Topic: Applicable modifiers in classes declarations.
OOP concepts involved: Classes, Class Attributes, Class Access Modifiers, Inheritance.
Programming generic concepts involved: Variable declaration, Value assignment to variable, Primitive data types, Data capture.

➢ Theoric introduction

CLASS MODIFIERS

In Java, **modifiers** are keywords that can be utilized when defining a class which will affect its runtime behaviour. There are multiple modifiers that can be used when declaring a class, some of which are:

- Access Modifiers
- Final
- Abstract
- Static

Not all modifiers are allowed on all classes, for example an interface cannot be final and an enum cannot be abstract. A class can be declared using one or more Modifiers using this basic form:

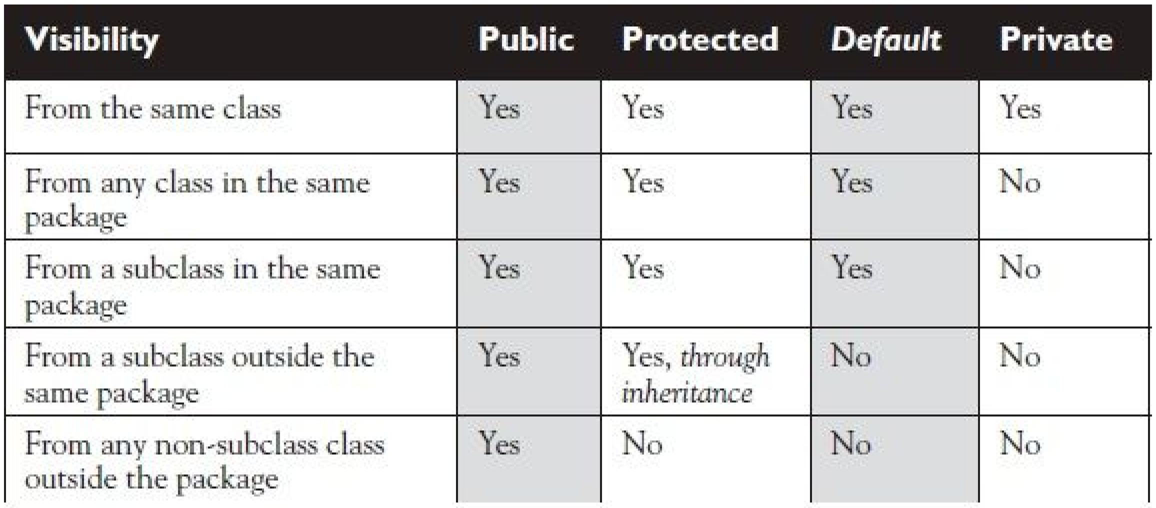
```
[Access Modifier] [Modifier(s)] class identifier{}
```

ACCESS MODIFIERS

Access Modifiers, like the name suggests, are modifiers used to restrict the scope of a class, that is to say, who can access and modify the values in it. This type of modifiers aren't exclusive to classes: constructors, variables, methods and data members can use this modifier too.

There are four types of Access Modifiers, each of whom, allow access to a certain level.

- ❏ Default : When no access modifier is specified for a class, constructor, variable, method or data member, it is said to be having the **default** access modifier. The classes, variables, methods or data members with this access modifier, are only accessible within their own package.
- ❏ Private: Any method, variable or data member declared as **private** will only be accessible within the same class. Any other class in the same package will not be able to access these members. Classes and interfaces cannot be declared **private**.
- ❏ Protected: The methods or data members declared as **protected** are accessible within the same package or subclasses in different packages.
- ❏ Public: This access modifier has the widest scope among all of them. Classes, methods, variables or data members declared as **public** are accessible from everywhere in the program. There is no restriction in who can access these members.



Visibility	Public	Protected	*Default*	Private
From the same class	Yes	Yes	Yes	Yes
From any class in the same package	Yes	Yes	Yes	No
From a subclass in the same package	Yes	Yes	Yes	No
From a subclass outside the same package	Yes	Yes, *through inheritance*	No	No
From any non-subclass class outside the package	Yes	No	No	No

FINAL MODIFIER

The final modifier can be used on a Class, Method or Variable to prevent its value from being changed, depending on what it is used, it will work in a different way.

- ❏ Final **variables**: These variables can only be initialized once, and cannot be reassigned to refer to a different object, however, this does not prevent the data within the object from being changed. In other words, the state of the object can be changed, but not the reference.

