

# 클래스, 객체

생성자, this, toString()

# 생성자(Constructor)

생성자

- 클래스 이름과 동일한 이름
- 반환 자료형 표기 없음

**Student.java**

```
public class Student {  
    String    name;  
    int       score;  
}
```

**Student.java**

```
public class Student {  
    String    name;  
    int       score;  
    public Student(String _name, int _score) {  
        name=_name;  
        score=_score;  
    }  
}
```



**Test.java**

```
public class Test {  
    public static void main(String[] args) {  
        Student student;  
        student=new Student();  
        student.name="김철수";  
        student.score=89;  
        System.out.println(student.name);  
        System.out.println(student.score);  
    }  
}
```

**Test.java**

```
public class Test {  
    public static void main(String[] args) {  
        Student student;  
        student=new Student("김철수", 89);  
        System.out.println(student.name);  
        System.out.println(student.score);  
    }  
}
```



# 클래스 String, 문자열 객체, 문자열 리터럴

## 클래스 String

- 자바 JDK에서 제공되는 클래스
- 문자열(문자들의 나열)을 저장하고 처리하는 클래스

```
public final class String ... {  
    ...  
}
```

```
public class Test {  
    public static void main(String[] args) {  
        String name1 = new String();  
        String name2 = new String("홍길동");  
        String name3 = "메르켈";  
        String name4 = "";  
    }  
}
```

- ← 길이 0 문자열 객체 생성
- ← 문자열 "홍길동"에 대한 객체 생성
- ← 문자열 "메르켈"에 대한 객체 생성
- ← 길이 0 문자열 객체 생성

String 리터럴  
(큰 따옴표로 둘러싸인 문자들)

# this

## *this*

- 현재 객체에 대한 참조값이 저장된 변수
- 생성자, instance method 등에서 사용

### Student.java

```
public class Student {  
    String    name;  
    int       score;  
    public Student(String _name, int _score) {  
        name=_name;  
        score=_score;  
    }  
}
```



### Student.java

```
public class Student {  
    String    name;  
    int       score;  
    public Student(String name, int score) {  
        this.name=name;  
        this.score=score;  
    }  
}
```

# 기본생성자(default constructor)

## Student.java

```
public class Student {  
    String    name;  
    int       score;  
}
```

## Test.java

```
public class Test {  
    public static void main(String[] args) {  
        Student student;  
        student=new Student();  
        student.name="김철수";  
        student.score=89;  
        System.out.println(student.name);  
        System.out.println(student.score);  
    }  
}
```

## Student.java

```
public class Student {  
    String name;  
    int score;  
  
    public Student() {  
    }  
}
```

### 생성자

- 클래스 이름과 동일한 이름을 가짐
- 반환 자료형 표기 없음
- 기본생성자(default constructor)는 파라미터 없는 생성자
- 클래스 정의문 내 생성자가 없으면 컴파일러가 자동으로 기본생성자를 생성

# 기본생성자(default constructor)

## Student.java

```
public class Student {  
    String    name;  
    int      score;  
    public Student(String name, int score) {  
        this.name=name;  
        this.score=score;  
    }  
}
```

## Test.java

```
public class Test {  
    public static void main(String[] args) {  
        Student student;  
        student=new Student(); ←  
    }  
}
```

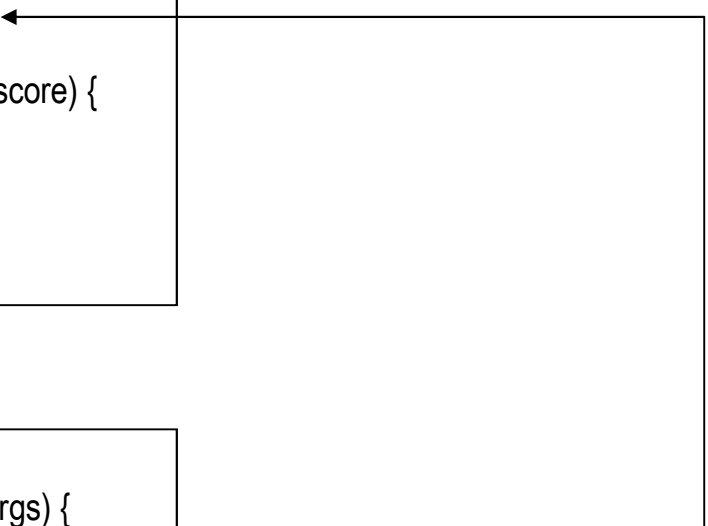
오류

- Student 클래스에 생성자가 존재하므로 기본생성자 Student()는 생성되지 않음
- 즉 기본생성자 Student()는 존재하지 않음

# 기본생성자(default constructor)

## Student.java

```
public class Student {  
    String    name;  
    int      score;  
    public Student() {  
    }  
    public Student(String name, int score) {  
        this.name=name;  
        this.score=score;  
    }  
}
```



