

Forum: [Hack Forums](#)

Thread: [Methods / Class](#)

CompSec - November 27, 2017, 3:18 am

Can anyone help me understand Methods and Classes in the most simple way?

```
public class Car
{
    private double mileage;
    public String manufacturer = "";
    private double petrol;

    public Car(String manufacturer)
    {
        this.manufacturer = manufacturer;
        mileage = 0;
        petrol = 0;
    }
}
```

What is the purpose in the second part of this? Why not just declare your String in the main method?

Aussie - November 27, 2017, 5:38 am

What you have there is a class called Car and a CONSTRUCTOR

Traditionally speaking, it is good software design to not have main class setting values of other types.

Code #shit pseudocode

```
main-class {
    main-method {
        functionY();
        Car car = new Car("mercedes");
    }
}
```

the reason why we have constructors is so we can make classes and effectively give them attributes (e.g. manufacturer, brand... etc), that pertain to the OBJECT type you are modelling (i.e. cars). This might seem foreign or counterintuitive to you initially, but when you develop a game or a bigger project it'll be consolidated, THIS is very important.

Imagine instantiating an enemy like... in mario e.g.

Enemy enemy = new Enemy(Type.GOOMBA, "bobby"); //declaring an enemy type goomba, his name bobby (irrelevant), rather than doing

```
enemy.type = TYPE.GOOMBA;
enemy.name = "bobby"
```

which is inefficient, also remember getters and setters for better encapsulation

CompSec - November 27, 2017, 9:39 am

(Nov 27, 2017 06:38 AM)n31ln3t Wrote: [\[CITE\]](#) What you have there is a class called Car and a CONSTRUCTOR

Traditionally speaking, it is good software design to not have main class setting values of other types.

Code #shit pseudocode

```
main-class {
  main-method {
    functionY();
    Car car = new Car("mercedes");
  }
}
```

the reason why we have constructors is so we can make classes and effectively give them attributes (e.g. manufacturer, brand... etc), that pertain to the OBJECT type you are modelling (i.e. cars). This might seem foreign or counterintuitive to you initially, but when you develop a game or a bigger project it'll be consolidated, THIS is very important.

Imagine instantiating an enemy like... in mario e.g.

Enemy enemy = new Enemy(Type.GOOMBA, "bobby"); //declaring an enemy type goomba, his name bobby (irrelevant), rather than doing

```
enemy.type = TYPE.GOOMBA;
enemy.name = "bobby"
```

which is inefficient, also remember getters and setters for better encapsulationAhh that definitely helps. Thank you ?

Ciphered - November 27, 2017, 9:25 pm

PHP Code

```
<?php
class Car {
```

```
}
```

This is your class, a class is just an object, inside we can put properties (variables) or methods like:

PHP Code

```
<?php
public class Car {
```

```
    public String car_type;
    public int speed;
```

```

public Car(String car_type, int speed) {
    // This is your constructor, this will be called when you initialize this object (see below)
    // this. keyword is used to differentiate between our speed value above, and the speed value with
in our Car(String car_type, int<b>speed</b>)
    this.car_type = car_type;
    this.speed = speed;
}

public void printCarType()
    // this is a void method, we don't need to return anything.
    System.out.println(car_type);
}

public String getMake() {
    // this method needs to return a string, because it has String instead of void
    return car_type;
}

public int getSpeed() {
    // this method is needs to return an integer, because it has int instead of void
    return this.speed;
}

public void increaseSpeedBy(int mph) {
    // this. means you are referring to this object
    this.speed += mph;
}

public void decreaseSpeedBy(int mph) {
    // this.speed means you are referring to the speed of this Car
    this.speed -= mph;
}
}

```

This is how you would use this class in your public static void main(String[] args) method.

PHP Code

```

<?php
class Program {

public static void main(String[] args) {
    // Initialize object:
    Car car = new Car("BMW", 0);
    // Call void methods
    car.printCarType();
    car.increaseSpeedBy(5);
    car.increaseSpeedBy(10);
    car.decreaseSpeedBy(15);
    // Call int method
    System.out.println("The speed of your car is: " + car.getSpeed());
}

}

```

The point of coding like this is because if we wanted to have multiple cars, it makes it easy to keep track of the different makes and speeds.

