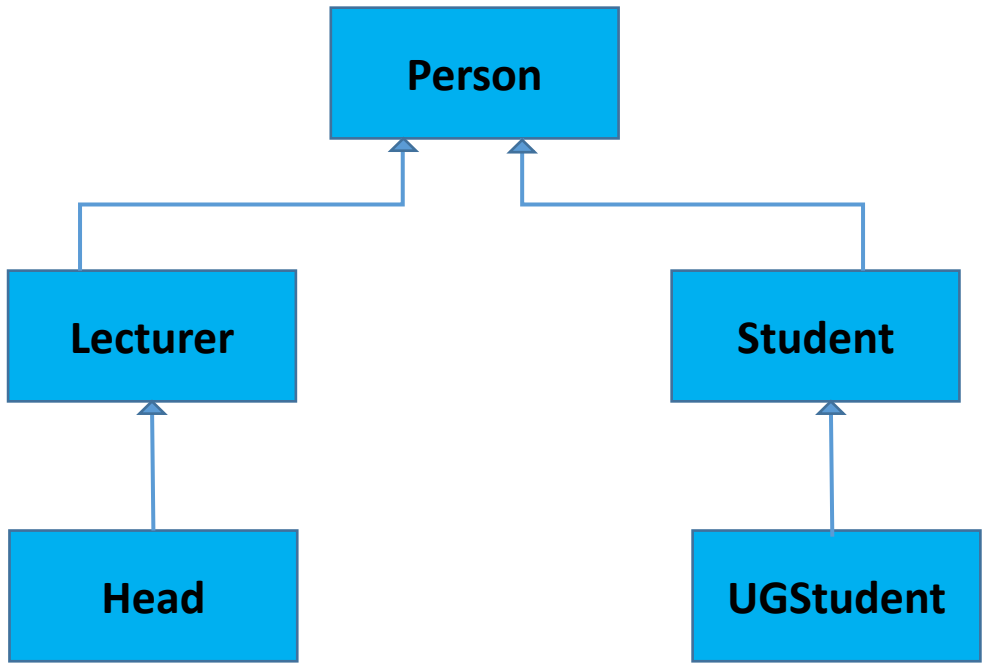
UFCFB6-30-2 Object-oriented Software Development Lab
Week 6(19)

Today you are free to work either in a group or independently.

Objectives of today's lab:

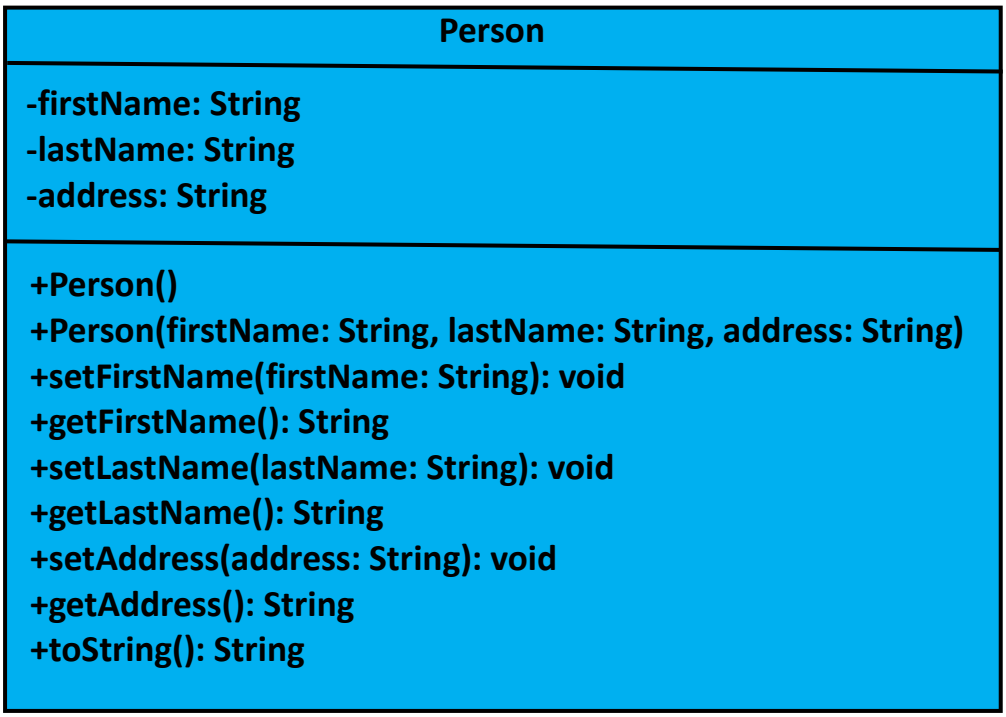
- design2code and Java/Inheritance recap

