JDK 7 출시 기념 (2011.7)
JDK 7 소개
#3 NIO2

김용환

knight76.tistory.com

Knight76 at gmail.com

NIO2

서론

Java SE 7 에 추가된 것

- http://download.oracle.com/javase/7/docs/technotes/guides/io/enhancements.html#7
- http://java.sun.com/developer/technicalArticles/javase/nio/
- http://www.oracle.com/us/technologies/java/file-system-api-428204.pdf
- File i/o파일 시스템에 대한 대안 제시
  java.nio.file 패키지,
  java.nio.file.attribute 패키지

Specification

- JSR 203
  - http://jcp.org/en/jsr/detail?id=203

JSRs: Java Specification Requests

JSR 203: More New I/O APIs for the Java™ Platform ("NIO.2")

Stage	Access	Start
Final Approval Ballot	View results	05 Jul, 201
Proposed Final Draft	Download page	17 Jun, 201
Public Review Ballot	View results	12 Apr, 201
Public Review	Download page	16 Feb, 20
Early Draft Review	Download page	12 Apr, 200
Expert Group Formation		22 Jan, 200
JSR Review Ballot	View results	07 Jan, 200

Status: In Progress

배경

- 기존의 API는 확정성을 염두한 것 아님
- Java.io.File 이슈
  - 현재는 메소드 호출시 fail이 일어날 때 exception 대신 boolean 변수를 return하는 구조임. 왜 문제가 생겼는지 알 길이 없었음 (File의 public boolean delete메소드)
  - Symbolic link, permission, attribute, 메모리 파일 통제 불가능
  - 안 좋은 성능 디자인이 있었음. Last modified time 또는 file의 type 정보를 얻어올 때 파일을 여러 개의 method가 호출됨
  - renameTo() 메소드에 문제 발생이 많았음
- 파일을 다루면서 좀더 더 좋은 기능을 넣고 싶음
- 내용상으로는 보완재가 아닌 완전 대체재의 개념
  - 그러나 기존 java.io.File은 deprecated된 것은 많지 않음 (하위 호환성)

주의

- Oracle jdk 싸이트에서 제공하지 아닌 것은 믿지 말것!!! (오직 oracle jdk!)
  - Api 설명이 다른 것들이 많음
  - 바뀐 oracle jdk7 api 설명이 반영되지 않은 인터넷 자료들이 많음
- Oracle Java Tutorial 정보도 틀린 것이 있음 (2011.7.20현재)
  - Path와 Files API ..

Concept

- Path
  - 파일시스템 안에 있는 파일
- Files
  - 파일과 디렉토리에 대한 static 메소드 지원
- FileSystem
  - 파일 시스템에 대한 handle
- FileStore
  - 볼륨, 실제 파일시스템에 대한 스토리지 정보

Path 및 File 다루기

File(java6이하)-> Path(java7)

Java6 이하

```
java.io.File file = new java.io.File("c:\\test\\1.png");
if (!file.delete()){
    System.out.println("???");
}
```

Java7

```
Path path = Paths.get("C:\\test\\.\\1");

try {
    path.delete();
} catch (IOException e) {
    //log.error(e, e);
}
```

Path : String, URI, java.io.File을 통해서 생성
File.toPath() : File-> Path
Path.toFile() : Path -> File

New API

- Java.nio.file
- Java.nio.attribute
- Java.nio.file.spi

java.nio.file

Interfaces
CopyOption
DirectoryStream
DirectoryStream.Filter
FileVisitor
OpenOption
Path
PathMatcher
SecureDirectoryStream
Watchable
WatchEvent
WatchEvent.Kind
WatchEvent.Modifier
WatchKey
WatchService

Classes
Files
FileStore
FileSystem
FileSystems
LinkPermission
Paths
SimpleFileVisitor
StandardWatchEventKin

Enums
AccessMode
FileVisitOption
FileVisitResult
LinkOption
StandardCopyOption
StandardOpenOption

java.nio.file.attribute

Interfaces
AclFileAttributeView
AttributeView
BasicFileAttributes
BasicFileAttributeView
DosFileAttributes
DosFileAttributeView
FileAttribute
FileAttributeView
FileOwnerAttributeView
FileStoreAttributeView
GroupPrincipal
PosixFileAttributes
PosixFileAttributeView
UserDefinedFileAttributeView
UserPrincipal

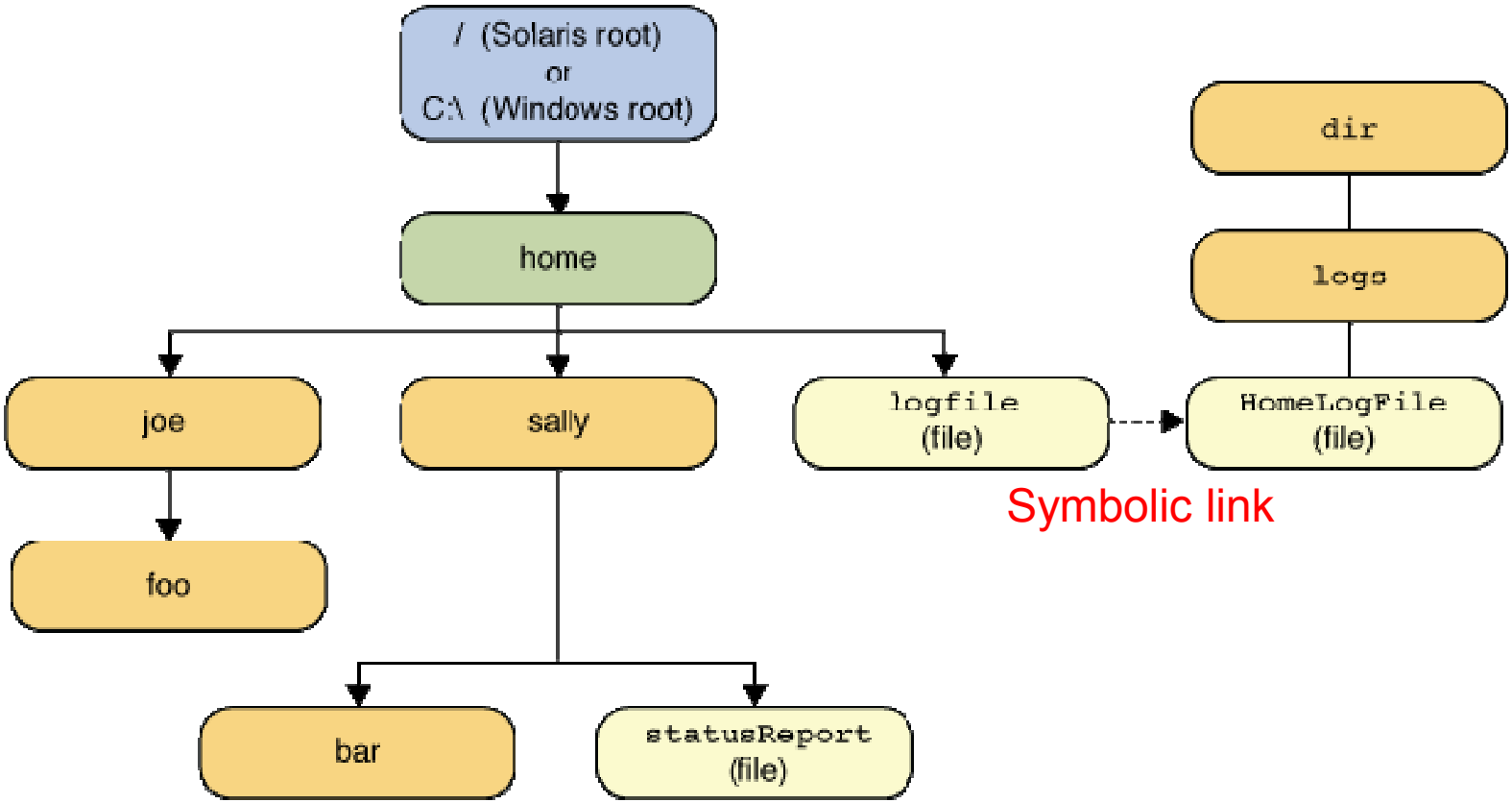
Classes
AclEntry
AclEntry.Builder
FileTime
PosixFilePermissions
UserPrincipalLookupService