PHP Code

```
<?php
public static void main(String[] args) {

    Car bmw = new Car("BMW", 0);
    Car telsa = new Car("Tesla", 0);
    Car subaru = new Car("Subaru", 0);
    telsa.increaseSpeedBy(60);
    subaru.increaseSpeedBy(75);
    bmw.increaseSpeedBy(40);
    // What if we wanted to find the car that has the highest speed?
    Car[] cars = new Car[] { bmw, telsa, subaru };
    Car highestSpeedCar;
    for (int i = 0; i < cars.Length; i++) {
        if (highestSpeedCar == null) {
            highestSpeedCar = cars[i];
        } else {
            if (highestSpeedCar.getSpeed() < cars[i].getSpeed()) {
                highestSpeedCar = cars[i];
            }
        }
    }

    System.out.println("The fastest car is: " + highestSpeedCar.printCarType());
}
```

king mala - November 29, 2017, 8:46 pm

One of the best explanations of classes I ever received was this:

Assume you have a general item like a bike. Using the class Bike, we can set some specifics to create different types of bikes. For example, we may want this bike to have many spokes, or perhaps we want a bike to have a larger seat. This would be an iteration of that case:

```
public class Bicycle {

    private int numSpokes;
    private String bikeName;
    private int numWheels;

    public Bicycle (int spokes, String name, int numWheels) {

        numSpokes = spokes;
        bikeName = name;
        this.numWheels = numWheels;

    }

    //Accessor Methods
    public int getSpokes() {
```

```

        return numSpokes;
    }

    public String getName() {

        return bikeName;
    }

    public int getNumWheels() {

        return numWheels;
    }
}

```

Now below, suppose we have our main class that we want to create instances of this bike class within:

```

public class BikeMaker {

    public static void main(String[] args) {

        Bicycle bike1 = new Bicycle(10, "Tricycle", 3);
        Bicycle bike2 = new Bicycle(3, "Motorbike", 2);

        System.out.println("Bike 1 is a " + bike1.getName() + " with " + bike1.getNumWheels() + "
wheels and " + bike1.getSpokes() + " spokes.");
        System.out.println("Bike 2 is a " + bike2.getName() + " with " + bike2.getNumWheels() + "
wheels and " + bike2.getSpokes() + " spokes.");

    }

}

```

This makes our main program much easier to read and simplifies the main method. You can imagine in a much more complicated program, you'll have 10s or even hundreds of classes working together, and if we were to force those all into one class, we'd have a very long and very complex program to follow should there be any bugs to fix or things to update.

CompSec - December 1, 2017, 7:29 am

(Nov 29, 2017 09:46 PM)king mala Wrote: [\[CITE\]](#) One of the best explanations of classes I ever received was this:

Assume you have a general item like a bike. Using the class Bike, we can set some specifics to create different types of bikes. For example, we may want this bike to have many spokes, or perhaps we want a bike to have a larger seat. This would be an iteration of that case:

```

public class Bicycle {

```

```

private int numSpokes;
private String bikeName;
private int numWheels;

public Bicycle (int spokes, String name, int numWheels) {

    numSpokes = spokes;
    bikeName = name;
    this.numWheels = numWheels;

}

//Accessor Methods
public int getSpokes() {

    return numSpokes;

}

public String getName() {

    return bikeName;

}

public int getNumWheels() {

    return numWheels;

}

}

```

Now below, suppose we have our main class that we want to create instances of this bike class within:

```

public class BikeMaker {

    public static void main(String[] args) {

        Bicycle bike1 = new Bicycle(10, "Tricycle", 3);
        Bicycle bike2 = new Bicycle(3, "Motorbike", 2);

        System.out.println("Bike 1 is a " + bike1.getName() + " with " + bike1.getNumWheels() + "
wheels and " + bike1.getSpokes() + " spokes.");
        System.out.println("Bike 2 is a " + bike2.getName() + " with " + bike2.getNumWheels() + "
wheels and " + bike2.getSpokes() + " spokes.");

    }

}

```

This makes our main program much easier to read and simplifies the main method. You can

imagine in a much more complicated program, you'll have 10s or even hundreds of classes working together, and if we were to force those all into one class, we'd have a very long and very complex program to follow should there be any bugs to fix or things to update.

Ahh wow, that is a great way of simplifying it for me. Thanks a lot ?

matt - December 1, 2017, 7:34 am

(Dec 1, 2017 08:29 AM)CompSec Wrote: [\[CITE\]](#) Ahh wow, that is a great way of simplifying it for me. Thanks a lot ?

That was the absolute worst way of describing it
wat

CompSec - December 1, 2017, 8:07 am

(Dec 1, 2017 08:34 AM)Midnight Blue™ Wrote: [\[CITE\]](#) That was the absolute worst way of describing it

wat

Haha ?

I must be backwards, it actually helped