SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

삼성 청년 SW 아카데미

Java

SAMSUNG SW ACADEMY FOR YOUTH

Programming Language

객체지향 프로그래밍

- 객체지향 프로그래밍
- 클래스
- 생성자

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

객체지향 프로그래밍

객체지향 프로그래밍

- 객체지향 프로그래밍 (OOP, Object Oriented Programming)
  - 객체(客體) : 사물과 같이 유형적인 것과 개념이나 논리와 같은 무형적인 것들
  - 지향(指向) : 작정하거나 지정한 방향으로 나아감
  - 객체 모델링 : 현실세계의 객체를 SW 객체로 설계하는 것
- 클래스(Class)
  - 객체(客體)를 만드는 설계도(Blueprint)
- 인스턴스 (Instance)
  - 클래스를 통해 생성된 객체

객체지향 프로그래밍

- 객체지향 프로그래밍 특징 (A PIE)
  - Abstraction (추상화)
  - Polymorphism (다형성)
  - Inheritance (상속)
  - Encapsulation (캡슐화)

- 객체지향 프로그래밍 장점
  - 모듈화된 프로그래밍
  - 재사용성이 높다

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

클래스

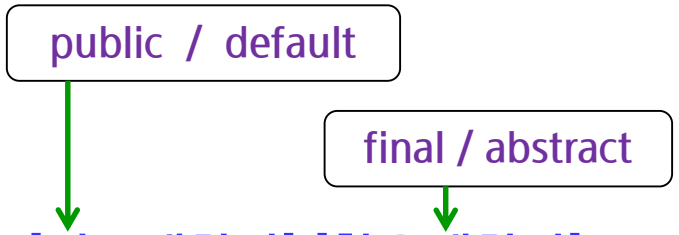
클래스

- 관련 있는 변수와 함수를 묶어서 만든 사용자정의 <자료형>
- 모든 객체들의 생산처
- 클래스 == 객체를 생성하는 틀
- 프로그래밍이 쓰이는 목적을 생각하여 어떤 객체를 만들어야 하는지 결정한다.
- 각 객체들이 어떤 특징(속성과 동작)을 가지고 있을지 결정한다.
- 클래스를 통해 생성된 객체를 인스턴스라고 한다.
- 객체들 사이에서 메시지를 주고 받도록 만들어 준다.

클래스 구성

- 속성 (Attribute) - 필드
- 동작 (Behavior) - 메소드
- 생성자 (Constructor)
- 중첩 클래스(클래스 내부의 클래스)

클래스 선언 및 객체의 생성




```
[접근제한자] [활용제한자] class 클래스명 {

    속성 정의 (필드)
    기능 정의 (메소드)
    생성자

}
```

```
public class Person {
    String name;
    int age;

    public void eat(){
    }

    public Person(){
    }
}
```

- 클래스명 변수명 = new 클래스명();
- 변수명.필드명
- 변수명.메서드명();

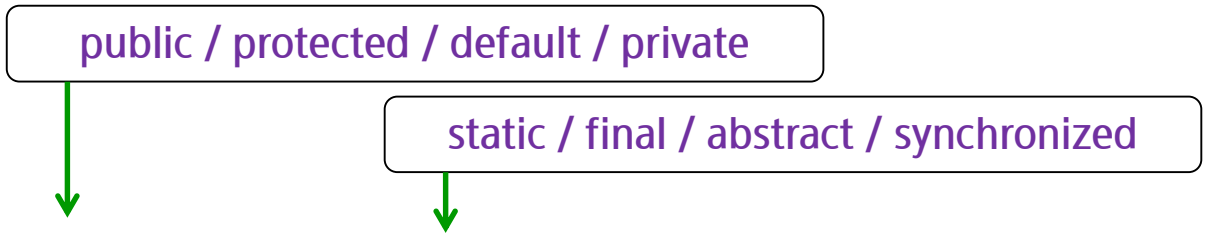
변수

- 클래스 변수(class variable)
  - 클래스 영역 선언 (static 키워드)
  - 생성시기 : 클래스가 메모리에 올라 갔을 때
  - 모든 인스턴스가 공유함.
- 인스턴스 변수(Instance variable)
  - 클래스 영역 선언
  - 생성시기 : 인스턴스가 생성되었을 때 (new)
  - 인스턴스 별로 생성됨.
- 지역 변수(local variable)
  - 클래스 영역 이외 (메서드, 생성자 … 등)
  - 생성시기 : 선언되었을 때

