

자바의 개요 및 프로그래밍 환경구축

1. 자바의 개요

- 1991년 Sun Microsystems 사의 James Gosling에 의해 개발
- 가전제품들의 상이한 중앙처리장치나 운영체제에도 동작하도록 개발
- 처음에는 "오크(Oak)"라고 불렀으나, 1995년 JAVA로 개명
- Netscape 와 Explorer 등의 웹 브라우저가 자바를 지원하면서 크게 발전

2. 자바의 플랫폼

- 자바 언어로 작성된 프로그램을 실행시킬 수 있는 환경
- 운영체제에 독립적

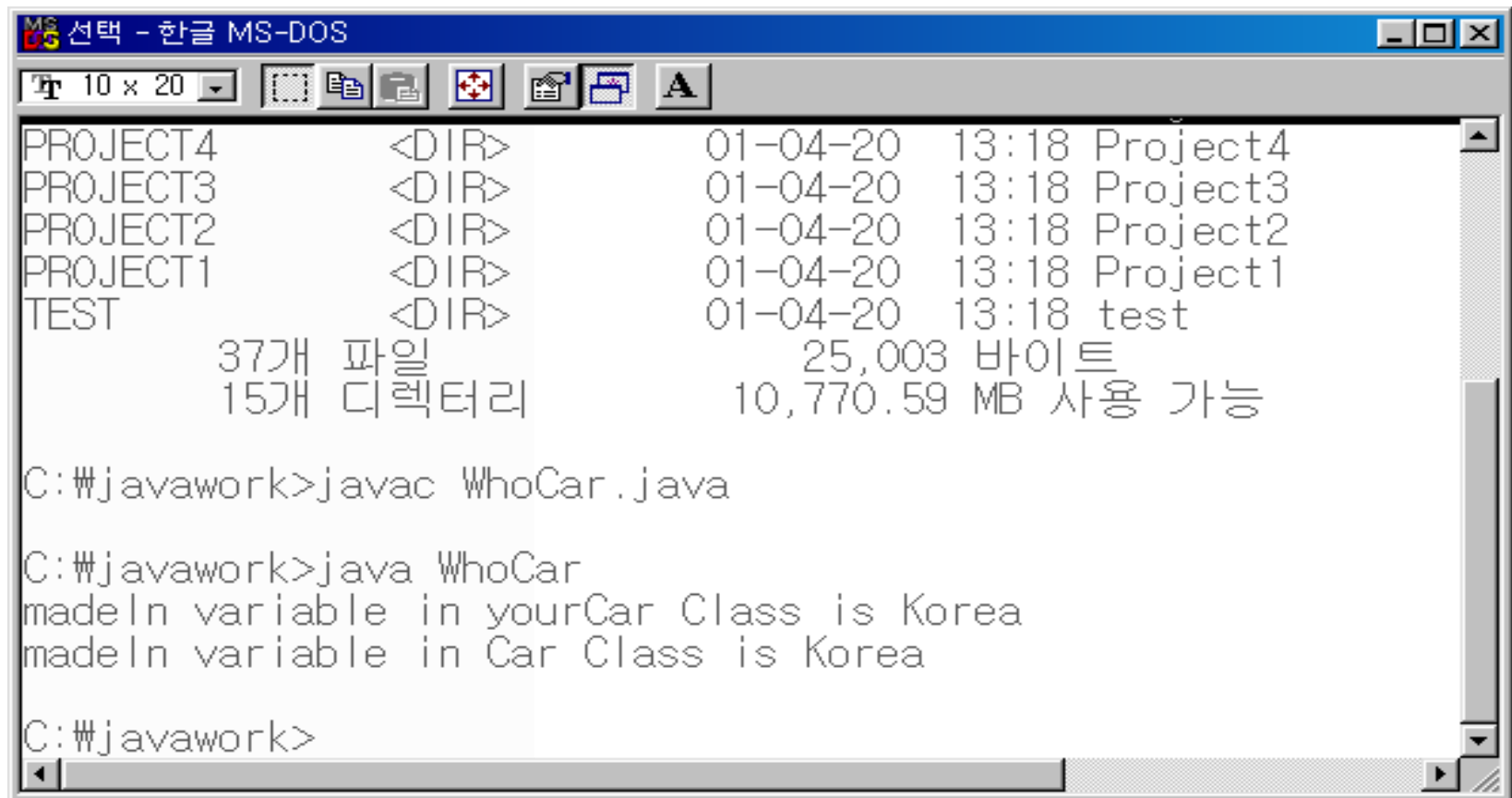


- 자바 프로그램:
 - 자바 언어로 된 프로그램
 - 확장자: 소스파일 .java
실행파일 .class
 - 컴파일: JDK 사용
- 자바가상머신:
 - 실행파일을 실행
 - JDK의 bin 디렉토리에 포함
- 운영체제: 윈도우, 유닉스, 리눅스 등
- JDK(Java Development Kit)

3. 자바의 컴파일과 실행

- 컴파일: 소스 코드 -> 바이트 코드
javac WhoCar.java
 -> WhoCar.class 생성
- 실행: JVM이 인터프리팅
 - 어플리케이션인 경우: java WhoCar
 - 애플릿인 경우: 애플릿을 포함하는
html 파일을 브라우저에서 실행 또는
appletviewer에서 실행

어플리케이션의 예(DOS에서 실행)



The screenshot shows a Windows-style window titled "MS 선택 - 한글 MS-DOS". The command prompt displays the output of a `dir` command, listing five directories: PROJECT4, PROJECT3, PROJECT2, PROJECT1, and TEST. Below the directory listing, it shows summary statistics: 37 files, 25,003 bytes, 15 directories, and 10,770.59 MB available. The user then runs `javac WhoCar.java` and `java WhoCar`, which outputs two lines of text: "madeIn variable in yourCar Class is Korea" and "madeIn variable in Car Class is Korea".

