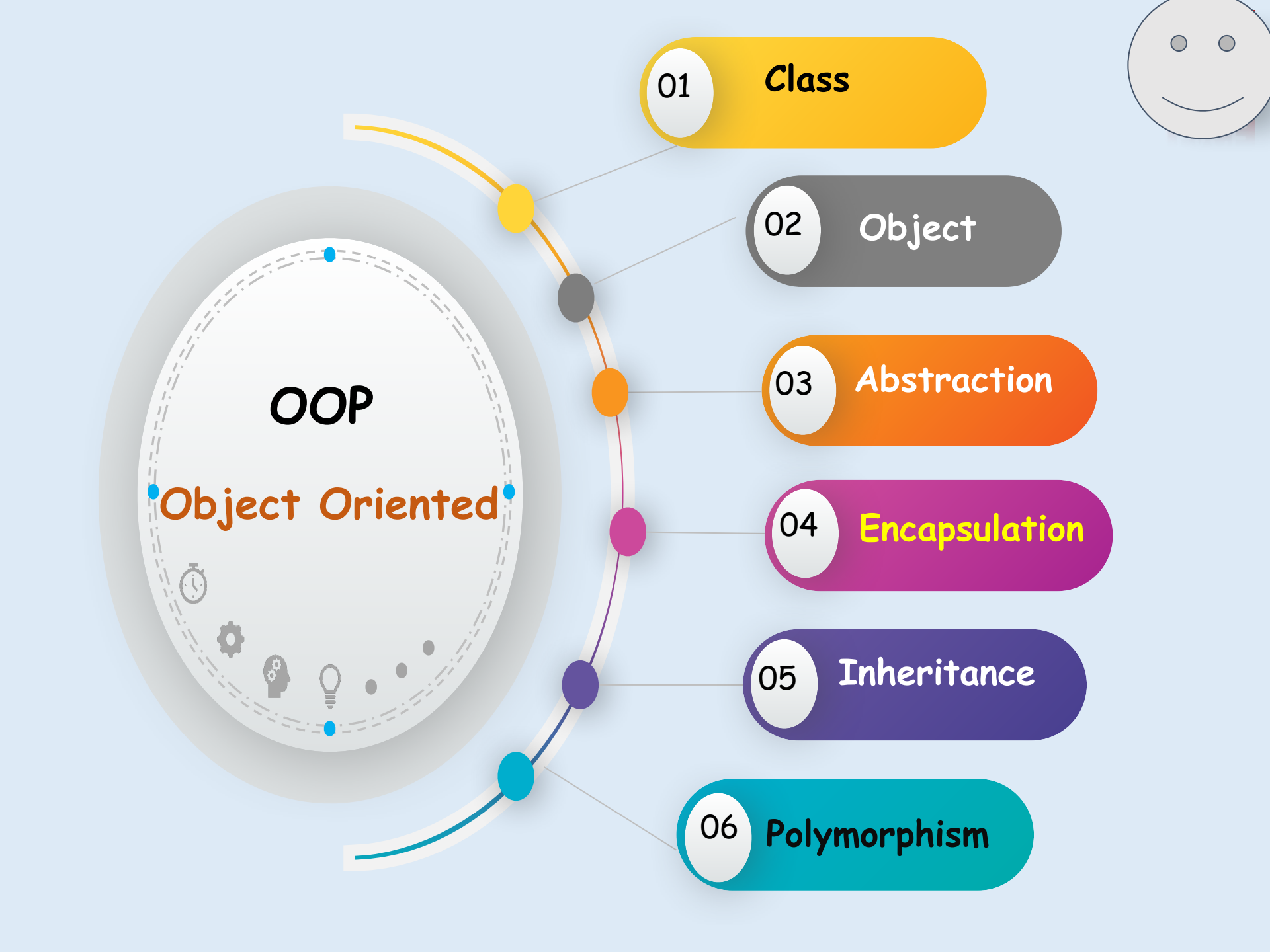
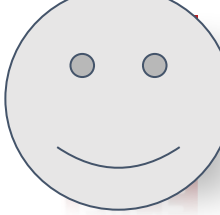
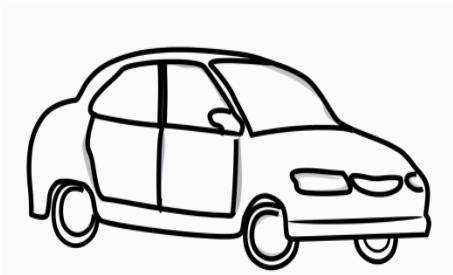
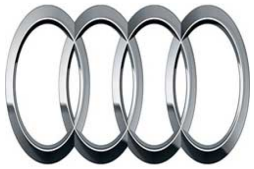