Enums
AclEntryFlag
AclEntryPermission
AclEntryType
PosixFilePermission

java.nio.file.spi

Classes
FileSystemProvider
FileTypeDetector

Path



-Symbolic link : reference에 대한 특수한 파일
-Original file : /home/logfile
-Target file : dir/logs/HomeLogFile
-Resolving a link : symbolic link가 가르키는 실제 파일 (logfile 은 dir/log/HomeLogFile을 의미)

Path

코드

```
Path path = Paths.get("C:\\test\\1");
System.out.format("toString: %s%n", path.toString());
System.out.format("getFileName: %s%n", path.getFileName());
System.out.format("getName(0): %s%n", path.getName(0));
System.out.format("getNameCount: %d%n", path.getNameCount());
System.out.format("subpath(0,2): %s%n", path.subpath(0,2));
System.out.format("getParent: %s%n", path.getParent());
System.out.format("getRoot: %s%n", path.getRoot());
System.out.format("getFileSystem : %s%n", path.getFileSystem());
System.out.format("getFileSystem : %s%n", path.isAbsolute());
System.out.format("%s%n", path.toUri());
```

결과

```
toString: C:\test\1
getFileName: 1
getName(0): test
getNameCount: 2
subpath(0,2): test\1
getParent: C:\test
getRoot: C:\
getFileSystem : sun.nio.fs.WindowsFileSystem@cf829d
getFileSystem : true
file:///C:/test/1/
```

Files

Methods	Comment
size(Path)	Returns the size of the specified file in bytes.
isDirectory(Path, LinkOption)	Returns true if the specified Path locates a file that is a directory.
isRegularFile(Path, LinkOption...)	Returns true if the specified Path locates a file that is a regular file.
isSymbolicLink(Path)	Returns true if the specified Path locates a file that is a symbolic link.
isHidden(Path)	Returns true if the specified Path locates a file that is considered hidden by the file system.
getLastModifiedTime(Path, LinkOption...) setLastModifiedTime(Path, FileTime)	Returns or sets the specified file's last modified time.
getOwner(Path, LinkOption...) setOwner(Path, UserPrincipal)	Returns or sets the owner of the file.
getPosixFilePermissions(Path, LinkOption...) setPosixFilePermissions(Path, Set<PosixFilePermission>)	Returns or sets a file's POSIX file permissions.
getAttribute(Path, String, LinkOption...) setAttribute(Path, String, Object, LinkOption...)	Returns or sets the value of a file attribute

개념 구분

- Path : 파일이나 디렉토리 자체
- Paths : Path를 만들어냄
- Files : 파일 처리 (복사, 지움, 수정 등등)

java.nio.file.Path

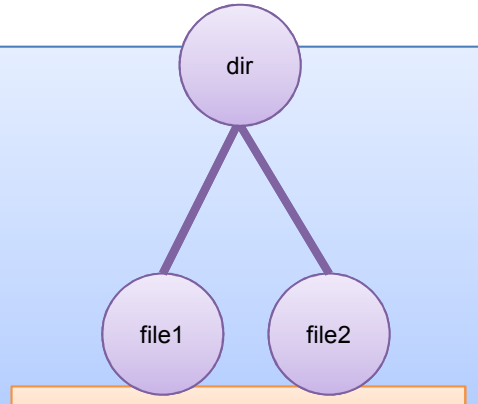
```
Path path = Paths.get("C:\\test\\.\\1");
System.out.format("normalize: %s%n", path.normalize());
```

getFileSystem : C:\test\1

```
Path p1 = Paths.get("C:\\home\\joe\\foo");
System.out.format("%s%n", p1.resolve("bar"));
```

C:\home\joe\foo\bar

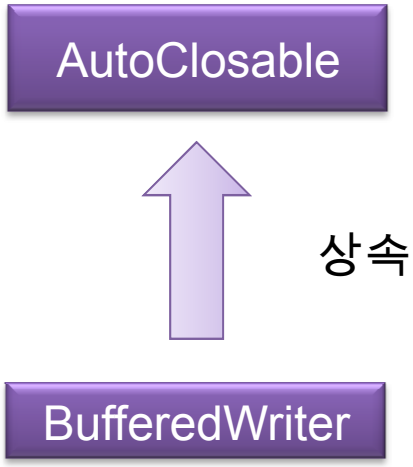
```
Path p1 = Paths.get("file1");
Path p2 = Paths.get("file2");

Path p1_to_p2 = p1.relativize(p2);
Path p2_to_p1 = p2.relativize(p1);

System.out.format("%s%n", p1_to_p2);
System.out.format("%s%n", p2_to_p1);
```