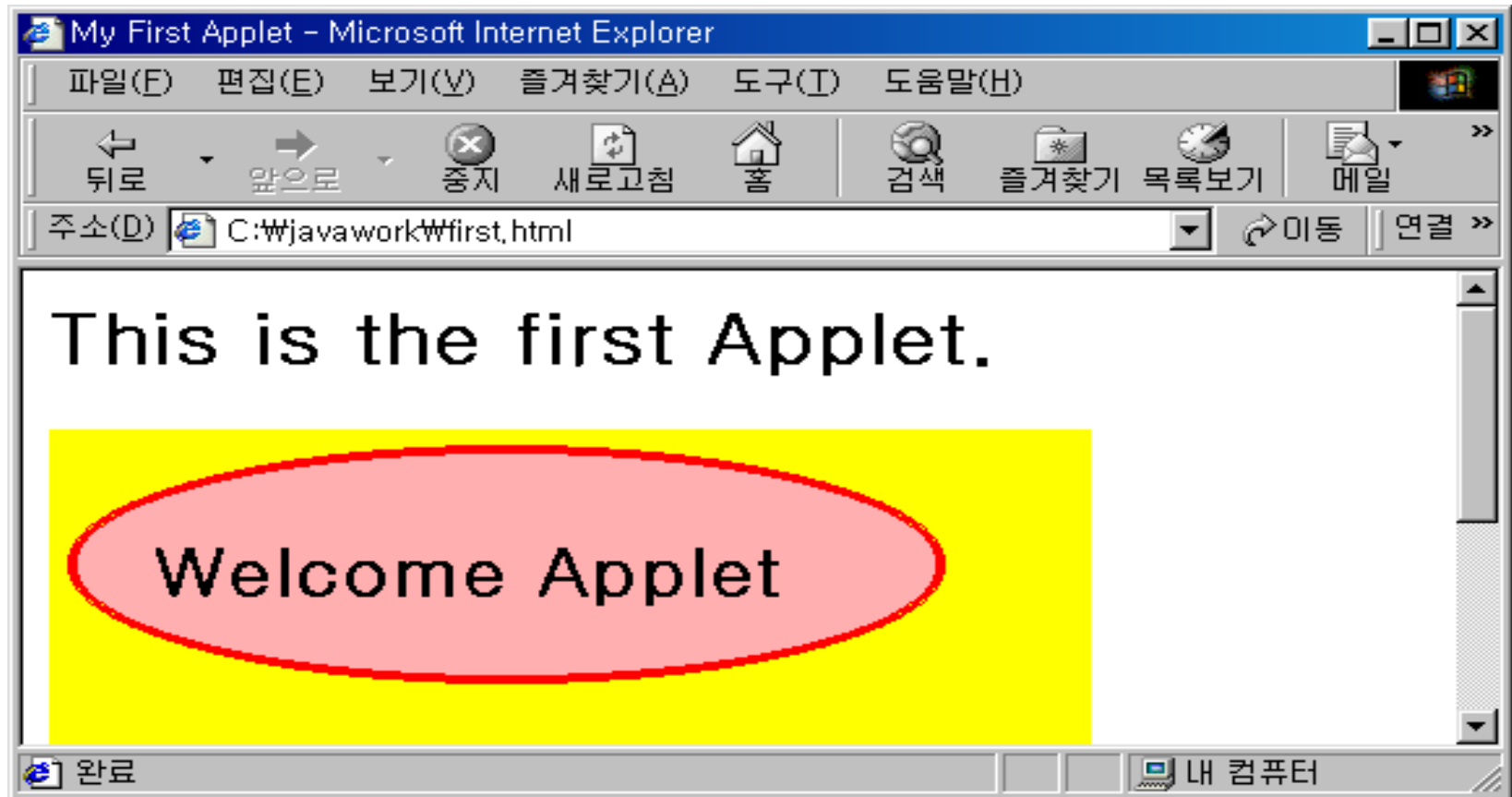
```
MS 선택 - 한글 MS-DOS
10 x 20
PROJECT4      <DIR>          01-04-20   13:18 Project4
PROJECT3      <DIR>          01-04-20   13:18 Project3
PROJECT2      <DIR>          01-04-20   13:18 Project2
PROJECT1      <DIR>          01-04-20   13:18 Project1
TEST          <DIR>          01-04-20   13:18 test
              37개 파일              25,003 바이트
              15개 디렉터리          10,770.59 MB 사용 가능

C:\javawork>javac WhoCar.java

C:\javawork>java WhoCar
madeIn variable in yourCar Class is Korea
madeIn variable in Car Class is Korea

C:\javawork>
```

애플릿의 예(웹브라우저 사용)



4. 자바 프로그래밍의 종류

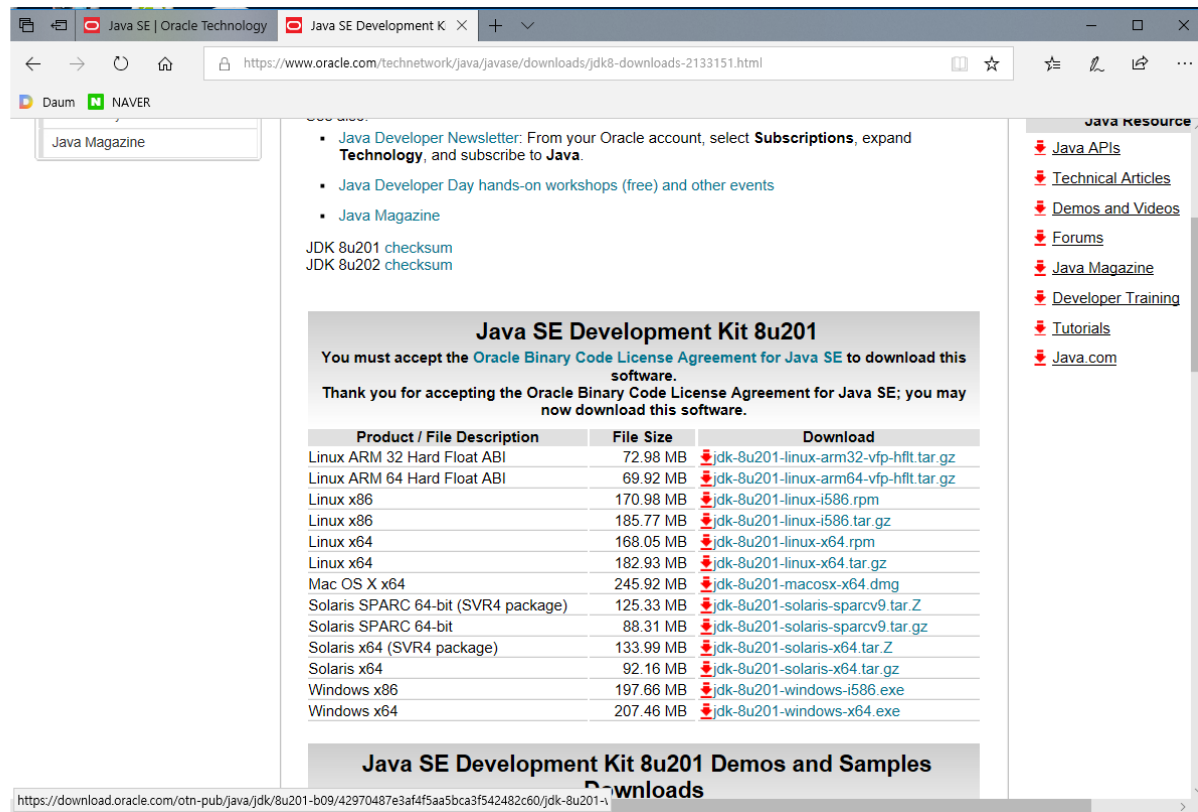
- 자바 애플리케이션: 독립적인 실행 형태로 가상머신이 위치한 곳에서 실행(스마트폰의 앱)
- 자바 애플릿: 클라이언트에서 웹 브라우저를 이용하여 실행
- 자바 서블릿: 웹서버에서 실행
- JSP(Java Server Page): 웹서버에서 실행되는 스크립트
- ** 자바스크립트: 클라이언트에서 실행되는 스크립트

