JAVA PROGRAMING

API

What is API?

API(Application Programming Interface)?

> API는 응용 프로그램이 운영체제나 데이터베이스 관리 시스템과 같은 시스템 프로그램과 통신할 때 사용되는 언어나 메시지 형식을 가지며, API는 프로그램 내에서 실행을 위해 특정 서브루틴에 연결을 제공하는 함수를 호출하는 것으로 구현된다. 그러므로 하나의 API는 함수의 호출에 의해 요청되는 작업을 수행하기 위해 이미 존재하거나 또는 연결되어야 하는 몇 개의 프로그램 모듈이나 루틴을 가진다 .
>
> [출처]API [application programming interface] | 네이버 백과사전

API가 풍부하다?

- 제공되는 기능이 많다는 뜻
- 새로운 것을 개발하기에 많은 시간을 단축할 수 있다는 장점

Java의 API/ API docs

API를 활용한다는 뜻

- 프로그램을 작성할 때 다른 사람들이 만들어 둔 기존의 클래스를 활용한다.

API docs

- API를 어떻게 활용해야 하는지에 대한 설명서

API docs는 다운로드 받거나, 웹을 통해서, Eclipse에서는 F3/F2 단축키를 통해서

API

java.lang.String

문자열작업을 쉽게 할 수 있도록 제공되는 API 클래스

기존 언어들에서 char[ ]을 이용하는 방식 대신에 편리하게 사용할 수 있다는 장점

객체타입으로 사용한다는 특징

문자열의 생성 두 가지

상수처럼 손 쉬운 방식

```
String str = “AAA”;
```

new를 활용하는 객체 방식

```
String str = new String(“AAA”);
```

두 방식의 메모리 활용 방식의 차이가 존재

API

‘==‘과 String의 equals( )

‘==‘ 연산자
메모리상에 같은 공간을 가리키는 지를 확인하는 연산자

equals( )
두 개의 String 객체가 같은 문자열을 내용으로 가지는지를 확인하는 연산자

Immutable

Java에서 문자열은 ‘불변(immutable)’하게 사용되는 메모리

다른 String 객체라고 해도 문자열의 내용이 같다면 결국 같은 메모리를 가리키게 한다.

만일 한 쪽의 문자열이 변경되면 새로운 객체가 만들어 지면서 다른 객체들에 영향이 없게 하는 구조

String에서 가장 많이 사용하는 메소드들

equals( ): 문자열의 내용물 비교
charAt( ): 특정 위치의 char 찾기
trim( ): 문자열 앞.뒤의 공백 없애기
getBytes(): 문자열의 내용을 byte[ ]로 변환

API

StringBuffer/StringBuilder

java.lang.String이 immutable이므로 가지는 문제를 해결하기 위해서 객체 생성을 최소화 하는 클래스
- 빠른 속도 & 객체 생성 최소화
  Enterprise급 시스템에서는 String사용을 가급적 자제

주요 메소드
- append( )
- reverse( )

날짜 관련 클래스 Date, Calendar

System.currentTimeMills( ) : long타입의 1000분의 1초
Date : JDK1.0에서부터 지원되는 날짜 처리 클래스
Calendar : JDK1.1이후에 변경된 날짜 처리 클래스
- GregorianCalendar

날짜 관련해서 가장 많이 쓰이는 작업들

특정 날짜의 년,월,일,요일 확인하기
특정 월의 마지막 날짜 구하기
문자열을 Date/Calendar로 변환하기
날짜들 간의 시간 차이 구하기

API

Random클래스

무작위의 어떤 값을 만들어 내야 하는 경우에 사용
- Math.random( )
- Random클래스를 활용하는 방식

java.util.Scanner

외부에서 데이터를 빨아들여서 데이터를 읽어내는 기능을 제공
JDK1.5에서부터 제공
JDK1.4이전까지는 java.io.BufferedReader라는 클래스를 활용
주요 메소드
- next( )
- nextLine( )

Wrapper클래스

wrap: '포장하다', '감싼다'는 의미
기본형 자료(int, double...)등이 JDK1.4버전까지는 객체 자료형과 엄밀하게 구분되었기 때문에 사용되었던 클래스들
- Integer, Byte, Character, Double, Long, Boolean

