ISSN 2354-0082

Implementation of Model View Controller Architecture in Object Oriented Programming Learning

Ester Lumba[1], Alexander Waworuntu[2]
[1]Department of Informatics, Universitas Bunda Mulia, Jakarta, Indonesia
[2]Department of Informatics, Universitas Multimedia Nusantara, Tangerang, Indonesia
[1]10178@lecturer.ubm.ac.id
[2]alex.wawo@umn.ac.id

Accepted 23 December 2021
Approved 18 January 2022

***Abstract*—This study aims to provide an overview of the application of software design patterns, namely Model View Controller (MVC) in object-oriented programming learning. In the software development industry, most application development uses frameworks. MVC architecture is a design pattern that is widely used by various frameworks. Students as prospective programmers or software developers must master and be able to translate object-oriented programming concepts into programming languages. In this study, the Java programming language is used to apply the object-oriented concept and implement the MVC architecture. This research resulted in an increase in students' programming skills and abilities as well as being able to apply the MVC architecture in developing applications using Java.**

***Index Terms*—architecture, framework, Java, MVC, object-oriented**

I. INTRODUCTION

Bachelor of computer science, especially graduates majoring in Informatics should be able to make computer programs. But in reality, many informatics graduates do not master programming techniques, especially object-oriented programming. On the other hand, the software industry requires graduates who are qualified in their fields. Based on our observations, most graduates majoring in Informatics are not ready to apply their knowledge in the world of work. This is because there is a gap in the learning materials obtained during the study with the needs of the industry. Universities where students study have provided curricula and compulsory subjects related to computer science, including Programming Algorithms and Object-Oriented Programming which are applied to certain programming languages. However, in the learning process sometimes there are obstacles, including the lecturers have not mastered the material and the syllabus provided does not follow the needs of the industry. In addition, lecturers who teach are fixated on the provided teaching materials provided which only come from textbooks. The learning process and the materials presented are not oriented to the needs of the industry. This causes students majoring in Informatics do not have adequate competence.

In the digital era, the industry needs a lot of programmers for application development. Based on data from the national occupational map in the field of information and communication technology (ICT), the need for IT human resources has not been met in almost all categories [1]. Informatics graduates should be able to create computer programs or applications. However, the programming ability of Informatics graduates is generally inadequate. Based on the search results through the google.com search engine with the keywords "Informatics graduates cannot code" (Informatics graduates cannot code) shows that there are many blogs, forums and opinions in various online media, that the programming ability of most Informatics graduates are inadequate. On the other hand, programming languages continue to evolve [2][3]. Many programming languages are developed using an object-oriented paradigm [4][5][6]. Therefore, it is a must for a developer to master the concept of object-oriented programming and be able to apply it to programming languages such as Java, .NET, Python, PHP and so on.

This research uses the Java programming language. Java is a programming language that fully supports object-oriented programming concepts [7][8]. Referring to the Tiobe Programming Community website report, it is stated that Java is a language that has been continuously popular among programmers since late 2014 [9]. The Tiobe site indexes a wide variety of programming languages. The index is calculated from the number of search engine results based on a query containing the name of the programming language which is updated every month. This index includes searches on search engines, Wikipedia and Amazon. Fig. 1 shows the ranking of programming languages. Based on the graph in Fig. 1, it can be seen that Java is a programming language that is ranked at the top.

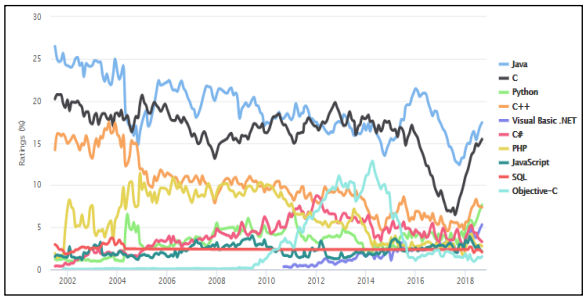

Fig. 1. Tiobe Programming Community Index

Computer applications are generally built using object-oriented programming languages [10][11]. Apart from being object-oriented, it also implements the Model View Controller (MVC) architecture. MVC is a design pattern that is widely used by various frameworks [12] and application development [13].

