Chapter 06

클래스

06-4. 메소드

혼자 공부하는 자바(개정판) (신용권 저)

❖ 목차

- 시작하기 전에
- 메소드 선언
- 리턴 문
- 메소드 호출
- 메소드 오버로딩
- 키워드로 끝내는 핵심 포인트

시작하기 전에

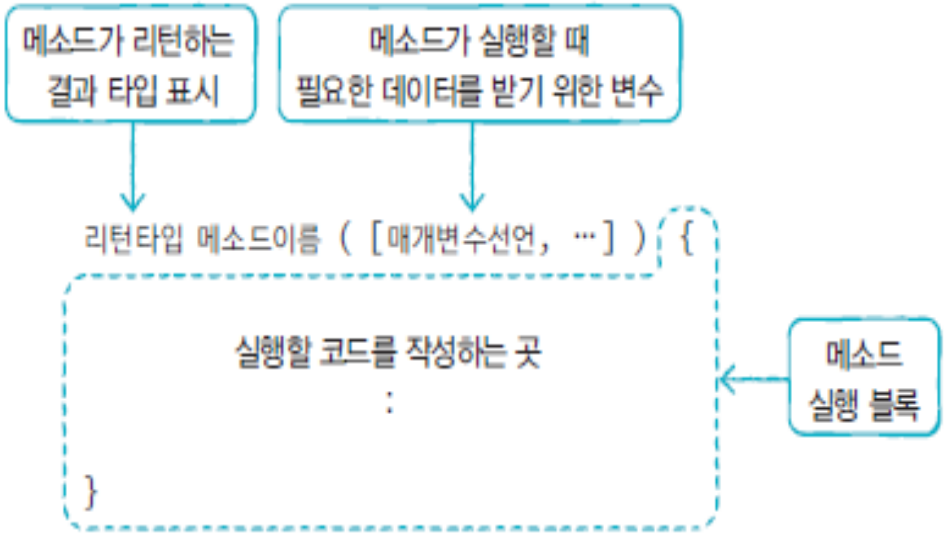
[핵심 키워드] : 선언부, void, 매개 변수, 리턴문, 호출, 오버로딩

[핵심 포인트]

메소드를 선언하고 호출하는 방법에 대해 알아본다.

❖ 메소드 선언부 (signature)

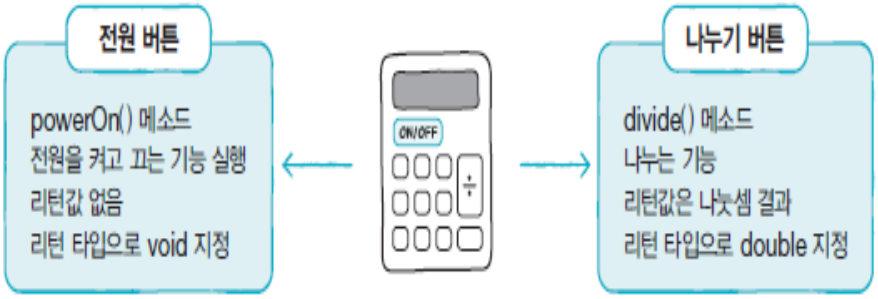
- 리턴 타입 : 메소드가 리턴하는 결과의 타입 표시
- 메소드 이름 : 메소드의 기능 드러나도록
- 식별자 규칙에 맞게 이름
- 매개 변수 선언 : 메소드 실행할 때 필요한
- 데이터 받기 위한 변수
- 메소드 실행 블록 : 실행할 코드 작성



메소드 선언

❖ 리턴 타입

- 메소드를 실행한 후의 결과값의 타입
- 리턴값 없을 수도 있음
- 리턴값 있는 경우 리턴 타입이 선언부에 명시




```
void powerOn() { … }
double divide( int x, int y ) { … }
```

- 리턴값 존재 여부에 따라 메소드 호출 방법 다름

```
powerOn();
double result = divide( 10, 20 );
```

```
int result = divide( 10, 20 );  //컴파일 에러
```

