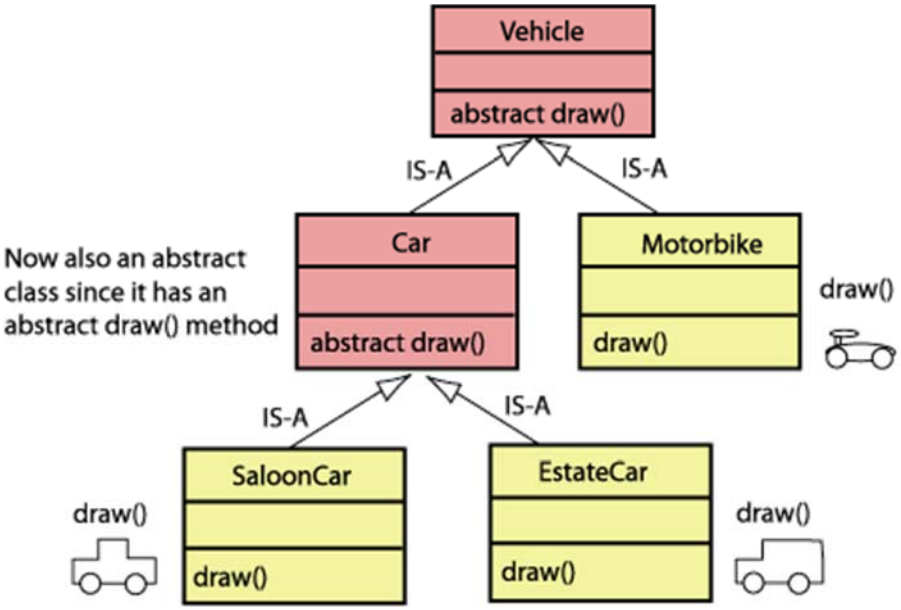
Object Orientated Programming

Lab Manual 9

Instructor: Mr. Samyan Qayyum Wahla **Learning Objectives:** • Interfaces • Abstract Classes • Multilevel Inheritance • Multiple Inheritance	**CLO:** **Registration Number:** 2020-CS-144 **Name:** Faraz Ahmad

Task1:

Implement the following design:



Q1. Implement the above design. And write the code here. In draw method, you can choose to just print some statements.

Q2. Why Vehicle and Car should be abstract?

There are unimplemented methods in vehicle and car (methods without body), so instead of changing it into an interface, we made it an abstract class containing abstract methods.

Q3. Is SaloonCar object can be manipulated using reference of Vehicle class. Give some examples.

Yes it can.

Vehicle saloonVehicle = new SaloonCar();

saloonVehicle.draw(); // Draws Saloon Car .//

Task 2:

Consider the example of Computer Science Department at UET Lahore. Apart from the teacher, Department also has some employee which are postgraduate student themselves and teaching. They are known as Graduate Assistants. Create the following classes.

1. Person (Base class of all)
2. Employee (Inherited from Person)
3. Student(Inherited from Person)
4. UndergraduateStudent(Inherited from Student)
5. PostgraduateStudent (Inherited from Student)
6. Teacher (Inherited from Employee)

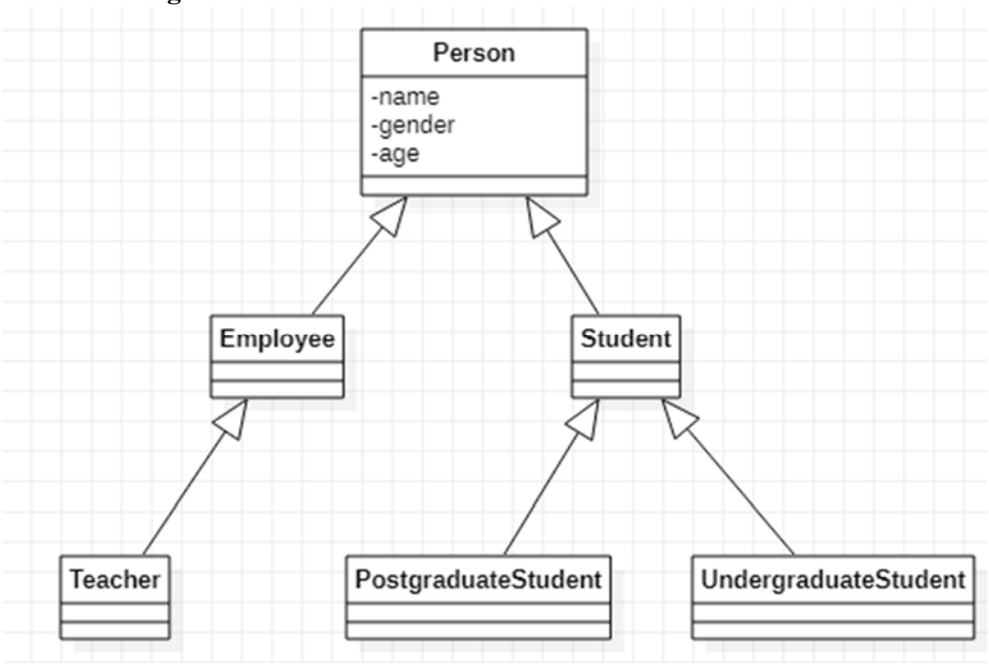
Q1. Add the reasonable attributes and functions in each class.
Q2. Show in the driver class that how the teacher object can be upcasted to Person object.

```
Person p = new Teacher();
p.print();
```

Q3. Show in the driver class that you can use the functions of Person class using the object of Undergraduate Student.

```
UndergraduateStudent ug = new UndergraduateStudent();
ug.getName();
```