Students of the Informatics Department as prospective programmers must master object-oriented programming concepts, design patterns such as MVC and be able to translate into programming languages. Therefore, we consider it necessary to introduce and apply this MVC architecture to Object Oriented Programming learning. To clarify the research, a research question was made as follows: "How to apply the MVC architectural design pattern in learning Object Oriented Programming using the Java programming language?" The purpose of this study is to provide an overview of how to apply the MVC architectural design pattern to desktop applications and how to implement object-oriented programming concepts in the Java programming language.

II. METHODOLOGY

The research process is divided into three main stages as shown in Fig. 2.

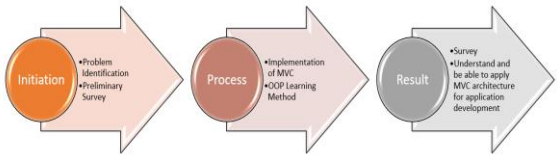

Fig. 2. Research Stages

A. Problem Identification and Preliminary Survey

In the first stage, the research team conducted a preliminary survey to identify problems. The research team is a lecturer who specializes in Object Oriented Programming (OOP) courses. At the beginning of the lecture, the researcher conducted a questionnaire to students who took OOP courses. The purpose of the researcher conducting this questionnaire is to obtain the needs and expectations of Informatics students after graduating and after attending OOP courses. The researcher made 8 questions which are presented in Table 1.

TABLE I. QUESTIONS AND QUESTIONNAIRE RESULTS BEFORE CLASS

No	Statement	Score				
		1	2	3	4	5
1	As a computer graduate candidate do you want to master programming techniques.	0	0	0	14	33
2	As a candidate for computer science, I want to have the ability to make computer programs	0	0	1	1	29
3	I now understand the concept of object-oriented programming	17	23	7	0	0
4	I want to master object-oriented programming	0	0	4	17	26
5	I understand how to implement object-oriented programming concepts in the Java programming language	15	26	6	0	0
6	Currently I have heard and understood the concept of Model View Controller (MVC) architecture.	24	17	6	0	0
7	I already understand how to implement the Model View Controller architecture in the Java programming language	34	12	1	0	0
8	I have been able to implement the Model View Controller architecture for desktop application development	37	9	1	0	0

A total of 47 students filled out the questionnaire as shown in table 1 above. In general, Informatics students want to master the techniques and concepts of object-oriented programming.

B. Object Oriented Programming Concept

The case study in this research uses Object Oriented Programming (OOP) courses. Researchers teach important concepts that must be understood in OOP, namely abstraction, encapsulation, inheritance and polymorphism. The programming language used is Java because it fully supports this concept. The coding technique is done by applying the MVC architecture. The researcher as well as a lecturer in the PBO course begins an explanation of classes, objects and properties using a diagram as shown in Fig. 3.

In figure 3, the property is used to identify the lecturer object. Where a lecturer (object lecturer) will have a NIDN, name and income (income). Meanwhile, to fill and change the value of each property using the set and get methods.

Based on the diagram in Fig. 3, it will then be elaborated into the Unified Modeling Language (UML) which is the standard modeling language [14]. The UML notation used is a class diagram. Class diagram is a static model that describes the structure of a system by showing the class names, attributes, operations or methods and the relationships between objects.