- ❏ Final **methods**: Final methods cannot be overridden by any subclasses, this makes it so that any class that inherits from a class with a final method, cannot change the contents of it.
- ❏ Final **classes**: These kind of classes are used when want to prevent a class from being subclassed. If a class is declared as final, no other class can inherit anything from it.

EXTENDS

When creating a class, we can use the keyword **extends**, to inherit all the attributes and methods defined in another class, this is a process called Inheritance. Below is an example of the basic form of a class extending another.

```
[Access Modifier] [Modifier(s)] class identifier extends identifier{}
```

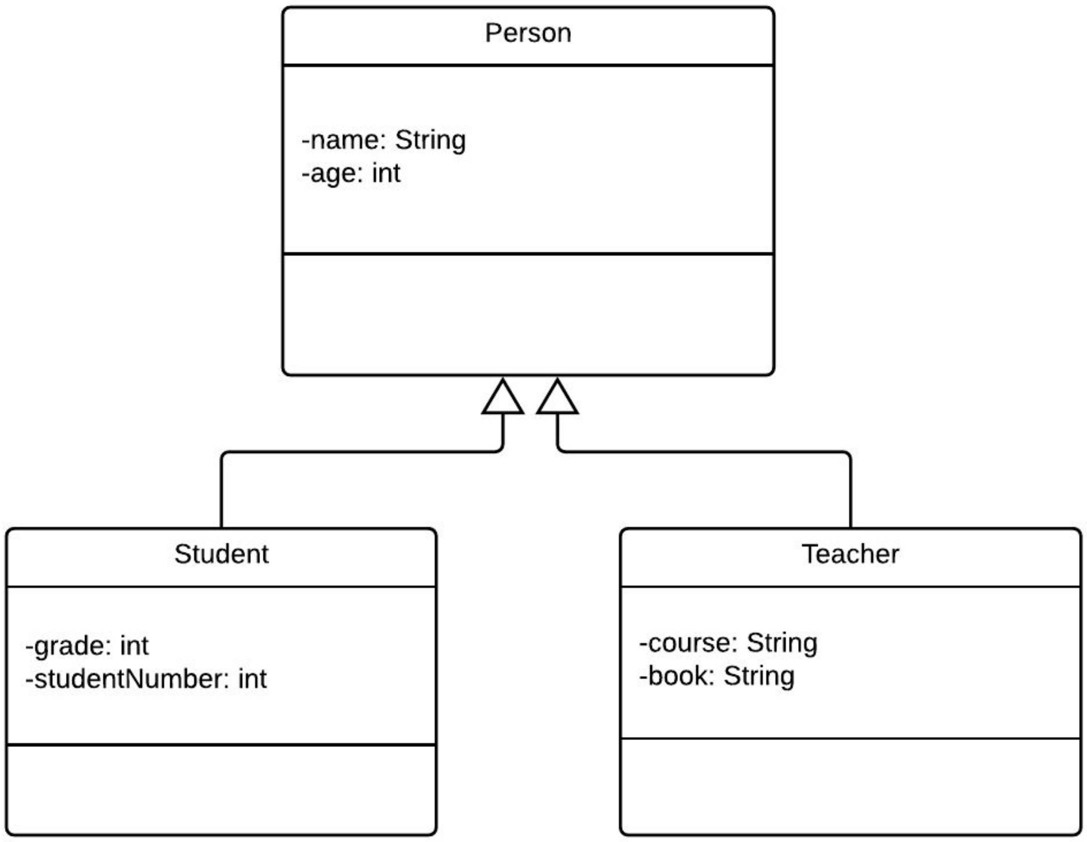
What this does, is it allows the class extending (also known as subclass or child class) to act as is it had attributes and methods from the class being extended (also known as the parent class). The subclass also has the option to override methods in the parent class, changing the contents in them, unless the final modifier is used.

➢ Statement

A school wants to register all its teachers into a database, and we need to know the name, school name and shift of each teacher. Also, all math teachers must also include their course and book used for it into the database.

Develop a program that manages the registration of all teachers, while giving a distinction between a regular Teacher and a Math Teacher.

