1. 자바 소개

자바(Java)의 정의

- C++ 에 기초한 선(Sun)사에 의해 개발된 프로그래밍 언어
- 객체 지향 언어
- 인터넷과 웹을 위한 프로그래밍 언어

자바의 재정의

- 최초의 범용 소프트웨어 플랫폼
- 언어, 자바 가상 컴퓨터와 클래스 라이브러리와 API들의 집합으로 구성
- 인터넷 컴퓨팅을 위한 플랫폼
  - 하드웨어와 독립
  - 확장성이 있음
  - 개방적임

자바의 현재

- 널리 채택되고 보편적 언어가 됨
- 안정이 되고 신뢰성이 높아짐
- 자바 기술이 공개되고 출판됨
- 일반 응용 프로그래밍 언어로 확장됨
- 많은 개발 도구들이 나옴
- 개발자,사용자,관리자 모두에게 혜택을 줌
- 이동 컴퓨팅 플랫폼으로 자리잡음

자바 2 플랫폼

- 표준형 플랫폼(J2SE)
  - 자바 언어와 연관된 도구들의 주류 플랫폼
- 기업용 플랫폼(J2EE)
  - 기업용 프로그램을 개발하기 위한 플랫폼
- 소형용 플랫폼(J2ME)
  - 소비자/임베디드 장비용 프로그램을 개발하기 위한 플랫폼

자바 프로그램 구조

- 자바 프로그램의 구조는 다음과 같다:
  - 자바 프로그램은 클래스들 (classes)로 구성된다.
  - 클래스는 하나 이상의 메소드(methods)들을 포함한다.
  - 메소드는 프로그램 문들(statements)을 포함한다.
- 이 용어들은 자세히 다루어질 것이다.
- 모든 자바 응용들은 main 메소드를 가진다.

자바 프로그램 구조

```
// 클래스에 대한 설명문들
public class SimpleProgram
{


}
```

클래스 머리부(header)

클래스 몸체(body)

설명문들(comments)은 거의 모든 곳에 추가될 수 있다

자바 프로그램 구조

```
// 클래스에 대한 설명문들
public class SimpleProgram
{

    //  메소드에 대한 설명문들
    public static void main (String[] args)
    {

    }

}
```

메소드 몸체

메소드 머리부

예제 프로그램

```
// 파일 이름 : Hello.java
// 목적: 자바 응용 프로그램의 기본 구조를
    보여준다
public class Hello
{
      // '안녕하세요'를 출력한다
     public static void main(String[] args)
     {
          System.out.println("안녕하세요");
     }
}
```

설명문(Comment)

- 설명문은 프로그램 내의 문서화를 제공한다.
- 설명문은 프로그램의 동작에 영향을 끼치지 않는다.
- 세 가지 종류의 설명문:
  - // 이 설명문은 이 줄의 끝까지 계속될 수 있다
  - /*
    이 설명문은 여러 줄에 걸쳐 계속될 수 있다
    */
  - /** 이것은 javadoc 설명문이다
    이 설명문은 여러 줄에 걸쳐 계속될 수 있다
    */

식별자(Identifier)

- 식별자는 프로그래머가 프로그램에서 구성요소들을 나타내기 위해 사용하는 단어이다.
- 식별자는 대문자들, 소문자들, 숫자들, 밑줄 문자(_)와 달러 기호($)의 조합이어야 한다.
- 식별자는 숫자로 시작할 수 없다.
- 자바는 대문자와 소문자를 구별한다.

식별자

- 세 가지 유형의 식별자들
  - 프로그래머가 선택한 단어들
  - 다른 프로그래머가 선택한 단어들
  - 예약어들(reserved words): 특별한 식별자들
- 식별자 이름은 나타내고자 하는 것을 잘 기술해야 하고 읽기가 쉬워야 한다.