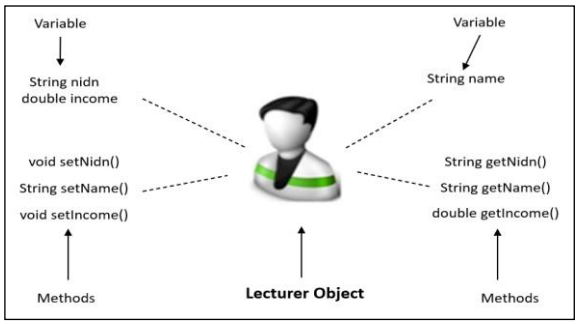

Fig. 3. Properties and methods of the lecturer object

In general, universities have lecturers in the category of permanent lecturers and part time lecturers. Researchers make class diagrams to explain the concept of inheritance. Fig. 4 shows a class diagram and the concept of inheritance that will be used to form the lecturer object.

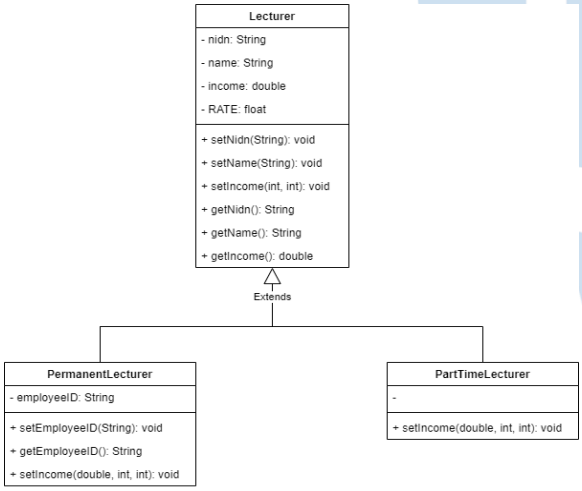

Fig. 4. Class diagram - inheritance concept

Next, the researcher translated the class diagram and the concept of inheritance on the lecturer object using the Java programming language. The coding technique uses the MVC architectural design pattern.

C. MVC Implementation

Model View Controller (MVC) is an architecture that is often used in web-based application development [5]. MVC architecture is basically a three-layer architecture (Fig. 5) that has different characteristics in each layer. The first layer is related to user input logic, the second layer is related to business logic and the third layer is used to implement user interface logic.

Many developers use MVC as the standard design pattern. The purpose of this design pattern is to create a program that can be used repeatedly for the same thing (code reuse). In MVC there are three types of classes that are placed in different packages, namely:

1. Model is a class used to implement data domain logic. These classes are used to retrieve, insert or update data into the database associated with the application.
2. View is used to create the application interface. The user interacts with the application using the interface design.
3. A controller is a class that is used to respond to user requests and relate them to the model.

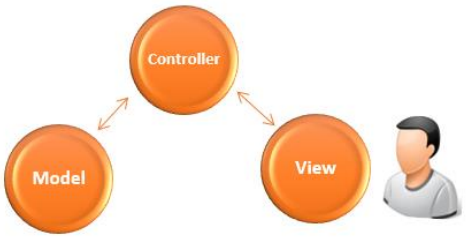

Fig. 5. MVC Architecture

MVC architecture really helps developers to control the complexity of application development by dividing it into three parts, namely model, view and controller. The MVC architecture makes it easy to maintain a program. Because all parts of the program have been mapped out in a clear structure. Changes to the design do not change the logic or data. On the other hand, changes to the logic or program can be made in a separate program section [15]. The steps for implementing MVC on desktop applications using the Java programming language and Apache Netbeans IDE version 12.0 are as follows:

The first step is to prepare packages to place program codes according to the MVC design pattern as shown in Fig. 6.

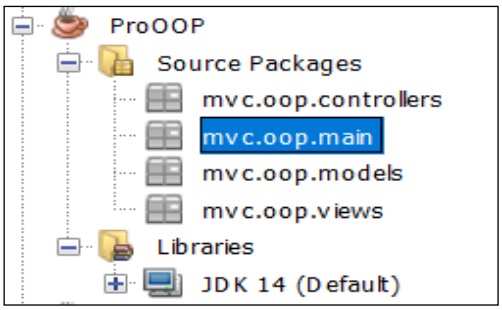

Fig. 6. Packages & libraries

In the source packages section there is a package mvc.oop.controllers used to accommodate all controllers involved in the application. Package mvc.oop.main contains the main program code that contains the main method, so that the program can be run or executed. The mvc.oop.models package contains classes or program codes related to the model. The mvc.oop.views package contains program code to generate output or design the application user interface. The libraries section is a collection of libraries that are used in the application. In the libraries section there is JDK 14 or Java Development Kit version 14.