JDK1.5이후에는 AutoBoxing/Unboxing이 지원되므로 필요성은 감소되었음

API

Auto Boxing/Unboxing

Auto Boxing
- 자동적으로 기본 자료형을 객체 자료형으로 변환해 주는 기능
- 자료 구조 사용시 유용하게 사용됨

Auto Unboxing
- 반대로 객체타입의 값을 기본 자료형으로 변환

Timer/TimerTask

약간의 시간(interval)을 두고 어떤 작업을 하고 싶은 경우에 사용
작업 순서
- TimerTask를 상속해서 하고 싶은 작업 작성
- Timer객체를 이용해서 Schedule 등록

Formatter

출력하는 형태를 format이라고 하고, Formatter는 데이터의 출력형태를 지정하는 것
문자열 포맷팅 - %S
소수 포맷팅- %.3f
날짜 포맷팅- %tY, %ty, %tm, %td, %tA, %ta

API

예제)http://www.oracle.com/technetwork/java/javase/documentation/api-jsp-136079.html

위는 ORACLE에서 제공하는 자바 API정보를 링크해주는 사이트이다. (영문)

API

특정 클래스를 사용하기위해서는, 직접 특정 자바 패키지(java package)를 import 해줘야합니다.

특정 클래스를 넣었는데 'unidentified' 등의 컴파일 오류가 발생할 경우, 해당 클래스 API 첫 화면의 상속 트리를 참조하여 import를 해보셔야 합니다.

팁) 사실 모든 메소드를 다 볼수는 없죠. 우선 API를 찾기 전에는 '혹시 이런 기능이 있을까?' 를 생각하고, 내가 필요한 기능이 뭔지 머리속으로 상상하면서, 메소드 명을 머리속으로 가상으로 만들어보면서 훑어보시면 빨리 찾을 수 있습니다.

예를 들어, 원하는 것은 단순히 JTextField의 Text 문자 창에 문자열을 쓰고, 읽는 기능이라면, 보통 어떤 값을 넣거나 가져올때는 Get이나 Set을 많이 씁니다. JTextField 메소드 요약 목록에서 Get과 Set 주변을 찾아보면 getText(), setText()를 금방 찾을 수 있습니다.

숙제) replace와 replaceall 매소드의 차이점을 설명하고, 실제 간단한 예제를 구현하시오.

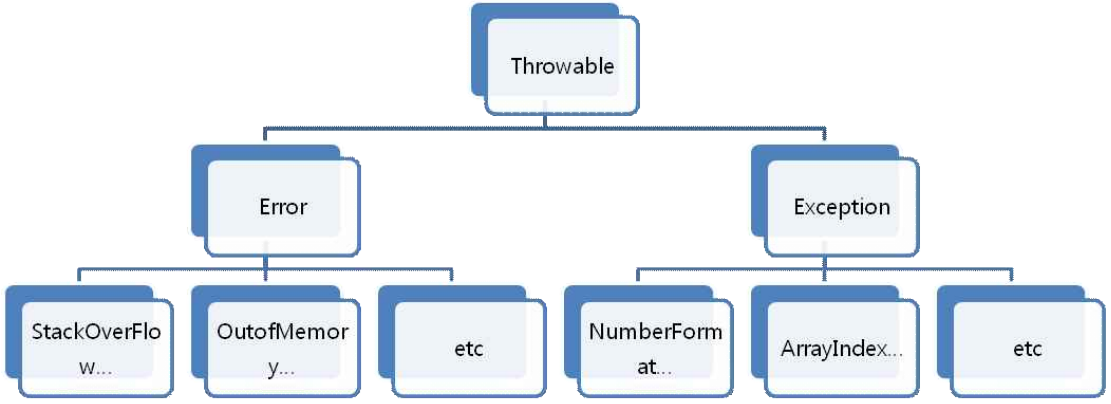
예외처리

Error/Exception

Error: 운영체제나 프로세스에 문제가 생겨서 프로그램적으로 어떤 조치도 취할 수 없는 상황
- OutofMemory
- StackOverflow
- etc

Exception: 프로그램 실행 중에 발생한 문제를 코드를 통해서 체크, 다른 코드들을 보호하기 위한 장치
- ArrayIndexOutofBoundsException
- NumberFormatException
- etc

Exception의 상속 구조



예외처리

예외의 종류
- checked VS unchecked Exception

코드를 작성할 때 컴파일러가 check하는 예외처리- checked Exception
- 주로 JVM과 외부 리소스와 통신하는 경우에 처리하도록 한다.
- Eclipse와 같은 IDE는 자동적으로 체크하는 기능을 가지고 있다.