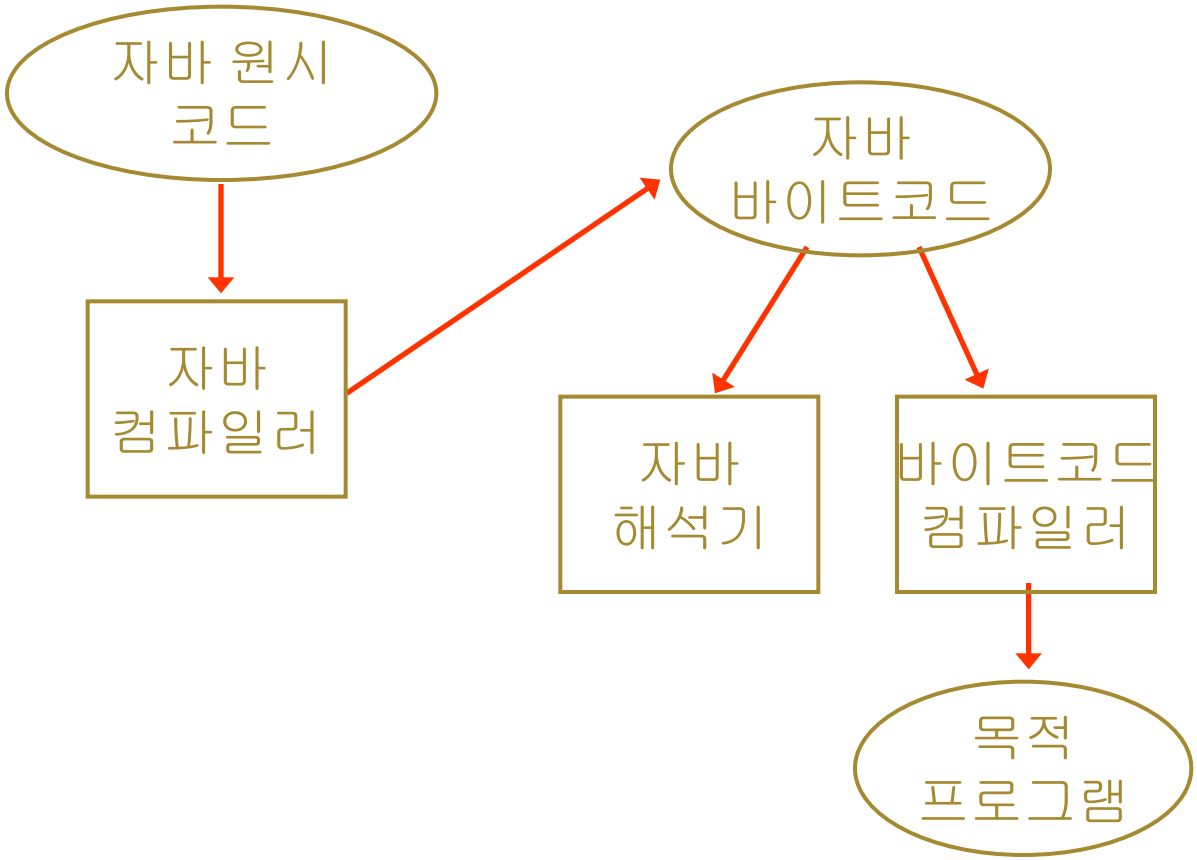
흰색 공간(White Space)

- 흰색 공간은 빈 칸(blank), 빈 줄(newline), 탭(tab)을 포함한다.
- 흰색 공간은 프로그램 안에 있는 단어들(words)과 기호들(symbols)을 분리하기 위해 사용한다.
- 우리는 프로그램을 여러 가지 방법으로 쓸 수 있다.
- 프로그램은 흰색 공간과 설명문을 적절히 사용하여 읽기 쉽게 작성해야 한다.

프로그래밍 언어 (Programming Language)

- 프로그램은 실행되기 위해 컴퓨터가 이해할 수 있는 기계어(machine language)로 바꾸어야 한다.
- 컴파일러(compiler)는 원시 프로그램(source code)을 기계어로 된 목적 프로그램(object code)으로 바꾸어 주는 소프트웨어이다.
- 자바 컴파일러는 자바 원시 프로그램을 중간 언어인 바이트코드(bytecode)로 바꾸어 준다.
- 자바 해석기(interpreter)가 바이트코드를 한 문장씩 읽고 특정 컴퓨터에서 실행한다.
- 따라서 자바 컴파일러는 특정 컴퓨터에 묶여 있지 않다.

