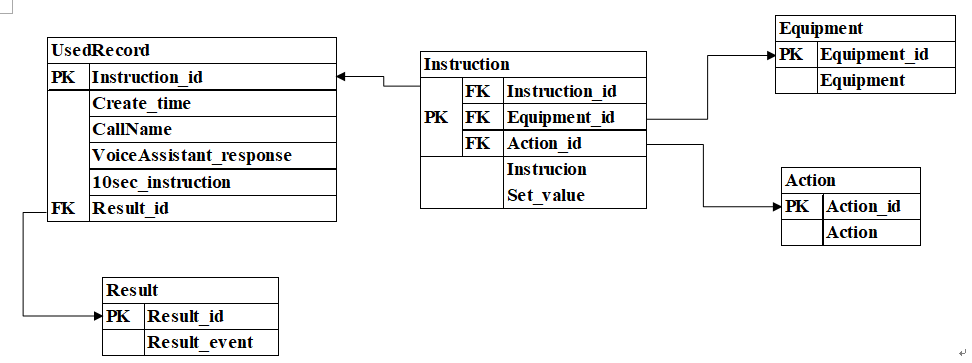


1. **Assume that you are going to adopt RDBMs to your project, please describe the referential integrity.**

****

RDBMs

Problem Domain



Instruction\_id is primary key in Used Record, is foreign key to Instruction.

Equipment\_id is primary key in Equipment, is foreign key to Instruction.

Action\_id is primary key in Action, is foreign key to Instruction.

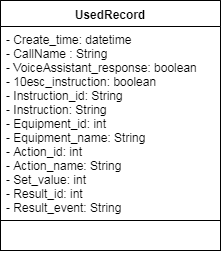
Result\_id is primary key in Result, is foreign key to UsedRecord.

Instruction has Composite primary key is combined with Instruction\_id, Equipment\_id, Action\_id, foreign key from UsedRecord, Equipment, Acion respectively.

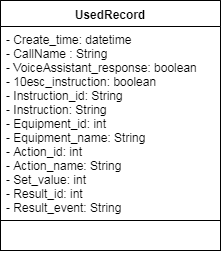
UsedRecord has a foreign key from Result.

1. **Using the steps of normalization, create a model that represents the file of your project in third normal form. Please make necessary assumptions to explain why the tables are related.**

0NF

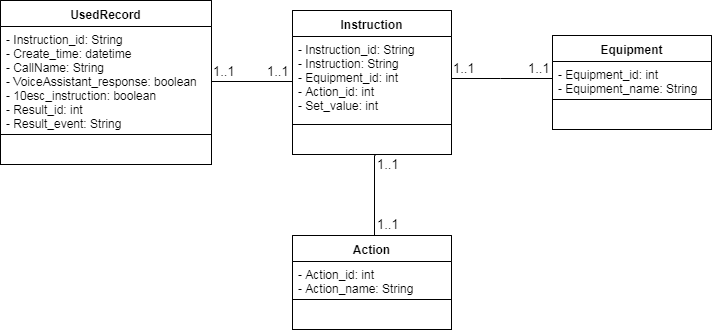


1NF



Because Instruction\_id, Instruction, Equipment\_id, Equipment\_name, Action\_id, Action\_name, Set\_value, Resault\_event, **these attributes are multi-valued or null,** but 1NF does not allow multi-value situation, so convert to 1NF must turn multi-valued into single value.

