■ 생성자

- 인스턴스가 생성될 때마다 호출되는 '**인스턴스 초기화 메소드**'
- 인스턴스 변수의 초기화 또는 인스턴스 생성시 수행할 작업에 사용
- 몇가지 조건을 제외하고는 메소드와 같다.
- **모든 클래스에는 반드시 하나 이상의 생성자가 있어야 한다.**
- new 연산자에 의해 호출되어 객체의 초기화 담당

```
new 클래스();
```

```
Card c = new Card();
```

1. 연산자 new에 의해서 메모리(heap)에 Card클래스의 인스턴스가 생성된다.
2. 생성자 Card()가 호출되어 수행된다.
3. 연산자 new의 결과로, 생성된 Card인스턴스의 주소가 반환되어 참조변수 c에 저장된다.

■ 생성자의 조건

- 생성자의 이름은 클래스의 이름과 같아야 한다.
- 생성자는 리턴값이 없다. (하지만 void를 쓰지 않는다.)

```
클래스이름(타입 변수명, 타입 변수명, ... ) {
        // 인스턴스 생성시 수행될 코드
      // 주로 인스턴스 변수의 초기화 코드를 적는다.
}
```

```
class Card {
      ...
          Card() {  // 매개변수가 없는 생성자.
              // 인스턴스 초기화 작업
          }

          Card(String kind, int number) { // 매개변수가 있는 생성자
              // 인스턴스 초기화 작업
          }

}
```

■ 기본 생성자

● 매개변수가 없는 생성자

클래스이름() { }
Card() { } // 컴파일러에 의해 추가된 Card클래스의 기본 생성자. 내용이 없다.

● 클래스에 생성자가 하나도 없으면 컴파일러가 기본 생성자를 추가한다.

(생성자가 하나라도 있으면 컴파일러는 기본 생성자를 추가하지 않는다.)

소스 파일(Car.java)

```
public class Car {


}
```

→

바이트 코드 파일(Car.class)

```
public class Car {
   public Car() { }   //자동 추가
}          기본 생성자
```

```
Car myCar = new Car();
                 기본 생성자
```

■ 기본 생성자 사용

```
public class Car1 {
    String color;
    int door;

    public Car1() {
        color = "빨강";
        door = 2;
    }
}
```

```
public class Car1Main {
    public static void main(String[] args) {
        Car1 car = new Car1();
        System.out.println(car.color);
        System.out.println(car.door);
    }
}
```

■ 매개변수가 있는 생성자

```
class Car {
      String color;          // 색상
      String gearType;       // 변속기 종류 - auto(자동), manual(수동)
      int door;              // 문의 개수

      Car() {}  // 생성자
      Car(String c, String g, int d) { // 생성자
            color = c;
            gearType = g;
            door = d;
      }
}
```

```
Car c = new Car();
c.color = "white";
c.gearType = "auto";
c.door = 4;
```

```
Car c = new Car("white","auto",4);
```

■ 매개변수가 있는 생성자 사용

```
public class Car2 {
    String color;
    int door;

    public Car2(String c, int d) {
        color = c;
        door = d;
    }
}
```

```
public class Car2Main {
    public static void main(String[] args) {
        Car2 car = new Car2("빨강", 2);
        System.out.println(car.color);
        System.out.println(car.door);
    }
}
```

■ 생성자를 여러가지 형태로 만드는 이유

- 객체 생성할 때 외부 값으로 객체를 초기화할 필요
- 외부 값이 어떤 타입으로 몇 개가 제공될 지 모름 - 생성자도 다양화

■ 생성자 오버로딩(Overloading)

- 매개변수의 타입, 개수, 순서가 다른 생성자 여러 개 선언

```
public class 클래스 {
    클래스 ( [타입 매개변수, ...] ) {
        ...
    }
    클래스 ( [타입 매개변수, ...] ) {
        ...
    }
}
```

[생성자의 오버로딩]
매개변수의 타입, 개수, 순서가 다르게 선언

```
Car(String model, String color) { ... }
Car(String color, String model) { ... } //오버로딩이 아님
```

```
public class Car {
    Car() { ... }
    Car(String model) { ... }
    Car(String model, String color) { ... }
    Car(String model, String color, int maxSpeed) { ... }
}
```

```
Car car1 = new Car();
Car car2 = new Car("그랜저");
Car car3 = new Car("그랜저", "흰색");
Car car4 = new Car("그랜저", "흰색", 300);
```

