

Forum:[Hack Forums](#)

Thread: [\[TUT\] - Java 3 \(Heredity\)](#).

terkitos - May 14, 2009, 10:53 am

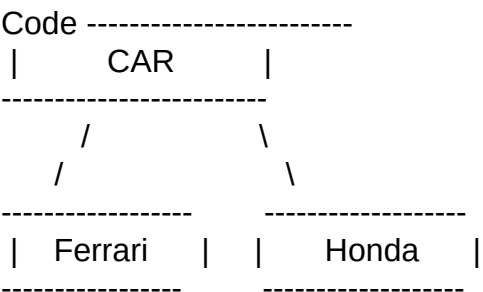
Hi! I'm here again to explain you more about java and it's programming, in this part of the tutorial I'm gonna explain some about heredity and how it works. This part it's easy when you know what are the objects and classes, I explained it in my second tutorial, Java 2 (Classes, objects and methods) .

The heredity it's a good part for not to do COPY & PASTE of a bunch of code, so this will help you not to write a lot and do the program easy to understand. This is one of the most important parts and it's the base of the objects programming, so it's useful because of the heredity and polymorphism that I'm going to explain in the next tutorial, that maybe I'm going to do it today, so you will be able to learn a lot today!

LET'S START!!!!!!!!!!!!!!

The heredity is useful when you have to make a program about something that have some kind of that something, this is hard but it's easy to understand when you learn an example, here is:

A program like a shop car, there is a Class named Car, (Yes!! Like I wrote in the second tutorial :D) and a kind of car are the Ferrari, that are more powerful than Honda for example, so, there is a DAD class and two or more SON classes :D see:



(So It's like a family, yes :D)

Now, the methods that are written in the class CAR, the "æson" classes are going to know it's methods so it's not needed to write it AGAIN!!! (woooooow it's for lazy people!! Yeah, for me :D) then, MAKE ME AN EXAMPLE, yes! I was going to do it now!

```
Code class Car {
    protected int tyres, doors, seats; //initializing variables to set a value later. and protected for not
    to be changed out this method
    Car (int tyres, int doors, int seats) {
        this.tyres = tyres;
        this.doors = doors;
        this.seats = seats;
    }
    public int getTyres() { return tyres; } //get the number of tyres, doors or seats
    public int getDoors() { return doors; }
    public int getSeats() { return seats; }
```

```

    public int setTyres(int tyres) { this.tyres = tyres; } //change the number of tyres
}

```

```

class Ferrari extends Car {
    int power;
    Ferrari(int power) {
        this.power = power;
    }
    public getPower() { return power; }
}
class Honda extends Car {
    int space;
    Honda (int space) {
        this.space = space;
    }
    public int getSpace() { return space; }
}

```

Easy, no?

BUT HOW CAN I USE IT?!?!?! Shhh, calm, I'm going to do a main class to show people how to use it! heheheh

```

Code class Car {
    protected int tyres, doors, seats; //initializing variables to set a value later. and protected for not
    to be changed out this method
    Car (int tyres, int doors, int seats) {
        this.tyres = tyres;
        this.doors = doors;
        this.seats = seats;
    }
    public int getTyres() { return tyres; } //get the number of tyres, doors or seats
    public int getDoors() { return doors; }
    public int getSeats() { return seats; }
    public int setTyres(int tyres) { this.tyres = tyres; } //change the number of tyres
}

```

```

class Ferrari extends Car {
    int power;
    Ferrari(int tyres, int doors, int seats, int power) {
        super(tyres, doors, seats); //This is for calling the builder of the SUPERCLASS (dad class :D)
        this.power = power;
    }
    public getPower() { return power; }
}
class Honda extends Car {
    int space;
    Honda (int tyres, int doors, int seats, int space) {
        super(tyres,doors,seats);
        this.space = space;
    }
}

```

```

    public int getSpace() { return space; }
}
class Main {
    public static void main(String[] args) {
        Ferrari testarrosa = new Ferrari(4,5,2,200); //tyres,doors,seats,power
        Honda honda = new Honda(4,4,5,100); //tyres,doors,seats,space
        System.out.println("Seats: " + testarrosa.getSeats());
        System.out.println("Seats: " + honda.getSeats());
        //And now lets print the other "option"?
        System.out.println("Power: " + testarrosa.getPower());
        System.out.println("Space: " + honda.getSpace());
    }
}

```

This is the end with a full example explaining and now let's make your own!!

See you in the next tutorial!! about polymorphism!!! :D

+rep is welcome :D

terkitos - May 14, 2009, 1:46 pm

This was my last tut. No one reads them xD

Code King - May 14, 2009, 1:52 pm

asking for +rep does the opposite

terkitos - May 14, 2009, 1:55 pm

If I don't get rep but people reads and like it, i'll continue but if not...

and I'm not asking, I only say it's welcome :D

klez - May 15, 2009, 4:20 pm

it's inheritance, not heredity. parent, not father. child, not son XD

btw, explaining inheritance to someone who knows almost nothing about OOP can be quite difficult. anyway, since it's a very important part of oop (and, as you said, a good way to avoid duplicated code) it's good to have tuts on it (java, c++ or smalltalk, it doesn't matter, the point is the concept itself, not the implementation).

one thing I'd like to add: c++ allows multiple inheritance (meaning a class can inherit from one or more classes), while java doesn't. and this, imho, isn't much of a loss ;-)

terkitos - May 15, 2009, 4:23 pm

I'd like to see you speaking in english, how many mistakes you'd have :)

and thank you for telling me my mistakes ^^