```
2021-10-31 -- Scratchpad of CSE213 (Sec-1)
==========================================

Parameterized constructor:
==========================
- It is not mandatory to define a parameterized constructor for ALL
fields

public class SomeClass{
        private int x, y, z;
        public SomeClass(){...} //default
        public SomeClass(int val1){ x=val1; y=z=0; }
        public SomeClass(int val1, int val2){ x=val1; y=val2; z=0; }
        public SomeClass(int val1, int val2, int val3){ x=val1;
y=val2; z=val3; }
}


Introduction to Inheritance:
============================
 - Inheritance allows us to extend a class to get a new class.

 - Class which is being extended is called: super/base/parent class

 - Newly extended class is called: sub/derived/child class

 - Subclass is a more specific-type of superclass, coz it establish
"is-a"
        relationship with the superclass

 - Subclass inherits ALL fields and methods from superclass, and on
top of that
        it can have its own additional fields and methods

- If the superclass author declares the fields as private, then the
subclass
        need to use getter/setter of superclass to use those inherited
fields.
        Also, subclass parameterized constructor needs to indicate to
compiler
        to fire superclass parameterized constructor, where some of
the
        parameters received by subclass-constructor will be specified
as
        parameters of superclass-constructor

- If superclass author anticipates that this class may need to be
extended in
        future, then the fields should be declared as protected
(instead of
        private) to make it convenient for subclass author

======================================================================
==========
Parameterized constructor in subclass (inheritance)
---------------------------------------------------
//see netbeans project for full code
public Student(int id, float cgpa, String name, String dob, String
gender) {
        this.id = id;
        this.cgpa = cgpa;
        this.name = name;
        this.dob = dob;
        this.gender = gender;
}

After break:

Method overriding:
==================
- If subclass is not happy with the definition of an inherited method
of
        superclass, then the subclass author can redefine the method
as its own
        method. In that case, there will method-overloading violation
(because
        both inherited and own version will have SAME parameter list).
Therefore
        the compiler should deactivate/nullify inherited definition so
that there
        will be no method-overloading voilation.
        This is called method OVERRIDING.

Case-1: Superclass method/s is/are not overridden
-------------------------------------------------
  public class Vehicle{
        protected fields: brand, model, cc, price
        public void show(){
                //print the fields
        }
  }


  public class Car extends Vehicle{
        additional private fields: regNo, noOfSeat,..
        //show method is inherited from Vehicle class
  }

  public class MainClass{
    p s v main(...){
        Car myCar = new Car();
        myCar.setInfo();
        myCar.show();   //Problem: NOt ALL fields are printed, if not
Overridden
    }
  }


Case-2: Superclass method/s is/are overridden in subclass
---------------------------------------------------------
  public class Vehicle{
        protected fields: brand, model, cc, price
        public void show(){
                //print the fields
        }
  }
  public class Car extends Vehicle{
        additional private fields: regNo, noOfSeat,..
        //inherited show method is deactivated
        //@Override
        public void show(){
                //code to print ALL fields (inherited + additional)
        }
  }
  public class MainClass{
    p s v main(...){
        Car myCar = new Car();
        myCar.setInfo();
        myCar.show();   //Solved:  ALL fields are printed, Overridden
    }
  }
```

```
c++:
class A{
        private:
        int x; string str; float f;
        public:
        void m1(){...}
        void m1(){...}
};

Java:
public class A{
        private int x,y; private string str; private float f;
        public void m1(){...}
        public void m1(){...}
};
```

Object class: The ADAM of Java species

```
User: Faculty
  Goal: View Report [incorrect, not specific]
  Goal: View Report on withdrawal rate of a course for a given
timeframe [specific]
        input: select a course taught by the faculty
        input: starting semester & ending semester
        output: barchart, each bar will show withdrawal % of the
selected course
                no of bar = no of semesters within that timeframe
```