메소드

메소드 (Method)

- 객체가 할 수 있는 행동을 정의
- 어떤 작업을 수행하는 명령문의의 집합에 이름을 붙여 놓은것
- 메소드의 이름은 소문자로 시작하는 것이 관례



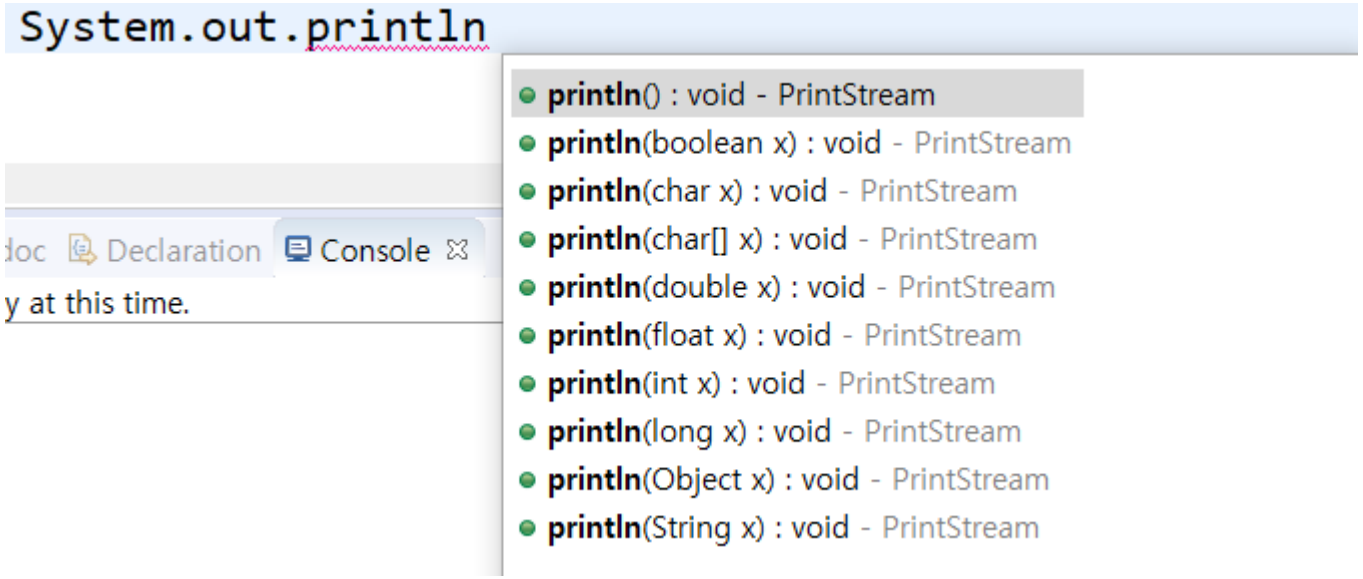

```
[접근제한자] [ 활용제한자] 반환값 메소드이름([매개변수들]) {

        행위 기술…

}
public static void main(String [] args) { }
```

메소드 오버로딩 (Overloading)

- 이름이 같고 매개변수가 다른 메소드를 여러 개 정의하는 것
- 중복 코드에 대한 효율적 관리 가능
- 파라미터의 개수 또는 순서, 타입이 달라야 할 것 (파라미터 이름만 다른 것은 X)
- 리턴 타입이 다른 것은 의미 X



클래스와 객체

- 클래스 : 관련 있는 변수와 함수를 묶어 만든 사용자 정의 자료형

- 객체 : 하나의 역할을 수행하는 '메소드와 변수(데이터)'의 묶음

- 객체지향 프로그래밍 : 프로그램을 단순히 데이터와 처리 방법으로 나누는 것이 아니라, 프로그램을 수많은 '객체(object)'라는 기본 단위로 나누고 이들의 상호작용으로 서술하는 방식