After preparing the package, then translating the class diagram in Fig. 4 into Java program code and placing it in the appropriate package.

1) Creating Model

The first class is the Lecturer class which is a super class of the Permanent Lecturer and Part Time Lecturer sub classes. The Lecturer class which is a super class is shown in the Fig. 7 program listings.

```
12  public abstract class Lecturer {
13      private String nidn;
14      private String name;
15      double income;
16      final float RATE = 100000;
17
18      public String getNidn() {
19          return nidn;
20      }
21      public void setNidn(String nidn) {
22          this.nidn = nidn;
23      }
24      public String getName() {
25          return name;
26      }
27      public void setName(String name) {
28          this.name = name;
29      }
30      public void setIncome(int credit, int numOfMt){
31          this.income = RATE * credit * numOfMt;
32      }
33      public double getIncome(){
34          return income;
35      }
36  }
```

Fig. 7. Program Listing: Lecturer Class

The Lecturer class declaration on the first line uses the abstract keyword. The abstract keyword means that it does not allow instantiating objects of this class. Object instantiation must be performed on its child class. In this class, the attribute or field declarations that are owned by each lecturer, both permanent lecturers and part time lecturers are declared. This class also contains methods for both changing values and reading values from attributes. After creating the base class, the next step is to create a Permanent Lecturer sub class and a Part Time Lecturer class which is a derivative of the Lecturer class. Fig. 8 is a listing of the Permanent Lecturer class program.

```
12  public class PermanentLecturer extends Lecturer{
13      private String EmployeeID;
14
15      public String getEmployeeID() {
16          return EmployeeID;
17      }
18      public void setEmployeeID(String EmployeeID) {
19          this.EmployeeID = EmployeeID;
20      }
21      public void setIncome(double bs,int credit, int numOfMt){
22          income = bs + RATE * credit * numOfMt;
23      }
24  }
```

Fig. 8. Code Listing: PermanentLecturer Class

In the program code, the extends keyword in the first line indicates that the Lecturer class is a subclass of the Lecturer class. In the next line, a specific attribute is declared to identify the permanent lecturer object, namely EmployeeID. The setIncome() method is a method used to calculate the income of a full-time lecturer. The setIncome() method overloads the setIncome() method in the super class. Method overloading occurs because in the super class the setIncome() method has 2 parameters while the sub class has 3 parameters. Still in the model section, then create a PartTimeLecturer sub class. Listing of the LecturerTransactional class program is shown in Fig. 9.

```
12  public class PartTimeLecturer extends Lecturer{
13      public void setIncome(int credit, int numOfMt) {
14          income = RATE * credit * numOfMt;
15      }
16  }
```

Fig. 9. Code Listing: PartTimeLecturer Class

Just like the Lecturer class, the presence of the extends keyword in the first line indicates that the PartTimeLecturer class is an instance of the Lecturer class. This class contains only one method, namely setIncome(). The setIncome() method in this class overrides the setIncome() method in the super class. The override method occurs because the setIncome() method in the sub class and in the super class has the same number of parameters. Method overloading and override is a way to implement the concept of polymorphism in object-oriented programming. In the Lecturer class and the PermanentLecturer class there is a private keyword that is used in the attribute declaration. The private keyword is a way to implement the concept of encapsulation in programs.