## Test.java

```
public class Test {  
    public static void main(String[] args) {  
        Student student;  
        student=new Student();  
    }  
}
```

- 다른 생성자가 있는 경우 기본생성자는 자동 생성되지 않음
- 다른 생성자가 정의되어 있지만 Test.java의 예와 같이 기본생성자 사용도 필요하다면 Student 클래스 정의문에 기본생성자 정의를 추가해야 함

# 클래스 실습 A

- 다음 조건에 따라 클래스 Location을 정의하시오
  - 필드변수명(설명,자료형): country(국가명,String), city(도시명,String)
  - 국가명과 도시명을 파라미터로 전달받아 대응하는 필드에 대입하는 생성자
- 다음은 Location 객체를 생성하고 그 내용을 출력하는 코드와 실행 결과를 보인 것이다. 이 코드가 정상 동작하도록 Location 클래스를 정의하시오.

```
public class Test {  
    public static void main(String[] args) {  
        Location loc;  
        loc=new Location("한국","부산");  
        System.out.println("국가:"+loc.country);  
        System.out.println("도시:"+loc.city);  
    }  
}
```

실행결과

국가:한국  
도시:부산



# 클래스 실습 B

- 다음 조건에 따라 클래스 DateInfo를 정의하시오
  - 필드변수명(설명,자료형): year(연도,int), month(월, int), day(일, int)
  - 연,월,일을 파라미터로 전달받아 대응하는 필드에 대입하는 생성자
- 다음은 DateInfo 객체를 생성하고 그 내용을 출력하는 코드와 실행 결과를 보인 것이다. 이 코드가 정상 동작하도록 DateInfo 클래스를 정의하시오.

```
public class Test {  
    public static void main(String[] args) {  
        DateInfo d=new DateInfo(2020,4,13);  
        System.out.println(d.year+"년"+d.month+"월"+d.day+"일");  
    }  
}
```

실행결과

2020년4월13일

# 클래스 실습 C

- 다음 조건에 따라 클래스 Car를 정의하시오
  - 필드변수명(설명,자료형): id(고유번호,String), length(길이,int), weight(중량,double), fuelType(연료유형,char), exportOnly(수출용여부,boolean)
  - 고유번호, 길이, 중량, 연료유형, 수출용여부 값을 파라미터로 전달받아 대응하는 필드에 대입하는 생성자
- 다음은 Car 객체를 생성하고 그 내용을 출력하는 코드와 실행 결과를 보인 것이다. 이 코드가 정상 동작하도록 Car 클래스를 정의하시오.

```
public class Test {  
    public static void main(String[] args) {  
        Car car=new Car("KSC-123", 3850, 1500.8, 'D', false);  
        System.out.println("고유번호="+car.id);  
        System.out.println("길이(mm)="+car.length);  
        System.out.println("중량(kg)="+car.weight);  
        System.out.println("연료유형="+car.fuelType);  
        System.out.println("수출용="+car.exportOnly);  
    }  
}
```

실행결과

```
고유번호=KSC-123  
길이(mm)=3850  
중량(kg)=1500.8  
연료유형=D  
수출용=false
```

# 클래스 실습 D

- 다음 조건에 따라 클래스 StudentScore를 정의하시오
  - 필드변수명(설명,자료형): id(학번,String), score(성적값들, int 배열)
  - 학번,과 세 개 성적 값을 파라미터로 전달받아 대응하는 필드에 대입하는 생성자
- 다음은 StudentScore 객체를 생성하고 그 내용을 출력하는 코드와 실행 결과를 보인 것이다. 이 코드가 정상 동작하도록 StudentScore 클래스를 정의하시오.

```
public class Test {  
    public static void main(String[] args) {  
        StudentScore s=new StudentScore("S-123", 95, 100, 91);  
        System.out.println("학 번:"+s.id+", 성적:"+s.score[0]+","+s.score[1]+","+s.score[2]);  
    }  
}
```

실행결과

학번:S-123, 성적:95,100,91

# 클래스 실습 E

- 다음 조건에 따라 클래스 DateInfo를 정의하시오
  - 필드변수명(설명,자료형): year(연도,int), month(월, int), day(일, int)
  - 연,월,일을 파라미터로 전달받아 대응하는 필드에 대입하는 생성자
- 다음 조건에 따라 클래스 Employee를 정의하시오
  - 필드변수명(설명,자료형): id(직원번호,String), joinDate(입사일, DateInfo)
  - 직원번호, 연도, 월, 일 값을 파라미터로 전달받아 대응하는 필드에 대입하는 생성자
- 다음은 Employee 객체를 생성하고 그 내용을 출력하는 코드와 실행 결과를 보인 것이다. 이 코드가 정상 동작하도록 Employee 클래스를 정의하시오.

```
public class Test {  
    public static void main(String[] args) {  
        Employee e=new Employee("E-123", 2020, 3, 2);  
        System.out.println("직원번호:"+e.id+", 입사일:"+e.joinDate.year+"년"+e.joinDate.year+"월"+e.joinDate.day+"일");  
    }  
}
```