자바 프로그램의 번역(compilation) 및 실행(execution) 과정



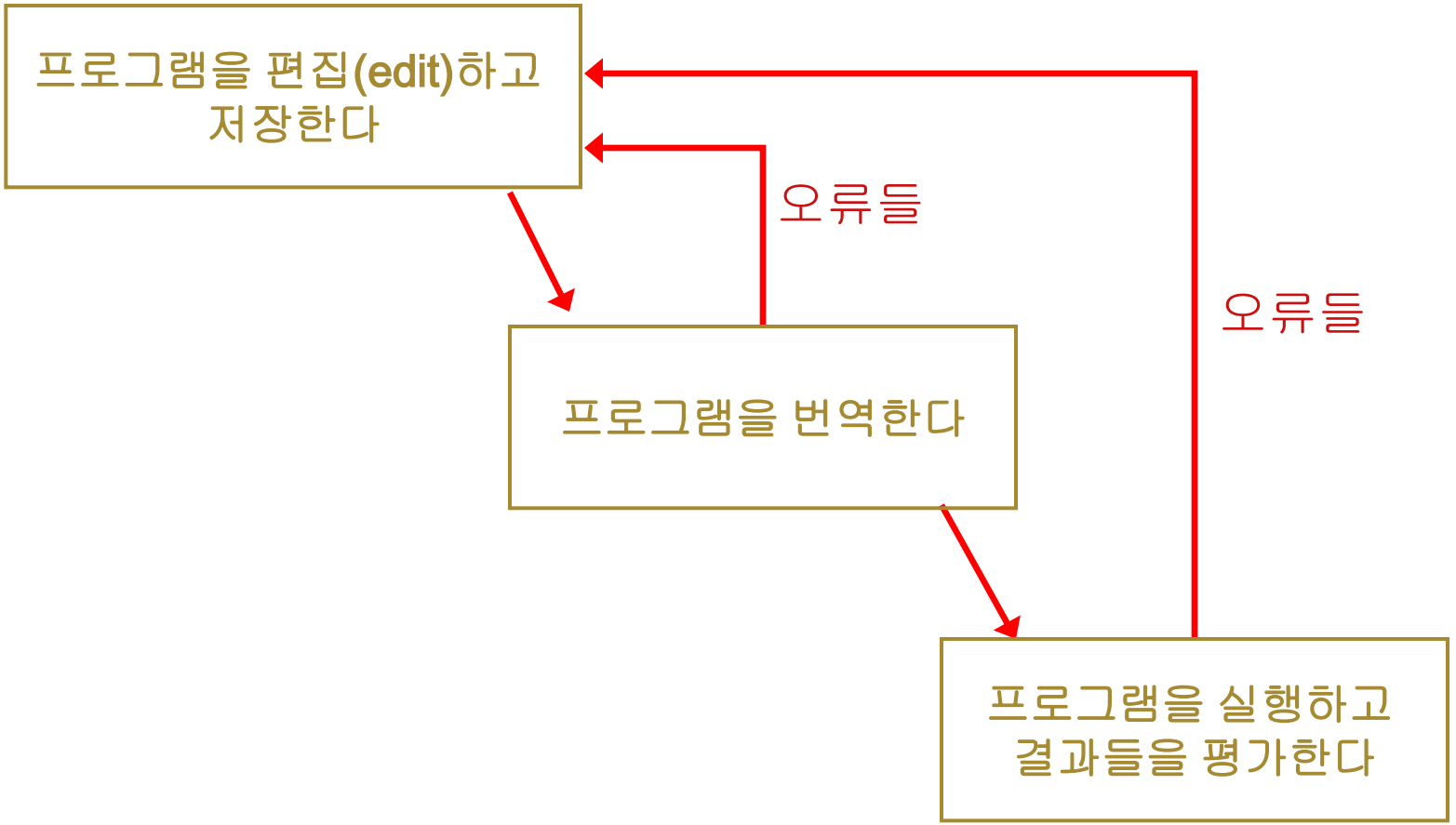
통사론(Syntax)과 의미론(Semantics)

- 프로그래밍 언어의 통사론 규칙들(syntax rules)은 언어의 요소들(elements)이 프로그램 문들을 만들기 위해 어떻게 결합될 수 있는지를 명확하게 기술한다.
- 이 규칙들은 프로그램을 작성하기 위해서 반드시 따라야 한다.
- 프로그램이 규칙(들)을 따르지 않는다면 컴파일러는 오류메시지 (error message) 를 만들어 낸다.
- 프로그램 문의 의미론은 그 문이 실행될 때 일어나는 것을 정의한다.

오류(Error)

- 오류는 세 가지 유형으로 나눌 수 있다:
  - 번역 오류 (compile-time error)
  - 실행 오류 (run-time error)
  - 논리 오류 (logical error)
- 번역 오류는 자바의 통사론 규칙을 따르지 않아서 발생하는 오류이다.
- 실행 오류는 프로그램 실행 중에 발생하는 오류이다.
- 논리 오류는 프로그램의 실행 결과가 틀린 경우이다.

기본적인 프로그램 개발 과정



자바에 대해 배워야 할 두 가지

- 자바 언어 그 자체를 배워야 함
  -- 필요한 클래스들과 메소드들을 여러분 자신이 프로그램할 수 있기 위해
- 자바 클래스 라이브러리(library)에 있는 클래스들과 메소드들을 사용하는 방법을 배워야 함
  -- 프로그램 성능과 이식성(portability)을 개선하기 위해
  -- 많은 다양한 클래스 라이브러리가 인터넷상에 있다.