■ 여러가지 형태의 생성자 사용

```
public class Car3 {
    String color;
    int door;

    public Car3() {
        color = "파랑";
        door = 4;
    }
    public Car3(String c) {
        color = c;
        door = 4;
    }
    public Car3(String c, int d) {
        color = c;
        door = d;
    }
}
```

■ 여러가지 형태의 생성자 사용

```
public class Car3Main {
    public static void main(String[] args) {
        Car3 car = new Car3();
        System.out.println(car.color);
        System.out.println(car.door);

        Car3 car2 = new Car3("빨강");
        System.out.println(car2.color);
        System.out.println(car2.door);

        Car3 car3 = new Car3("노랑", 2);
        System.out.println(car3.color);
        System.out.println(car3.door);
    }
}
```

■ 생성자에서 다른 생성자 호출하기 – this()

- this() – 생성자, 같은 클래스의 다른 생성자를 호출할 때 사용
  - 초기화 내용을 한 생성자에 몰아 작성
  - 다른 생성자는 초기화 내용을 작성한 생성자를 this(...)로 호출
- 다른 생성자 호출은 생성자의 첫 문장에서만 가능

```
1  class Car {
2      String color;
3      String gearType;
4      int door;
5
6      Car() {
7          color = "white";
8          gearType = "auto";
9          door = 4;
10     }
11
12     Car(String c, String g, int d) {
13         color = c;
14         gearType = g;
15         door = d;
16     }
17
18 }
19
```

* 코드의 재사용성을 높인 코드

```
Car() {
    //Card("white","auto",4);
    this("white","auto",4);
}
```

■ 생성자에서 다른 생성자 호출하기 (1 / 2)

```
public class Car4 {
    String color;
    int door;

    public Car4() {
        this("파랑", 4); // 자신의 생성자 호출
    }
    public Car4(String c) {
        this(c, 4);
    }
    public Car4(String c, int d) {
        color = c;
        door = d;
    }
}
```

■ 생성자에서 다른 생성자 호출하기 (2 / 2)

```
public class Car4Main {
    public static void main(String[] args) {
        Car4 car = new Car4();
        System.out.println(car.color);
        System.out.println(car.door);

        Car4 car2 = new Car4("빨강");
        System.out.println(car2.color);
        System.out.println(car2.door);

        Car4 car3 = new Car4("노랑", 2);
        System.out.println(car3.color);
        System.out.println(car3.door);
    }
}
```

■ 참조변수 this

- 인스턴스 자신을 가리키는 참조변수
- 객체 내부에서 인스턴스 멤버임을 명확히 하기 위해 사용
- 매개변수와 필드명이 동일할 때 인스턴스 필드임을 명확히 하기 위해 사용

```
1  class Car {
2      String color;
3      String gearType;
4      int door;
5
6      Car() {
7          //Card("white","auto",4);
8          this("white","auto",4);
9      }
10
11     Car(String c, String g, int d){
12         color = c;
13         gearType = g;
14         door = d;
15     }
16 }
17
```

```
Car(String color, String gearType, int door){
    this.color = color;
    this.gearType = gearType;
    this.door = door;
}
```

■ 참조변수 this (1 / 2)

```
public class Car5 {
    String color = null;
    int door = 0;

    public Car5(String color, int door) {
        this.color = color;
        this.door = door;
    }

    public Car5(String color) {
        this(color, 4);
    }
    public Car5() { }
    public void setColor(String color) {
        this.color = color;
    }

    public String getColor() {
        return color;
    }

    public void setDoor(int door) {
        this.door = door;
    }

    public int getDoor() {
        return door;
    }
}
```

■ 참조변수 this (2 / 2)

```
public class Car5Main {
    public static void main(String[] args) {
        Car5 car = new Car5();
        System.out.println(car.getColor());
        System.out.println(car.getDoor());

        Car5 car2 = new Car5("blue", 4);
        System.out.println(car2.getColor());
        System.out.println(car2.getDoor());
    }
}
```