OOP
Object Oriented
01 Class
02 Object
03 Abstraction
04 Encapsulation
05 Inheritance
06 Polymorphism



class **Car**



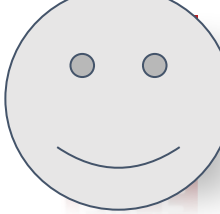
Object1



Object2

Object3

01 Class



CAR

Properties:

- 4_Wheels
- Speed limit
- Mileage

Functions of a car:

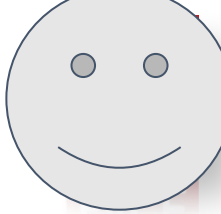
- Increase_Speed()
- Apply_Brakes()

```
class CAR
{
    // Data Members or Properties
    // Member functions
};
```

01 Class

- A class is only a **prototype**.
- **No storage** is assigned when we define a class.

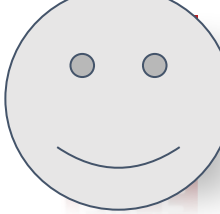
```
class Car
{
    public:
        int mileage;
        int speed_limit;
};
```



01 Class & Object

- To use the data and access functions defined in the class, we need to create **Objects**

```
class Car
{
    public:
        int mileage;
        int speed_limit;
} Audi ;
```



Audi - An object

Audi.mileage = 16

Audi.speed_limit = 250

01 Class & Object

Audi, Ford, Benz - are objects

Audi.mileage = 16

Audi.speed_limit = 250

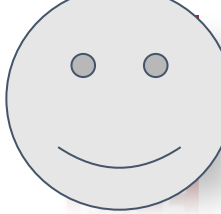
Ford.mileage = 20

Ford.speed_limit = 230

Benz.mileage = 19

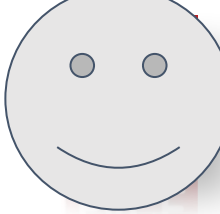
Benz.speed_limit= 240

```
class Car
{
   public:
      int mileage;
      int speed_limit;
} Audi, Ford, Benz ;
```



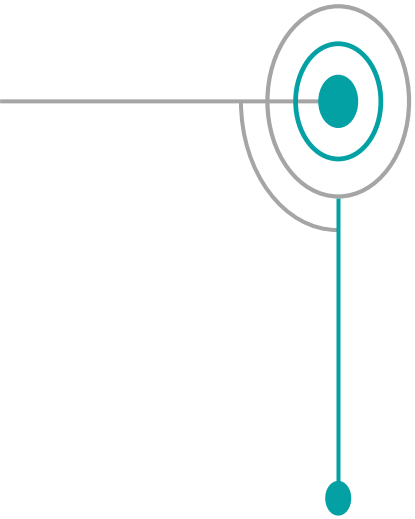
- **Any number of Objects** can be created

OOP



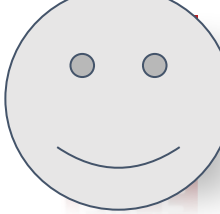
Class

```
class  ClassName
{
        // Access specifiers
        // Data Members
        // Member functions
};
```