2) Creating View

To create output that can be seen by the user or users in the Java programming language use the System.out.println() command and to create a graphical display use the JFrame object. The view will be designed based on the properties of the lecturer object that have been defined in the model section. Fig. 10 is a program listing of the LecturerView class.

```
12    public class LecturerView {
13      public void lecturerInfo(String nidn, String name,
14          String employeeID, double income){
15          System.out.println("Permanent Lecturer Information");
16          System.out.println("------------------------------");
17          System.out.println("NIDN \t\t: "+nidn);
18          System.out.println("Name \t\t: "+name);
19          System.out.println("Employee ID \t: "+employeeID);
20          System.out.println("Income \t\t: "+income);
21      }
22      public void lecturerInfo(String nidn, String name,
23          double income){
24          System.out.println("\nPart Time Lecturer Information");
25          System.out.println("------------------------------");
26          System.out.println("NIDN \t\t: "+nidn);
27          System.out.println("Name \t\t: "+name);
28          System.out.println("Income \t\t: "+income);
29      }
30    }
```

Fig. 10. Code Listing: LecturerView Class

The LecturerView class has two methods with the same name, namely the lecturerInfo() method. This method is an overloading method because it has a different number of parameters. The first lecturerInfo() method is used to display permanent lecturer information and the second lecturerInfo() method is used to display part time lecturer information.

3) Creating Controller

In the MVC architecture, the controller is the link between the model and the view. Fig. 11 is a listing of the Controller class.

```
8   import mvc.oop.models.PartTimeLecturer;
9   import mvc.oop.models.PermanentLecturer;
10  import mvc.oop.views.LecturerView;
11
12  /**...4 lines */
16  public class LecturerController {
17    private PermanentLecturer model;
      private LecturerView view;
19    private PartTimeLecturer model1;
20
21    public LecturerController(PermanentLecturer
22        model, LecturerView view) {
23        this.model = model;
24        this.view = view;
25    }
26     public LecturerController(PartTimeLecturer
27         model1, LecturerView view) {
28        this.model1 = model1;
29        this.view = view;
30    }
31    public void viewInfoPermanentLect(){
32        view.lecturerInfo(model.getNidn(),
33            model.getName(), model.getEmployeeID(),
34            model.getIncome());
35    }
36    public void viewInfoPartTimeLect(){
37        view.lecturerInfo(model1.getNidn(),
38            model1.getName(), model1.getIncome());
39    }
40  }
```

Fig. 11. Code Listing: LecturerController Class

The LecturerController class above is a liaison between the PermanentLecturer class, PartTime Lecturer in the mvc.oop.models package and the LecturerView in the mvc.oop.views package. In this class, the model and view object declarations are declared. Then define the constructor that will be used to initialize the permanent lecturer object and the part time lecturer object.

In the viewInfoPermanentLect() and viewInfoPartTimeLect() methods, the view object calls the lecturerInfo() method which corresponds to the number of parameters in the LecturerView class. The parameter in the lecturerInfo() method consists of a model object that calls the get method contained in the Lecturer and Permanent Lecturer classes.

4) Creating Main Program

To run the program, you must create a class that contains the main() method. In this class, each lecturer object is instantiated, both permanent lecturers and part time lecturers (Fig. 12).

```
8   import mvc.oop.controllers.LecturerController;
9   import mvc.oop.models.PartTimeLecturer;
10  import mvc.oop.models.PermanentLecturer;
11  import mvc.oop.views.LecturerView;
12
13  /**...4 lines */
17  public class MainMenu {
18    public static void main (String args[]){
19      PermanentLecturer model = permanentLectInfo();
20      PartTimeLecturer model1 = partTimeLectInfo();
21      LecturerView view = new LecturerView();
22      LecturerController c = new LecturerController(model, view);
23      LecturerController c1 = new LecturerController(model1, view);
24      c.viewInfoPermanentLect();
25      c1.viewInfoPartTimeLect();
26    }
27    private static PermanentLecturer permanentLectInfo(){
28      PermanentLecturer permanentL = new PermanentLecturer();
29      permanentL.setNidn("0309111901");
30      permanentL.setName("Brino Jefferson");
31      permanentL.setEmployeeID("K-0140");
32      permanentL.setIncome(5000000, 10, 4);
33      return permanentL;
34    }
35    private static PartTimeLecturer partTimeLectInfo(){
36      PartTimeLecturer partTimeL = new PartTimeLecturer();
37      partTimeL.setNidn("0309111902");
38      partTimeL.setName("Blessy Jeniffer");
39      partTimeL.setIncome(10, 4);
40      return partTimeL;
41    }
42  }
```

