Practical 5 – OO Principles & UML

Exercise 1:

Draw a UML class diagram that models the relationship between the following classes: Bank, ATM, Customer, Account, ATM Transactions, Current Account and Saving Account.

Use the tool https://www.draw.io/ to create a class diagram to solve the above program.

Exercise 2

Nicosia municipality in an attempt to offer a climate-friendly solution to move around the city is starting a bike renting program. The competent authority needs an information system to be able to store the information regarding the rentals.
The following requirements must be taken into consideration:

- It should be possible to store the ID number, name, surname and address for every client. The ID must be different for every client and all clients should have at least a ID and full name.
- The database should also contain information about the bicycle models that can be rented- Each model has a unique name, a type (that can only be road, mountain, bmx or hybrid) and the number of gears.
- Several different points will be set up where bicycles can be picked up and returned. Each one of these points is identified by a unique name and has an address.
- The system will also offer the opportunity to make an online reservation.
- When a reservation is made, the following data must be known: which client did the reservation, when will he pick up the bike (day), which bike model he wants and where will he pick up the bike (store).
- When a bike is picked up, the actual bike that was picked up must be stored in the database.

- When a bike is returned, the return date should be stored in the database.

Use the tool https://www.draw.io/ to create a class diagram to solve the above program.

Exercise 3

Create an **abstract class Animal** with:

1. An abstract method ***abstract void sleep();***
2. The method ***public void makeSound()*** that prints “I can make every sound”.
3. With the method ***public void eat()*** that prints “I can eat!”
4. Create an ***interface Pet,*** this interface must have the **private** fields **name, weight, age ownerName, hasMicrochip and microchipCode.** Create **setter** and **getter** methods for each field and the method **addMicrochip(String code)**.
5. Create a class **Dog** that extends the Animal class and implements the Pet interface.
6. The Dog class must implement the **sleep**() .
7. Create another method ***public void makeSound()*** that prints the dog’s sound. What type of Polymorphism is that? Write your answer in a comment.
8. Create another method ***public void eat(String food)*** that prints “I can eat” + food +”!”. What type of Polymorphism is that? Write your answer in a comment.
9. Also, the Dog class must implement each method of the Pet interface. The method addMicrochip(String code) must set the variable hasMicrochip as true and the value of microchipCode equal to the given code.
10. In a **Driver** class create:
    a. Animal dog1= new Dog();
    b. Dog dog2 = new Dog();
    c. Initialise the needed values of each object considering that dog1 doesn’t have a microchip but dog 2 has. Then call **addMicrochip** to add a microchip to dog1.
    d. Call ***makeSound***() method for both dogs and explain in comments the output result.