컴파일러가 체크하지 않지만 실행할 때 발생하는 Exception – unchecked Exception
- 주로 문자열 작업, 배열, 숫자처리 등에서 발생한다.
- 개발자가 직접 처리해야만 한다.

NullPointException (Unchecked)

객체 자료형에서만 발생
어떤 변수에 실제 객체의 레퍼런스(리모컨)이 없는 상태에서 변수를 통해서 어떤 작업을 하는 경우
Java의 경우에는 반드시 '.'에서만 발생한다.

ArrayIndexOutOfBoundsException (Unchecked)

배열의 인덱스 번호가 배열의 범위를 초과해서 접근하는 경우
- int[ ] arr = {1,2,3}; //index번호는 0,1,2만 사용가능
- arr[3]; //실행할 때 문제 발생

예외처리

NumberFormatException (Unchecked)

문자나 날짜 데이터를 숫자형태로 변환할 때 발생하는 Exception

checked Exception의 원칙

주로 JVM과 외부의 리소스가 통신을 하는 경우에 일어난다.
- 네트워크를 통한 통신
- 파일 시스템을 사용하는 작업 등

checked Exception은 실행하는 데 있어서 외부의 간섭이 일어나는 경우에 일어난다.
- 운영체제의 설정이나 상황이 간섭을 일으킬 수 있는 경우

Exception 처리의 기본 코드

위험할 수 있는 코드를 try~ catch { }을 활용해서 보험처리를 한다.

```
try{
  문제가 발생할 수 있는 코드
  보호가 필요한 코드
}catch(Exception e){
  잘 못 되었을 때 처리할 코드
}
```

'{ }'은 별도의 메모리 공간을 차지하기 때문에 실행되는 코드는 별도의 공간에 실행

예외처리

try ~ catch(Exception)블록

catch(Exception e) { ... }을 보면 마치 메소드처럼 파라미터로 처리되는 것과 유사

실제로 JVM은 예외가 발생하면 예외를 객체로 만들어서 처리

Exception 은 모든 예외 객체들의 부모 클래스이므로 다형성처럼 모든 하위의 예외객체들을 처리할 수 있는 구조

throws Exception

예외를 현재 코드에서 직접 처리하지 않고, 코드가 호출된 곳으로 전달해 버리는 방식
다른 의미로 표현하자면 호출한 장소에 예외 객체를 전달하는 방식
일종의 리턴 타입처럼 설계하는 데 유용하게 사용될 수 있다.

사용자 정의 예외

주로 Exception을 상속해서 개발자가 직접 Exception을 설계할 수 있다.
클래스를 정의하는 것과 비슷하게 설계
주로 비즈니스 로직에 대한 잘못된 상황을 전파하는 목적으로 설계하는 것이 일반적이다.

예외처리

IDE툴에서는 자동으로 예외처리를 생성해 준다.

```
java.io.File file = new java.io.File("c:\\file.txt");
try{
    file.getCanonicalFile();
catch (IOException e) {
    e.printStackTrace();
}
```

한가지 주위해야 하는 것은 위와 같이 "e.pringStackTrace()"를 툴에서 자동으로 생성해 주었다고, 예외 처리가 모두 끝나는 것이 아닙니다.

처리해야할 예외에 대한 후 작업이 있다면 별도 코드를 작성해야 하며 예외를 Log에 남기는 부분도 필요합니다.

Logging또한 "e.printStackTrace()"등의 IDE에서 만들어주는 코드를 그대로 남기는데, 어떤 경우에 예외가 호출되었는지에 대한 정확한 정보를 남기기 위해서 개발자가 나중에 알수 있는 형태의 메시지 형태로 재정의 해서 남겨야 합니다.

프레임워크

Java에서는 C언어와 달리 자주 사용되는 자료구조들을 API로 제공
주로 interface와 상속등의 다형성을 잘 설계한 구조로 Collections Framework이라고 통칭
JDK1.6까지 지속적인 알고리즘이나 자료구조가 추가됨
- JDK1.6에서는 Deque interface추가

java.util패키지의 Collections Framework

Interface를 이용한 설계의 극치
각 구현해야 하는 자료구조들은 Interface로 설계하고 공통코드는 상속의 구조를 이용하는 설계 방식

java.lang.Object
  └ java.util.AbstractCollection<E>
    └ java.util.AbstractList<E>
      └ **java.util.ArrayList<E>**

All Implemented Interfaces:
Serializable, Cloneable, Iterable<E>, Collection<E>, List<E>, RandomAccess

Direct Known Subclasses:
AttributeList, RoleList, RoleUnresolvedList

JDK1.5 이후의 Generic 문법

Collections Framework이 기존에는 모든 객체자료형들을 처리하기 위해서 java.lang.Object타입을 사용
JDK1.5이후에는 컴파일 시점에 자료구조에서 사용되는 Type을 체크하는 Generic문법을 사용하는 방식으로 변화

프레임워크

프레임워크(framework)란?