자바 개발 환경

- TextPad: 자바 지원 텍스트 편집기, 사용하기 쉬움
- Sun's JDK: 명령어 방식 인터페이스
  - 최신 버전: JDK 6 Update 18 (Sun의 웹 사이트: http://java.sun.com/javase/)
- 소프트웨어 업체들이 많은 자바 개발 환경(IDE) 소프트웨어를 출시
  - 이클립스 컨소시움의 Eclipse
  - Microsoft의 Visual J++
  - Sun의 NetBeans
  - Symantec의 Visual Café
  - Inprise의 JBuilder

자바 프로그램의 분류

- 응용 프로그램(Java Application,보통 프로그램): 브라우저와 무관하게 실행되는 독립적 프로그램

- 애플릿(Java Applet, 웹을 통해 전송 가능): 자바를 지원하는 브라우저(browser)에서 실행되는 자바 프로그램(HTML 페이지에 포함)

응용 프로그램의 구조

- 각 프로그램은 여러 클래스들로 구성된다.
- 호출되는 첫번째 메소드는 main 메소드:

  public static void main(String[] argv)

  (복잡해 보일지 모르나 추후 설명!)
- main 메소드가 프로그램의 흐름(flow)을 통제한다.

애플릿의 구조

- 응용과 비슷하나 실행되는 첫번째 메소드가 init이다:

  public void init()

- 애플릿의 나머지는 사용자 행위등과 같은 사건들에 대응하는 일련의 사건 처리기(event handler)이다.
- 브라우저나 애플릿 뷰어(applet viewer)에서 실행되는 자바 해석기에 의해 실행된다.

응용 프로그램의 작성 및 실행

- TextPad와 같은 편집기를 사용하여 자바 원시 프로그램을 작성한다.
- 원시 프로그램을 자바 컴파일러를 사용하여 바이트코드로 번역한다.
- 바이트코드를 자바 해석기를 사용하여 실행한다.

응용 프로그램의 작성

아래와 같은 `Hello.java`라는 파일을 만든다:

```
// 파일 이름 : Hello.java
// 목적: 자바 응용 프로그램의 기본 구조를 보여준다
public class Hello
{
     // '안녕하세요'를 출력한다
    public static void main(String[] args)
    {
        System.out.println("안녕하세요");
    }
}
```

응용 프로그램의 번역

- 명령 프롬프트에서 다음 명령어를 입력한다:

  javac Hello.java

- TextPad 사용시
  - TextPad를 시작한다
  - *File* 메뉴의 *Open* 메뉴 항목을 선택한 후 Hello.java 원시 코드를 불러 들인다
  - *Tools* 메뉴의 *External Tools* 메뉴 항목 중 *Compile Java* 를 선택한다

주: 컴파일이 성공하면 컴파일러는 Hello.class라는 (바이트코드) 파일을 생성한다

응용 프로그램의 실행

- 명령 프롬프트에서 다음 명령어를 입력한다:
  java Hello
  - 화면 상에 "안녕하세요"라는 문구가 보여야 한다
- TextPad 사용시
  - *Tools* 메뉴의 *External Tools* 메뉴 항목 중 *Run Java Application* 을 선택한다
    -- 출력을 위해 또 다른 창을 띄운다
    -- 그 창 위에 "안녕하세요"라는 문구가 보여야 한다

애플릿의 작성 및 실행

- 애플릿 원시 프로그램을 작성한다.
- 원시 프로그램을 번역한다.
- 애플릿을 포함하는 HTML 파일을 만든다.
- HTML 파일을 애플릿 뷰어나 브라우저안으로 불러 들인다.

애플릿의 작성

아래와 같은 *HelloWorld.java* 라는 파일을 만든다:

```
import java.applet.Applet;
import java.awt.*;
public class HelloWorld extends Applet {
        public void paint(Graphics g) {
                g.drawString("안녕하세요", 50, 25);
        }
}
```

HTML 파일의 생성

아래와 같은 *HelloWorld.html*라는 파일을 만든다(*HelloWorld.class*를 포함하는 디렉토리와 같은 위치에):

```
<HTML>
<HEAD>
<TITLE>Hello to Everyone! </TITLE>
</HEAD>
<BODY>
애플릿의 출력 결과:
<APPLET CODE="HelloWorld.class" WIDTH=150 HEIGHT=25>
</APPLET>
</BODY>
</HTML>
```

HTML 파일 불러 들이기

- 브라우저안으로 HTML 파일을 불러 들인다
  - 브라우저의 위치에 당신이 만든 HTML 파일의 URL을 입력한다
  - 예: 내 C 드라이브에 있는 파일의 경우 다음과 같이 입력

    file:///C|/myhome/HelloWorld.html
- 혹은, HTML 파일을 JDK 안에 제공된 애플릿 뷰어 안으로 불러 들인다
  - 명령 프롬프트에서 다음 명령어를 입력한다:

    appletviewer HelloWorld.html

주: 여러분은 브라우저나 애플릿 뷰어에서 다음과 같은 것을 보아야 한다:

애플릿의 출력 결과: 안녕하세요

요약

- 자바 정의
- 자바 프로그램 구조
- 프로그램의 번역 및 실행 과정
- 자바 개발 환경
- 자바 프로그램의 작성 및 실행