Lab Exercise-2

Advanced Programming Lab-II

1. Using polymorphism

Make a java class 'Voice'. Inside this class, create a method prepareVoice ( ). This method will have an array of type 'Animal'. The size of the array should be 5. Create five objects: cow, dog, pig, goat and lion, and assign them as the array elements. Make another method hear( ). This method will contain a loop that will display different voices on your terminal. Make another class called 'Test' where you will test your system of classes. This means that Test will contain main method, and therefore, it will be an executable class.

Learning: observe that hear method will have only one statement containing makeVoice(), but makeVoice ( ) will be called for all the animals, depending upon the content of the reference type animal.

2. Writing beautiful code

With reference to the above program, if you call hear () before prepareVoice ( ) or only call hear () but no prepareVoice( ), the program will not work as expected. So you may want to enforce some way on users to use your program. One way to do so is by creating another method inside Voice that will call the methods in required sequence. You can compare this approach to making a special dish according to a recipe. Recipe is a procedure; if you don't follow it you will not get the perfect taste. This is a very-very special way of writing beautiful codes when your procedures are to be called in a strict order. In the world of Software Design, this way of writing codes is called 'Template Methods'.

Write another class Voice2 having a templateMethod ( ) to call the required procedures in sequence.

Learning: Observe that you may enforce some strict order, a recipe, for the methods of your class to achieve some goal—displaying voices of animals, for instance.

3. About Protected-access specifier

Create a class named ‘Pack1’. Define a method named ‘protected void display ( ){….}’ inside this class. Declare this class inside a package “com.juet”. Compile it. Make another class ‘Pack2’ having main method in itself inside a package “com.jiet”. Try creating object of ‘Pack1’ inside the main method of Pack2. Notice the errors and rectify them. Again compile the class.

Learning: A class can only be qualified as public or non-public, it can’t be protected. A method declared as protected inside one class in a package, cannot be accessed inside another class in a different package. So, either declare the member ‘public’ in class one, or make another class extending the class containing protected member. But in the later case it can be accessed only via object of second class.