메소드 선언

❖ 메소드 이름

- 숫자로 시작하면 안 되고, $와 _ 제와한 특수문자 사용 불가
- 메소드 이름은 관례적으로 소문자로 작성
- 서로 다른 단어가 혼합된 이름일 경우 뒤이어 오는 단어의 첫 글자를 대문자로 작성

```
void run()  { … }
void startEngine() { … }
String getName() { … }
int[] getScores() { … }
```

메소드 선언

❖ 매개 변수 선언

- 메소드 실행에 필요한 데이터를 외부에서 받아 저장할 목적

```
double divide( int x, int y ) { … }
```

```
double result = divide( 10, 20 );
```

```
byte b1 = 10;
byte b2 = 20;
double result = divide( b1, b2 );
```

- 잘못된 매개값 사용하여 컴파일 에러 발생하는 경우

```
double result = divide( 10.5, 20.0 );
```

메소드 선언

❖ 매개 변수의 개수를 모를 경우

- 매개 변수를 배열 타입으로 선언

```
int sum1(int[] values) {  }
```

```
int[] values = { 1, 2, 3 };
int result = sum1(values);
int result = sum1(new int[] { 1, 2, 3, 4, 5 });
```

- 배열 생성하지 않고 값의 목록만 넘겨주는 방식

```
int sum2(int ··· values) {  }
```

```
int result = sum2(1, 2, 3);
int result = sum2(1, 2, 3, 4, 5);
```

```
int[] values = { 1, 2, 3 };
int result = sum2(values);
int result = sum2(new int[] { 1, 2, 3, 4, 5 });
```

리턴(return)문

- 리턴값이 있는 메소드
  - 메소드 선언에 리턴 타입 있는 메소드는 리턴문 사용하여 리턴값 지정

```
return 리턴값;
```

  - return문의 리턴값은 리턴타입이거나 리턴타입으로 변환될 수 있어야 함

```
int plus(int x, int y) {
  int result = x + y;
  return result;
}
```

```
int plus(int x, int y) {
  byte result = (byte) (x + y);
  return result;
}
```

