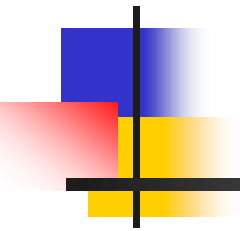


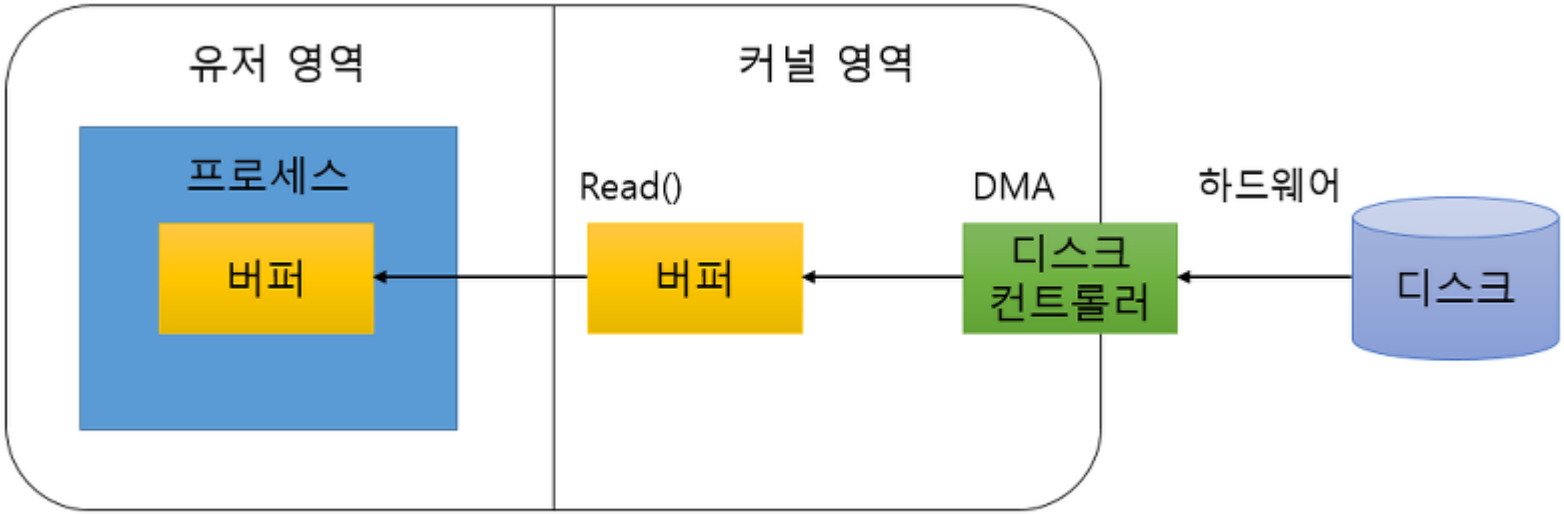
NIO(New I/O)

경복대학교
소프트웨어융합과
배 희호 교수

NIO

- Java NIO는 기존 IO Package를 개선하기 위해 나온 새로운 Package
- JAVA 4부터 새로운 입출력(New Input Output)이라는 뜻에서 java.nio Package에 포함
- JAVA 7로 Version을 올리면서 JAVA IO와 NIO 사이의 일관성 없는 클래스 설계를 바로 잡고 비동기 채널 등의 Network 지원을 대폭 강화한 NIO.2 API가 추가 되었음
- NIO.2는 java.nio의 하위 패키지(java.nio.channels, java.nio.charset, java.nio.file)에 통합되어 있음

NIO



By jdk, http://jeong-pro.tistory.com

NIO

- IO와 NIO의 차이점

구분	IO	NIO
입출력 방식	단방향 스트림 방식	양방향 채널 방식
버퍼 방식	버퍼 지원 안함 필터 스트림을 사용	버퍼 사용을 기본으로 함
데이터 전송의 기본 단위	바이트	버퍼
비동기 방식	지원 안함 (동기 방식)	지원
블로킹 / 넌블로킹 방식	블로킹 방식만 지원	블로킹 / 넌블로킹 방식 모두 지원

NIO

- IO Program 절차
  - File/Folder 주소 및 파일 처리 (File)
  - 입출력 Stream 생성
  - 입출력 실행
  - Stream 닫음
- NIO Program 절차
  - File/Folder 주소 (Path 클래스)
  - File 처리 (Files 클래스)
  - 입출력 Channel 생성 (Channel)
  - 입출력 생성 (Buffer 클래스)