사전적 의미로는 '복잡한 문제를 해결하거나 서술하는데 사용하는 기본 개념 구조'라고 정의되어 있습니다. **정의에 따르면 '특정한 목적에 사용되는 사고체계'라는 뜻으로 소프트웨어 개발에 해당하는 것은 동통으로 사용하는 라이브러리, 공통으로 사용하는 개발 도구, 공통으로 사용하는 인터페이스 등**이 있습니다.
프레임워크는 만들고자 하는 구조물의 기본 골격으로 아키텍처와 마찬가지로 소프트웨어 공학뿐만 아니라 건축, 비행기, 선박, 다리같은 구조물을 만드는 모든 분야에서 발견할 수 있는 개념입니다. 프레임워크는 **실체가 있는 구조물 뿐만 아니라 실체가 없는 정책, 전략에도 쓰입니다.** 예를 들어 경영 전략을 수립할때 쓰이는 4C(Customer, Company, Channel, Competitor)나 4P(Price, Place, Promotion, Product)같은 프레임워크는 현실을 분석해내는 전략 사고의 틀을 제공합니다. 또한 정통부에서 2003.06월에 발표한 정부정책 라이프사이클 프레임워크(GPLC framework)는 정책수립에서 정책집행까지 정책에 관련된 모든 과정의 틀을 제공합니다. 그 외에도 수많은 프레임워크가 있지만, 모든 프레임워크들은 자신이 다루는 대상의 틀을 결정한다는 공통점을 지닙니다.

프레임 워크의 특징

프레임워크는 만들고자 하는 최종 구조물을 완벽하게 결정하지 못하지만 최종 구조물 대강(大綱, outline, 틀)을 결정

틀을 결정하는 프레임워크의 특성은 '재사용'이라는 미덕을 더해줍니다. 프레임워크는 개별 문제를 해결하는 재사용이 아니라 전체 문제를 해결하는 재사용이기 때문에 패턴언어의 인스턴스로 생각할 수 있습니다.

프레임워크

프레임워크는 개념으로 끝나지 않고 반드시 설계가 있어야만 합니다. 모든 프레임워크는 프레임워크 구성요소, 각 구성요소가 다루는 내용과 갖춰야 할 형식, 구성요소 사이의 관계, 사용 예제를 담고 있어야 합니다.

프레임워크가 중요한 이유는 **객체 지향 개발을 하게 되면서 개발자의 취향에 따라 다양한 프로그램이 나오게 되어있습니다.** 프로그램 개발에 투입되는 개발자도 점점 늘어남에 따라 전체 시스템의 통합성, 일관성이 부족하게 되었기때문입니다. 그래서 **개발자의 자유를 제한하기 위해 프레임워크가 도입된 이유**입니다

프레임워크가 가져야할 특징으로는

☆ 개발자들이 따라야 하는 가이드라인을 가지게 됩니다.

☆ 개발할 수 있는 범위가 정해져 있습니다..

☆ 개발자를 위한 다양한 도구들이 지원됩니다.

Collections Framework

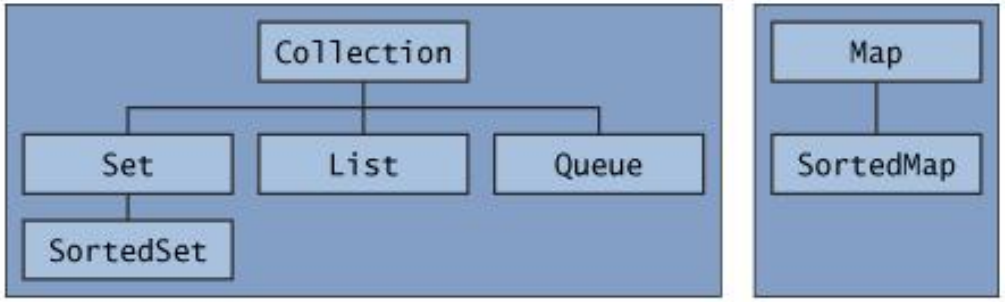
Collections Framework에서 중요한 사항들

자료구조들의 종류는 결국은 어떤 구조에서 얼마나 빨리 원하는 데이터를 찾는가에 따라 결정된다.