5. 자바의 개발 환경

- Sun 사의 JDK를 이용하여 DOS 환경에서 컴파일 및 실행
- Borland사의 Jbuilder, WebGain사의 VisualCafe 등
- 이클립스 재단의 이클립스(eclipse) - 무료로 사용 가능
- 자바 개발환경의 설치:

JDK 설치

- Oracle사의 홈페이지에서 JDK를 다운 받는다 (java.sun.com 에서 윈도우 x86 버전 -> 32비트).



Java SE Development Kit 8u201

You must accept the [Oracle Binary Code License Agreement for Java SE](#) to download this software.

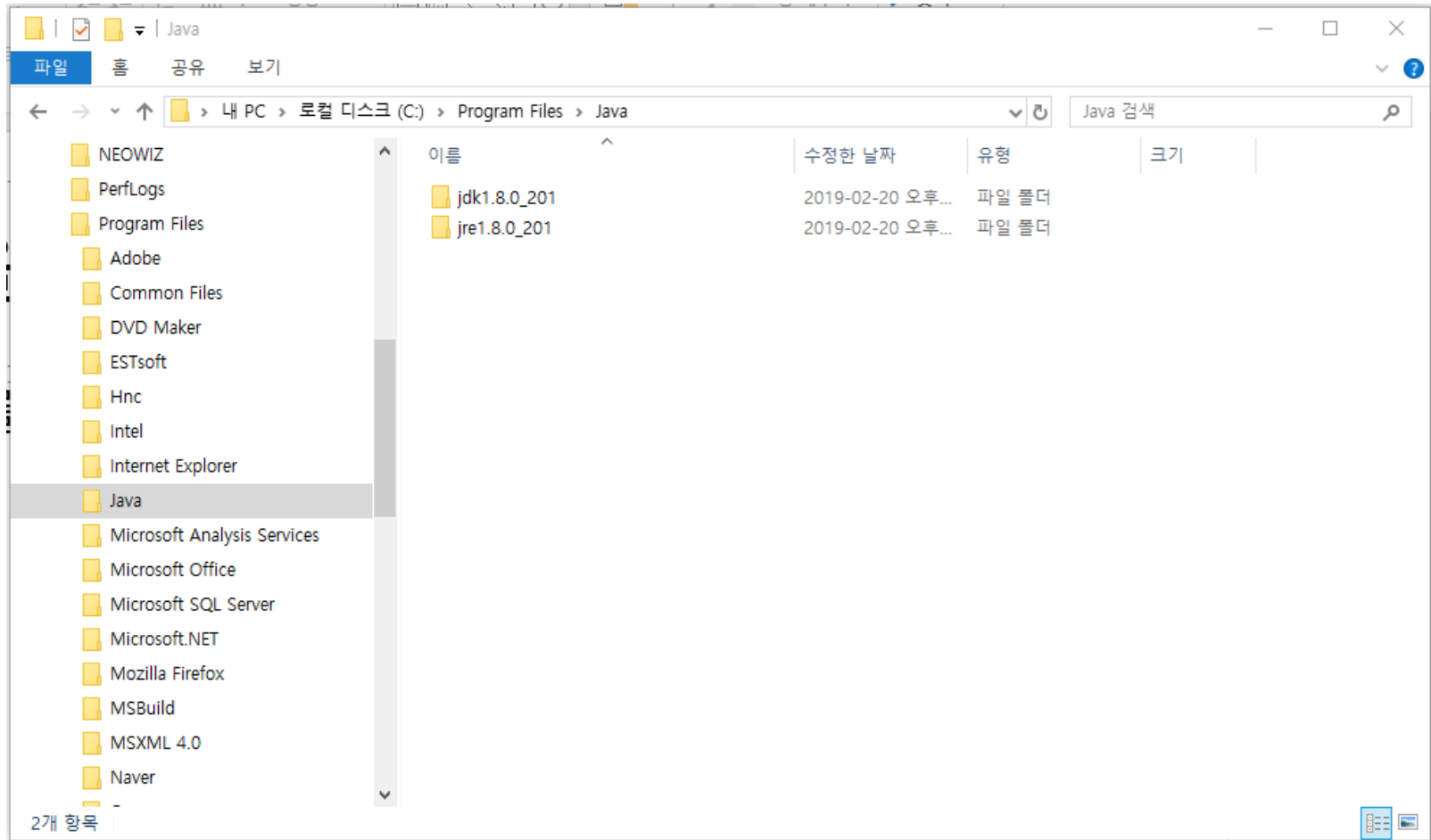
Thank you for accepting the Oracle Binary Code License Agreement for Java SE; you may now download this software.