Consider the following class diagram. Suppose you are developing a software system for FET-UWE, and you have already designed, written, coded and tested a class called Person. An object in this class holds personal information about somebody using his name and address. A Person also has methods that can construct a new person from given data, print the information in a standard format, set and get the name and address and so on. Now FET-UWE also wants to implement a *Lecturer* class, a *Head* class, a *Student* class, and a *UGStudent* class that are all different kinds of People.



UFCFB6-30-2 Object-oriented Software Development Lab
Week 6(19)

A complete UML class diagram of the Person class and its corresponding Java code implementation are shown below. Note that all the member variables are private and methods are public.




```
public class Person {
    private String firstName;
    private String lastName;
    private String address;

    public Person() {
    }

    public Person(String firstName, String lastName, String address)
   {
        this.firstName = firstName;
        this.lastName = lastName;
        this.address = address;
    }

    public String getFirstName() {
        return firstName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public String getLastName() {
        return lastName;
    }
```

```
    public void setLastName(String lastName) {
        this.lastName = lastName;
    }

    public String getAddress() {
        return address;
    }

    public void setAddress(String address) {
        this.address = address;
    }


    @Override
    public String toString() {
        return "Person's name is " + getFirstName() +" "+
getLastName() +"\n Address is "+ getAddress();
    }

    public static void main(String args[])
   {

       Person p = new Person();
       System.out.println(p);
       p.setFirstName("James");
       p.setLastName("Lear");
       p.setAddress("FET-CSCT");
       System.out.println(p);
       Person p1 = new Person("Kun", "Wei", "FET-CSCT");
       System.out.println(p1);

   }
}
```

Task 1: Study the Person class diagram and the corresponding Java code. Write, save, compile, and run the above code.

/* file name should be `Person.java`*/

A successful run of your program can display the following output:

```
Person's name is null null
Address is null
Person's name is James Lear
Address is FET-CSCT
Person's name is Kun Wei
Address is FET-CSCT
```

UFCFB6-30-2 Object-oriented Software Development Lab
Week 6(19)

Task 2. Design complete UML diagrams of other four classes *Lecturer*, *Head*, *Student*, and *UGStudent* using Astah. The classes *Lecturer* and *Student* extend Person, *Head* extends *Lecturer*, and *UGStudent* extends *Student*. Your four complete UML classes contain the following:

Lecturer class:

- One integer data field representing lecturer ID
- A no-argument constructor that creates a default lecturer
- A constructor that creates a lecturer with the specified name, address, and the lecturer ID (note that you need to access name and address from the base class)
- All the appropriate setters and getters methods
- A method named toString() that returns a string description for the lecturer

Head class:

- One string data field representing HOD of a specific department [e.g., HOD-CSCT]
- A no-argument constructor that creates a default HOD
- A constructor that creates a HOD with the specified name, address, the lecturer ID, and HOD of a specific department (note that you need to access required fields from the base class)
- All the appropriate setters and getters methods
- A method named toString() that returns a string description for the HOD

Student class:

- One integer data field representing student ID
- A no-argument constructor that creates a default student
- A constructor that creates a student with the specified name, address, and the student ID (note that you need to access name and address from the base class)
- All the appropriate setters and getters methods
- A method named toString() that returns a string description for the student

UGStudent class:

- One String data field representing degree name name (e.g., BSc (Hons) in CS)
- A no-argument constructor that creates a default UG student
- A constructor that creates a UG student with the specified name, address, the student ID, and the degree name (note that you need to access required fields from the base class)
- All the appropriate setters and getters methods
- A method named toString() that returns a string description for the UG student

Task 3. Implement your classes.

Task 4. Read the SOLID principles covered in lecture 5, Justify whether your class design and implementation conform to the open-closed principle.