Fig. 12. Code Listing: Main Class

Line 19 is the declaration of the model object which is initiated by the call to the permanentLectInfo() method defined in line 27. Line 20 is the declaration of the model1 object which is initiated by the call to the partTimeLectInfo() method defined in line 35. Next line 21 declares the view object. Lines 22-23 are instantiating the LecturerController object as well as initiating the model and view objects. Lines 24-25 are method calls by LecturerController objects, namely c and c1, where both viewInfoPermanentLect() and viewInfoPartTimeLect() methods have been defined in the LecturerController class.

After finishing coding on each part of the MVC architecture, the program can be run. The class that is executed is the Main Menu class. If there is no error (error), it will display the results as shown in Fig. 13.

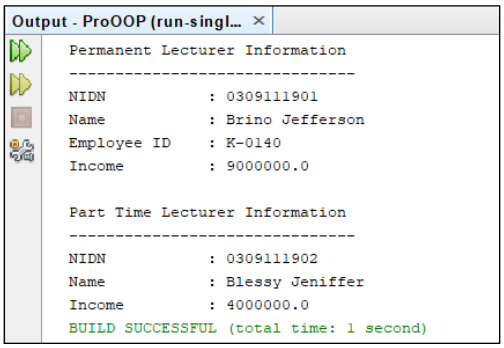

Fig. 13. Program Result

III. Result and Discussion

This research produces a computer application that implements:

1. The concepts of abstraction, inheritance, encapsulation and polymorphism in object-oriented programming using the Java language.
2. The MVC architectural design pattern is shown in Fig. 14. The entire package will contain program codes that are interconnected.

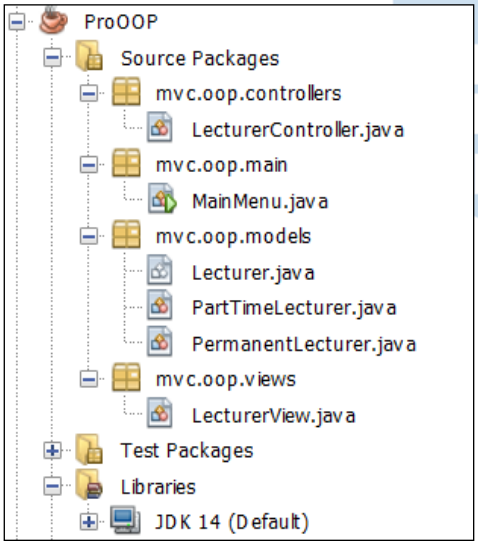

Fig. 14. Package Structure

After attending the lecture, the researcher distributed questionnaires to get feedback from the lecture process and tested whether the lecture process met the expectations of the students. The researcher presents the same questions that were done before the lecture process. Table 2 presents the questions and the results of the questionnaire. A total of 45 students filled out the questionnaire and the difference were 2 people from the respondent's data before the lecture.

To make it clearer, we present the data from the questionnaire which has been processed using a spreadsheet application in the form of a bar chart. For example, the researcher presents two questions, namely question 1 and question 8. The results displayed are data before the lecture (meeting 1) and after the lecture is over (meeting 13). The results on the bar chart of Fig. 15 show that informatics students have a desire to master programming techniques. Furthermore, the same question is asked after following the lecture process and the results are shown in Fig 16. Based on student feedback conducted through questions on the questionnaire that after attending lectures and doing assignments with various topics, the average student understands how to implement MVC architecture in the Java programming language (Fig. 17, 18).