Product / File Description	File Size	Download
Linux ARM 32 Hard Float ABI	72.98 MB	jdk-8u201-linux-arm32-vfp-hflt.tar.gz
Linux ARM 64 Hard Float ABI	69.92 MB	jdk-8u201-linux-arm64-vfp-hflt.tar.gz
Linux x86	170.98 MB	jdk-8u201-linux-i586.rpm
Linux x86	185.77 MB	jdk-8u201-linux-i586.tar.gz
Linux x64	168.05 MB	jdk-8u201-linux-x64.rpm
Linux x64	182.93 MB	jdk-8u201-linux-x64.tar.gz
Mac OS X x64	245.92 MB	jdk-8u201-macosx-x64.dmg
Solaris SPARC 64-bit (SVR4 package)	125.33 MB	jdk-8u201-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	88.31 MB	jdk-8u201-solaris-sparcv9.tar.gz
Solaris x64 (SVR4 package)	133.99 MB	jdk-8u201-solaris-x64.tar.Z
Solaris x64	92.16 MB	jdk-8u201-solaris-x64.tar.gz
Windows x86	197.66 MB	jdk-8u201-windows-i586.exe
Windows x64	207.46 MB	jdk-8u201-windows-x64.exe

Java SE Development Kit 8u201 Demos and Samples

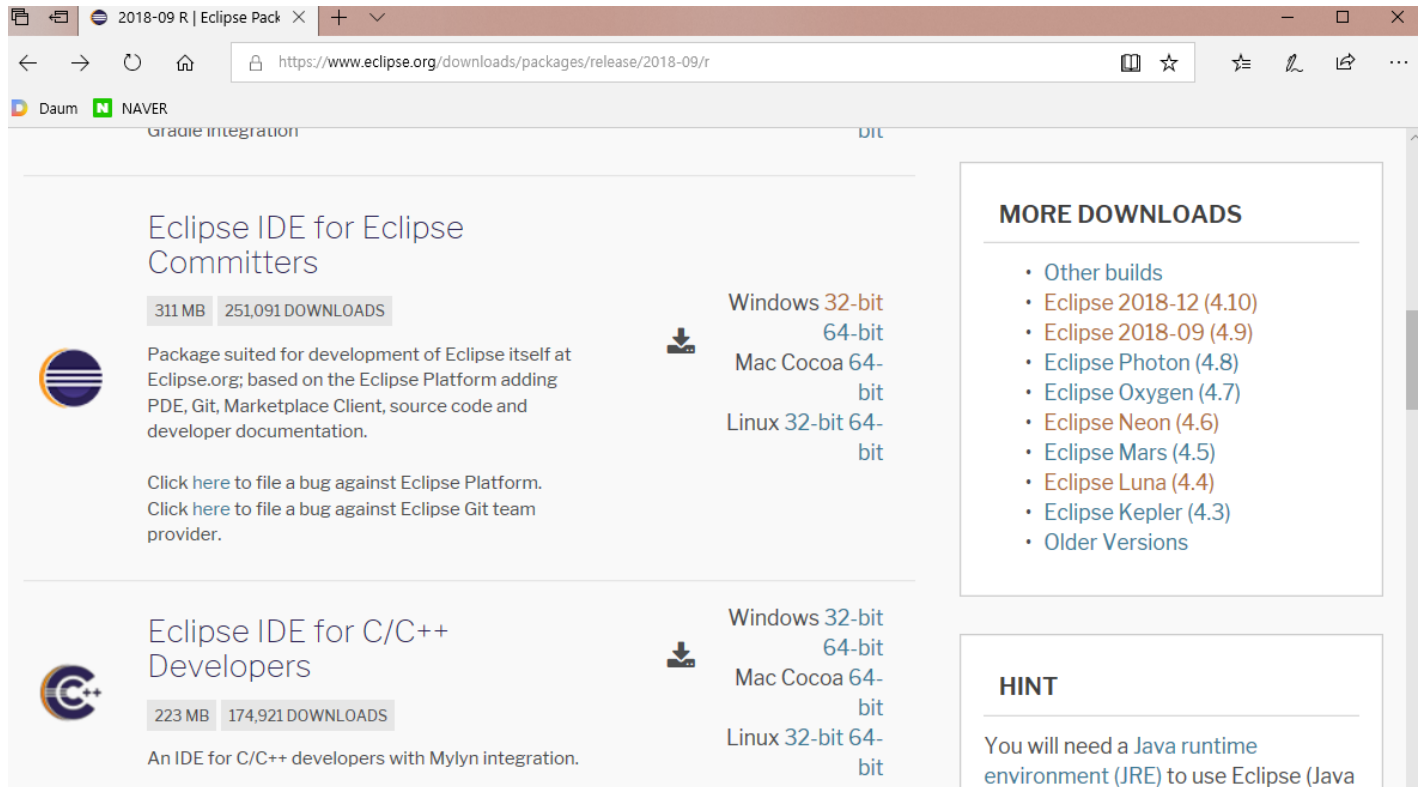
Downloads

- 설치 완료되면 C:\Program Files\Java 디렉터리에 두 개의 디렉터리가 생성



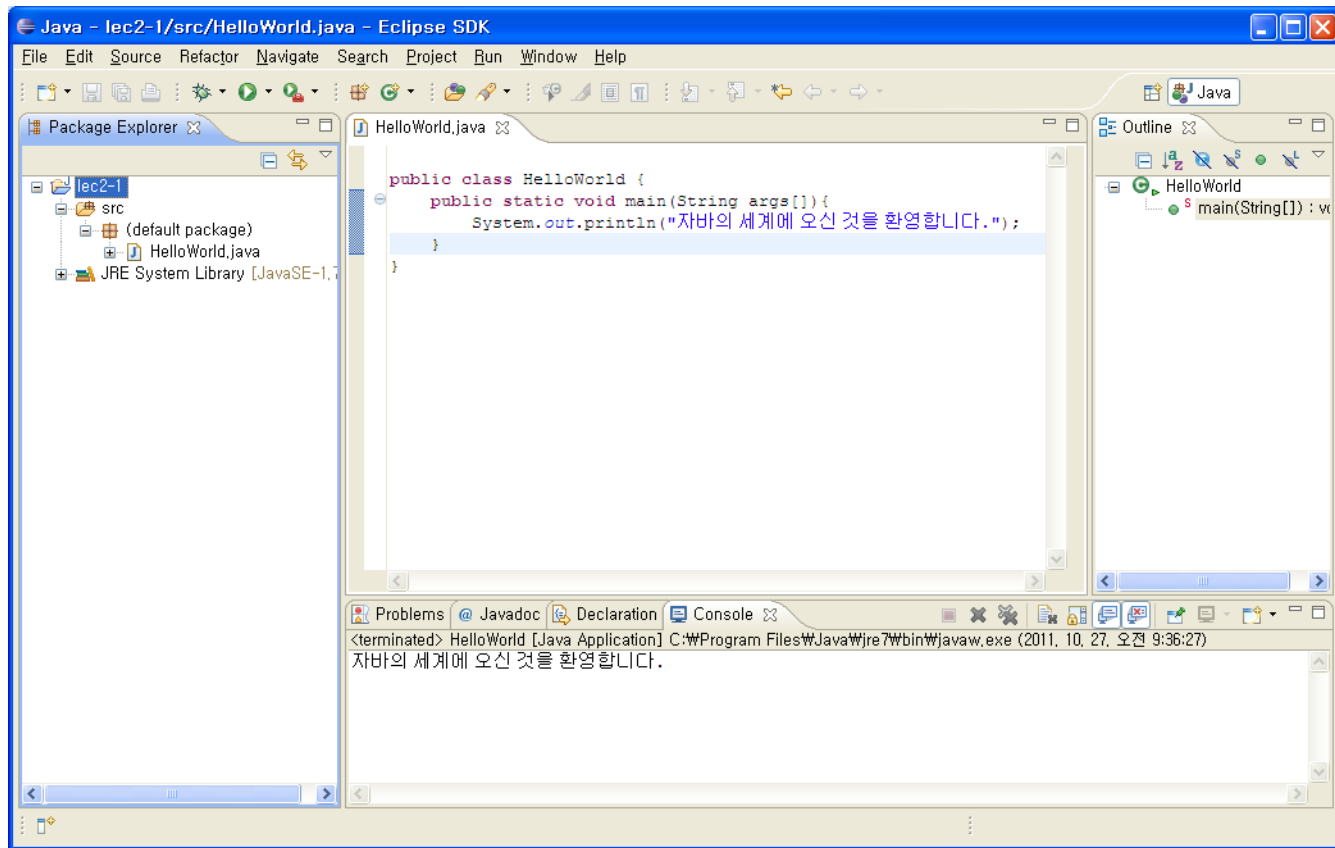
이클립스 설치

- 이클립스 홈페이지(www.eclipse.org) 방문
- Eclipse IDE Window 32 Bit 선택



이클립스 실행

- 압축파일을 푼 후에 eclipse 실행



표준입출력의 예

The screenshot displays the Eclipse IDE with the following components:

- Package Explorer:** Shows a project structure with packages 'lec2-1' and 'lec2-2'. Under 'lec2-2', there is a 'src' folder containing a '(default package)' and a file 'StandardIODemo.java'.