- 순서를 유지할 것인가?
- 중복을 허용할 것인가?
- 몇 번 만에 데이터를 찾아낼 수 있는가?
- 다른 자료구조들에 비해서 어떤 단점과 장점을 가지고 있는가?

코어 Interfaces

Collection Frameworks의 가장 중심이 되는 Interfaces



Collections Framework

equals( )와 hashCode( )

자료구조 안에 들어가는 객체들은 단순히 메모리상의 위치 비교를 하는 것이 아니다.

때로는 메모리가 달라도 검색이 가능해야 하는데 이를 위해서 java.lang.Object의 equals( )와 hashCode( )메소드가 사용된다.

equals( ): 메모리상의 위치가 다르다고 해도 검색이 가능하도록 설계하려면 override해 준다.

hashCode( ): 해싱 알고리즘을 사용할 때 객체들을 분류하고 보관할때 기준이 되는 hash value를 만들어 내는 메소드

List 계열

특징: 순서가 있고, 중복을 허용 (배열과 유사)
장점: 가변적인 배열
단점: 원하는 데이터가 뒤쪽에 위치하는 경우 속도의 문제
방식: equals( )를 이용한 데이터 검색
구현 클래스
- ArrayList
- LinkedList

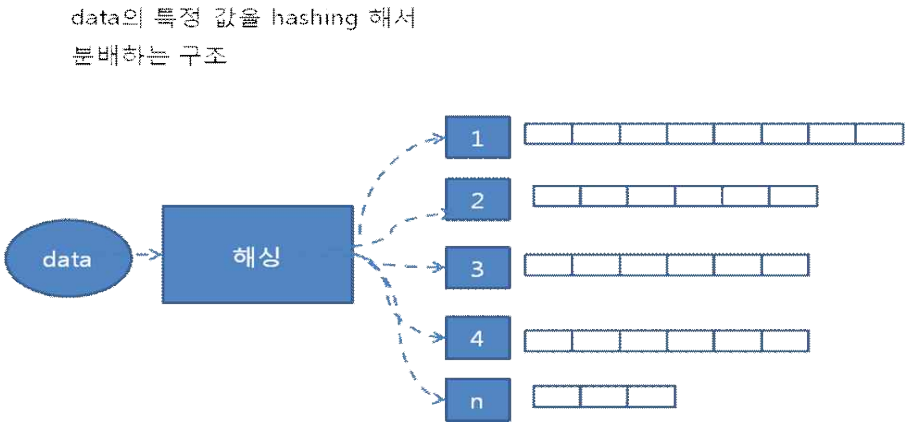
Collections Framework

Set 계열

특징: 순서가 없고, 중복을 허용하지 않음
장점: 빠른 속도
단점: 단순 집합의 개념으로 정렬하려면 별도의 처리가 필요하다.
구현 클래스
- HashSet
- TreeSet:

Map계열

특징: Key(키)와 Value(값)으로 나뉘에 데이터 관리, 순서는 없으며, 키에 대한 중복은 없음
장점: 빠른 속도
단점: Key의 검색 속도가 검색 속도를 좌우
구현 클래스
- HashMap
- TreeMap

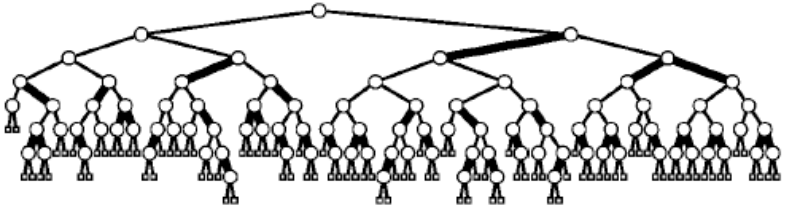


Collections Framework

Tree 계열

각 Element(Node)들의 균형된 분포가 관건
Balanced Tree - Node들의 분포가 고른 Tree구조
- 어떤 Node를 찾아도 검색 depth가 일정하게 구성

구현 클래스
- TreeSet
- TreeMap



Stack 클래스

특징: 아래가 막혀있는 순서 있는 구조
장점: 위쪽의 데이터를 빈번하게 접근하는 경우에 유리
단점: 선형 리스트의 단점
메소드: push( ), pop( ), peek( )