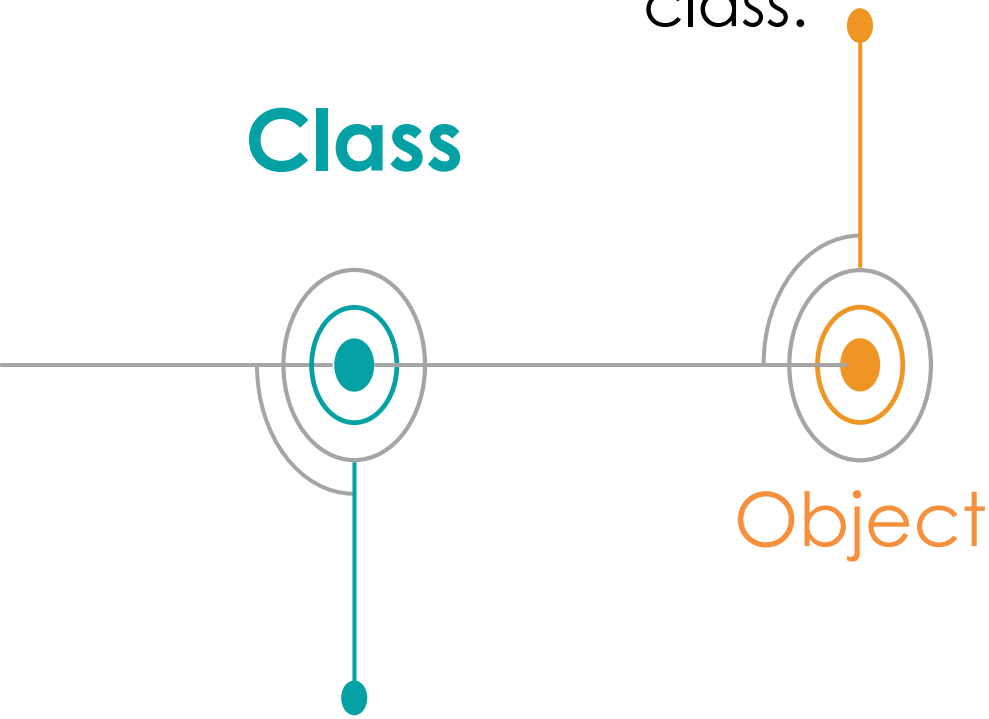


A class is a blueprint for the object.

