오버로딩은 동일한 이름의 메소드를 쓰면서 매개변수의 개수 또는 타입만 달리한다.

오버로딩을 쓰는 이유?

기능은 동일하지만 여러 다른 데이터 타입을 받아야 할 때 메소드 이름이 각각 달라지면 복잡해지기 때문에 오버로딩을 사용하여 메소드의 이름을 동일하게 하여 복잡성을 줄여서 사용한다.

java.lang.system

※ 오버로딩과 오버라이딩 잘 구분할 것

클래스 구성요소 중 마지막 구성요소이다.

※ 생성자(Constructor)

생성자는 메소드다. 객체가 생성될 때 호출되는 특별한 메소드이며, 주로 자원을 액세스하거나 자원을 사용하기 위해서 초기화하거나 자원관련 작업을 하거나 객체가 가지고 있는 멤버변수를 초기화 하는 작업을 정의한다. (자원 – DBMS, 네트워크, 파일시스템….)

[객체생성]

클래스타입 변수 = new 생성자메소드()

------------ --------------

^ ^

|\_사용할클래스 |\_클래스 안에 미리 정의되어 있는 생성자 메소드를 호출

일반 메소드 처럼 생성자 메소드도 매개변수로 외부에서

값을 전달 받아 사용할 수 있다.

[규칙]

1. 생성자 메소드명은 클래스명과 대소문자가 정확하게 동일한 이름으로 정의해야 한다.

=>리턴타입을 명시하지 않는다.

2. 생성자 메소드를 정의하지 않으면 컴파일러가 기본 생성자를 제공한다.

=> 기본 생성자: 매개변수가 없는 생성자

=> 생성자 메소드를 개발자가 정의하면 컴파일러가 기본 생성자를 제공하지 않는다.

=> 처리되는 일이 없다고 하더라도 무조건 기본 생성자는 정의해야 한다.

3. 생성자 메소드도 일반 메소드 처럼 매개변수를 정의하고 외부에서 값을 전달받아 사용할 수 있다.

=>주로 객체에 정의된 멤버변수의 값을 초기화하는 작업

4.생성자 메소드도 일반 메소드처럼 오버로딩을 허용한다.

5.생성자 메소드 내부에서 다른 생성자 메소드를 호출할 수 있다.

=> this(매개변수…)

=>반드시 첫 번째 문장에서 호출해야 한다.

java.lang.String

생성자 메소드도 오버로딩이 가능하다.

Deprecated 업데이트된 현재 버전부터는 쓰지 않겠다.

지금까지 우리가 썼던 것은 String(String Original)이다.

문자열 조작은 문자열에 대한 작업이다.

string(byte[] bytes) 사용해 볼 것이다.

객체 또는 인스턴스가 생성될 때 호출되는 ‘인스턴스 초기화 메소드’이다.

변수 초기화 작업에 주로 사용되며, 인스턴스 생성 시 실행되어야 할 작업 위해 사용

1. 생성자의 이름은 클래스의 이름과 같아야 한다. ; 클래스명과 동일

-객체가 생성될 때 호출되는 메소드

그래야 new 연산자가 실행될 때 어떤 객체(인스턴스)를 heap에 올릴 것인지 판단할 수 있기 때문이다.

객체가 생성될 때 실행되는 시점이 딱 한 번이다.

2. 생성자는 리턴 값이 없다.

※ 연산자 new가 인스턴스를 생성하는 것이지 생성자가 인스턴스를 생성하는 것이 아니다.

ex) Card c = new Card( );

1. 연산자 new에 의해서 메모리에 Card클래스의 인스턴스가 생성된다.

2. 생성자 Card()가 호출되어 수행된다.

3. 연산자 new의 결과로, 생성된 Card인스턴스의 주소가 반환되어 참조변수 c에 저장된다.

기본 생성자가 컴파일러에 의해서 추가되는 경우는 클래스에 정의된 생성자가 하나도 없을 때 생성된다.