➢ Class design (UML)



➢ Program Code

Person.java

```
public class Person{
        //Defining object attributes (Its state)
        private String name;
        private int age;

        // Constructor Method
        public Person(){
        }

        public void printInfo(){
                System.out.println("\nInformation about the Person:");
                System.out.println("Name:  " + this.getName());
                System.out.println(" Age:  " + this.getAge());
        }

        //Getters and Setters are set to final to avoid subclasses changing the contents of
them.
        public final String getName(){
                return this.name;
        }

        public final int getAge(){
                return this.age;
        }

        public final void setAge(int age){
                this.age = age;
        }

        public final void setName(String name){
                this.name = name;
        }
}
```

Student.java

```
public final class Student extends Person{
        //By making the class final, we prevent any subclass from inheriting from this class
        private int grade;
        private int studentNumber;

        // Constructor Method
        public Student(){
                super();
        }

        //We can override the contents in a method in the subclass
        public void printInfo(){
                System.out.println("Information about the Student:");
                System.out.println("Name: " + this.getName());
                System.out.println("Age: " + this.getAge());
                System.out.println("Grade: " + this.getGrade());
                System.out.println("Student Number: " + this.getNumber());
        }

        //Getters and Setters
        public final void setGrade(int grade){
                this.grade = grade;
        }

        public final void setNumber(int studentNumber){
```

```java
                this.studentNumber = studentNumber;
        }

        public final int getGrade(){
                return this.grade;
        }

        public final int getNumber(){
                return this.studentNumber;
        }
}
```

Teacher.java

```java
public final class Teacher extends Person{
        //By making the class final, we prevent any subclass from inheriting from this class
        private String course;
        private String book;

        // Constructor Method
        public Teacher(String course){
                super(); //This line of codes calls the constructor of the parent class
                 this.course = course;
        }

        //We can override the contents in a method in the subclass
        public void printInfo(){
                System.out.println("Information about the Teacher:");
                System.out.println("Name: " + this.getName());
                System.out.println("Age: " + this.getAge());
                System.out.println("Course: " + this.getCourse());
                System.out.println("Book used: " + this.getBook());
        }

        //Getters and Setters
        public final void setCourse(String course){
                this.course = course;
        }

        public final void setBook(String course){
                this.book = book;
        }

        public final String getCourse(){
                return this.course;
        }

        public final String getBook(){
                return this.book;
        }
}
```

Main.java

```
import java.util.Scanner;
import java.lang.Integer;

public class Main {

        public static void main(String[] args) {
                // Declaring and instantiating a Teacher and a Student object
                Teacher t1 = new Teacher(“POO”);
                Student s1 = new Student();

                // Declaring the object to allow data reading predefining object System.in
               //  for standard data input from the keyboard
                Scanner sc = new Scanner(System.in);

                // Capturing data through the console and assigning them
                System.out.println("What's the Teacher's name?");
                t1.setName(sc.nextLine());
                System.out.println("What's the Teacher's age?");
                t1.setAge(Integer.parseInt(sc.nextLine()));
                System.out.println("What's the Teacher's course?");
                t1.setCourse(sc.nextLine());
                System.out.println("What's the Book used in the course?");
                t1.setBook(sc.nextLine());
                t1.printInfo();

                System.out.println("What's the Student's name?");
                s1.setName(sc.nextLine());
                System.out.println("What's the Student's age?");
                s1.setAge(Integer.parseInt(sc.nextLine()));
                System.out.println("What's the Student's grade?");
                s1.setGrade(Integer.parseInt(sc.nextLine()));
                System.out.println("What's the Student's number?");
                s1.setNumber(Integer.parseInt(sc.nextLine()));
                s1.printInfo();


                sc.close(); // Closing data flow
        }
}
```

➢ Program execution

The objective of the program is to create two extended class that inherits all of the attributes a basic Person class. The program also uses use of the Final modifier, to avoid changes in certain variables, methods and to avoid the subclass from being Inherited.

At the end of the program, we see the data we’ve collected for our two “specialized” classes, Teacher and Student.

```
What's the Teacher's name?
Reyes Juarez
What's the Teacher's age?
40
What's the Teacher's course?
Object Oriented Programming
What's the Book used in the course?
Effective Java
Information about the Teacher:
Name: Reyes Juarez
Age: 40
Course: Object Oriented Programming
Book used: Effective Java

What's the Student's name?
John Doe
What's the Student's age?
19
What's the Student's grade?
4
What's the Student's number?
234134
Information about the Student:
Name: John Doe
Age: 19
Grade: 4
Student Number: 234134
```

➢ Conclusions

All modifiers are great for limiting what others can do with certain variables, methods or classes. This gives the programmer control with the data and how it can be accessed, protecting the information.

While some of them, like **extends**, allow us to inherit attributes and methods from a class to create a subclass and have more control with our classes and save time re-coding what we already have.