

## Object is real world entity



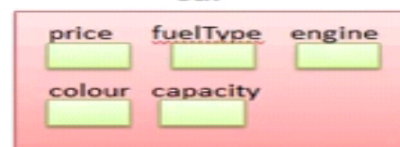
### Properties

price  
fuel type  
engine  
colour  
capacity

### Methods

setPrice()  
setFuelType()  
setEngine()  
setColour()  
setCapacity()  
getPrice()  
getFuelType()  
getEngine()  
getColour()  
getCapacity()

### car



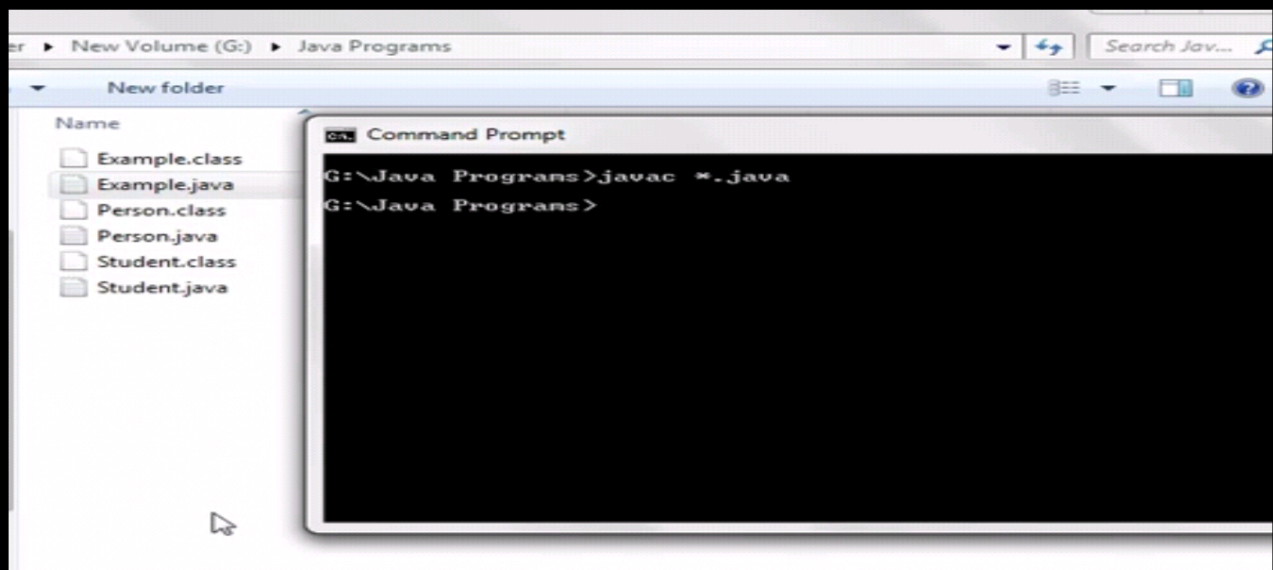
## Syntax

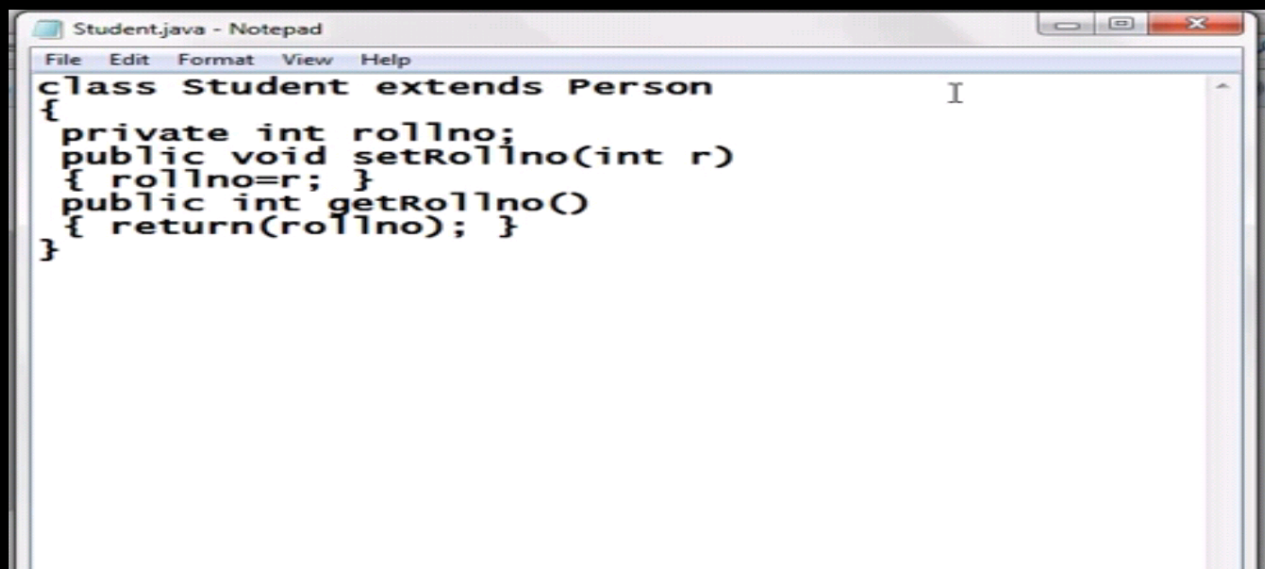
---

```
class SubClass extends SuperClass  
{  
  
}
```