TABLE II. Questions and Questionnaire Results After Class

No	Statement	Score				
		1	2	3	4	5
1	As a computer graduate candidate do you want to master programming techniques.	0	0	2	8	35
2	As a candidate for computer science, I want to have the ability to make computer programs	0	0	3	15	27
3	I now understand the concept of object-oriented programming	0	0	12	25	8
4	I want to master object-oriented programming	0	0	6	12	27
5	I understand how to implement object-oriented programming concepts in the Java programming language	0	0	16	22	7
6	Currently I have heard and understood the concept of Model View Controller (MVC) architecture.	0	0	6	20	19
7	I already understand how to implement the Model View Controller architecture in the Java programming language	0	0	11	27	7
8	I have been able to implement the Model View Controller architecture for desktop application development	0	1	12	24	8

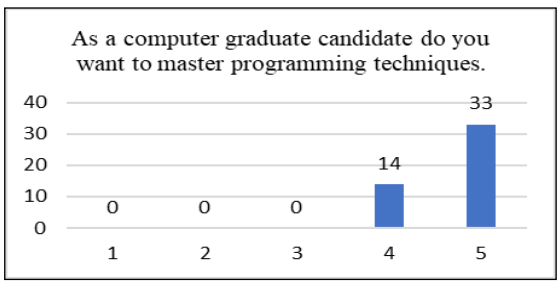

Fig. 15. Before class questionnaire result (Question 1)

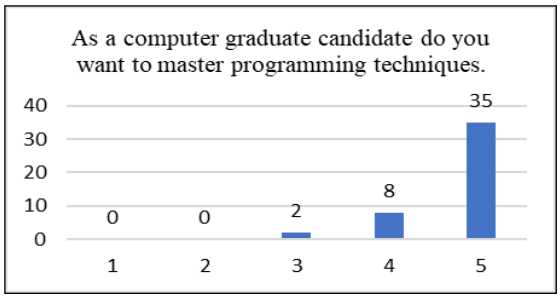

Fig. 16. After class questionnaire result (Question 1)

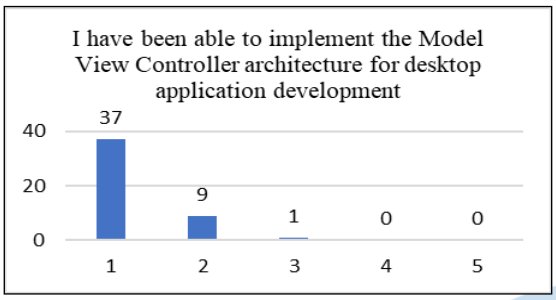

Fig. 17. Before class questionnaire result (Question 8)

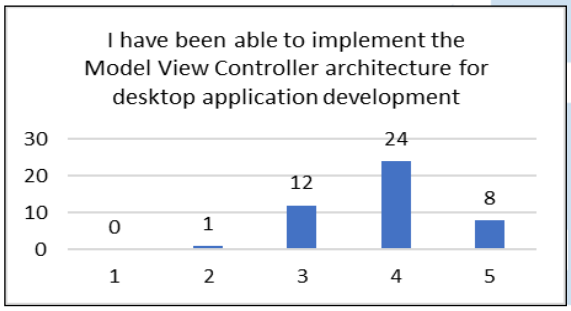

Fig. 18. After class questionnaire result (Question 8)

IV. CONCLUSION

Based on the results of the questionnaire after the lecture, 78% of students strongly agreed and had a desire to master programming techniques, this result increased by 8% from the questionnaire before the lecture. Furthermore, the ability to implement the MVC architecture increases after going through the lecture process. The results of the questionnaire before the lecture showed as many as 79% of students answered strongly disagree, which means that they have not been able to apply the MVC architecture to software development. However, after attending the lecture, 27% answered agree and 53% answered strongly agree. This shows that there is an increase in students' ability to apply MVC architecture in software development. Learning materials in core competency courses must be taught according to industry needs. Understanding how to implement the MVC architectural design pattern can facilitate the development of neater applications, making it easier to detect errors and build complex applications.