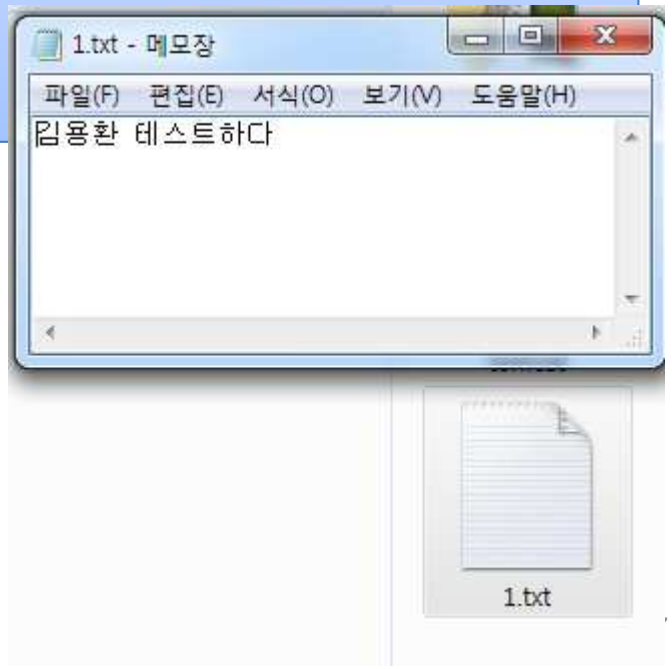


..\file2
..\file1

try-with-resources

```
Path file = Paths.get("c:\\test\\1.txt");
Charset charset = Charset.forName("UTF8");
String s = "김용환 테스트하다";
try (BufferedWriter writer = Files.newBufferedWriter(file, charset)) {
    writer.write(s, 0, s.length());
} catch (IOException x) {
    System.err.format("IOException: %s%n", x);
}
```





Files.move

```
// file move
Path source = Paths.get("c:\\test\\1.txt");
Path target = Paths.get("c:\\test\\2.txt");
Files.move(source, target, REPLACE_EXISTING, ATOMIC_MOVE);
```

public static Path move(Path source, Path target, CopyOption... options)

- Copy도 비슷

Atomic operation이 가능

Enum Constant Detail

REPLACE_EXISTING

public static final StandardCopyOption REPLACE_EXISTING

Replace an existing file if it exists.

COPY_ATTRIBUTES

public static final StandardCopyOption COPY_ATTRIBUTES

Copy attributes to the new file.

ATOMIC_MOVE

public static final StandardCopyOption ATOMIC_MOVE

Move the file as an atomic file system operation.

Files.delete

```
Path path = Paths.get("c:\\test\\21.txt");
try {
    Files.delete(path);
} catch (NoSuchFileException x) {
    System.err.format("%s: no such file or directory%n", path);
} catch (DirectoryNotEmptyException x) {
    System.err.format("%s not empty%n", path);
} catch (IOException x) {
    //File permission problems are caught here.
    System.err.println(x);
}
```

'파일이 있으면 지워라' 메소드도 존재

static boolean	**deleteIfExists**(Path path)

Method Chaining

```
// like String Buffer append method
StringBuffer sb = new StringBuffer();
sb.append("Hello ")
  .append("  ")
  .append("! Welcome!");

String value = Charset.defaultCharset()
                      .decode(buf)
                        .toString();

Path file = Paths.get("c:\\test\\1.txt");
UserPrincipal group = file.getFileSystem()
                          .getUserPrincipalLookupService()
                          .lookupPrincipalByName("me");
```

Glob

- Glob?
  - 일종의 pattern matching 방법
  - 문법
    - http://openjdk.java.net/projects/nio/javadoc/java/nio/file/FileSystem.html#getPathMatcher(java.lang.String)
    - *.java -> *.java
    - *.{java,class} -> *.java, *.class
    - C:* -> c:\ 드라이브의 모든 파일 시스템
    - /home/**
    - Test.? -> Test.a, Test.b
    - [a-z]
    - *[0-9]*
  - 예전에 Apache ORO(http://jakarta.apache.org/oro/)에서 text 검색을 위해서 Perl5, AWK, glob 등을 처리했음 (2010년 9월 retired됨)

Glob

코드

```
Path file1 = Paths.get("Test1.java");
Path file2 = Paths.get("Test2.text");

PathMatcher matcher =
        FileSystems.getDefault().getPathMatcher("glob:*.{java,class}");

System.out.println(file1.toString() + " : " + matcher.matches(file1));
System.out.println(file2.toString() + " : " +matcher.matches(file2));
```

결과

```
Test1.java : true
Test2.text : false
```

기존 사용 regular expression은 다음과 같이 사용

FileSystems.getDefault().getPathMatcher("regex:…..");

Checking File Accessibility

코드

```
Path file = Paths.get("c:\\test\\2.txt");
boolean isRegularExecutableFile = Files.isRegularFile(file) &
                                  Files.isReadable(file) &
                                  Files.isExecutable(file);
System.out.print(isRegularExecutableFile);
```

결과

```
true
```

Checking Same File

코드

```
Path p1 = Paths.get("c:\\test\\2.txt");
Path p2 = Paths.get("c:\\test\\a.class");

if (Files.isSameFile(p1, p2)) {
    System.out.print("same");
} else {
    System.out.print("not same");
}
```

결과

```
not same
```

Attributes

```
Path file = ...;

// modified time 변경
Files.setLastModifiedTime(file, ft);

// set attribute (View이름:속성이름), 이 정보가 없으면 “basic”에 저장
Files.setAttribute(file, "dos:hidden", true);

// read attribute
DosFileAttributes attr = Files.readAttributes(file, DosFileAttributes.class);
System.out.println("isReadOnly is " + attr.isReadOnly());
```

Attributes

- 특정 파일시스템에 맞는 Attribute을 지정 (예, Posix)

```
PosixFileAttributes attrs = Files.readAttributes(path, PosixFileAttributes.class);


UserPrincipal owner = attrs.owner();
UserPrincipal group = attrs.group();
Set<PosixFilePermission> perms = attrs.permissions();


Files.createFile(newPath,
PosixFilePermissions.asFileAttribute(perms));
```