- Editor:** Displays the source code of 'StandardIODemo.java'. The code imports 'java.io.*' and 'java.util.StringTokenizer', defines a 'StandardIODemo' class, and implements a 'main' method that reads two integers from the user and prints them back.
- Outline:** Shows the class structure with 'import declarations', 'java.io.*', 'java.util.StringTokenizer', and the 'StandardIODemo' class with its 'main' method.
- Console:** Shows the output of the program execution. It indicates the application was terminated and shows the input and output messages.

```
import java.io.*;
import java.util.StringTokenizer;

public class StandardIODemo {

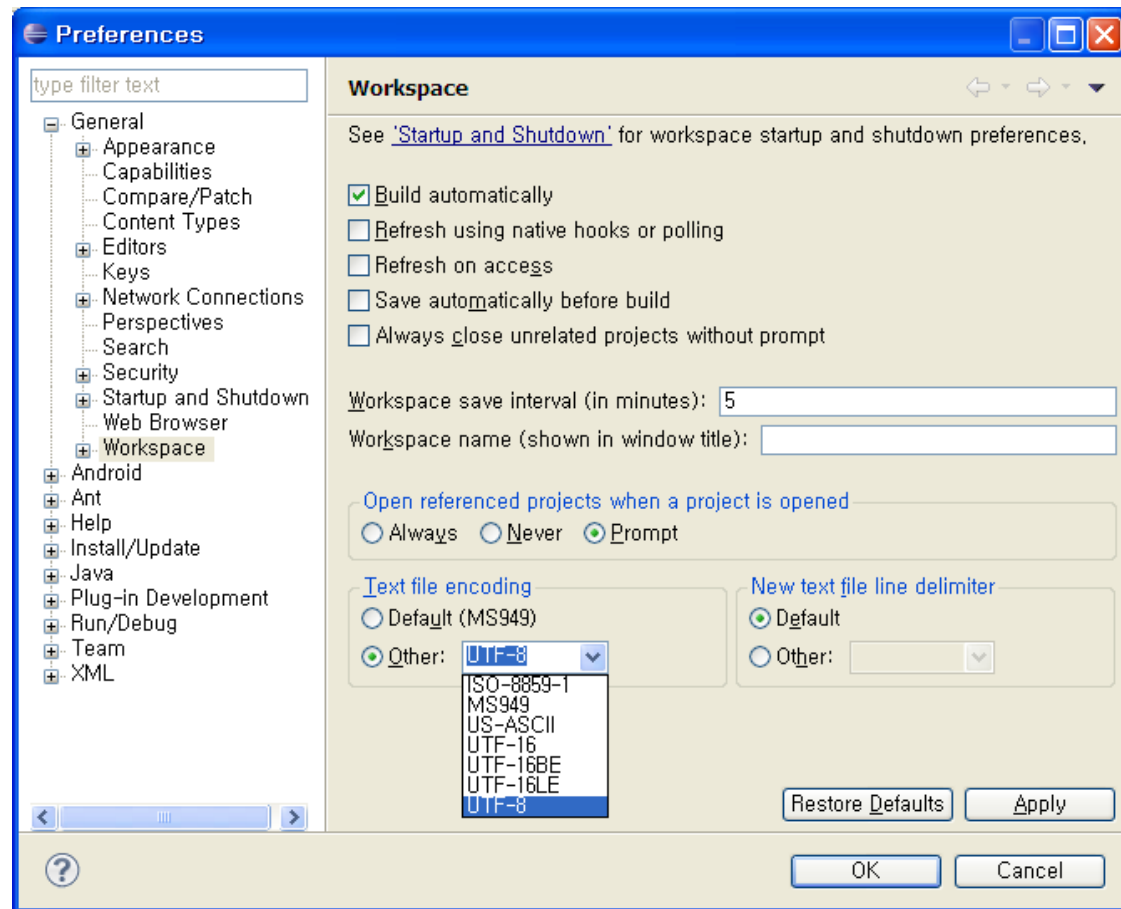
    public static void main(String args[]) {
        String input = "";
        BufferedReader in = new BufferedReader(
            new InputStreamReader(System.in));

        System.out.print ("Input two Integers: ");
        System.out.flush ();
        try {
            input = in.readLine ();
        } catch (IOException e) {
            System.out.println (e); }
        // Get tokens
        StringTokenizer st = new StringTokenizer(input);
        if (st.hasMoreTokens()) {
            String aToken1 = st.nextToken();
            int i = Integer.parseInt(aToken1) ;
            System.out.println("First integer is " + i);
        }
        if (st.hasMoreTokens()) {
            String aToken2 = st.nextToken();
            int j = Integer.parseInt(aToken2) ;
            System.out.println("Second integer is " + j);
        }
    } // end of main ()
}
```