NIO

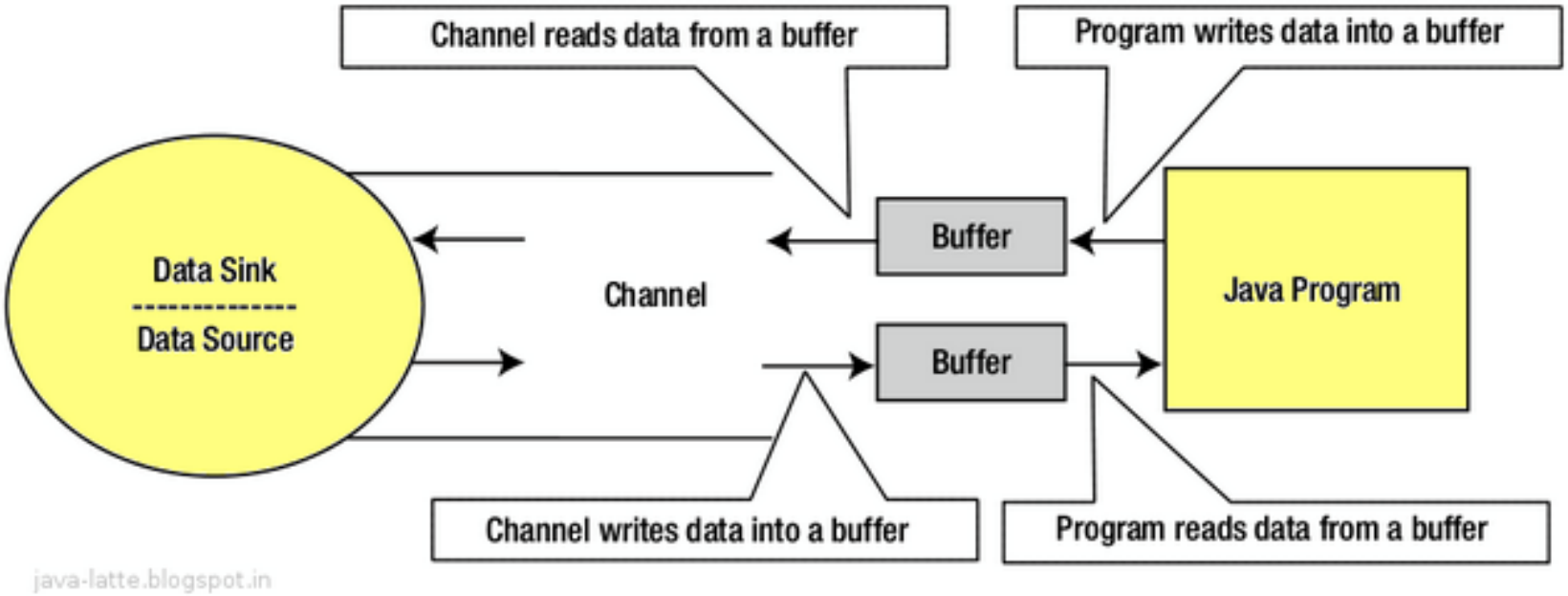
- Stream과 Channel
  - IO는 Stream 기반
    - Stream은 입력 Stream과 출력 Stream이 구분되어 있음
    - Data를 읽기 위해 입력 Stream을 생성, Data 출력 위해 출력 Stream 생성 해야함
  - NIO는 Channel 기반
    - Channel은 Stream과는 다르게 양방향 입출력 가능
    - 입력과 출력을 위한 별도의 Channel을 만들 필요 없음

NIO

- Buffer와 NonBuffer
  - IO에서는 출력 Stream이 1 Byte를 쓰면 입력 Stream이 1 Byte를 읽음
    - 이런 System은 대체로 느림
    - Buffer를 사용하여 복수 개의 Byte를 한꺼번에 입력 받고, 출력하는 것이 좋음
    - IO는 Buffer를 제공해주는 보조 Stream인 BufferedInputStream, BufferedOutputStream을 연결해서 사용하기도 함
  - NIO는 기본적으로 Buffer를 사용하여 입출력을 함
    - IO보다 성능이 좋음
    - Channel은 Buffer에 저장된 Data를 출력하고 입력된 Data를 Buffer에 저장

NIO

- Channel



Interaction between a channel, buffers, a Java program, a data source, and a data sink

NIO

- Buffer와 NonBuffer
  - IO는 Stream에서 읽은 Data를 즉시 처리하므로 Stream으로 부터 입력된 전체 Data를 별도로 저장하지 않으면 입력된 Data의 위치를 이동해 가며 자유롭게 이용할 수 없음
  - NIO는 읽은 Data를 무조건 Buffer에 저장
    - Data의 위치를 이동해 가며 필요한 부분만 읽고 쓸 수 있음