User Defined Attributes

- NTFS Alternative Data Stream, EXT3, ZFS와 같이 사용자가 지정할 수 있는 파일 시스템을 위함

```
Path file = ...;
UserDefinedFileAttributeView view =
        Files.getFileAttributeView(file, UserDefinedFileAttributeView.class);
view.write("user.mimetype", Charset.defaultCharset().encode("text/html");
```

ContentType

코드

```
Path dir = Paths.get("c:\\test\\2.JPG");
String type = Files.probeContentType(dir);
System.out.format("'%s' filetype.%n", type);
// custom한 타입을 추가하기 위해서는
// java.nio.file.spi.FileTypeDetector 이용
```

결과

```
'image/jpeg' filetype.
```

Default File System & Path 구분자

코드

```
PathMatcher matcher =
FileSystems.getDefault().getPathMatcher("glob:*.*");
```

코드

```
String separator = File.separator;  (posix : /, window : \)
String separator = FileSystems.getDefault().getSeparator();
```

FileStore

코드

```
Path path = Paths.get("c:\\test");
FileStore store = Files.getFileStore(path);

long total = store.getTotalSpace() / (1024 * 1024);
long used = (store.getTotalSpace() - store.getUnallocatedSpace()) /
(1024 * 1024);
long avail = store.getUsableSpace() / (1024 * 1024);

System.out.println("total (mb): " + total);
System.out.println("used (mb): " + used);
System.out.println("avail (mb): " + avail);
```

결과

```
total (mb): 76316
used (mb): 52939
avail (mb): 23377
```

FileSystem

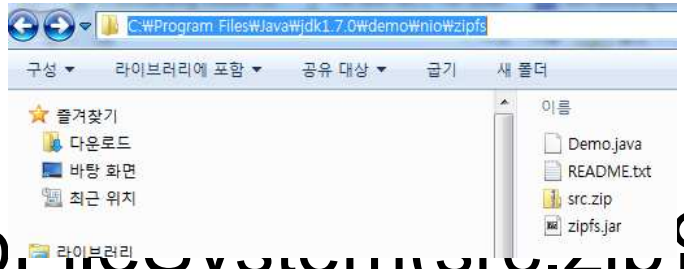
- 특정 OS나 플랫폼에 맞는 FileSystem 이용/직접 개발 가능
  - 메모리 기반 FileSystem
  - Desktop FileSystem
  - Hadoop FileSystem
- java.nio.file.spi의 FileSystemProvider abstract 클래스을 이용 (Factory)

```
FileSystem fs = FilesSystems.getDefault();
```

```
class CustomFileSystem extends FileSystem {
..
}
```

NIO.2 filesystem provider for zip/jar archives

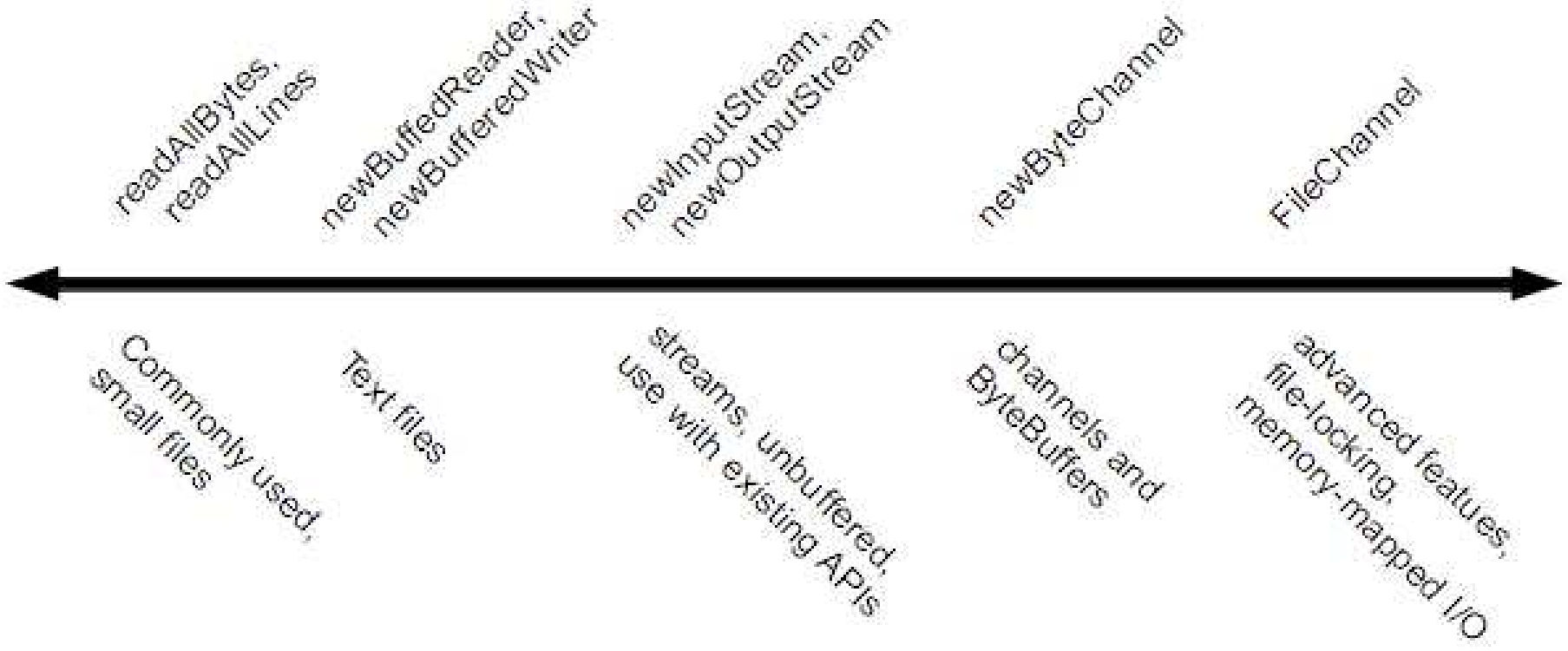
- Demo
  - C:\Program Files\Java\jdk1.7.0\demo\nio\zipfs



- Zip.FileSystem(src.zip안에 있음)은 Zip또는 Jar 파일을 java.nio.file.FileSystem 으로 사용 가능

```
Path jarfile = Paths.get("foo.jar");
FileSystem fs = FileSystems.newFileSystem(jarfile, null);
Path mf = fs.getPath("/META-INF/MANIFEST.MF");
InputStream in = mf.newInputStream();
```