Console Output:

```
<terminated> StandardIODemo [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (2011. 10. 27. 오전 9:45:13)
Input two Integers: 10 20
First integer is 10
Second integer is 20
```

- 강의자료 program 폴더의 샘플 프로그램을 이클립스에서 볼때 한글이 깨지면, <window 메뉴><preference 서브메뉴>후 아래의 팝업 창에서, <General><Workspace><Text file encoding> <Other>에서 UTF-8 을 선택후 OK 버튼을 누른다.



JAVA 프로그래밍 언어 개요

자바와 C++의 차이점

- 구조체 대신 오직 클래스만을 지원한다.
- 자바에서는 모든 것을 클래스에서 처리하므로 클래스 밖의 함수는 없다.
- 다중 상속을 지원하지 않으며, 인터페이스를 통해 다중 상속을 구현할 수 있다.
- 연산자 중복을 지원하지 않는다.

- 자바는 포인터를 객체의 참조 형태로만 지원한다.
- 포인터 연산을 지원하지 않는다.
- 메모리가 더 이상 필요하지 않을 때 자동 Garbage Collection을 수행한다.
- 자바 실행시 시스템은 모든 배열의 첨자가 배열내에 있는지 첨자를 검사한다.

1. 프로그램의 구성요소

- 프로그램의 구성

```
package test.example; // package 선언
import java.lang.*;    // import 문
import java.util.* ;

public class HelloWorld{

    public static void main(String args[]){

        System.out.println("자바의 세계에 오신 것을 환영합
니다.");

    }

}
```

주석

- 3가지 형식

`/* 주석 */`

`// 주석`

`/** 주석 */` javadoc에서 사용

식별자

- 변수, 상수, 배열, 클래스, 메소드 등의 이름
- 대소문자를 구별
- 예약어를 식별자로 사용할 수 없음
- 알파벳, \$, _, 숫자 만을 사용
- 첫글자로 0 ~ 9 의 숫자를 사용못함
- 공백을 포함할 수 없음
- 첫글자로 \$과 _을 사용할 수 있음

자료형 (8가지 기본자료형)

	데이터형	크기	데이터	설명
논리형	boolean	1 bit	true 또는 false	진리값
문자형	char	16 bit	₩u0000 ~ ₩uFFFF	유니코드 문자
정수형	byte	8 bit	-128 ~ 127	부호있는 정수
	short	16 bit	-32768 ~ 32767	
	int	32 bit	$-2^{31} \sim 2^{31}-1$	
	long	64 bit	$-2^{63} \sim 2^{63}-1$	
실수형	float	32 bit	Float.Min_Value ~	부동소숫점
	double	64 bit	Double.Min_Value ~	

문자열(string)

- c에서의 NULL 문자로 끝나는 문자형 배열이 아니라 java.lang.String 클래스의 인스턴스이다.
- null 종료문자로 끝나지 않는다.
- 연결 연산자로 +를 사용한다.
- 예:

```
String str = "Enter an integer value: ";
```

리터럴 (literal): 상수

- int형: 032(8진수), 32(10진수),
0x32(16진수)
- long형: 59L
- float형: 8.15f, 3.12E+12f
- double형: 2.75, 2.75d, 5.73E+12d
- boolean형: true, false
- char형: 'a', '\n', '\t', '\t\t'
- 문자열형: "Hello, Java!"

변수와 배열 선언

- 변수: `int i, j = 10;`
`float a;`
- 배열: 데이터 형이 아니라 객체이며,
배열들끼리 연산이 불가능
`int a[]; // 선언`
`a = new int[3]; // 생성 및 자동 초기화`
또는
`int a[] = {0, 0, 0};`
- 자동 초기화: `boolean`형(`false`), `int`와 `long`형(`0`),
그외(`null`)

형 변환(type cast)

- 자동 형 변환:

예: `int i; float f = 12.3;`
`i = f ;`

- 명시적 형 변환:

치역 = (type) 정의역;

예: `i = (int) f;`

2. 클래스

- 선언:

```
접근제한자 class 클래스이름 {  
    // 멤버 변수들  
    // 생성자  
    // 멤버 메소드들  
}
```

- 접근제한자:

public – 어떤 객체라도 접근 가능

final – 더 이상 서브 클래스를 만들 수 없음

abstract – 추상 클래스로 선언

예:

```
public class Date {  
    // 멤버 변수들  
    int day, mon, year;  
    // 멤버 메소드  
    public void Date(int d, int m, int y)  
        { day=d, mon=m, year=y;} // 생성자  
    public void setDay(int n)  
        { day += n; }  
}
```

객체의 생성과 소멸

- 생성:

`Date today = new Date(1, 1, 2000);`

- 자바는 자동적으로 쓰레기 수집을 하기 때문에 소멸자를 지원하지 않는다.
`finalize` 메소드라는 소멸자가 존재하나 시스템에 의해 쓰레기 수집 때 자동으로 호출된다.

