7강

JAVA

객체 지향 프로그램의 이해

객체의 이해

❖ 객체란
- 하나 이상의 성질을 가지고 있으며 어떠한 행동을 할 수 있는 것
- 사람, 얼굴, 컴퓨터 등

❖ 얼굴의 성질
- 눈, 코, 입, 귀, 눈썹

❖ 얼굴의 행동
- 웃는다 : 눈의 끝과 입의 끝이 만나기 위해 노력하는 것
- 화낸다 : 눈 끝이 하늘을 향하며 입을 최대한 크게 벌려 상대를 위협하는 것

❖ 객체지향 프로그램의 구조
- 객체 : class
- 성질 : field
- 행동 : method

객체 지향 프로그램의 특징

- **캡슐화**
  - 객체 내부에서 동작되는 여러 내용들을 숨기는 것
  - 외부에 노출된 내용만으로도 객체를 활용할 수 있음
  - Class 내의 field와 method를 모르더라도 공개된 method 만으로 class를 활용하는 것
- **상속**
  - 상위의 객체가 하위 객체에게 특징을 전달하는 것.
  - 기존에 개발한 내용을 추가로 개발할 필요 없음.
  - 상속 후 내용을 추가하여 사용할 수도 있음.
- **다형성**
  - 모양을 다양하게 갖는 것.
  - 같은 이름으로 여러 형식을 구현할 경우 사용.
  - 달린다(사람, 자동차, 오토바이)

클래스 이해

class 만들기

❖ 구조

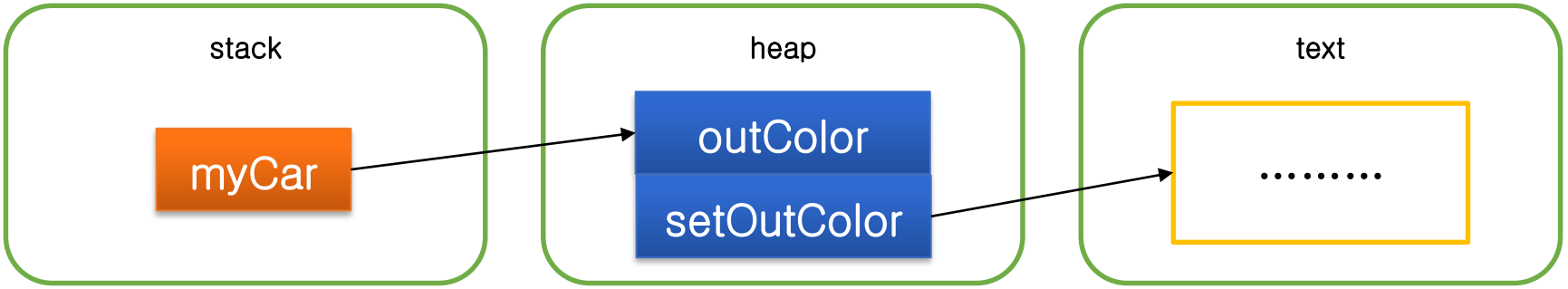
```
class 클래스명{
        //field
        자료형     필드명;
        자료형     필드명;
        ……

        //method
        반환자료형 메소드명(인자1, 인자2){
                구현
        }

        반환자료형 메소드명(인자1, 인자2){
                구현
        }
}
```

인스턴스 생성

```
class car{
    String  outColor;
    void setOutColor(String outColor){……}
}
public class Ex01 {
    public static void main(String[] args){
        car myCar = new car();
    }
}
```



생성자(Constructor)

❖ 이해하기
- 객체가 생성될 경우 즉, 인스턴스화 될 경우 실행
- 생성시 필요한 정보 처리
- 예) 게임을 실행하면 인증 화면으로 이동

❖ 기본 생성자
- 클래스가 만들어지면 자동으로 만들어지는 생성자
- 생략되어 있으며 명시화하여 구현할 수 있음.
- 아래 코드에서 new car()가 기본 생성자임.
- 클래스 명과 같은 이름으로 메소드 구현

```
public class Ex01 {
    public static void main(String[] args){
        car myCar = new car();
    }
}
```

메소드(method)

❖ 이해하기
- class 내에서 동작을 구현
- 차 문을 연다, 핸들을 조작한다.

❖ 메소드 구조
- 반환자료형은 기본타입과 참조타입 모두 사용 가능
- 매개변수는 생략 가능하며 여러 개를 사용할 수도 있음.

```
반환자료형 메소드이름(매개변수자료형 매개변수명){
        구현 내용
    }
}
```

오버로딩(Overloading)

❖ 이해하기

- 같은 이름으로 여러 기능을 구현할 경우 사용
- 매개 변수의 종류, 개수로 구분

```
void println() { .. }
void println(boolean x) { .. }
void println(char x) { .. }
void println(char[] x) { .. }
void println(double x) { .. }
void println(float x) { .. }
void println(int x) { .. }
void println(long x) { .. }
void println(Object x) { .. }
void println(String x) { .. }
```

this 연산자

❖ 이해하기
- Class 자체를 의미함.
- Class 내에서 생성자를 호출할 경우 사용
- Class 내에서 멤버 필드를 지정할 경우 사용

```
public class Member {
    private String picture;
    public String getPicture() {
        return picture;
    }
    public void setPicture(String picture) {
        this.picture = picture;
    }
}
```