Details of reading, writing, creating, and opening files



Commonly Used Methods for Small Files

코드

```
//write
Path file = Paths.get("c:\\test\\created.txt");
String s = "DEADBEEF";
byte[] b = s.getBytes();
Files.write(file, b);

//read
byte[] fileArray;
fileArray = Files.readAllBytes(file);
System.out.print(new String(fileArray));
```

결과

```
DEADBEEF
```

Buffered I/O Methods for Text Files

코드

```
Path file = Paths.get("c:\\test\\created.txt");
String s = "DEADBEEF";
Charset charset = Charset.forName("UTF-8");
//write
try (BufferedWriter writer = Files.newBufferedWriter(file, charset)) {
    writer.write(s, 0, s.length());
} catch (IOException x) {
    System.err.format("IOException: %s%n", x);
}
//read
try (BufferedReader reader = Files.newBufferedReader(file, charset)) {
    String line = null;
    while ((line = reader.readLine()) != null) {
        System.out.println(line);
    }
} catch (IOException x) {
    System.err.format("IOException: %s%n", x);
}
```

결과

```
DEADBEEF
```

Methods for buffered Streams

코드

```
import static java.nio.file.StandardOpenOption.*;
Path file = Paths.get("c:\\test\\created.txt");
String s = "DEADBEEF";
byte data[] = s.getBytes();
//write
try (OutputStream out = new BufferedOutputStream(
     Files.newOutputStream(file, CREATE, CREATE_NEW))) {
     out.write(data, 0, data.length);
} catch (IOException x) {
    System.err.println(x);
}
//read
try (InputStream in = Files.newInputStream(file);
            BufferedReader reader = new BufferedReader(new InputStreamReader(in))) {
    String line = null;
    while ((line = reader.readLine()) != null) {
        System.out.println(line);
    }
} catch (IOException x) {
    System.err.println(x);
}
```

결과

```
DEADBEEF
```

Methods for Channels and ByteBuffers

코드

```
import static java.nio.file.StandardOpenOption.*;

Path file = Paths.get("c:\\test\\created.txt");
String s = "DEADBEEF";
//write
Set<StandardOpenOption> options = new HashSet<StandardOpenOption>();
options.add(APPEND);
options.add(CREATE);
byte data[] = s.getBytes();
ByteBuffer bb = ByteBuffer.wrap(data);
try (SeekableByteChannel sbc = Files.newByteChannel(file, options)) {
    sbc.write(bb);
} catch (IOException x) {
    System.out.println("exception thrown: " + x);
}
//read
try (SeekableByteChannel sbc = Files.newByteChannel(file)) {
    ByteBuffer buf = ByteBuffer.allocate(10);
    String encoding = System.getProperty("file.encoding");
    while (sbc.read(buf) > 0) {
        buf.rewind();
        System.out.print(Charset.forName(encoding).decode(buf));
        buf.flip();
    }
} catch (IOException x) {
    System.out.println("caught exception: " + x);
}
```

결과

```
DEADBEE
F
```

Methods for File Channel

코드

```
Path file = Paths.get("c:\\test\\created.txt");
String s = "DEADBEEF";
byte data[] = s.getBytes();
ByteBuffer buffer = ByteBuffer.wrap(data);
//write
try (AsynchronousFileChannel fileChannel =
    AsynchronousFileChannel.open(file, StandardOpenOption.CREATE,
                                 StandardOpenOption.TRUNCATE_EXISTING,
                                 StandardOpenOption.WRITE)) {
    fileChannel.write(buffer, 0);
    fileChannel.close();
} catch (IOException e) {
    System.out.println("exception thrown: " + e);
}
//read
try (AsynchronousFileChannel fileChannel = AsynchronousFileChannel.open(file)) {
    ByteBuffer buf = ByteBuffer.allocate(100);
    java.util.concurrent.Future<Integer> pendingResult = fileChannel.read(buf, 0);
    pendingResult.get();
    buf.rewind();
    if (pendingResult.isDone()) {
        byte[] bytes = new byte[buf.position()];
        buf.get(bytes);
        String encoding = System.getProperty("file.encoding");
        System.out.print(Charset.forName(encoding).decode(buf));
    }
} catch (IOException e) {
    System.out.println("exception thrown: " + e);
}
```

결과

```
DEADBEEF
```

Methods for Creating Regular and Temporary Files

코드

```
try {
    Path tempFile = Files.createTempFile(null, ".temp");
    System.out.format("The temporary file has been created: %s%n",
tempFile);
} catch (IOException x) {
    System.err.format("IOException: %s%n", x);
}
```

결과

```
The temporary file has been created:
C:\Users\knight\AppData\Local\Temp\647926783459236273 4.temp
```

기타

```
Path newLink = ...;
Path target = ...;

// 심볼릭 링크 만들기
Files.createSymbolicLink(newLink, target);

// detecting
boolean isSymbolicLink = Files.isSymbolicLink(newLink);

// 원 파일 찾기
System.out.format("Target of link '%s' is '%s'%n", link,
Files.readSymbolicLink(newLink));

// 하드링크 만들기
Files.createLink(newLink, existingFile);
```

디렉토리 다루기

Getting Root Directories

코드

```
Iterable<Path> dirs = FileSystems.getDefault().getRootDirectories();
for (Path name: dirs) {
    System.err.println(name);
}
```

결과

```
C:\
D:\
E:\
F:\
```

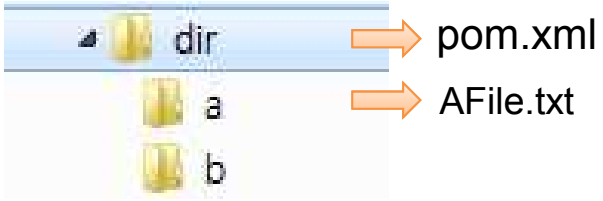
Getting Root Directories

```
// create & delete
Files.createDirectories(Paths.get("c:\\test\\test"));
Files.delete(Paths.get("c:\\test\\test"));

// list
Path dir = Paths.get("c:\\test\\dir");
try (DirectoryStream<Path> stream = Files.newDirectoryStream(dir)) {
    for (Path file: stream) {
        System.out.println(file.getFileName());
    }
} catch (IOException | DirectoryIteratorException x) {
    System.err.println(x);
}

// filter
try (DirectoryStream<Path> stream = Files.newDirectoryStream(dir,
"*.{xml,class,jar}")) { // glob
    for (Path entry: stream) {
        System.out.println(entry.getFileName());
    }
} catch (IOException x) {
    System.err.println(x);
}
```