NIO

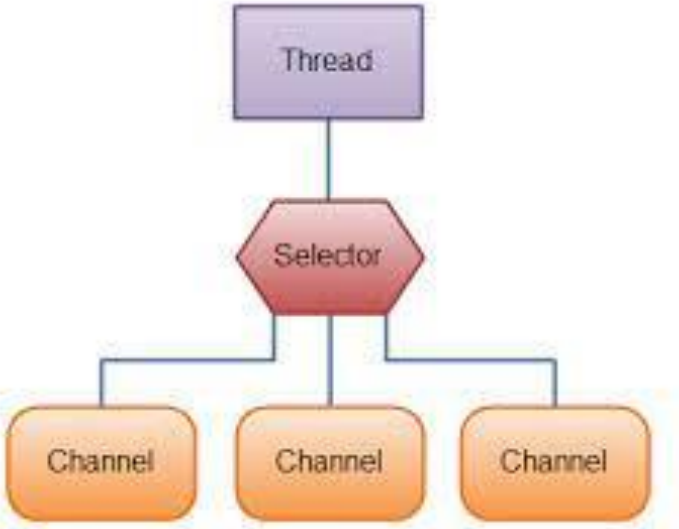
- Blocking과 NonBlocking
  - IO는 Blocking 됨
    - 입력 Stream의 read()를 호출하면 Data가 입력되기 전까지 Thread는 Blocking(대기 상태)되고 출력 Stream의 write()를 호출하면 Data 출력 전까지 Thread는 Blocking 됨
    - IO Thread가 Blocking되면 다른 일을 할 수 없고 Blocking을 빠져나오기 위해 Interrupt 할 수도 없음
    - 유일한 방법은 Stream을 닫는 것임

NIO

- Blocking과 NonBlocking
  - NIO는 Blocking과 NonBlocking 특징을 모두 가짐
    - NIO Blocking은 Thread를 Interrupt함으로써 빠져나올 수 있음
    - NonBlocking : 입출력 작업 시 Thread가 Blocking되지 않는 것
    - NIO의 NonBlocking은 입출력 작업 준비가 완료된 Channel만 선택해서 작업 Thread가 처리하므로 작업 Thread가 Blocking되지 않음
    - NIO NonBlocking의 핵심 객체는 멀티플렉서인 셀렉터(Selector)
    - Selector는 복수 개의 Channel중에서 준비 완료된 채널을 선택하는 방법을 제공

NIO

- Selector
  - Network Programming의 효율을 높이기 위한 것
  - Client 하나당 Thread 하나를 생성해서 처리하기 때문에 Thread가 많이 생성될수록 급격한 성능 저하를 가졌던 단점을 개선하는 Reactor 패턴의 구현체
  - Selector는 어느 Channel set이 IO event를 가지고 있는지를 알려줌



NIO

- IO와 NIO의 선택
  - NIO는 불특정 다수의 클라이언트 연결 or 멀티 파일들을 넌블로킹이나 비동기 처리할 수 있음
    - 과도한 스레드 생성을 피하고 스래드를 효과적으로 재사용한다는 장점이 있음
  - 운영체제의 버퍼를 이용한 입출력이 가능하므로 입출력 성능 향상
  - NIO는 연결 클라이언트 수가 많고 하나의 입출력 처리 작업이 오래 걸리지 않는 경우에 사용하는 것이 좋음
  - 스레드에서 입출력 처리가 오래 걸린다면 대기하는 작업의 수가 늘어나므로 제한된 스레드로 처리하는 것이 불편할 수 있음

NIO

- IO와 NIO의 선택
  - 대용량의 데이터 처리의 경우 IO가 좋음
  - NIO는 버퍼 할당 크기가 문제가 되고, 모든 입출력 작업에 버퍼를 무조건 사용해야 하므로 즉시 처리하는 IO보다 조금 더 복잡
  - 연결 클라이언트 수가 적고 전송되는 데이터가 대용량이면서 순차적으로 처리될 필요성이 있는 경우 IO로 서버를 구현하는 것이 좋음

NIO

- NIO에서 제공하는 패키지

NIO 패키지	포함되어 있는 내용
java.nio	다양한 버퍼 클래스
java.nio.channels	파일 채널, TCP 채널, UDP 채널 등의 클래스
java.nio.channels.spi	java.nio.channels 패키지를 위한 서비스 제공자 클래스
java.nio.charset	문자셋, 인코더, 디코더 API
java.nio.charset.spi	java.nio.charset 패키지를 위한 서비스 제공자 클래스
java.nio.file	파일 및 파일 시스템에 접근하기 위한 클래스
java.nio.file.attribute	파일 및 파일 시스템의 속성에 접근하기 위한 클래스
java.nio.file.spi	java.nio.file 패키지를 위한 서비스 제공자 클래스