



COMP-455
OOB using Java Language

EXERCISES/3RD SET

Lecturer:	<i>Dr. Constandinos X. Mavromoustakis</i>
Skills Examined:	<i>Java Classes and Java methods and types/ Object-Oriented Programming: Inheritance</i>
Date of release:	<i>25/11/2009</i>

Exercise 1:

(a) A subclass can inherit "interface" or "implementation" from a superclass. How do inheritance hierarchies designed for inheriting interface, differ from those designed for inheriting implementation (behavioral characteristics of a class)?

(b) What are abstract methods? (i) Describe the circumstances in which an abstract method would be appropriate. Mention a real time example. (ii) Then from that precise example, give the real code (but do not write the source for the methods/only the "spine" of the code) with no source code implementation.

(c) Create a sample final class called MySample with a final method. Inherit from that class and attempt to override that method. What it then occurs after the compilation? What happens during execution? Show the source code and the outcome. (You need to write and provide : The error-free source code and the executable along with a precise documentation explaining the given code and the outcome of it).

Exercise 2:

Create a class with a constructor or a method that is overloaded three times. Inherit by extending this to a new class called MyInheritance, and then, to the class MyInheritance add new overloaded versions of the methods. Show that all methods are available in a/the derived class.

(You need to submit : The error-free source code and the executable along with a precise documentation explaining the given code).

Exercise 3:

Create a class interface called Amphibian. From this, inherit/implement a class called Bronchus. Put appropriate methods (of your choice) in the base bronchus class. Then, create a class Frog which inherits and obeys to all characteristics of Amphibian and Bronchus (see the Diagram 1). In a TestClassBronchus, in the static main() function, create a Frog and upcast it to Amphibian, and demonstrate that all the methods still work (of all sides).

Create your own scenario for this exercise, and put practical outputs so that there are proofs that all inherited methods are still working. (You need to write and provide: The error-free source code and the executable along with a precise documentation explaining the inheritance characteristics of the given code).

Diagram 1 for the exercise 3 above:

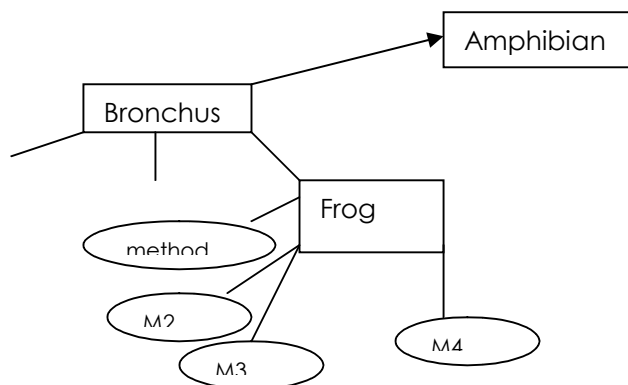


Diagram 1

Exercise 4:

Create a class interface called *Building*. From this, inherit/implement a class called *UniversityBuilding*. Put appropriate methods (of your choice) in the base *UniversityBuilding* class. Then, create the classes *EuropaBuilding* and *MainBuilding* which inherit and obey to all characteristics of *Building* and *UniversityBuilding* (see the Diagram 2). In a *TestClassUniversityBuilding*, in the static *main()* function, create *EuropaBuilding* and *MainBuilding* and upcast it to *UniversityBuilding*, and demonstrate that all the methods declared work perfectly (of all sides).

Use your imagination and create your own scenario for this exercise, and put practical outputs so that there are proofs that all inherited methods are still working.

Diagram 2 for the exercise 4:

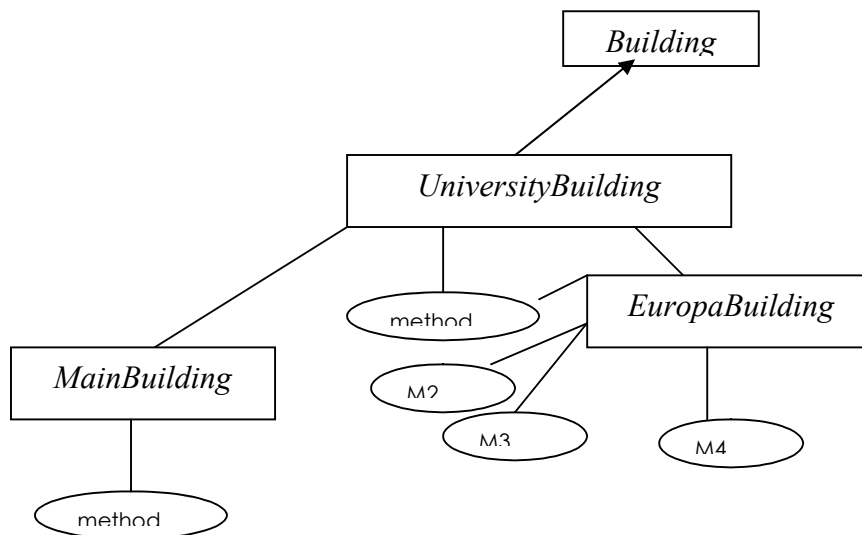
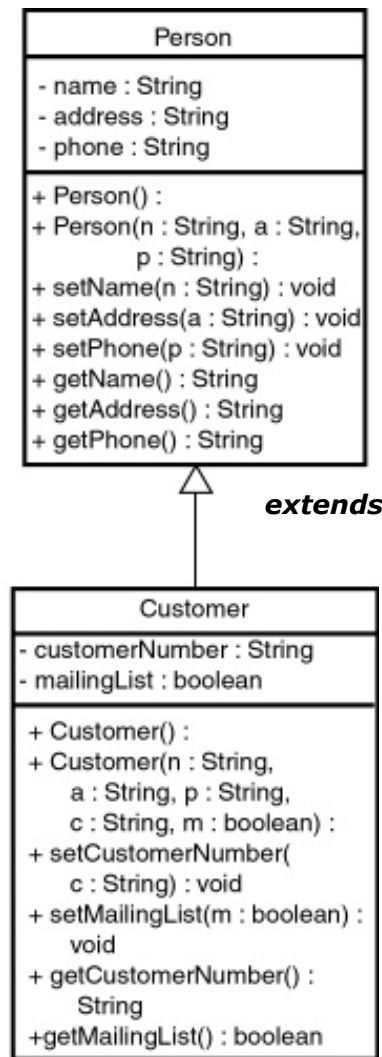


Diagram 2

(You need to write and provide: The class diagram, the error-free source code and the executable screenshots along with a precise documentation explaining the inheritance characteristics of the given code/ Please ONLY hardcopy should be submitted. Additionally -as it is mentioned- for each exercise support, screenshots of the possible output are needed).

Exercise 5:

Create a class called Person which has the following (class diagram) attributes:



Then write a class called Customer which *inherits* the characteristics of the Person (since it is referred to as a Person) with the attributes as shown above and as follows:

```
private String customerNumber; // Customer number
private boolean mailingList; // Mailing list flag

/**
 * Constructor
 * Initializes fields with default values
 */

public Customer()
{
}
```

```

/**
 * Constructor
 * Initializes fields with argument values
 */

public Customer(String n, String a, String p,
                String c, boolean m)
{
}
/**
 * setCustomerNumber method
 */
public void setCustomerNumber(String c)
{
    customerNumber = c;
}

/**
 * setMailingList method
 */
public void setMailingList(boolean m)
{
    mailingList = m;
}
/**
 * getCustomerNumber method
 */
public String getCustomerNumber()
{
    return customerNumber;
}
/**
 * getMailingList method
 */

public boolean getMailingList()
{
    return mailingList;
}
}

```

By using your own code scenario in the methods and for the constructors, thereafter create a class **CustomerClassDemo** (which has *public static void main(String[] args)*) by which you test your program if it operates and obeys to your own scenario.

(You need to write and provide: The error-free source code and the executable screenshots along with a precise documentation explaining the inheritance characteristics of the given code).

Exercise 6:

*Create an inheritance hierarchy of **Rodent**: **Mouse**, **Gerbil**, **Hamster**, etc. In the base class, provide methods that are common to all **Rodents**, and **override** these in the derived classes, to perform different behaviours depending on the specific type of **Rodent**. Create an array of **Rodent**, fill it with different specific types of Rodents, and call your base-class methods to see what happens.*

(Hint: Create a class diagram for this exercise)

(You need to write and provide: The class diagram, the error-free source code and the executable along with a precise documentation explaining the inheritance characteristics of the given code expressing the above class scenario).