Lab 5: Inheritance and Method Overriding

Lab Objective: After this lab, the student should be able to:

- Use inheritance in Java
- Override methods
- Understand assignments between objects of superclasses and subclasses
- Use ***instanceof*** operator
- Test objects equality
- Use polymorthism

Part 1:

We want to design a hierarchical model of the IGEE employees. Code a class named **Person** which has name, address, phone number, and email address as fields. A person may be an employee or a student. A student has a class status (freshman, sophomore, junior, or senior). Define the status as a constant. An employee has an office, salary, and hiring date. We distinguish two types of employees, faculty and staff. A faculty member belong to a department, has office hours and a rank. A staff member has a title, may be at administration or support. Each class has a method work showing a custom message of the object job.

Ensure that you define a coherent hierarchy of classes using inheritance.

For each class,

- Define a default constructor in which you initialize the fields to default values then print the name of the class (this will serve to trace the constructors call when testing your code).
- Define a copy constructor. The copy constructor takes an instance of the same class as its argument. If the argument is null set the fields to their default values, otherwise make a deep copy of each object field.
- Declare the mutators and assessors as needed.
- Override the method **toString** (inherited from Object class) in each class to display the class name and the person's name.
- Override the method **equals** (inherited from Object class) which checks whether the instance provided as argument is equal to the object calling equals. Check for null parameter, use ***instanceof*** operator to check that the two objects are co-variant, compare the references and compare all the fields' values.

In the main code:

1. Instantiate an Object of each class using the default constructor. For each instance, what are the called constructors, in which order and why?
2. Assign the created objects mutually, and tell when it works (fill the table below). For the cases where you cannot assign, use casting, does it work now? why?

	Person	Employee	Student	Faculty	Staff
Person	works				
Employee		works			
Student			works		
Faculty				works	
Staff					works

   - Can we assign a subclass object to superclass reference?
   - Can we assign a superclass object to a subclass reference?
   - Can we assign an object to a reference of class which does not inherit from the class of the object?
3. Check all the objects mutually with the ***instanceof*** operator. Fill the results in the following table.

	Person	Employee	Student	Faculty	Staff
Person	true				
Employee		true			
Student			true		
Faculty				true	
Staff					true

1. Create a copy of a person object with copy constructor and test your equals method.
2. Type the following code

```
Person p=new Person(),pe,ps,pf,pt;
pe= new Employee();
ps= new Student();
pf= new Faculty();
pt= new Staff();
```

- Call toString() method on each object and explain the result.
- Call equals on `p` objet four times, taking each time one of the other objects (pe, ps, pf and pt) as parameter. Explain the result.
- Create an array of type `Person` fill it with objects from the four subclasses. Iterate over the array and call the method works for each object. Explain how the polymorphism concept being used here.

Part 2:

We want to represent geometric shapes and some operations that can be performed on them. The shapes application will consist of the following classes: **Shape, Circle, Square, Cube, Sphere, Cylinder,** and **Glome.**

Our design choice involves that Shapes in higher dimensions inherit data from lower dimensional shapes. For example

- A cube is a three dimensional square.
- A sphere is a three dimensional circle
- A glome is a four dimensional circle.
- A cylinder is another kind of three dimensional circle.

Note: the inheritance relationship is said to be "**is a**"**,** which means a subclass object is also an object of the superclass.

The circle, sphere, cylinder, and glome all share attribute radius. The square and cube share the attribute side length.

All shapes inherit getName() from the superclass Shape.

Shapes have a method called draw() which simulates drawing the shape.

Each shape has an area. High dimensional shapes have a volume.

Design and implement the classes involved in representing the shapes