- ❑ **extends** is a keyword
- ❑ Base class means Super Class
- ❑ Derived Class means Sub Class



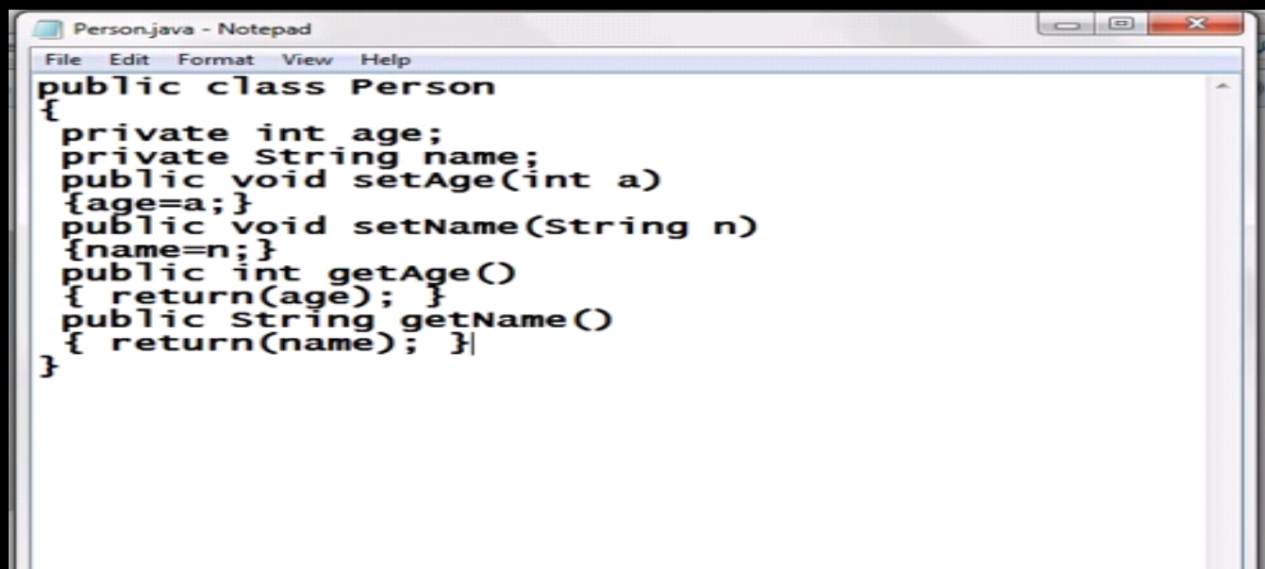




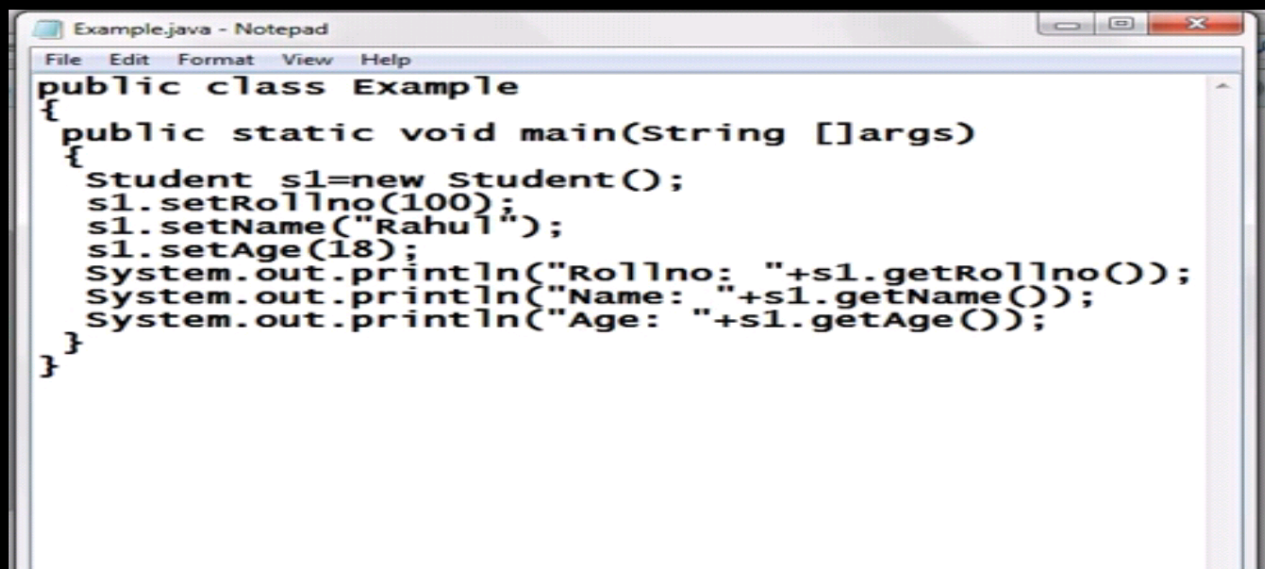
The image shows a screenshot of a Notepad window titled "Student.java - Notepad". The window has a menu bar with "File", "Edit", "Format", "View", and "Help". The code inside the window is as follows:

```
class Student extends Person
{
    private int rollno;
    public void setRollno(int r)
    { rollno=r; }
    public int getRollno()
    { return(rollno); }
}
```

The code defines a class named "Student" that extends the "Person" class. It contains a private integer variable "rollno" and two public methods: "setRollno(int r)" which sets the roll number, and "getRollno()" which returns the roll number. The code is formatted with indentation for the methods and their bodies.

A screenshot of a Notepad window titled "Person.java - Notepad". The window has a menu bar with "File", "Edit", "Format", "View", and "Help". The text inside the window is a Java class definition for "Person". The code is as follows:

```
public class Person
{
    private int age;
    private String name;
    public void setAge(int a)
    {age=a;}
    public void setName(String n)
    {name=n;}
    public int getAge()
    { return(age); }
    public String getName()
    { return(name); }
}
```



