1. 배열

* 변수에는 하나 값만 담을 수 있다.
* 여러 개 값을 사용하는 변수를 만들고 싶으면 배열을 사용

1. 배열 선언

* 배열을 선언하는 방식은 두 가지 방식이 있다

변수타입[] 배열의 이름;

변수타입 배열의 이름[];

ex) int[] score;

int score[];

1. 배열 초기화

* 배열의 초기화 방식에는 두 가지 방식 있음

배열의 값들을 직접 입력

변수타입[] 배열의 이름 = {값1, 값2, 값3, 값4….}

ex) int[] score = {100, 90, 80};

new 키워드 사용하여 초기화

변수타입[] 배열의 이름 = new 변수타입[] {값1, 값2, 값3, 값4…..}

ex) int[] score = new int[] {100, 90, 80};

new 키워드를 사용하여 배열의 길이만 지정 후 값을 초기화는 방식

변수타입에 따라 초기화되는 값이 달라짐

변수타입[] 배열의 이름 = new 변수타입[배열의 길이]

ex) int[] score = new int[3];

1. 배열의 복사

* for문을 이용한 복사

ex) int[] score1 ={100, 90, 80};

int[] score2 = new int[5];

for(int i = 0; i < score1.length; i++) {

score2[i] = score1[i];

}

* System.arraycopy() 메소드 이용한 복사

System.arraycopy(원본 배열의 이름, 복사를 시작할 인덱스, 복사될 대상의 배열의 이름(새 배열의 이름), 새 배열에 복사가 시작되는 인덱스, 복사될 개수);

ex) String[] str1 = {"java", "hello"};

String[] str2 = new String[3];

System.arraycopy(str1, 0, str2, 0, 2);

1. 향상된 for문

* 배열을 이용하여 초기화된 변수없이 for 사용가능

int[] scores = {100, 90, 80};

for(int score : scores) {

}

배열 요소의 개수만큼 요소를 하나씩 꺼내서 사용하는 방식

1. 열거 타입(Enum)

* 데이터가 가지수가 한정돼있을 때 사용
* 요일이나 학점 등 데이터의 개수가 한정돼있을 때 미리 상수로 지정하여 사용

ex) public enum Week {

MONDAY,

TUESDAY,

WEDNESDAY,

…

}

열거타입에 요소들은 대문자로 선언

* 열거타입의 사용

열거타입 변수명 = 열거타입.열거상수

Week today = Week.MONDAY;

1. 객체

* 객체란 물리적으로 존재하거나 추상적으로 생각할 수 있는 것 중에 자신의 속성 값을 가지고 있는 것
* 객체는 속성과 동작을 가지고 있음

물리적 객체 : TV, 냉장고, 차, 리모컨, 컴퓨터..

추상적 객체 : 강의, 주문..

* 객체의 예시

사람 {

나이,

신장,

이름

밥먹는다,

잔다,

웃는다..

}

1. 객체와 클래스의 관계

* 클래스는 객체의 설계도

Class Human {

int age;

int height;

String name;

void eat(){}

void sleep(){}

void smile(){}

}

1. 클래스의 선언

* 클래스 명명 규칙

하나 이상의 문자로 이뤄져야 됨

첫 글자에는 숫자가 올 수 없음

$, \_이외의 특수문자 사용 불가능

자바 예약어는 클래스 명으로 사용할 수 없음

보편적인 클래스 명은 첫 글자는 대문자 나머지는 소문자

ex) public Class Car {

}

public 접근 제한자 : 어디서나 선언된 클래스에 접근가능

1. 객체의 생성과 객체 초기화

* new 키워드를 이용하여 진행

ex) Car car = new Car();

1. 객체의 속성(필드)

* 현재 객체의 상태 값을 저장하는 변수, 배열, 다른 객체
* 속성의 사용은 객체명.속성명 접근해서 사용

1. 객체의 생성자 메소드

* 생성자 메소드는 객체를 생성할 때 객체를 초기화해주는 메소드
* 클래스를 만들면 자동으로 기본 생성자 메소드 제공
* 기본 생성자 메소드의 형태

public 클래스명() {

}

기본 생성자 메소드는 처음 클래스를 만들면 생략되어 있음

* 기본 생성자 메소드의 사용

객체를 생성할 때 new 키워드와 함께 호출

* 사용자 정의 생성자 메소드

public 클래스명(인자 값1, 인자 값2,…..) {‘

속성 값1 = 인자 값1;

속성 값2 = 인자 값2;

}

* 이름이 같은 메소드의 인자 값의 개수만 변경하여 메소드를 재정의 하는 것을 오버로딩

1. this() 생성자

* 클래스 내에서 다른 생성자 호출하고 싶을 때 this() 생성자 메소드 사용

1. 메소드

* 동작들을 구현해놓은 함수 블록
* 메소드의 선언

반환 값 타입 메소드 명(매개변수1, 매개변수2…..) {

return 반환 값;

}

* 반환 값이 없을 경우에는 void 타입을 지정
* 메소드 명명규칙

숫자로 시작하면 안됨

$, \_이외의 특수문자 사용 불가능

CamelCase표기법을 따름 – 첫 번째 단어는 소문자로 두 번째 단어의 첫 글자는 대문자로 표기

* 매개변수의 개수를 모를 때의 메소드 선언

배열을 이용해서 매개변수를 받도록 메소드 선언

ex) int add(int[] nums) {

}

1. 난수 발생

* Math.random() 메소드를 이용하여 0.0이상 1.0미만의 난수발생