멤버 변수와 메소드

- 형식:

접근제한자 변수명;

접근제한자 반환형 메소드명(파라미터){
 구현부;
}

- 접근제한자:

public, private, protected: C++과 같음

default: 같은 패키지 내의 클래스 (생략가능)

static: 모든 인스턴스에서 공유

final(변수): 초기값으로 고정됨(상수변수)

abstract(메소드): 추상 함수(C++에서 가상함수)

- static 변수: 해당 클래스의 모든 인스턴스들에 의해 공유되는 변수

```
class Thing {  
    static int count;    // static 변수  
    String name;  
    Thing(String name) {  
        this.name = name;  
        ++count; }  
}  
  
class StaticVariable {  
    public static void main(String args[]) {  
        Thing t1 = new Thing("Bowling Ball");  
        System.out.println(t1.name + " " + t1.count);  
        Thing t2 = new Thing("Ping Pong Ball");  
        System.out.println(t2.name + " " + t2.count);  
        Thing t3 = new Thing("Football");  
        System.out.println(t3.name + " " + t3.count); }  
}
```

- static 변수를 사용시 객체를 생성하지 않고 클래스명과 직접 사용 가능(전역 변수 기능)

```
class StaticBag {  
    static boolean flag;  
    static int i, j = 2, k = 3, l, m;  
    static double array[] = { -3.4, 8.8e100, -9.2e-100 };  
    static String s1, s2 = new String("Hello");}  
class StaticBagTest {  
    public static void main(String args[]) {  
        System.out.println(StaticBag.flag);  
        System.out.println(StaticBag.i);  
        System.out.println(StaticBag.j);  
        System.out.println(StaticBag.k);  
        for(int i = 0; i < StaticBag.array.length; i++)  
            System.out.println(StaticBag.array[i]);  
        System.out.println(StaticBag.s1);  
        System.out.println(StaticBag.s2);    }  
}
```


- **static** 메소드: 클래스의 객체와는 무관하게 동작한다.

```
class LinearEquation {  
    static double solve(double a, double b) {  
        return -b/a; }  
}  
  
class StaticMethod {  
    public static void main(String args[]) {  
        System.out.println(LinearEquation.solve(2, 2)); }  
}
```

- **public static void main(String[] args)**

main 메소드는 **static**으로 선언되었기 때문에 이것을 호출하기 위해 클래스의 인스턴스를 생성할 필요가 없다. 자바 프로그램 수행시 자바 인터프리터는 **main**이라고 선언된 메소드를 찾아 실행한다.

클래스 상속

- 단일 상속만 허용한다.

```
class derived-class extends super-class {  
...  
}
```

- 추상(abstract) 클래스는 반드시 하위 클래스를 가지며 추상 클래스를 상속받은 하위 클래스는 반드시 상위 클래스의 추상 메소드를 재정의 하여야 한다.

```
abstract class SumExample {  
    public int sum(int x, int y) {  
        return ( x+y ); }  
    abstract public void ab_method();  
}
```

```
class MyTest extends SumExample {  
    public void ab_method() {  
        int result;  
        result = sum(10, 20);  
        System.out.println(result); }  
    public static void main(String args[])  
        MyTest test = new MyTest();  
        test.ab_method();  
        System.out.println(test.sum(30, 40)); }  
}
```

메소드 오버로딩(중복)

```
class Car {  
    int speed;  
    int gearNum, carDoor;  
    String carName, carColor;  
    public void speedUp() {    // 일반 메소드의 중복  
        speed = speed + 100; }  
    public void speedUp(int turbo) {  
        speed = speed * turbo; }  
    public Car(String name) {    // 생성자의 중복  
        carName = name; }  
    public Car(String Color, String name)    {  
        carName = name;  
        carColor = Color; }  
    public Car(int Door) {  
        carDoor = Door; }
```

this와 super 키워드

- 그림자 변수(shadow variable):
부모 클래스와 자식 클래스에 같은 이름의 변수가 존재할 때, 자식 클래스에서 그 이름을 참조하면 자식 클래스의 변수가 사용된다. 이때 부모 클래스에 있는 변수를 그림자 변수라 한다.
- super: 부모 클래스의 변수나 메소드를 나타낼 때 사용(예: super.변수명, super.메소드명)
- this: 클래스 내에서 자기 자신의 변수나 메소드를 나타낼 때 사용

3. JAVA의 연산자 및 메소드의 구현

- 산술연산자: +, -, *, /, %
- 증가, 감소 연산자: ++, --
- 비교연산자: ==, !=, <, >, <=, >=
- 논리연산자: &&, ||, !
- 비트연산자: &, |, ^, ~, <<, >> 등
- 캐스트연산자: (type)
- 우선순위: 산술 > 비교 > 논리

메소드 구현

- 형식:

접근제한자 반환형 메소드이름(파라미터들) {
 몸체;
}

- 반환값이 없을 때 반환형으로 void를 사용
- 몸체 내에서 값을 반환할 때 return 문을 사용
- 배열을 반환할 때 반환형 다음에 []을 명시

```

class Range{
    int[] aRange(int lower, int upper) {
        int arr[] = new int[(upper - lower) + 1];
        for(int i=0; i < arr.length; i++){
            arr[i] = lower++; }
        return arr; }
    public static void main(String arguments[]){
        int theArray[];
        Range theRange = new Range();
        theArray = theRange.aRange(1,10);
        System.out.print("The array:");
        for (int i = 0; i< theArray.length; i++){
            System.out.print(theArray[i] + ""); }
        System.out.println(""); }
}

```

- 결과

The array:[1 2 3 4 5 6 7 8 9 10]