Collections Framework

List 인터페이스 예제

```
package java.util;

public interface List extends Collection {
    // Query Operations
    int size();
    boolean isEmpty();
    boolean contains(Object o);
    Iterator iterator();
    Object[] toArray();
    Object[] toArray(Object a[]);

    // Modification Operations
    boolean add(Object o);
    boolean remove(Object o);         // Bulk Modification Operations
    boolean containsAll(Collection c);
    boolean addAll(Collection c);
    boolean addAll(int index, Collection c);
    boolean removeAll(Collection c);
    boolean retainAll(Collection c);
    void clear();

    // Comparison and hashing
    boolean equals(Object o);
    int hashCode();
```

Collections Framework

```
// Positional Access Operations
      Object get(int index);
     Object set(int index, Object element);
     void add(int index, Object element);
     Object remove(int index);

      // Search Operations
      int indexOf(Object o);
     int lastIndexOf(Object o);

      // List Iterators
      ListIterator listIterator();
     ListIterator listIterator(int index);

      // View
      List subList(int fromIndex, int toIndex);
 }
```

Toolkit

자바에는 AWT, Swing 과 SWT라는 GUI 툴킷이 존재한다. 다양한 종류가 있는 만큼 각 툴킷의 장단점이 존재한다.

1. AWT (Abstract Windows Toolkit)
Java에서 기본적으로 지원하는 GUI 툴킷이다. AWT는 모든 자바 버전에서 제공하는 표준으로 가장 안정적이다 그리고 AWT를 사용하기 위한 별도의 설치 없이도 자바 런타임 환경이 있는 곳 어디에서나 사용할 수 있다.
반면에 제한된 GUI 컴포넌트, 레이아웃, 이벤트를 가진 매우 단순한 툴킷이다. 그래서 더 많은 컴포넌트가 필요하다면 처음부터 구현해야 한다는 문제가 있다.
AWT 컴포넌트는 "쓰레드 보안"이 되어있다. 따라서 애플리케이션에서 어떤 쓰레드가 GUI를 업데이트 하는지 신경 쓰지 않아도 된다. 이로써 GUI 업데이트 문제가 없어진다. 하지만 AWT GUI는 느리게 실행된다.

2. Swing
AWT의 단점을 보완하기 위해서 만든 툴킷이다. Swibg은 매우 유연하고 강력한 GUI 툴킷이지만, 복잡한 구조로 사용하기 위해서 많은 학습이 필요하다. Swing은 AWT를 기반으로 구현된다. 모든 Swing들의 부분은 AWT의 일부이기도 한다. Swingd은 AWT의 이벤트를 사용하며 Colors, Images, Graphics와 같은 클래스를 지원한다.
Swing은 하드웨어 GUI 가속장치와 특별한 호스트 GUI 연산을 활용하지 못할 수도 있다. 따라서 Swing 애플리케이션은 원시 GUI 보다 느리다. AWT와는 달리 Swing 컴포넌트는 쓰레드 보안이 되지 않는다. 어떤 쓰레드가 GUI 업데이트 하는지를 신경 써야 한다. 쓰레드 사용을 잘못 할 경우 사용자 인터페이스 고장 같은 예외상황이 발생한다.

3. SWT
SWT는 저수준 GUI 툴킷이다. JFace라는 향상된 컴포넌트와 유틸리티 서비스 세트로 SWT 구현이 쉽다. SWT는 AWT와 Swing의 단점을 제거하고 장점만을 가진 시스템을 구현하였다.

GWT

Google Web Toolkit(GWT)는 Google Maps와 Gmail과 같은 AJAX 기반 웹 어플리케이션을 쉽게 개발할 수 있도록 도와주는 오픈 소스 자바 개발 프레임워크입니다.