OOP

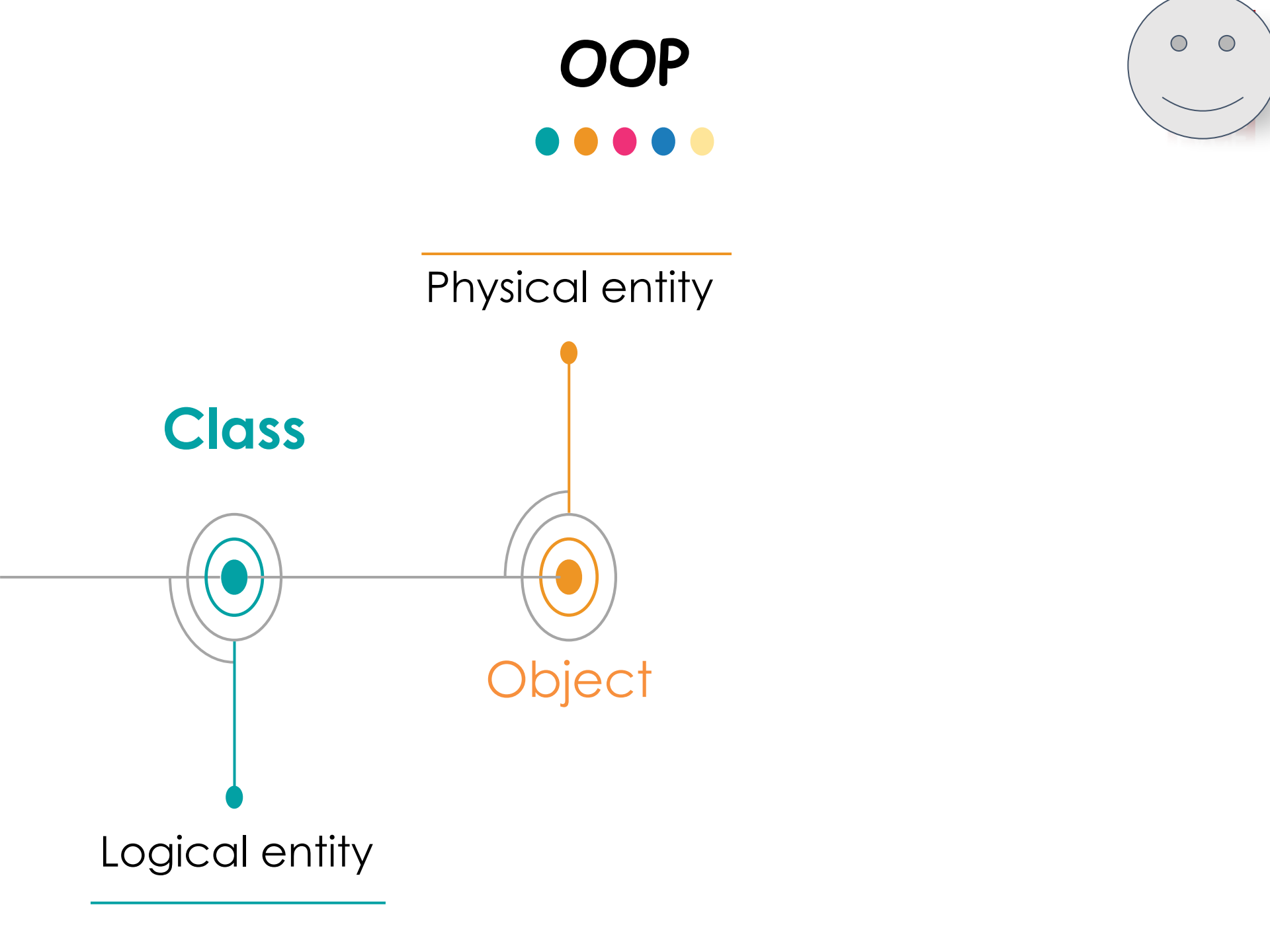


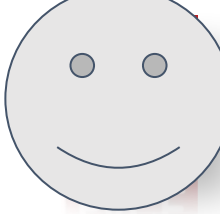
An object is an instance of a class.



A class is a blueprint for the object.

OOP





Class

- Container - collection of variables and functions
- No memory is allocated -during class declaration
- One class definition - only once in the program.

Object

- Object is a instance of class
- Memory is allocated - during object declaration
- For one class multiple objects can be created.

Syntax for creating objects in java:

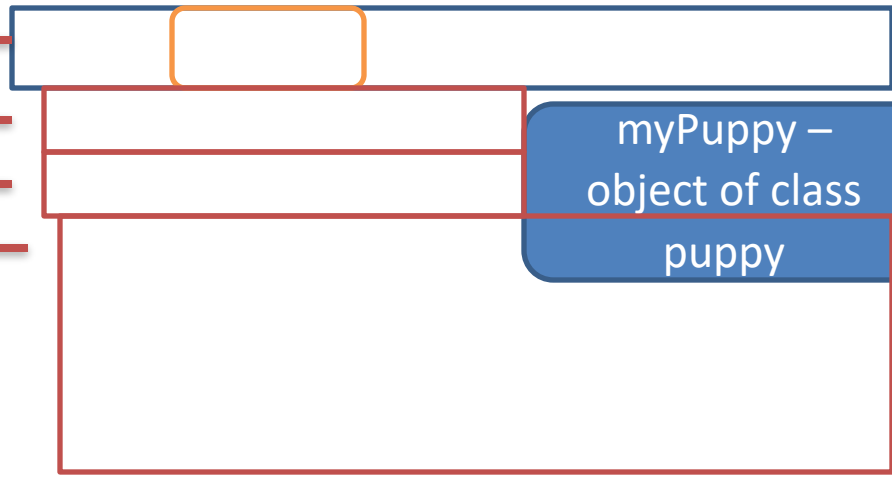
Class_name object_name = new Class_name();

```
public class Puppy
{
   int puppyAg
   public P
   {
      Syst                    );
   }
   public void setAg(          )
   {
      puppyAge = age;
   }
   public int getAge( )
   {
      System.out.println
       ("Puppy's age is :"+ puppyAge );
      return puppyAge;
   }


  public static void main(String []args)
   {
     Puppy myPuppy = new Puppy("tommy");
      myPuppy.setAge( 2 );
     myPuppy.getAge( );
       System.out.println
          ("call using obj :" +
   }
}
```

Class - Puppy

Class name should always start with capital letter



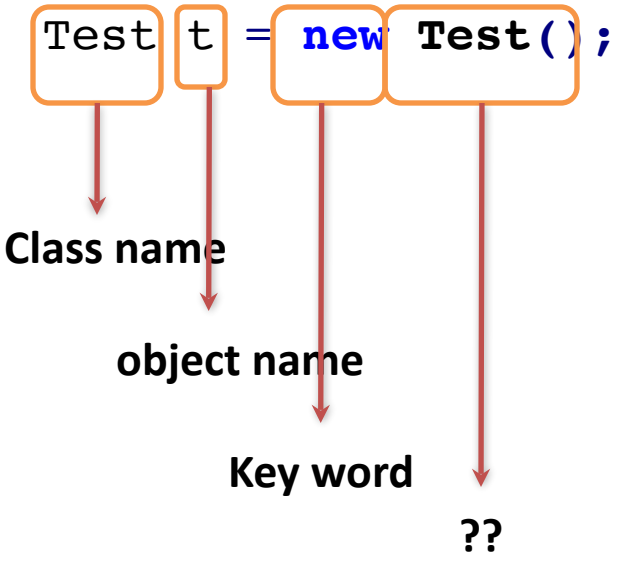
Output:
Name: tommy
Puppy's age is :2
call using obj :2