Walking File Tree

```
Path dir = Paths.get("c:\\test\\dir");
PrintFiles pf = new PrintFiles();
Files.walkFileTree(dir, pf);

class PrintFiles extends SimpleFileVisitor<Path> {
    public FileVisitResult visitFile(Path file, BasicFileAttributes attr) {
        System.out.println( "File :" + file.getFileName());
        return CONTINUE;
    }

    public FileVisitResult postVisitDirectory(Path dir, IOException ioe) throws
IOException {
        System.out.printf("Post visit directory: %s\n", dir.getFileName());
        return CONTINUE;
    }

    public FileVisitResult preVisitDirectory(Path dir, BasicFileAttributes bfa) throws
IOException {
        System.out.printf("Pre visit directory: %s\n", dir.getFileName() );
        return CONTINUE;
    }
}
```

Walking File Tree - 결과




```
Pre visit directory: dir
Pre visit directory: a
File :AFile.txt
Post visit directory: a
Pre visit directory: b
Post visit directory: b
File :pom.xml
Post visit directory: dir
```

Files.walkFileTree

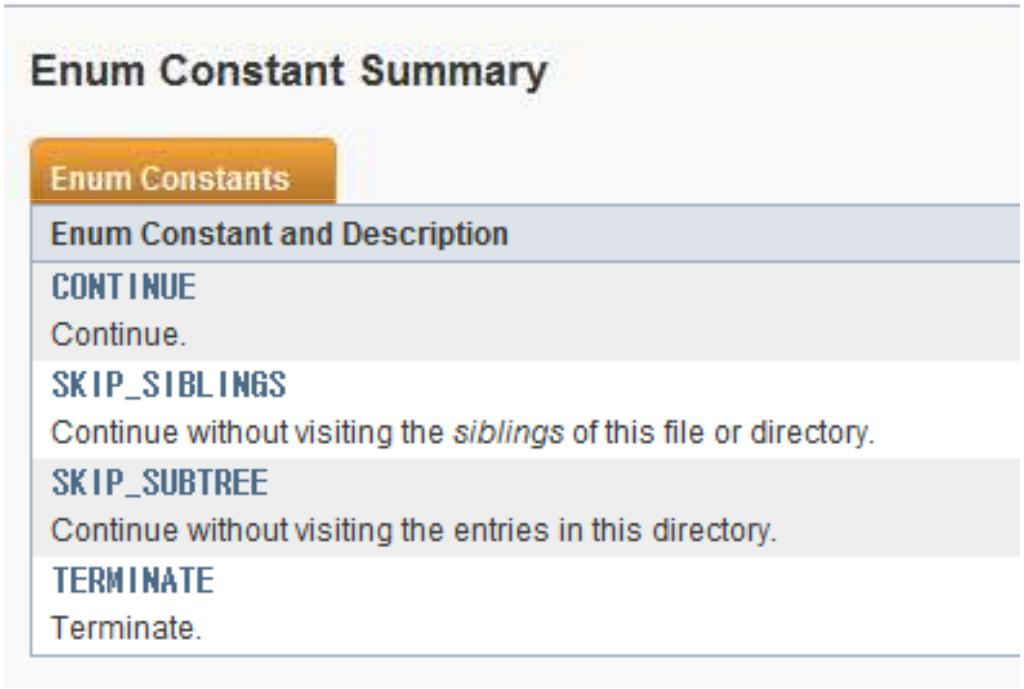
- 주어진 Path를 기준으로 file tree를 iterate
- 파일이나 디렉토리를 만나면 FileVisitor를 호출 (invoke)
- 깊이 우선(Depth First)
- 각 디렉토리당 pre/post 가 각각 호출된다
- 디폴트로 심볼링 링크는 제외

Methods

Modifier and Type	Method and Description
`FileVisitResult`	`postVisitDirectory(T dir, IOException exc)` Invoked for a directory after entries in the directory, and all of their descendants, have been visited.
`FileVisitResult`	`preVisitDirectory(T dir, BasicFileAttributes attrs)` Invoked for a directory before entries in the directory are visited.
`FileVisitResult`	`visitFile(T file, BasicFileAttributes attrs)` Invoked for a file in a directory.
`FileVisitResult`	`visitFileFailed(T file, IOException exc)` Invoked for a file that could not be visited.

FileVistResult

- 메소드 결과



Enum Constant Summary

Enum Constants	
Enum Constant and Description	
CONTINUE	Continue.
SKIP_SIBLINGS	Continue without visiting the *siblings* of this file or directory.
SKIP_SUBTREE	Continue without visiting the entries in this directory.
TERMINATE	Terminate.

Files.walkFileTree

- 제한을 걸어서 walkFileTree를 사용가능
- 만약 심볼링 링크가 되면 잘못해서 cycle이 될 수 있는 상황에 대해서 대비 가능

```
import static java.nio.file.FileVisitResult.*;

Path startingDir = ...;
// 심볼링 링크도 가능하게 함
EnumSet<FileVisitOption> opts = EnumSet.of(FOLLOW_LINKS);

Finder finder = new Finder(pattern);
Files.walkFileTree(startingDir, opts, Integer.MAX_VALUE, finder);
```

```
FileVistor interface 를 상속받은 클래스
public FileVisitResult visitFileFailed(Path file, IOException exc) {
    if (exc instanceof FileSystemLoopException) {
        System.err.println("cycle detected: " + file);
    } else {
        System.err.format("Unable to copy: %s: %s%n", file, exc);
    }
    return CONTINUE;
}
```

파일 변경에 대해서 Event 받기

WatchService

```
import static java.nio.file.StandardWatchEventKinds.*;

Path dir = Paths.get("c:\\test\\");
WatchService watcher = FileSystems.getDefault().newWatchService();
WatchKey key = dir.register(watcher, ENTRY_CREATE, ENTRY_DELETE,
                            ENTRY_MODIFY);

for (;;) {
        key = watcher.take();
        for (WatchEvent<?> event : key.pollEvents()) {
                if (event.kind() == ENTRY_CREATE) {
                        Path name = (Path) event.context();
                        System.out.format("%s created%n", name);
                } else if (event.kind() == ENTRY_DELETE) {
                        Path name = (Path) event.context();
                        System.out.format("%s deleted%n", name);
                } else if (event.kind() == ENTRY_MODIFY) {
                        Path name = (Path) event.context();
                        System.out.format("%s modified%n", name);
                }
        }
        key.reset();
}
```