실행결과

직원번호:E-123, 입사일:2020년2020월2일

# toString()

## toString()

- 객체의 정보들을 하나의 문자열로 만들어 반환하는 메소드 (객체에 대한 문자열 표현을 String으로 반환)

### Student.java

```
public class Student {  
    String    name;  
    int       score;  
    public Student(String name, int score) {  
        this.name=name;  
        this.score=score;  
    }  
}
```



### Student.java

```
public class Student {  
    String    name;  
    int       score;  
    public Student(String name, int score) {  
        this.name=name;  
        this.score=score;  
    }  
    @Override  
    public String toString() {  
        return name+","+score;  
    }  
}
```

### Test.java

```
public class Test {  
    public static void main(String[] args) {  
        Student student;  
        student=new Student("이영희", 89);  
        System.out.println(student.name+" "+student.score);  
    }  
}
```



### Test.java

```
public class Test {  
    public static void main(String[] args) {  
        Student student;  
        student=new Student("이영희", 89);  
        System.out.println(student.toString());  
        System.out.println(student);  
    }  
}
```

# 클래스 실습 F

- 다음 조건에 따라 클래스 YMD를 정의하시오
  - 필드변수명(설명,자료형): year(연도,int), month(월, int), day(일, int)
  - 연,월,일을 파라미터로 전달받아 대응하는 필드에 대입하는 생성자
  - 연,월,일 값을 다음 예와 같은 형식의 문자열로 반환하는 toString() 메소드 (예: **2019년12월31일**)
- 다음은 YMD 객체를 생성하고 그 내용을 출력하는 코드와 실행 결과를 보인 것이다. 이 코드가 정상 동작하도록 YMD 클래스를 정의하시오.

```
public class Test {  
    public static void main(String[] args) {  
        YMD date=new YMD(2019, 12, 31);  
        System.out.println(date.toString());  
        System.out.println(date);  
    }  
}
```

실행결과

```
2019년12월31일  
2019년12월31일
```

# 클래스 실습 G



다음 조건에 따라 클래스 User를 정의하시오

- 필드변수명(설명,자료형): email(메일주소,String), registrationDate(가입일-8자리숫자문자열,String)
- 메일주소, 가입일을 파라미터로 전달받아 대응하는 필드에 대입하는 생성자
- 메일주소, 가입일을 다음 예와 같은 형식의 문자열로 반환하는 toString() 메소드 (예: **메일주소:gdhong@ks.ac.kr, 등록일:2019/11/23**)



다음은 User 객체를 생성하고 그 내용을 출력하는 코드와 실행 결과를 보인 것이다. 이 코드가 정상 동작하도록 User 클래스를 정의하시오.

```
public class Test {  
    public static void main(String[] args) {  
        User u=new User("gdhong@ks.ac.kr", "20191123");  
        System.out.println(u);  
    }  
}
```

실행결과

메일주소:gdhong@ks.ac.kr, 등록일:2019/11/23

# 클래스 실습 H



다음 조건에 따라 클래스 Name을 정의하시오

- 필드변수명(설명,자료형): firstName(성 제외 이름,String), lastName(성, String)
- 성, 성 제외 이름을 파라미터로 전달받아 대응하는 필드에 대입하는 생성자
- 성, 성 제외 이름을 다음 예와 같은 형식의 문자열로 반환하는 toString() 메소드 (예: **이름:길동,홍**)



다음 조건에 따라 클래스 Person을 정의하시오

- 필드변수명(설명,자료형): id(아이디, String), name(이름,Name)
- 아이디, 성, 성 제외 이름을 파라미터로 전달받아 대응하는 필드에 대입하는 생성자
- 아이디, 성, 성 제외 이름을 다음 예와 같은 형식의 문자열로 반환하는 toString() 메소드 (예: **아이디:gdhong, 이름:길동,홍**)



다음은 Person 객체를 생성하고 그 내용을 출력하는 코드와 실행 결과를 보인 것이다. 이 코드가 정상 동작하도록 Person 클래스를 정의하시오.

```
public class Test {  
    public static void main(String[] args) {  
        Person    p=new Person("gdhong", "홍","길동");  
        System.out.println(p);  
    }  
}
```

실행결과


아이디:gdhong, 이름:길동,홍



## References

 <http://docs.oracle.com/javase/7/docs/api/>

 <https://docs.oracle.com/javase/tutorial/java/>

 김윤명. (2008). 뇌를 자극하는 Java 프로그래밍. 한빛미디어.

 남궁성. 자바의 정석. 도우출판.

 황기태, 김효수 (2015). 명품 Java Programming. 생능출판사.