리턴(return)문

- 리턴값이 없는 메소드 : void
  - void 선언된 메소드에서 return문 사용하여 메소드 실행 강제

```
return;
```

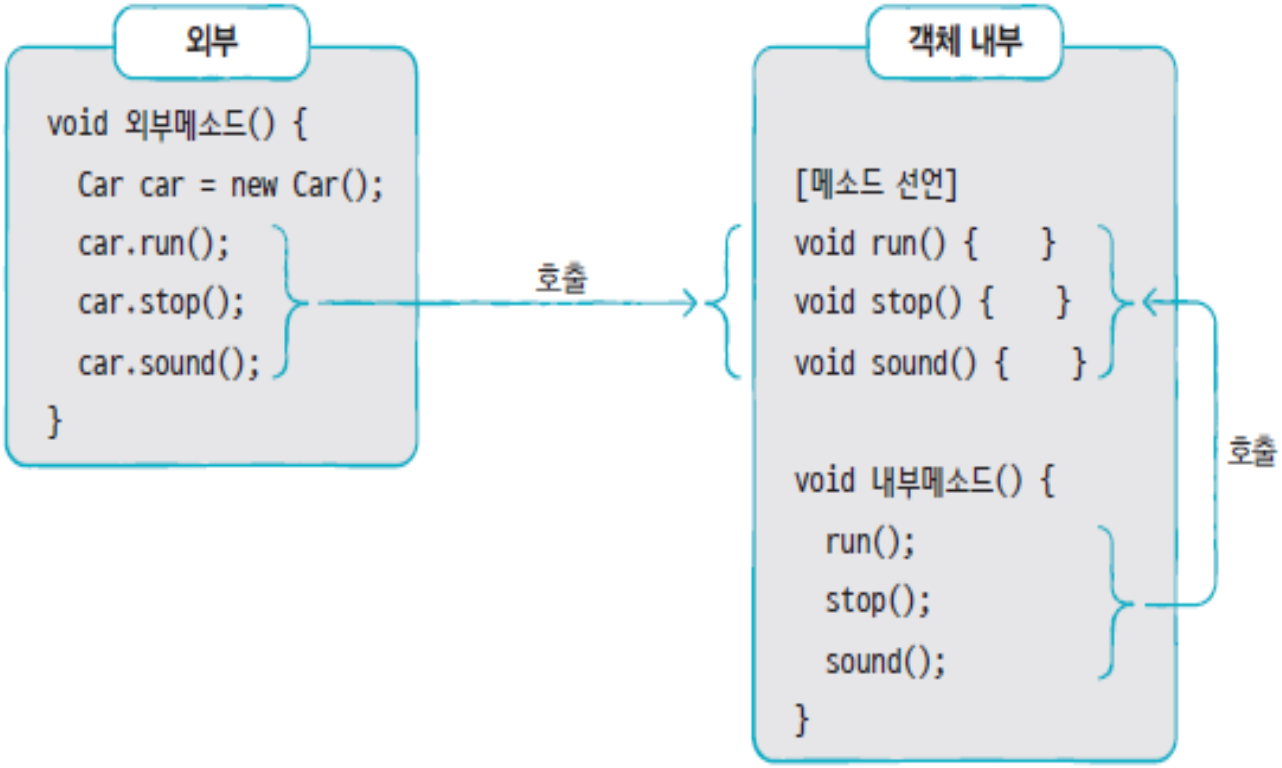
```
void run() {
  while(true) {
    if(gas > 0) {
      System.out.println("달립니다.(gas잔량:" + gas + ")");
      gas -= 1;
    } else {
      System.out.println("멈춥니다.(gas잔량:" + gas + ")");
      return;
    }
  }
}
```

run() 메소드 실행 종료

메소드 호출

❖ 메소드 호출

- 클래스 내외부의 호출에 의해 메소드 실행
  - 내부의 경우 단순히 메소드 이름으로 호출
  - 외부의 경우 클래스로부터 객체 생성한 뒤 참조 변수 사용하여 메소드 호출



메소드 호출

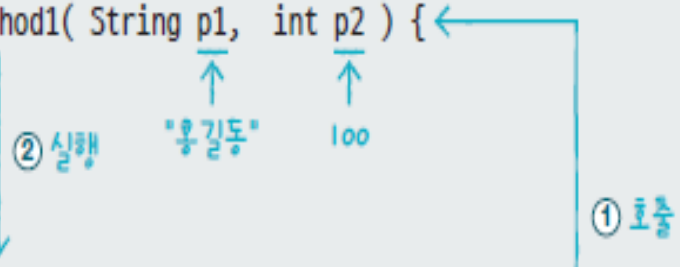
❖ 객체 내부에서 호출

- 메소드가 리턴값 없거나(void) 있어도 받고 싶지 않은 경우

```
메소드( 매개값, … );
```

```
public class ClassName {
  void method1( String p1,  int p2 ) {
  }

  void method2() {
    method1( "홍길동", 100 );
  }
}
```



메소드 호출

- 리턴값 있는 메소드 호출하고 리턴값 받고 싶은 경우

```
타입 변수 = 메소드(매개값, … );
```

리턴값

```
public class ClassName {
  int method1(int x, int y) {
    int result = x + y;
    return result;
  }

  void method2() {
    int result1 = method1(10, 20);      //result1에는 30이 저장
    double result2 = method1(10, 20);   //result2에는 30.0이 저장
  }
}
```