WatchService

<윈도우>
c:\\test 디렉토리 안에서 '새폴더' 생성->
이름을 '김용환'으로 변경 -> 이름을 '김용환-1'로 변경

```
새 폴더 created
새 폴더 deleted
김용환 created
김용환 deleted
김용환-1 created
```

<윈도우>
c:\\test 디렉토리 안에서 '2.txt' 라는 파일의 내용을 수정
-> '2.txt' 파일 삭제

```
2.txt modified
2.txt deleted
```

WatchService

- Event를 받을 Watch를 등록
- Create/delete/modify 되면 event가 전달되게 함
- 디렉토리에서만 사용가능, 파일은 안됨
  - Exception in thread "main" java.nio.file.NotDirectoryException: c:\test\2.txt
- RC 버전에 WatchService 관련해서 버그발생
  - WatchEvent.context() 의 Path Instance의 절대위치가 Java 를 실행시키는 현재 디렉토리 기준으로 된 파일이 나옴. 기대치는 주어진 Path
  - Oracle Bugzilla에 올림

WatchService

```
..          for (WatchEvent<?> event : key.pollEvents()) {
                    if (event.kind() == ENTRY_CREATE) {
                            Path name = (Path) event.context();
                        System.out.println("name : " + name.toAbsolutePath());
                            System.out.format("%s created%n", name);
                    } else if (event.kind() == ENTRY_DELETE) {
                            Path name = (Path) event.context();
                        System.out.println("name : " + name.toAbsolutePath());
                            System.out.format("%s deleted%n", name);
                    } else if (event.kind() == ENTRY_MODIFY) {
                            Path name = (Path) event.context();
                        System.out.println("name : " + name.toAbsolutePath());
                            System.out.format("%s modified%n", name);
                    }
            }
…
```

C:\test\ 디렉토리에
why 디렉토리생성했더니??

name : e:\jdk1.7\test\bin\why
why created

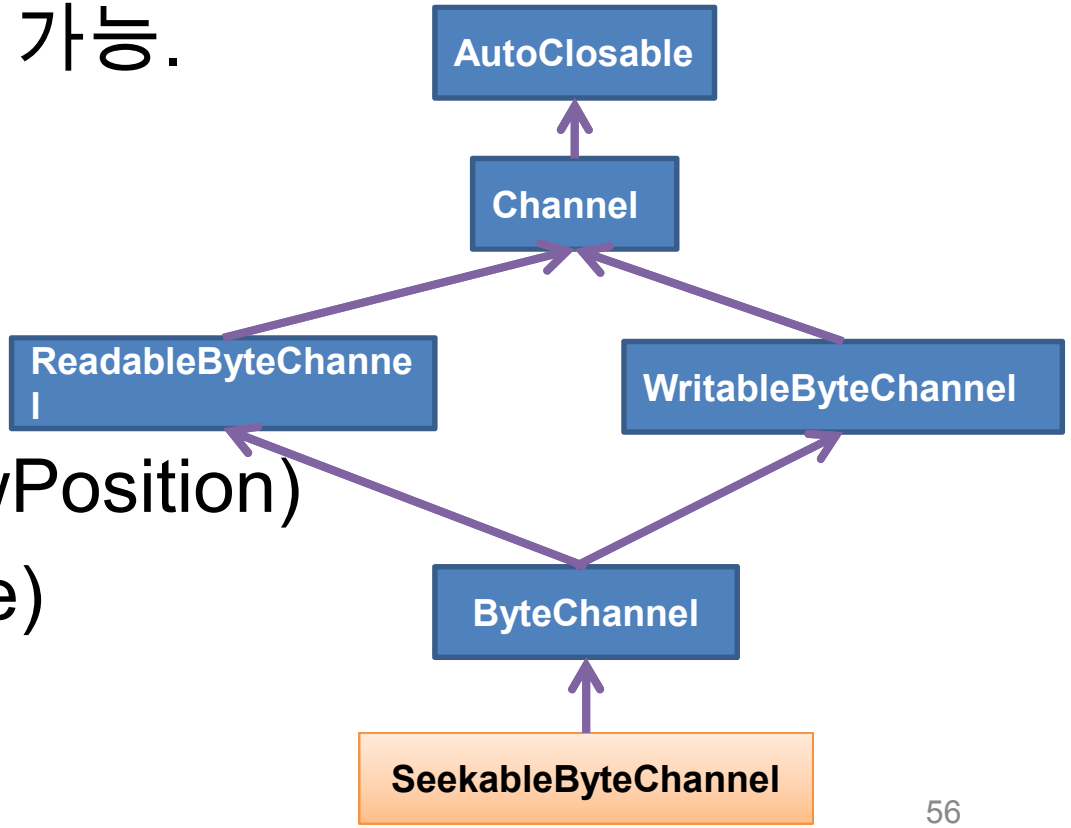
새로운 API

New API

- SeekableByteChannel (random access)
- MulticastChannel
- NetworkChannel
- Asynchronous I/O

SeekableByteChannel

- Random access
  - 현재 position을 유지하며, 이 position 을 기준으로 읽고 쓰기가 가능.
- Method
  - long position()
  - long size()
  - position(long newPosition)
  - truncate(long size)



SeekableByteChannel

코드

```
import static java.nio.file.StandardOpenOption.*;

Path file = Paths.get("c:\\test\\created.txt");
String s = "DEADBEEF";
//write
Set<StandardOpenOption> options = new HashSet<StandardOpenOption>();
options.add(APPEND);
options.add(CREATE);
byte data[] = s.getBytes();
ByteBuffer bb = ByteBuffer.wrap(data);
try (SeekableByteChannel sbc = Files.newByteChannel(file, options)) {
    sbc.write(bb);
} catch (IOException x) {
    System.out.println("exception thrown: " + x);
}
//read
try (SeekableByteChannel sbc = Files.newByteChannel(file)) {
    ByteBuffer buf = ByteBuffer.allocate(10);
    String encoding = System.getProperty("file.encoding");
    while (sbc.read(buf) > 0) {
        buf.rewind();
        System.out.print(Charset.forName(encoding).decode(buf));
        buf.flip();
    }
} catch (IOException x) {
    System.out.println("caught exception: " + x);
}
```

결과

DEADBEEF

NetworkChannel

- Network 소켓, binding, socket option을 셋팅하고, local address를 알려주는 interface
- AsynchronousServerSocketChannel, AsynchronousSocketChannel, DatagramChannel, ServerSocketChannel, SocketChannel 상속받음

Methods

Modifier and Type	Method and Description
NetworkChannel	bind(SocketAddress local) Binds the channel's socket to a local address.
SocketAddress	getLocalAddress() Returns the socket address that this channel's socket is bound to, or null if the socket is not bound.
<T> T	getOption(SocketOption<T> name) Returns the value of a socket option.
<T> NetworkChannel	setOption(SocketOption<T> name, T value) Sets the value of a socket option.
Set<SocketOption<?>>	supportedOptions() Returns a set of the socket options supported by this channel.