The image shows a screenshot of a Notepad window titled "Example.java - Notepad". The window contains the following Java code:

```
public class Example
{
    public static void main(String []args)
    {
        Student s1=new Student();
        s1.setRollno(100);
        s1.setName("Rahul");
        s1.setAge(18);
        System.out.println("Rollno: "+s1.getRollno());
        System.out.println("Name: "+s1.getName());
        System.out.println("Age: "+s1.getAge());
    }
}
```

The code defines a public class named `Example`. Inside the class, there is a `main` method that takes an array of strings as an argument. The `main` method creates a new `Student` object named `s1` and sets its `rollno` to 100, `name` to "Rahul", and `age` to 18. It then prints the roll number, name, and age of the `s1` object to the console.

## Remember

---

- ❑ In the Java programming language, each class is allowed to have one direct superclass, and each superclass has the potential for an unlimited number of *subclasses*
- ❑ Private members of the superclass are not accessible by the subclass and can only be indirectly accessed.
- ❑ Members that have default accessibility in the superclass are also not accessible by subclasses in other packages



## Java Supports

---

- ☐ Single Inheritance
- ☐ Multilevel Inheritance
- ☐ Hierarchical Inheritance





## Java Supports

- ☐ Single Inheritance
- ☐ Multilevel Inheritance
- ☐ Hierarchical Inheritance