메소드 호출

❖ 객체 외부에서 호출

- 우선 클래스로부터 객체 생성

```
클래스 참조변수 = new 클래스( 매개값, … );
```

- 참조 변수와 도트 연산자 사용하여 메소드 호출

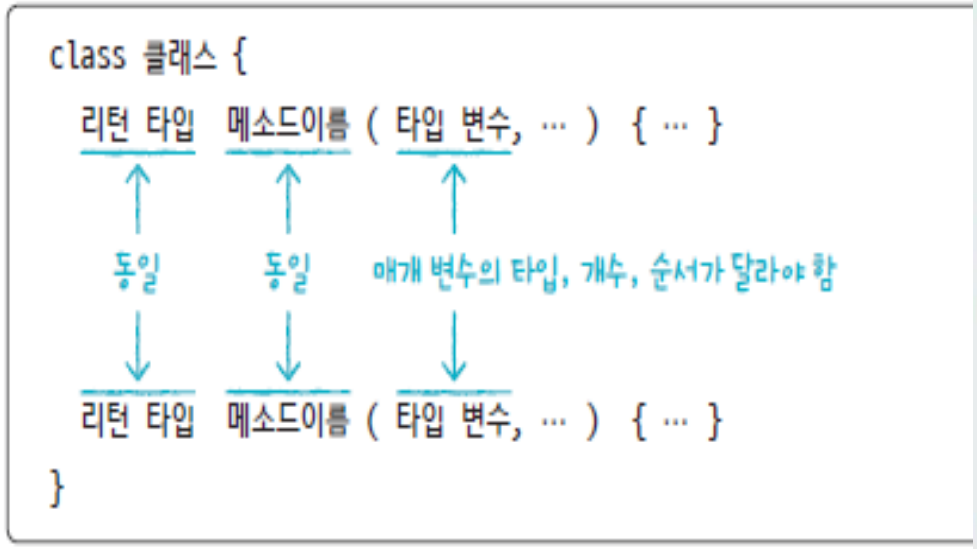
```
참조변수.메소드( 매개값, … );  //리턴값이 없거나, 있어도 리턴값을 받지 않을 경우
타입 변수 = 참조변수.메소드( 매개값, … );  //리턴값이 있고, 리턴값을 받고 싶을 경우
```

```
Car myCar = new Car();
myCar.keyTurnOn();
myCar.run();
int speed = myCar.getSpeed();
```

메소드 오버로딩

❖ 메소드 오버로딩 (overloading)

- 같은 이름의 메소드를 여러 개 선언
- 매개값을 다양하게 받아 처리할 수 있도록 하기 위함
- 매개 변수의 타입, 개수, 순서 중 하나가 달라야




```
int plus(int x, int y) {
  int result = x + y;
  return result;
}
double plus(double x, double y) {
  double result = x + y;
  return result;
}
```

메소드 오버로딩

- 오버로딩된 메소드 호출하는 경우 JVM은 매개값 타입 보고 메소드를 선택

```
plus(10, 20);
```

- plus(int x, int y)가 실행

```
plus(10.5, 20.3);
```

- plus(double x, double y)가 실행

```
int x = 10;
double y = 20.3;
plus(x, y);
```

- plus(double x, double y)가 실행

```
int plus(int x, int y) {
  int result = x + y;
  return result;
}
double plus(double x, double y) {
  double result = x + y;
  return result;
}
```

메소드 오버로딩

- 매개 변수의 타입, 개수, 순서 같은 경우 매개변수 이름 달라도 메소드 오버로딩 아님에 주의

```
int divide(int x, int y) { … }
double divide(int boonja, int boonmo) { … }
```

- System.out.println() 메소드

```
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) { … }
```

키워드로 끝내는 핵심 포인트

- 선언부 : 리턴 타입, 메소드 이름, 매개 변수 선언
- void : 리턴값이 없는 메소드는 리턴 타입으로 void를 기술해야 함
- 매개 변수 : 메소드 호출 시 제공되는 매개값이 대입되어 메소드 블록 실행 시 이용됨
- 리턴문 : 메소드의 리턴값을 지정하거나 메소드 실행 종료를 위해 사용할 수 있음.
- 호출 : 메소드를 실행하려면 '메소드 이름(매개값. ...)' 형태로 호출
- 오버로딩 : 클래스 내에 같은 이름의 메소드 여러 개 선언하는 것을 말함

Thank You!