2NF







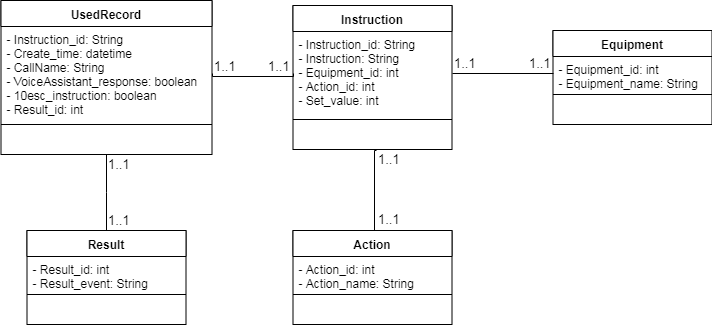


If there are some partial dependencies, it will cause waste of space, add, delete, update problems, because in 1NF,

1. Create\_time, CallName, VoiceAssistant\_response, 10sec\_instruction, Result\_id, Result\_event only need to rely on the Instruction\_id primary key,
2. Equipment\_name only need to rely on the Equipment\_id primary key,
3. Action\_name only need to rely on the Action\_id primary key, to solve partial dependencies problem in 1NF, so divided into Instruction, Equipment, Action.

Make a Composite key with Instruction\_id, Equipment\_id and Action\_id into Instruction.

3NF



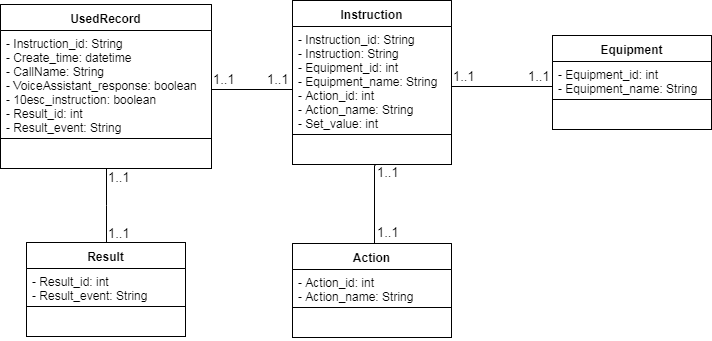






3NF does not allow the transitive dependency, but in 2NF through Instruction\_id can find the Result\_id, to find Result\_event, so we divided it with Record\_id and Result\_event in Result table.

1. **Describe how you would denormalize the model that you created in question 9. Draw the new class diagram based on your suggested changes.**



IF

(1) If user or developer wants to know what happened in the used record, he often through the Result\_id to search corresponding Result\_event. This steps are very miscellaneously to user, so that we decide to put Result\_event in UsedRecord Table.

(2) When querying the used instruction, we need to search the action name and equipment name by joining with Action\_id and Equipment\_id. Therefore, we put Action\_name and Equipment\_name into instruction table for improving effectiveness of the system, reducing the count of join.

1. **Examine the model that you created in question 10. Develop the inter-file clustering and index strategies. Describe how your clustering strategy will improve the performance of the database. List possible indices you would recommend and describe the reasons.**

inter-file clustering

When storing the data, we classify it under two types – success and error, and let them related to each other physically in the storage. It will more easily to load the related error or correct instruction, improving the performance of system.

intra-file clustering

index strategies

The indexing strategy is indicated in the diagram. We create an index which pointer to the error result of record by Result\_id. It can reduce searching time, helping user or developer to find record rapidly. User will view the instruction information within short time.

Assignments and participation

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Name | Score | Work |
| B10423011 | Lily | 100 | Participation, Normalization, Denormalize,  Inter-file clustering, Index strategies |
| B10423034 | Ging | 100 | Participation, Code |
| B10523004 | Rick | 100 | Participation, CRC card, Contract, Method specification Form, Activity diagram-base, Algorithm Specification |
| B10523007 | Bess | 100 | Participation, Law of Demeter (LoD), Coupling, Cohesion, Connascence, Evaluate project |
| B10523032 | Xavier | 100 | Participation, CRC card, Contract, Method specification Form, Activity diagram-base, Algorithm Specification |
| B10523034 | Kenny | 70 | PPT |
| B10523057 | Helen | 100 | Participation, Law of Demeter (LoD), Coupling, Cohesion, Connascence, Evaluate project |
| X10698051 | Andy | 100 | Participation, Normalization, Denormalize,  Inter-file clustering, Index strategies |
| A10523050 | Ian | 70 | PPT |