ACKNOWLEDGMENT

We would like to express our deepest gratitude to the DRPM of the Ministry of Research, Technology and Higher Education who has supported and financed this research and publication.

REFERENCES

[1] B. Agung, "Indonesia Darurat Tenaga Programmer," CNN Indonesia, 28 July 2017. [Online]. Available: https://www.cnnindonesia.com/teknologi/20170728094848-185-230919/indonesia-darurat-tenaga-programmer. [Accessed 23 October 2019].

[2] S. Perugini, "Emerging languages: An alternative approach to teaching programming languages," Journal of Functional Programming, vol. 29, no. 13, 2019.

[3] S. Singh and S. Kaur, "A systematic literature review: Refactoring for disclosing code smells in object oriented software.," Ain Shams Engineering Journal, vol. 9, no. 4, pp. 2129-2151, 2018.

[4] S. Baset and K. Stoffel, "Object-oriented modeling with ontologies around: A survey of existing approaches," International Journal of Software Engineering and Knowledge Engineering, vol. 28, no. 11n12, pp. 1775-1794, 2018.

[5] R. Setiyawan, W. Gata, Elbiansyah, D. Ardiansyah, D. Yuliandari, D. Wijayanti, F. P. Harmono, Y. Komalasari, M. Lase and M. H. Fakhriza, "Evaluation of PHP Framework Measured Using Object-Oriented Metrics with the Analytic Hierarchy Process," IOP Conference Series: Materials Science and Engineering, vol. 874, no. 1, 2020.

[6] Y. Yan, "Design and Implementation of a Teaching Assistance Platform for College Students Based on ASP. NET," International Journal of Emerging Technologies in Learning, vol. 14, no. 12, 2019.

[7] H. Schildt , Java: The Complete Reference, Eleventh Edition, New York: McGraw-Hill Education, 2018.

[8] K. K. Zaw, W. Zaw, N. Funabiki and W.-C. Kao, "An informative test code approach in code writing problem for three object-oriented programming concepts in java programming learning assistant system," IAENG International Journal of Computer Science, vol. 46, no. 3, pp. 445-453, 2019.

[9] "TIOBE Index for October 2019," TIOBE Programming Community Index, October 2019. [Online]. Available: https://www.tiobe.com/tiobe-index/. [Accessed 15 October 2019].

[10] A. Prajapati, A. Parashar and J. K. Chhabra, "Restructuring Object-Oriented Software Systems Using Various Aspects of Class Information," Arabian Journal for Science and Engineering, vol. 45, no. 12, pp. 10433-10457, 2020.

[11] J. Bräuer, R. Pläsch , M. Saft and C. Körner , "Measuring object-oriented design principles: The results of focus group-based research," Journal of Systems and Software, vol. 140, no. June 2018, pp. 74-90, 2018.

[12] P. Ouyang, W. Cao, M. Wu, C. Gan and F. Wang, "Design of Intelligent Drilling System Software Framework and Data Architecture Based on MVC Pattern," in 2019 Chinese Control Conference (CCC), 2019.

[13] L. Christopher and A. Waworuntu, "Java Programming Language Learning Application Based on Octalysis Gamification Framework," IJNMT (International Journal of New Media Technology), vol. 8, no. 1, pp. 65-69, 2021.

[14] F. Ciccozzi, I. Malavolta and B. Selic , "Execution of UML models: a systematic review of research and practice," Software & Systems Modeling, vol. 18, no. 3, pp. 2313-2360, 2019.

[15] E. Lumba and A. Waworuntu, "Application of Lecturer Performance Report in Indonesia with Model View Controller (MVC) Architecture," in 2nd Asia Pacific Information Technology Conference, 2020