MulticastChannel

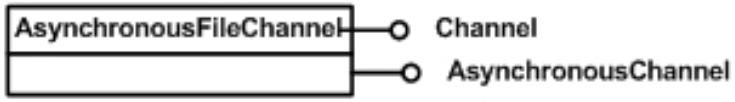
- IP multicasting(ip group단위로 IP datagram을 주고 받음) 을 지원하는 network channel
- NetworkChannel을 상속하였고, DatagramChannel이 상속 받음

Methods

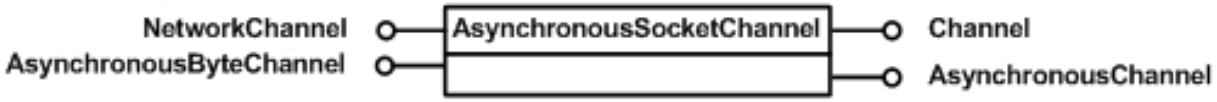
Modifier and Type	Method and Description
void	**close**() Closes this channel.
MembershipKey	**join**(**InetAddress** group, **NetworkInterface** interf) Joins a multicast group to begin receiving all datagrams sent to the group, returning a membership key.
MembershipKey	**join**(**InetAddress** group, **NetworkInterface** interf, **InetAddress** source) Joins a multicast group to begin receiving datagrams sent to the group from a given source address.

상속

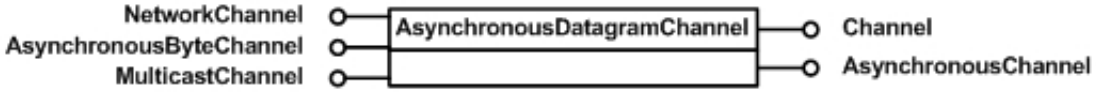
- AsynchronousByteChannel, AsynchronousFileChannel



- AsynchronousSocketChannel



- AsynchronousServerSocketChannel



Asynchronous I/O

- Mina like API
- Socket과 파일에 대해서 Asynch 작업을 가능케 함
  - java.util.concurrent.Future를 리턴
  - completionHandler를 이용

AsynchronousFileChannel

```
AsynchronousFileChannel channel =
                    AsynchronousFileChannel.open(Paths.get("c:\\test\2.txt"));
ByteBuffer buffer = ByteBuffer.allocate(1024);
```

1

```
Future result = channel.read(buffer, 100);
boolean done = result.isDone();
int bytesRead = result.get();
```

2

```
Future result = channel.read(buffer, 100);
boolean done = result.isDone();
int bytesRead = result.get(10, TimeUnit.SECONDS)
```

IO 요청과 완료간의 문맥을 유지하는 attachment.
편한 처리를 위해서 Connection 같은 것을 넣을 수 있음

3

```
String str = new String();
channel.read(buffer, 100, str , new CompletionHandler<Integer, String>(){
    public void completed(Integer result, String attachement) {  …  }
    public void failed(Throwable exception, String attachement) {  …   }
});
```

CompletionHandler

- V = type of result value
- A = type of object attached to I/O operation
  - Used to pass context
  - Typically encapsulates connection context

```
interface CompletionHandler<V,A> {
    void completed(V result, A attachment);
    void failed(Throwable exc, A attachment);
}
```

AsynchronousChannelGroup

- CompletionHandler는 AsynchronousChanneGroup 내부에 있는 Thread Pool에 의해 호출됨
- 내가 원하는 Thread Pool을 지정 가능

```
ExecutorService service = Executors.newFixedThreadPool(25);
AsynchronousChannelGroup channelGroup =
                        AsynchronousChannelGroup.withThreadPool(service);
AsynchronousSocketChannel socketChannel =
                        AsynchronousSocketChannel.open(channelGroup);
```

AsynchronousServerSocketChannel

```
ExecutorService service = Executors.newFixedThreadPool(25);
AsynchronousChannelGroup channelGroup =
                            AsynchronousChannelGroup.withThreadPool(service);

AsynchronousSocketChannel asyChannel =
                             AsynchronousSocketChannel.open(channelGroup);
AsynchronousServerSocketChannel listener =
                            AsynchronousServerSocketChannel.open().bind(
                            new InetSocketAddress(5000));


listener.accept(null, new CompletionHandler<AsynchronousSocketChannel, Object>() {
    @Override
    public void completed(AsynchronousSocketChannel result, Object attachment) {
    }

    @Override
    public void failed(Throwable exc, Object attachment) {
    }
});
```

DatagramChannel

```
NetworkInterface networkInterface = NetworkInterface.getByName("ifcfg0");
DatagramChannel dc = DatagramChannel.open(StandardProtocolFamily.INET)
        .setOption(StandardSocketOptions.SO_REUSEADDR, true)
        .bind(new InetSocketAddress(5000))
        .setOption(StandardSocketOptions.IP_MULTICAST_IF, networkInterface);

InetAddress group = InetAddress.getByName("225.4.5.6");
MembershipKey key = dc.join(group, networkInterface);
```

New API 추가에 따른 기존 코드 영향

- AsynchronousByteChannel
  - java.nio.channels.Channels
  - java.nio.channels.FileLock
- SeekableByteChannel
  - java.nio.channels.FileChannel 이 SeekableByteChannel을 상속
- MulticastChannel
  - java.nio.channels.DatagramChannel
- NetworkChannel
  - 아래 클래스들은 모두 NetworkChannel을 상속
    - java.nio.channels.DatagramChannel
    - java.nio.channels.SocketChannel
    - java.nio.channels.ServerSocketChannel
- Path
  - java.util.Scanner
  - java.io.FilePermission
  - java.io.File

To be continued #4