오늘날 동적인 웹 어플리케이션을 개발하는 것은 싫증나는 작업이고 에러가 나기 쉬운 작업입니다. 이 작업은 당신이 당신의 일할 시간 중의 90%를 웹브라우저와 플렛폼간의 호환성 문제로 소비하게 합니다. 그리고 자바스크립트의 모듈화가 부족한 점은 공유, 테스트, AJAX 컴포넌트의 재사용을 어렵게 합니다.
GWT는 당신의 골치 아픈 작업들을 도와줄 것이며, 당신의 유저들에게 동적인, 표준-호환적인 경험을 제공할 것입니다. 당신은 자바로 프로그램을 작성하면, GWT 컴파일러가 당신의 자바 클래스를 브라우저에서 구동될 수 있는 자바스크립트와 HTML로 변환해 줄 것입니다.

http://code.google.com/intl/ko-KR/webtoolkit/overview.html#Why

* 이렇듯 프레임워크와 툴킷의 개념은 비슷한 점이 많습니다. 용어를 구분짓자면,
프레임워크는 제품을 만드는 규약, 지원을 말하며, 실제 완성될 제품에 대해 어느정도 알 수 있는 청그림입니다.

툴킷은 제품을 만들때 사용하는 도구들의 모음입니다. 제품을 만드는데 필요한 도구들이므로 이 도구를 사용하면 역시 실제 완성될 제품에 대한 청그림 및 규약이 있을 수 밖에 없습니다.

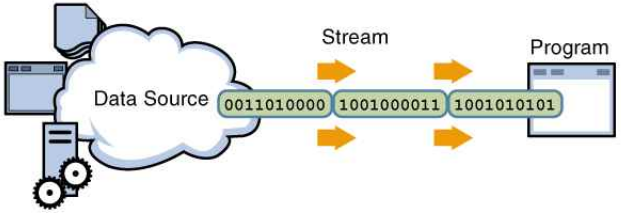
Java IO

I/O 프로그래밍

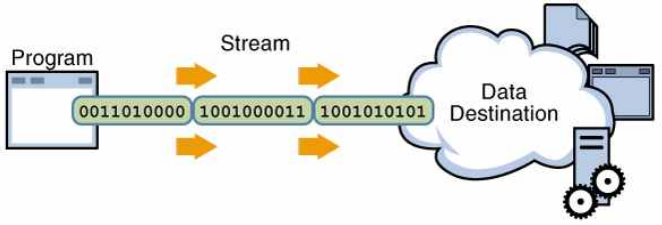
데이터를 읽어내는 Input 기능과 데이터를 출력해 주는 Output기능을 활용하는 프로그래밍
API에서 많은 Input방식/ Output방식 지원
데이터 입력의 대상
　　키보드, 파일, 네트워크를 통해 들어오는 데이터
데이터 출력의 대상
　　모니터, 파일, 외부 네트워크

I/O 프로그래밍의 핵심 키워드

In – Read
　　모든 작업 시에 in이나 read라는 단어가 있다면 '읽어'내는 기능을 의미한다.



Out – Write
　　모든 작업 시에 out이나 write라는 단어가 있다면 '쓰는' 기능을 의미한다.



Java IO

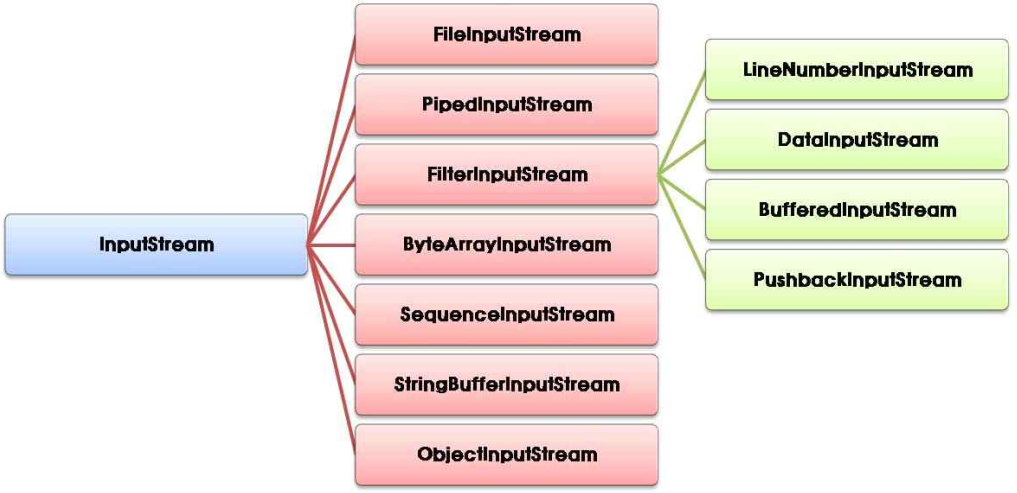
I/O 프로그래밍의 절차

원하는 대상을 정한다.
대상에 적절한 InputStream계열의 파이프나 OutputStream계열의 파이프를 연결한다.
대상에서 읽거나 쓰기 작업을 한다.
모든 작업 후에는 반드시 close( )해서 파이프 연결을 종료한다.