java.util.random

random() 사용

java.io.file

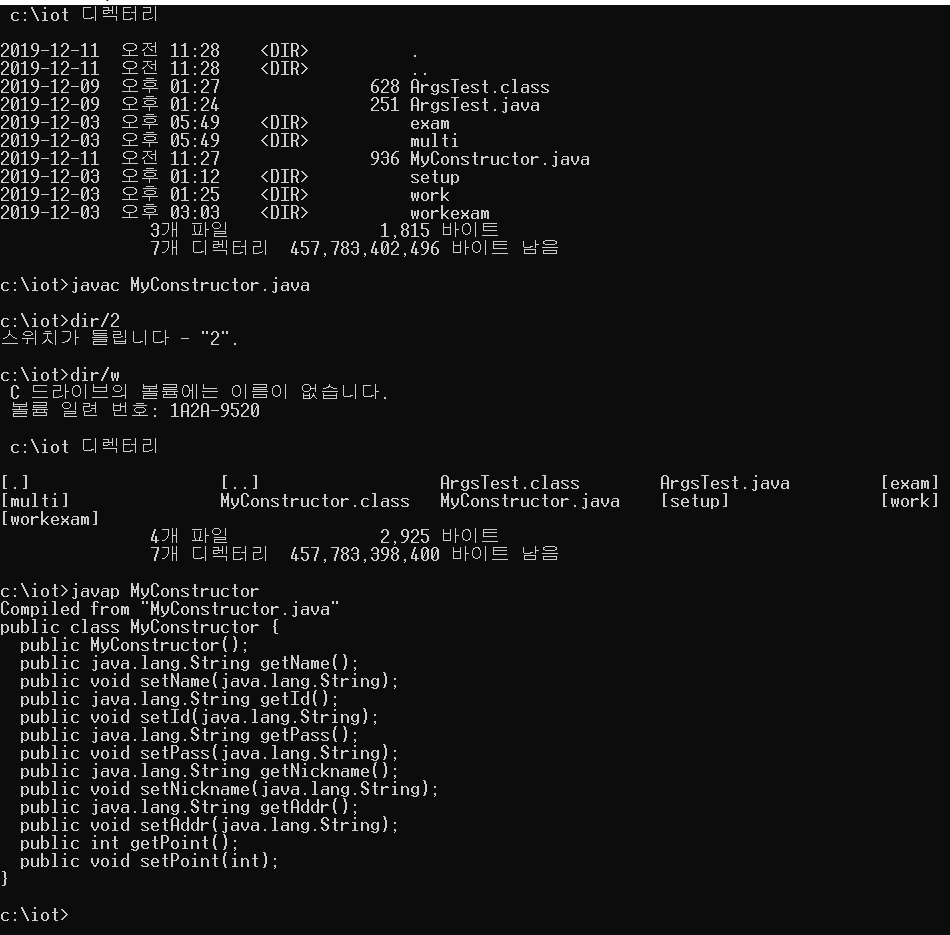
파일과 클래스를 모델링 해 놓은 것

모델링 과정은 다시 설명한다.

객체의 파일과 특성을 뽑아내는 작업

basicjava커서 오른쪽 클릭 → 파일 → test.txt 확인

File(String pathname)



주로 대부분의 생성자 메소드는 public을 사용한다.

생성자는 삽입용, setter와 getter는 하나씩 체크하고 수정할 때 사용한다. 결론은 둘 다 존재해야 한다.

처음에 실행하려고 했을 때

이미 스태택 멤버를 읽어서 올려놓는다.

static은 대기 상태의 멤버 변수이다.

new로 클래스 생성할 때마다 생성된 변수가 딱 한 번만 메모리에 올라간다.

static이 붙은 변수는 공유변수이다. 객체마다 달라지는 값은 인스턴스 변수에다가 저장한다.

회원관리가 가장 복잡한 로직이다.

가입은행정보 = 무적은행

똑같은 정보들은 메소드 영역에 생성하여

객체를 생성할 때마다 변수를 만들 필요없이

메소드 영역에서 가져다 쓰기위해 만들어졌다.

The static field StaticDemo.staticNum should be accessed in a static way

클래스 변수는 클래스 소속이지 인스턴스 소속이 아니다.

그래서 그 static으로 선언한 변수는 static이 속하는 클래스명으로 접근해야 한다.

일반 메소드에서 클래스 메소드 호출이 가능한 이유는 이미 클래스 메소드가 메모리에 올라와

있기 때문에 메모리에 객체 할당 없이 호출이 가능하다.

static 메소드는 클래스명.메소드 형식으로 쓴다.

클래스 메소드를 사용하는 시점에 일반 메소드는 안 올라와 있어서 일반적인 방법으로

사용이 불가능하다.

클래스에서 정의된 메소드라고 하더라도 무조건 객체생성 후에 호출해야 한다.

for문 자동 생성

for + ctrl +space

값을 넘긴다는 의미에서 call by value이고 참조형 주소를 넘긴다는 의미에서 call by reference이다.