《 Java: 상속과 생성자 호출순서 》

출처: www.youtube.com/@weekendcode // https://inf.run/zSrvA

```
// 부모 클래스
class Animal {
  String name;
  int age;

  void eat() {
    System.out.println(name + " is eating.");
  }

  void sleep() {
    System.out.println(name + " is sleeping.");
  }
}
// 자식 클래스
class Dog extendsAnimal {
  void bark() {
    System.out.println(name + " is barking.");
  }
}

public class Main {
  public static void main(String[] args) {
    // Dog 객체 생성
    Dog dog = new Dog();
    dog.name= "Buddy";
    dog.age= 3;

    // 상속받은 메서드 호출
    dog.eat(); // Buddy is eating.
    dog.sleep(); // Buddy is sleeping.

    // 자식 클래스의 메서드 호출
    dog.bark(); // Buddy is barking.
  }
}
```

```
// 부모클래스
class Car{
    String model;
    int year;

    void start() {
        System.out.println(model+ " isstarting.");
    }

    void stop() {
        System.out.println(model+ " isstopping.");
    }

    void refuel() {
        System.out.println(model+ " isrefueling.");
    }
}

// 자식클래스
class ElectricCarextends Car {
    void refuel() {
        System.out.println(model + " is charging.");
    }

    void checkBattery() {
        System.out.println(model + " battery level is checking.");
    }
}
```

```
// 부모클래스
class Car {
    String model;
    int year;

    void start() {
        System.out.println(model+ " isstarting.");
    }

    void stop() {
        System.out.println(model+ " isstopping.");
    }

    void refuel() {
        System.out.println(model+ " isrefueling.");
    }
}

// 자식클래스
class ElectricCarextends Car {
  void refuel() {
        System.out.println(model + " is charging.");
    }
    void checkBattery() {
        System.out.println(model + " battery level is checking.");
    }
}

public class Main {
  public static void main(String[] args) {
    Car car = new Car();  //부모 객체 생성
```

```
        car.model= "Generic Car";
        car.year= 2020;

        // Create an ElectricCarobject
        ElectricCar eCar= new ElectricCar();  //자식 객체 생성
        eCar.model= "Tesla Model 3";
        eCar.year= 2021;

        // Call methods of Car class
        car.start(); // Generic Car is starting.
        car.refuel(); // Generic Car is refueling.
        car.stop(); // Generic Car is stopping.

        // Call methods of ElectricCarclass
        eCar.start(); // Tesla Model 3 is starting.
        eCar.refuel(); // Tesla Model 3 is charging.
        eCar.stop(); // Tesla Model 3 is stopping.
        eCar.checkBattery(); // Tesla Model 3 battery level is checking.
    }
}
```

생성자가 있으면 어떻게 될까?

```
class Car {
  String model;
  int year;
  Car() {    System.out.println("Car constructor"); }

  void start() {  System.out.println(model + " start"); }

  void stop() {    System.out.println(model + " stop"); }
}
class ElectricCarextends Car {
  ElectricCar() {  System.out.println("ElectricCar constructor"); }

  void stop() {  System.out.println(model + " stop and power down");  }

  void checkBattery() {    System.out.println(model + " check battery");  }
}

public class Main {
  public static void main(String[] args) {
    ElectricCar eCar= new ElectricCar();
    eCar.model= "Tesla Model 3";
    eCar.year= 2021;

    eCar.start();
    eCar.stop();
    eCar.checkBattery();
  }
}
```

출력값:

```
Car constructor
ElectricCarconstructor
Tesla Model 3 start
Tesla Model 3 stop and power down
Tesla Model 3 check battery
```

부모의 생성자가 여러개 + 명시적 호출

```
class Car {
    String model;
    int year;

   Car() { System.out.println("Car()"); }

    Car(String model, int year) {
        this.model= model;
        this.year= year;
        System.out.println("Car(" + model + ", " + year + ")");
    }
}

class ElectricCarextends Car {
    int batteryCapacity;
    ElectricCar() {  System.out.println("ECar()");  }

   ElectricCar(String model, int year, int batteryCapacity) {
        super(model, year);
        this.batteryCapacity= batteryCapacity;
        System.out.println("ECar(" + batteryCapacity+ ")");
    }
}

public class Main {
    public static void main(String[] args) {
     ElectricCar tesla = new ElectricCar("Tesla", 2021, 75);
    }
}
```

출력값:

Car(Tesla, 2021)

ECar(75)

만약 super가 없다면?

```
class Car {
    String model;
    int year;

    Car() {  System.out.println("Car()");  }

    Car(String model, int year) {
        this.model= model;
        this.year= year;
        System.out.println("Car(" + model + ", " + year + ")");
     }
}

class ElectricCarextends Car {
    int batteryCapacity;
    ElectricCar() {  System.out.println("ECar()");  }
    ElectricCar(String model, int year, int batteryCapacity) {
        // super(model, year);
        this.batteryCapacity= batteryCapacity;
        System.out.println("ECar(" + batteryCapacity+ ")");
     }
}

public class Main {
     public static void main(String[] args) {
      ElectricCar tesla = new ElectricCar("Tesla", 2021, 75);
     }
}
```

출력값:

Car()

ECar(75)

부모 위에 또 부모가 있을 때 생성자

```
class Vehicle {
    Vehicle() {
        System.out.println("Veh()");
    }
}

class Car extends Vehicle {
    Car() {
        System.out.println("Car()");
    }
}

class ElectricCarextends Car {
    ElectricCar() {
        System.out.println("ECar()");
    }
}

public class Main {
    public static void main(String[] args) {
        ElectricCar tesla = new ElectricCar();
    }
}
```

출력값:

```
Veh()
Car()
ECar()
```