InputStream/ Reader

데이터를 읽어내는 기능
- InputStream 은 byte단위의 데이터 읽기
- Reader는 char단위의 데이터 읽기

InputStream은 추상클래스로 설계되어 공통적인 기능을 물려주는 역할



Java IO

InputStream의 가장 핵심 기능-read( )

연결된 파이프를 통해서 read( )해 주면 한 byte의 내용을 int타입으로 읽어낸다.
- 이 때 int의 값이 -1인 경우는 더 이상 읽어낼 데이터가 없는 경우

read( ) int: 한 byte에 쓰여진 데이터
read(byte[ ]) int: 읽어낸 데이터는 byte[ ]안으로 들어가고, int는 몇 개나 새로운 데이터가 읽어냈는지를 말한다.

OutputStream의 핵심 기능-write( )

연결된 파이프를 통해서 write(int) 해 주면 대상으로 데이터를 전송
write(int) 한 바이트의 데이터를 전송
write(byte[ ]) byte[ ]안에 있는 데이터들을 전송
write(byte[] , int, int) byte[ ]안에 있는 데이터들을 시작 위치에서부터 몇 개의 개수만큼 데이터들을 전송

파일 복사하는 프로그램 만들기의 절차

원본 파일에는 FileInputStream연결
복사해서 생성할 대상에는 FileOutputStream연결
원본에서 데이터를 읽어낸다(read).
대상에 읽어낸 데이터를 출력한다(write).
더 이상 읽을 데이터가 없으면 모든 연결을 close( )한다.

Java IO

문자열 데이터의 읽고 쓰기

String과 byte의 관계
- String 클래스의 getBytes( ) byte[ ] 반환
- String 클래스의 생성자 new String(byte[ ])

문자열을 보다 편하게 읽고 쓰기 위한 Tip
- 문자열이 끝날 때 '\n'을 기호로 삼아서 보낸다.
- Scanner를 이용해서 nextLine( )해 주면 '\n' 이전의 데이터들을 손쉽게 문자열로 가져온다.

원래 기능을 보다 편리하게

InputStream/OutputStream은 byte단위로 데이터를 처리하기 때문에 좀 더 많은 기능은 상속의 형태로 구현되어 있다.
원래의 In/Output Stream을 다른 클래스 안으로 결합시켜서 기능을 확장하는 방식

DataInputStream/DataOutputStream

InputStream/OutputStream을 좀 더 편리하게 사용하기 위해서 제공되는 확장된 클래스
ex> readInt( )의 경우는 한 번에 4 byte의 데이터를 읽어서 int 타입으로 변환하는 기능

기본자료형에 맞게 데이터들을 처리하거나 UTF-8 방식의 문자 데이터를 처리하는 기능사용시에 편리

Java IO

Reader/Writer

InputStream/OutputStream의 char처리 대용
char의 2bytes 기반의 처리 기능 지원
char기반의 처리라는 점을 제외하면 모든 처리는 InputStream/OutputStream과 동일

InputStreamReader/OutputStreamWriter

110v 를 220v로 변환해 주는 Adapter역할을 하는 클래스
InputStreamReader
- Reader 계열의 클래스이지만 생성시에 InputStream을 받아서 변환하는 역할

OutputStreamWriter
- Writer 계열의 클래스이지만 생성시에 OutputStream을 받아서 변환하는 역할

ObjectInputStream/ObjectOutputStream

객체가 가진 정보를 byte로 기록해 두었다가 다시 비어있는 객체에 정보를 넣는 방식의 기능을 지원하는 클래스들
객체의 정보를 byte로 변환할 때 java.io.Serializable 인터페이스 타입인지를 확인하고 JVM이 변환/복구 작업을 수행
Serializable 인터페이스는 개발자가 아무 작업을 할 수 없고, 단지 타입 정보만 추가하는 방식으로 사용

Java IO

RandomAccessFile

다른 클래스들과는 달리 '읽기,쓰기'기능이 동시에 존재하는 클래스
seek(long)라는 기능을 통해서 파일내에서 위치(cursor)를 마음대로 이동하는 기능

텍스트 파일을 생성하여, 1~100까지의 정수를 저장한 후,
임의의 위치에서 숫자를 읽어오는 프로그램을 작성하시오.