싱글톤 패턴 : 어플리케이션이 시작 될 때 어떤 클래스가 최초 한번만 메모리를 할당하고 그 메모리에 인스턴스를 만들어 사용하는 디자인 패턴

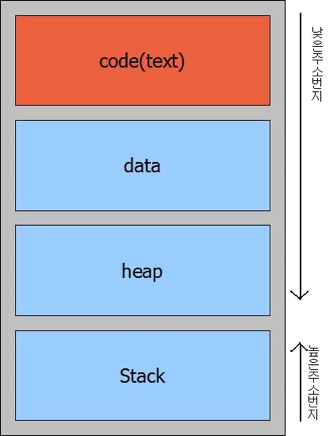
* 하나의 인스턴스를 메모리에 등록해서 여러 쓰레드가 동시에 해당 인스턴스를 공유하여 사용 할 수 있다.
* 주위해야 할점 : 싱글턴을 만들 때 동시성 문제를 고려해서 설계해야 한다.

싱글톤 패턴의 장점 :

* 고정된 메모리 영역을 얻으면서 한번의 new로 인스턴스를 사용하기 때문에 메모리 낭비를 방지 한다.
* 싱글톤으로 만들어진 클래스의 인스턴스는 전역이기 때문에 다른 클래스의 인스턴스들이 데이터를 공유하기 쉽다.

싱글톤 단점

* 싱글톤 인스턴스가 너무 많은 일을 하거나 많은 데이터를 공유시킬 경우에 다른 클래스의 인스턴스들 간에 결합도가 높아져 "개방-폐쇄 원칙을 위배된다"
* 싱글톤 인스턴스를 수정하게 되면 이것을 사용하는 클래스 모두 수정 해야 한다.
* 멀티쓰레드 환경에서 동기화처리를 안하면 인스턴스가 2개가 생성 될 수 있는 가능성이 생긴다.



프로그램을 크게 보면 code, data로 구성된다. 좀더 세분할 경우 code(text),data,heap,stack 으로 나뉜다. (각각 세그먼트라고 한다)

Code(text) : 프로그램의 code명령, 기계어 명령이 존재

* + Read only 이며 이곳에 데이터를 쓸려면 access violation을 발생시켜야 한다.
  + Cpu가 읽어 수행한다고 해서 text라고 부른다.

Data : 전역(global)변수, 정적(static)변수, 초기화된 배열과 그 구조들이 저장되는 영역

* + 프로그램이 실행될 때 생성되고 프로드램 종료시 시스템한테 반환한다.
  + Complie-time때 결정

Heap : 동적으로 할당되는 메모리 영역

* + New, 자바 오브젝트들이 저장되는 공간
  + 자바 경우 가비지컬렉터에 의해 자동으로 해체된다.
  + Run-time시 요구 할때마다 사용된다.

Stack : 지역(local)변수 및 매개변수, 복귀번지 등이 저장되는 영역

* + 함수 호출시 stack영역에 생성되고 사용된 후 시스템에 반환된다.
  + 함수로 인수를 보낼 때는 인수를 역순으로 보낸뒤 복귀번지에 저장한다. (선입후출)
* Code, data, heap은 하위 메모리로 부터 할당, stack은 상위 메모리로 부터 할당 된다.
* 처음 컴파일시 소스코드는 하위메모리에 할당하고 그다음 전역변수, 정적변수를 data영역에 할당된다.
* Heap영역은 런타임시에 필요에 따라 객체를 생성하면서 stack이 있는 방량으로 할당되거나 해체된다.
  + Runtime : 컴퓨터 프로그램이 실행되고 있는 동안의 동작을 말한다.
* stack은 높은 주소 번지에서 시작해서 낮은 번지로 할당되거나 해체 한다.
* 우리가 프로그램을 실행하면 os kernel에서 제어에 적합한 자료구조로 만들어 메모리로 읽어내어 동작하게 된다. ?
* Heap영역을 제외하고는 나머지 영역은 컴파일러에 의해서 결정된다. 따라서 code(text),data,stack 영역은 최소한 프로그램 실행 이전에 모든 정보를 운영체제에게 알려줘야 한다.
  + Complie-time에 data영역 및 stack영역의 크기를 계산해서 필요한 메모리 공가의 정보를 파일에 함께 갖게된다.
  + Complie : 자바소스코드에서 자바 바이트코드( .class) , jvm이 바이트코드를 읽는다)

끝으로 운영체제는 각각의 프로세스 간에 메모리 영역을 침범하지 목하도록 한다.