Constructors in Java

Consider an example!

```
Class Test()
{
    //class members and methods
}
To create object for Test Class,
Test t = new Test();
```

Constructor!!
Constructs memory for the object “t”



Whenever an object is created, constructor is called

Rules to declare a constructor in JAVA

```
public class Test
{
    Test()
    {
        System.out.print("Constructor");
    }
    public static void main(String args[])
    {
        Test m = new Test();
    }
}
```

1. Constructor name and class name should be same.
2. No return type

Applications of Constructor!

01

```
public class Test
{
    void m1()
    {
        System.out.println("m1-method");
    }
    public static void main(String args[])
    {
        Test t = new Test();
        t.m1();
    }
}
```

There is no predefined constructor! The compiler by itself generates a dummy constructor

```
Test
{}
```

Output:
m1 - method

Applications of Constructor!

01

```
public class Test
{
    void m1()
    {
        System.out.println("m1-method");
    }
    Test ()
    {
        System.out.println("0-arg constructor");
    }
    Test (int a)
    {
        System.out.println("1-arg constructor");
    }
    public static void main(String args[])
    {
        Test t = new Test();
        Test t1 = new Test(10);
        t.m1();
                        t1.m1();
    }
}
```

0 – arg constructor defined

1 – arg constructor defined

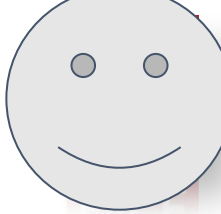
Output:

0-arg constructor

1-arg constructor
m1-method

m1-method

Try!



```
public class Test
{
    void m1()
    {
        System.out.println("m1-method");
    }
      Test (int a)
    {
        System.out.println("1-arg constructor");
    }
    public static void main(String args[])
    {
         Test t = new Test();
         Test t1 = new Test(10);
         t.m1();
                          t1.m1();
    }
}
```

1 – arg constructor defined

Output:

Compilation error

Advantages of Constructor!

01

Test t = new Test();

1. The logics inside the constructor are executed only during object creation.

```
public class Test
{
    int no;    //0
    String name;    //Null
    float sal; //0.0
    void disp()
    {
        System.out.println("Emp id :" +no);
        System.out.println("Emp name :" +name);
    }
    public static void main(String args[])
    {
     Test t = new Test();
     t.disp();
    }
}
```

We can initialize the constructors!!

Output:

Emp id :0

Emp name :null

```
public class Test
{
    int no;
    String name;
    float sal;
    Test()
    {
        no=1;
        name = "john";
        sal = 10345.56f;
    }
    void disp()
    {
        System.out.println("Emp id :" +no);
        System.out.println("Emp name :" +name);
        System.out.println("Emp sal :" +sal);
    }
    public static void main(String args[])
    {
     Test t = new Test();
     t.disp();
    }
}
```

Output:

Emp id :1
Emp name :john
Emp sal :10345.56

Advantages of Constructor!

Test t = new Test();

1. The logics inside the constructor are executed only during object creation.

```
public class Test
{
    int no;      //0
    String name;     //Null
    float sal;  //0.0
    void disp()
    {
        System.out.println("Emp id :" +no);
        System.out.println("Emp name :" +name);
    }
    public static void main(String args[])
    {
     Test t = new Test();
     Test t1 = new Test();
     t.disp();
     t1.disp();
    }
}
```

Default values

Therefore, pass parameters!!

```
Emp id :0
Emp name :null
Emp id :0
Emp name :null
```

```
public class Test
{
    int no;
    float sal;
    Test(int a,float b )
    {
        no=a;
        sal = b;
    }
    void disp()
    {
        System.out.println("Emp id :" +no);
        System.out.println("Emp sal :" +sal);
    }
    public static void main(String args[])
    {
     Test t = new Test(11, 10675.56f);
     Test t1 = new Test(12,13345.67f);
     t.disp();
     t1.disp();
    }
}
```

Output:

Emp id :11
Emp sal :10675.56
Emp id :12
Emp sal :13345.67


Thank
you!!