Thread

process

프로그램이 os로부터 메모리를 할당받아 실행된 상태

여러 프로세스가 동시에 실행되더라도 자신만으 메모리를 사용하고 있음

Thread

실제 작업(task)을 수행하는 단위

하나의 Process는 하나 이상의 thread를 가짐

Multi-Tasking

두 가지 이상의 작업(task)를 동시 처리하는 것

스레드의 생성과 실행

Thread 클래스를 상속 받아 생성

꼭 run() 메서드를 override 해줘야 함

class A extends Thread{

run(){ //override

}

}

Runnable 인터페이스를 상속 받아 생성

꼭 run() 메서드를 override 해줘야 함

class A implements Runnable{

run(){ //override

}

}

Runnable을 클래스 내부에 구현, 마찬가지로 run()메서드 override

Runnable A = new Runnable(){

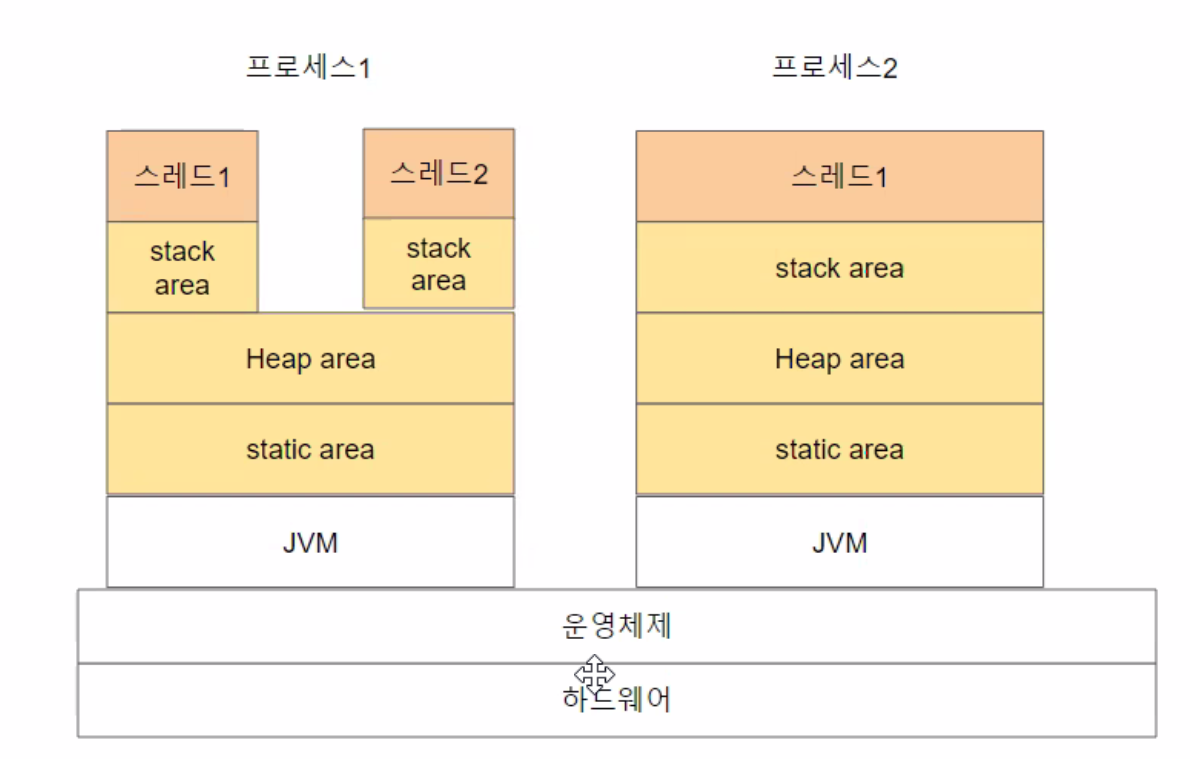
run(){ //override

}

람다식을 이용하여 구현, run() override 안 해도 됨

Runnable A = () ->{

}



Multi - Threading

여러 Thread가 동시에 수행되는 프로그래밍 : 여러 작업(task)가 동시에 수행되는

효과

Thread 각각 갖는 작업 공간(contex), 공유하는 자원(static area,Heap area) 있음

공유하는 자원에서 race condition이 발생할 수 있음

이렇게 여러 Thread가 공유하는 자원 중 경쟁이 발생하는 부분 : critical section

=> Thread 우선순위, 동기화 이용해 오류 해결해야 함

Thread 우선순위

Thread.MIN\_PRIORITY(=1) : 최솟값 1

Thread.MAX\_PRIORITY(=10) : 최댓값 10

Thread.NORM\_PRIORITY(=5) : 기본값 5

setPriority()/getPriority()

우선 순위가 높을수록 CPU에 배분될 확률이 높음 : 100퍼센트 아님

setflag(boolean flag) : flag가 true가 되면 스레드 종료

join()

동시에 두 개 이상의 Thread가 실행될 때

Thread의 결과를 참조하여 실행해야 경우 => join()함수 사용

join() 함수를 호출한 Thread가 not-Runnable 상태가 감

다른 Thread의 수행이 끝나면 runnable 상태로 돌아옴