메소드의 인수 전달

- Call By Value: 메소드를 호출할 때 인수의 값이 복사되어 파라미터에 전달된다. 즉 메소드 내에서 파라미터의 값을 변경하더라도 인수값은 변경되지 않는다.
- 그러나 배열이나 객체가 메소드에 전달될 때는 배열과 객체의 레퍼런스(주소)가 복사되어 전달되므로 메소드 내에서 파라미터를 변경하면 인수값도 변경된다.

```

class CallByValue {
    public static void main(String args[]) {
        int i = 5;
        int j[] = { 1, 2, 3, 4 };
        StringBuffer sb = new StringBuffer("abcde");
        display(i, j, sb);
        a(i, j, sb);
        display(i, j, sb); }
    static void a(int s, int t[], StringBuffer st) {
        s = 7;
        t[0] = 11;
        st.append("fghij"); }
    static void display(int i, int j[], StringBuffer sb) {
        System.out.println(i);
        for(int index = 0; index < j.length; index++)
            System.out.print(j[index] + " ");
        System.out.println("");
        System.out.println(sb); }
}

```

앞 프로그램의 결과

5

1 2 3 4

abcde

5

11 2 3 4

abcdefghijkl

6. 클래스 생성자

- 상위 클래스의 생성자는 상속되는 것이 아니라, 하위 클래스의 생성자가 호출될 때 그 클래스의 상위 클래스가 있으면 자동적으로 호출된다.
- 이때 상위 클래스의 생성자가 하위 클래스의 생성자보다 먼저 실행된다.
- 클래스에 생성자가 정의되어 있지 않으면 컴파일러가 자동으로 디폴트 생성자를 생성한다.
- 부모 클래스의 생성자에 파라미터가 있는 경우, 하위 클래스의 생성자를 정의하지 않으면 문제가 발생하며 하위 클래스의 생성자 내부에서 부모 클래스의 생성자를 파라미터와 함께 호출해 주어야 한다.

- 예러

```
class Car {  
    protected String name; //자동차의 이름을 나타내는 변수  
    protected int speed;    //속도를 나타내는 변수  
    protected int doorNum; //자동차의 문의 수를 나타내는 변수  
    public Car(String CarName) {  
        name = CarName; }  
    public void speeddown() {    //속도를 올리는 함수  
        speed = speed - 20;}  
    :  
}  
class subCar extends Car {  
    public static void main(String args[]) {  
        subCar myCar;  
        myCar = new subCar();  
        System.out.println(myCar.name); }  
}
```

- 해결방법 1: 부모 클래스에 디폴트 생성자를 둔다.

```
class Car {  
    protected String name; //자동차의 이름을 나타내는 변수  
    protected int speed;    //속도를 나타내는 변수  
    protected int doorNum; //자동차의 문의 수를 나타내는 변수  
    public Car() { }        // 디폴트 생성자  
    public Car(String CarName) {  
        name = CarName; }  
    public void speeddown() {    //속도를 올리는 함수  
        speed = speed - 20;}  
    :  
}
```

- 또는

```
public Car() {    // 디폴트 생성자 대신 매개변수 없는 생성자  
    this("붕고"); }
```

- 해결방법 2: 자식 클래스에서 부모 클래스의 생성자를 호출

```
class subCar extends Car {  
    public subCar() {  
        super("봉고");  
    }  
    public static void main(String args[]) {  
        subCar myCar;  
        myCar = new subCar();  
        System.out.println(myCar.name); }  
}
```

기타 중요한 클래스

- Object 클래스
- 래퍼(wrapper) 클래스:
Boolean, Byte, Character, Short,
Integer, Long, Float, Double

** 오토 박싱(auto boxing)

기본 자료형 데이터 → 대응 래퍼 클래스 객체로 자동 변환
하는 것


```
import java.util.*;
public class Wrapper {
    public static void main(String args[]){
        boolean a = true;
        Boolean wrapperA = new Boolean(a);
        int b = 300;
        Integer wrapperB = new Integer(b);
        float c = (float)300.3;
        Float wrapperC = new Float(c);
        System.out.println("WrapperA 값은 "+wrapperA.toString());
        System.out.println("WrapperB 값은 "+wrapperB.intValue());
        System.out.println("WrapperC 값은 "+wrapperC.floatValue());}
}
```

결과: WrapperA 값은 true
WrapperB 값은 300
WrapperC 값은 300.3

표준 입출력

- `java.lang.System` 클래스를 통해 이루어진다.
- 표준 입력: 키보드, `System.in`을 사용
출력: 화면, `System.out`을 사용
- 문자 스트림과 바이트 스트림을 사용할 수 있다.

표준입출력의 예(lec2-1)

```
import java.util.Scanner;    // 표준 입력 클래스
public class StandardIODemo {
    public static void main(String args[]) {
        int num;
        int arr[] ;
        String str ;
        Scanner sc = new Scanner(System.in);    // 표준 입력 클래스 생성
        System.out.println ("Input number of Integers: ");
        num = sc.nextInt();
        arr = new int[num] ;
        System.out.print ("Input " + num + " Integers: ");
        for (int i=0; i < arr.length; i++)
            arr[i] = sc.nextInt();
    }
}
```

```

    System.out.print ("Inputed Integers: ");
    for (int i=0; i < arr.length; i++)
        System.out.print(arr[i] + " ");
    System.out.println ("Input a Sring : ");
    str = sc.next();    // 하나의 문자열 읽기
    System.out.println("The inputed string is "+ str);
} // end of main ()
}

```

실행결과:

Input number of Integers:

3

Input 3 Integers: 10 20 30

Inputed Integers: 10 20 30

Input a Sring :

Algorithm

The inputed string is Algorithm

** 파일 입력의 예도 lec2-1 예제 프로그램에 포함되어 있음

레포트#1

- n 개의 정수를 입력 받아 오름차순으로 정렬하여 출력하는 자바프로그램을 작성하시오.

실행 예:

n 값 입력: 5

5 개의 정수 입력: 30 50 40 10 20

정렬결과 : 10 20 30 40 50

- 제출물 - 프로그램 소스(주석 포함)
실행결과(화면 캡취)