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

생성자

생성자

생성자 메서드

- new 키워드와 함께 호출하여 객체 생성
- 클래스명과 동일
- 결과형 리턴값을 갖지 않음
- 객체가 생성될 때 반드시 하나의 생성자 호출
- 멤버필드의 초기화에 주로 사용
- 하나의 클래스 내부에 생성자가 하나도 없으면 자동적으로 default 생성자가 있는 것으로 인지
  - default 생성자: 매개 변수도 없고 내용도 없는 생성자
- 매개변수의 개수가 다르거나, 자료형이 다른 여러 개의 생성자가 있을 수 있음(생성자 오버로딩)
- 생성자의 첫번째 라인으로 this() 생성자를 사용하여 또 다른 생성자를 하나 호출 가능

생성자 특징

- 클래스 명과 이름이 동일 (대·소문자)
- 반환타입이 없다. (void 작성 x)

```
public class Dog {
    public Dog() {
        System.out.println("기본 생성자!");
        System.out.println("클래스 이름과 동일하고 반환타입 x");
    }
}
```

생성자 특징

기본(디폴트) 생성자

- 클래스 내에 생성자가 하나도 정의되어 있지 않을 경우 JVM이 자동으로 제공하는 생성자
- 형태 : 매개변수가 없는 형태, 클래스 명() { }

```
class Dog {
    생성자가 하나도 없는 상태임
    JVM 이 자동으로 제공함
    ??????
    Dog( ) { }
}
```

```
class Main {
  public static void main(String [] a) {
        // 객체 생성
        Dog d = new Dog( );
   }
}
```

생성자 특징

파라미터가 있는 생성자

- 생성자의 목적이 필드 초기화
- 생성자 호출 시 값을 넘겨주어야 함.
- 해당 생성자를 작성하면 JVM에서 기본 생성자를 추가하지 않음.

```
class Dog {
    String name;
    int age;
    Dog( String n, int a){
        name = n;
        age = a;
    }
}
```

```
class Main {
    public static void main(String [] a) {
        Dog d1 = new Dog( );
        d1.name = "쫑";
        d1.age = 3;
        Dog d2 = new Dog("메리", 4);
    }
}
```

생성자 특징

- 생성자 오버로딩을 지원한다.
  - 클래스 내에 메소드 이름이 같고 매개변수의 타입 또는 개수가 다른 것

```
class Dog {
    Dog( ) {  }
    Dog(String name) {  }
    Dog(int age) {  }
    Dog(String name, int age) {  }
}
```

```
class Main {
    public static void main(String [] a) {
        Dog d = new Dog( );
        Dog d2 = new Dog("쫑");
        Dog d3 = new Dog(3);
        Dog d4 = new Dog("메리", 4);
    }
}
```

생성자 특징

- this
  - 참조 변수로써 객체 자신을 가리킴
  - this를 이용하여 자신의 멤버 접근 가능
  - 지역변수(매개변수)와 필드의 이름이 동일할 경우 필드임을 식별할 수 있게 함.
  - 객체에 대한 참조 이므로 static 영역에서 this 사용 불가

생성자 특징

- this의 활용
  - this.멤버변수
  - this ([인자값..]) : 생성자 호출
  - this 생성자 호출 시 제한사항
    - 생성자 내에서만 호출이 가능함
    - 생성자 내에서 첫번째 구문에 위치해야 함

```
class Dog {
    String name;
    int age;
    void info ( ) {
        System.out.print(this.name);
        System.out.println(this.age);
    }
}
```

생성자 특징

- this의 활용
  - this.멤버변수
  - this ([인자값..]) : 생성자 호출
  - this 생성자 호출 시 제한사항
    - 생성자 내에서만 호출이 가능함
    - 생성자 내에서 첫번째 구문에 위치해야 함

```
class Dog {
    String name;
    int age;
    Dog (  )  {
        this("쫑");
    }
    Dog ( String name ) {
    }
}
```

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

다음 방송에서 만나요!

삼성 청년 SW 아카데미