**COMPUTER PROGRAMMING 2**

**KNOWLEDGE TEST**. Write your answer in the blank/space provided. Note that enumeration types are in *all or nothing* form. (2pts each) **NO ERASURE.**

1. A Class diagram is an example of what communication tool? UNIFIED MODELING LANGUAGE
2. Composition represents a \_\_\_\_\_\_\_\_\_\_ relationship. HAS-A RELATIONSHIP
3. \_\_\_\_\_\_\_\_\_\_ is a collection of variables of the same data type. ARRAY
4. \_\_\_\_\_\_\_\_\_\_ is an instance of a class. OBJECT/S
5. What are the 3 Types of method?
   1. VOID
   2. RETURN
   3. CONSTRUCTOR

**APPLICATION TEST**. Provide the correct answer by converting the following words into Java naming conventions. (3pts each) **NO ERASURE.**

1. Change the word “library system” as if it is a class. ANSWER: LibrarySystem
2. Change the word “number of rooms” as if it is a variable. ANSWER: numberOfRooms
3. Change the word “get statistics” as if it is a method. ANSWER: getStatistics

**COMPREHENSION TEST**. Answer the following question with the best of your ability. (please use the next or back pages)

1. Explain the significance of enhanced-for loop?

Enhanced-for loop is a type of pre-test loop – iteration structure that checks first the condition of the loop before executing its instructions, implemented primarily to simplify the way of looping through elements of a Collection such as Arrays, Dictionary, Array-Lists, etc.

For instance, we have declared an array of integers named “x” with elements: 2,4,6,8,9.

Example: printing all the elements of an array.

* 1. int[] x = {2,4,6,8,9};
  2. for(int y : x){
  3. System.out.print(y);
  4. }

In the second line of the example code, we can see the syntax for Enhance-for loop. Specifically, the second and third line of code implies that for each integer inside array x, that will be referred to as y, the program has to display the value of y; giving us an output of 24689.

**ANALYSIS TEST** – Answer the following question with the best of your ability. (please use the next or back pages)

1. Differentiate no-arg methods to var-arg methods.

No-arg methods are instances of a method that has no argument. In programming, methods without argument simplify the way of invoking it and therefore, prevents the programmer to pass information to a method or function. Below is the example of no-arg method signature:

flyHigh()

On the other hand, var-arg methods are instances of a method that contains infinite numbers of arguments provided that these arguments possess the same data type. In addition, you may only implement one var-arg for each definition of a method and must be declared as your last argument inside a method signature.   
  
It is also good to take note that the way of declaring a var-arg is not the same with on how you declare the usual variables. We must write an ellipsis (…) next to the var-arg’s data type and follow by its identifier. Below is the example of method signature with var-arg:

saySomething(bool repeat, String… theWord)

**EVALUATION TEST** – Answer the following question with the best of your ability. (please use the next or back pages)

1. Justify the implementation of enumerators in lieu of strings.

Some algorithms require the use of strings as a form of a look-at value. For example, we created a class named Fries and gave it a Size state wherein the Size must be “small”, “medium” or “large”. The problem with the said example is that the Size (state) should only have one of the three and only three values (i.e. Size = “small” / Size = “medium”). But, by giving it a variable type of string, it goes to show that one may assign “tiny” or “extra\_large” as a value of Size.

Generally, if the value of String is finite (i.e Gender = “Male” / “Female”), it is better to use Enumerators instead. Because enum is a special type of class that holds multiple constants. Below is an example of implementation of an enum:  
  
public enum Size{

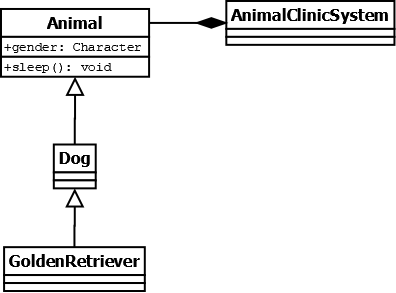
small, medium, large

}

Example assignment:

Size x = Size.small;

SS1 ANSWER:



**CN1 / AS1 / EN1**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| CRITERIA | WEIGHT | MASTERY (4) | PROFICIENT (3) | DEVELOPING (2) | NOVICE (1) |
| Quality of Response | 2 | In-depth understanding of the subject is very evident. Multiple good examples were also given. | In-depth understanding of the subject is evident. A good example was also given. | The response is good but there are some points that were not articulated. | The response is not fully thought out. |
| Mechanics: Spelling, Grammar, Punctuation | 1 | The content has been thoroughly proofread and contains no error. | The content has been thoroughly proofread and contains few errors. | The content has been thoroughly proofread and contains some errors. | The content is delivered sloppy and contains many errors. |
| Penmanship: Neatness, Letter Formation, Sizing | 1 | Always shows evidence of time and effort, use of appropriate letter size and formation. | Often shows evidence of time and effort, use of appropriate letter size and formation. | Often shows evidence of time and effort, but did not use appropriate letter size and formation. | There is no or least effort given. |

**SYNTHESIS TEST** – Develop the following requirements. (please use the back page for item SS1)

1. Create a class diagram with multi-level inheritance wherein the superclass has at least one state and one behavior and, must be a composition to the main class – class wherein main method is implemented.
2. Convert the class diagram in item SS1 to an actual java application.
3. Polymorph a chair class with 3 distinct overloaded constructors per object.

**SS1**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| CRITERIA | WEIGHT | MASTERY (4) | PROFICIENT (3) | DEVELOPING (2) | NOVICE (1) |
| Correctness | 3 | The class diagram fulfilled the specifications completely. | The class diagram fulfilled the specifications but with a few incorrect implementations. | The class diagram fulfilled the specifications with several incorrect implementations. | The class diagram failed to fulfill the specifications. |
| Mechanics:  Neatness, Naming Conventions | 1 | The class diagram is organized excellently and provided with completely correct naming conventions. | The class diagram is organized and provided with few incorrect naming conventions. | The class diagram is somehow organized and provided with several incorrect naming conventions. | The class diagram is not organized making it an ineffective communication tool for developers. |

**SS2**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| CRITERIA | WEIGHT | MASTERY (4) | PROFICIENT (3) | DEVELOPING (2) | NOVICE (1) |
| Coordination | 1 | The Java Application is completely the equivalent to what is specified in the class diagram. | The Java Application is more often equivalent to what is specified in the class diagram. | The Java Application is somehow equivalent to what is specified in the class diagram. | The Java Application does not complement the class diagram. |

**SS3**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| CRITERIA | WEIGHT | MASTERY (4) | PROFICIENT (3) | DEVELOPING (2) | NOVICE (1) |
| Correctness | 3 | The Java Application fulfilled the specifications completely. | The Java Application fulfilled the specifications but with a few incorrect implementations. | The Java Application fulfilled the specifications with several incorrect implementations. | The Java Application failed to fulfill the specifications. |
| Naming Conventions | 1 | The Java application is completely provided with correct naming conventions. | There are few incorrect naming conventions in the Java Application. | There are some incorrect naming conventions in the Java Application. | The Java Code Convention is not comprehended; making the implementation looks like from other convention. |