**초창기 시절에 JAVA 언어의 단점**

1. 기존 C/C++에 비해서 속도가 굉장히 느리다.

2. 리소스( 메모리,CPU )를 많이 사용한다.

**현재 JAVA 언어의 장점**

1. 객체 지향 언어로 기능을 부품화할 수 있다. -> 하나하나를 객체를 합쳐 큰 프로그램을 만들 수 있다는 것

2. JRE를 이용해서 운영체제로부터 자유롭다. -> 윈도우에서 코딩을 했지만 -> 리눅스용 JRE를 설치하면 자유롭게 할 수 있다는 점

3. 웹 및 모바일 프로그래밍이 쉽다.

4. GC를 통한 자동 메모리 관리를 지원한다. Garbage colllector를 통해

5. 실행속도가 많이 개선되어 빨라졌다.

**가비지 컬렉터 (Gabage Collector)**

* 프로그램 실행에 필요한 메모리를 Gabage Collector 가 자동으로 관리한다.

**C 계열 프로그램**  **Java 프로그램**

개발자가 직접 메모리를 관리해야함 개발자가 메모리에 접근할 수 없음.

만약 메모리관리를 잘못할 경우 따라서 개발자는 메모리관리를 할 수 없고

메모리 누수가 발생하고 가비지 컬렉터가 불필요한 메모리를 회수해서

타 프로그램 동작이 멈출 수 있다. 메모리를 최적화 함

**기본자료형과 객체자료형**

기본자료형은 데이터가 변수에 직접 저장되고, 객체 자료형은 객체 메모리 주소가 변수에 저장된다.

C 계열에서는 포인터라고 하고, Java에서는 레퍼런스라고 한다.

**객체란?**

세상에 존재하는 모는 것을 뜻하며, 프로그래밍에서 속성과 기능을 가지는 프로그램 단위이다.

객체 (인간세상) ㅜ 객체 (프로그램)

ex) 사람 속성: 키, 몸무게 , 기능: 의사 사칙연산 프로그램 속성 : \*,+,-,/

체중계 속성: 바늘, 눈금 기능: 몸무게측정 기능: 연산기능

**클래스란**?

객체를 생성하기 위한 틀로 모든 객체는 클래스로부터 나온다.

선수용 자전거, 사람용 자전거, SUV,등등 객체가 어떤걸 쓰느냐에 따라 달라질 수 있다.

붕어빵 기계 틀

객체를 하나만 만든다고 하더라도 클래스가 필요하다.

클래스만 만들고 객체를 뽑아내고 그럼 그건 메모리에 탑제된다.

**클래스 구성요소** (자전거)

속성 (멤버변수) = 인장, 핸들, 바구니, 기어, 페달

기능 (메서드) = 기어변속, 가속, 브레이크

**가비지 컬렉터**

가비지 컬렉터가 쓱 가져간다 -> 떠돌아다니는 변수나 객체를 처리하는 역할 ( 누구도 참조하지않고, 연결되어 있지 않은 것들은 처리하는 역할 )

**클래스 제작과 객체 생성**

클래스는 멤버변수 (속성) , 메서드(기능) , 생성자 등으로 구성된다.

Void ->반환값이 없다.

Public string run() -> 반환값이 string이다

**생성자**

클래스이름과 동일한 메서드

반환형이 없다.

생성할때 (가장 먼저 !!!) 필요한 기술을 사용하면 된다.

**메서드**

메서드 선언부 , 정의부로 나눌 수 있다.

접근자 반환형 메서드이름 메개변수

Public void getInfo () {

System.out.println( " i = " + i);

}

**중복 메서드 ( overloading)**

이름은 같고, 매개변수의 개수 또는 타입이 다른 메서드를 만들 수 있다.

Public void getInfo(){

System.out.println(" ---getInfo - I --");

}

Public void getInfo(String str, int ins){

System.out.println(" ---getInfo - II --");

}

**메모리에서 객체 생성(동적 생성)**

객체는 메모리에서 동적으로 생성되며, 객체가 더 이상 필요 없게되면 GC에 의해서 제거된다.

ChildClass child = new ChildClass();

**여기서 child 는 레퍼런스**

**->** 생성된 객체의 주소를 변수에 저장하는 것을 레퍼런스라고 한다.

**Null과 NullPointException**

레퍼런스에 null 저장되면 객체의 연결이 끊기며, 더 이상 객체를 이용할 수 없다.

회수해가면 없어진다.

**디폴트 생성자**

객체가 생성될 때 가장 먼저 호출되는 생성자로, 만약 개발자가 명시하지 않아도 컴파일 시점에 자동 생성된다.

사용자정의 생성자

디폴트 생성자가 없더라도 컴파일러가 자동으로 생성한다.

**소멸자**

객체가 GC에 의해서 메모리에서 제거 될 때 finalize() 메서드가 호출된다.

System.gc() ; // 이걸 사용한다고 해서 GC가 바로 작동하는 것이 아니라!(GC바쁨) ,가급적 빨리 작동하도록 요청하는 것이다.

java는 기본적으로 메모리를 개발자가 직접관리하지않으므로 일반적으로 System.gc(); 를 사용하는 경우는 드물다

**this 키워드**

현재 객체를 가리킬 때 this를 사용한다.

this.x = x; //나자신의 객체를 가리킴 -> 전역변수 , 클래스에 있는 개체를 말한다. !!!!!

this.y = y;