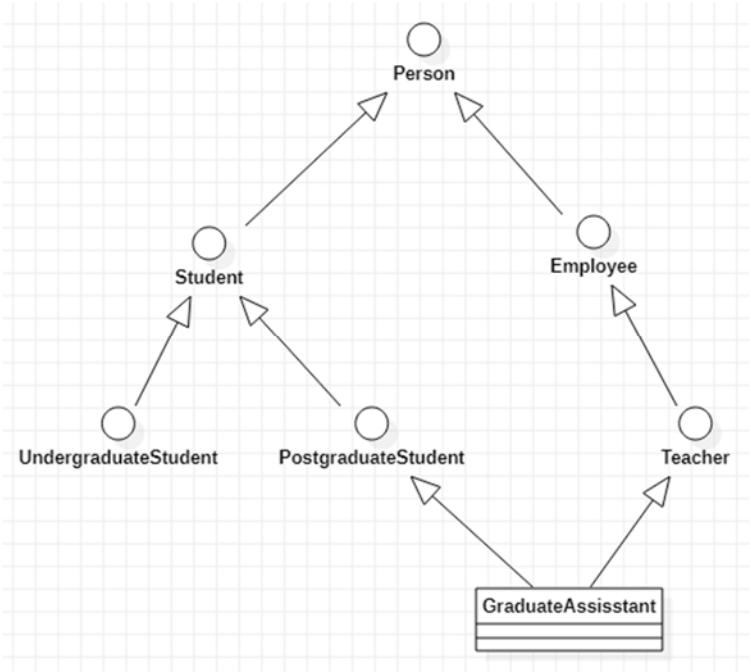
Q4. Create the class diagram of above classes in starUML.



Q5. Add a new class GraduateAssisstant and inherit it from PostgraduateStudent and Teacher. If you cannot inherit multiple classes, then what is the solution?

Ans) Change person class to interface and all child classes too

Q6. Draw the updated class diagram in startUML



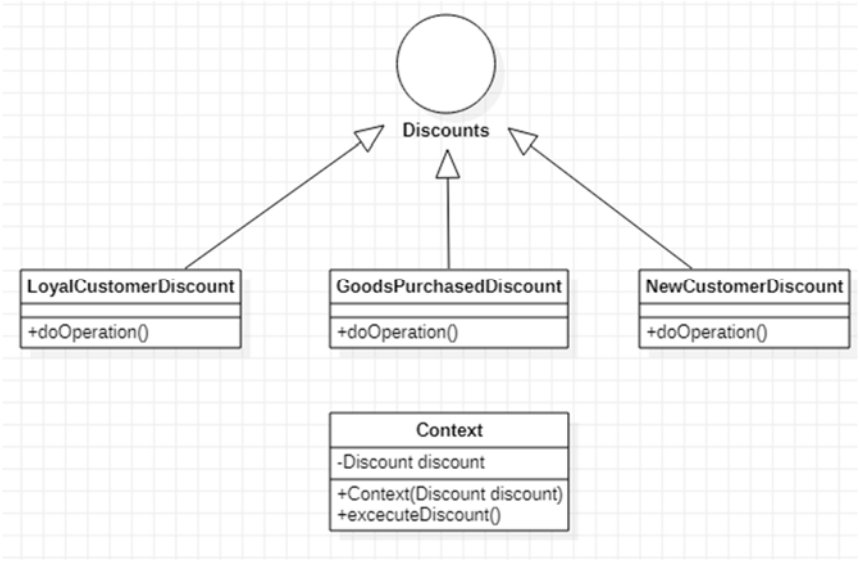
Q7. Add the complete code here.

Ans)

```
interface Person {
    String name = "";
    int age = 0;
    String gender = "";
    public default void print() {
        System.out.println("Person");
    }
}
interface Employee extends Person {
    int salary = 0;
}
```

```
interface Teacher extends Employee {
   String department = "Computer Science";
}
interface Student extends Person {
   int fees = 0;
   String regNo = "2020-CS-144";
}
interface PostgraduateStudent extends Student {
   public default void print() {
      System.out.println("Postgraduate Assisstant");
   }
}
interface UndergraduateStudent extends Student {
   public default void print() {
      System.out.println("Undergraduate Assisstant");
   }
}
class GraduateAssisstant implements Teacher, PostgraduateStudent {
   public void print() {
      System.out.println("Graduate Assisstant");
   }
}
public class Task2 {
public static void main(String[] args) {
      // Teacher p = (Teacher) new Person();
      // Person obj = new Teacher();
      // obj.print(); // Prints "Person" .//
   }
}
```

Task 3:




```
interface Discount {
    String doOperation();
}
class NewCustomerDiscount implements Discount {
    @Override
    public String doOperation() {
        return "New Customer Discount";
    }
}
class LoyalCustomerDiscount implements Discount {
    @Override
    public String doOperation() {
        return "Loyal Customer Discount";
    }
}
class GoodsDiscount implements Discount {
    @Override
    public String doOperation() {
```

```
        return "Discount for customers purchasing goods of more than Rs. 10000";
    }
}
class Context {
    private Discount discount;
    public Context(Discount discount) {
        this.discount = discount;
    }
    public String executeDiscount() {
        return discount.doOperation();
    }
}
public class Task3 {
    public static void main(String[] args) {
        Context context;
        // Prints new customer discount
        context = new Context(new NewCustomerDiscount());
        System.out.println(context.executeDiscount());
        // Prints loyal customer discount
        context = new Context(new LoyalCustomerDiscount());
        System.out.println(context.executeDiscount());
        // Prints Discounts for purchase more than 10000
        context = new Context(new GoodsDiscount());
        System.out.println(context.executeDiscount());
    }
}
```