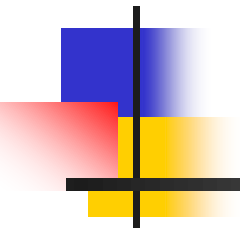


JAVA Programming 기초

경복대학교
소프트웨어융합과
배희호 교수

JAVA Program

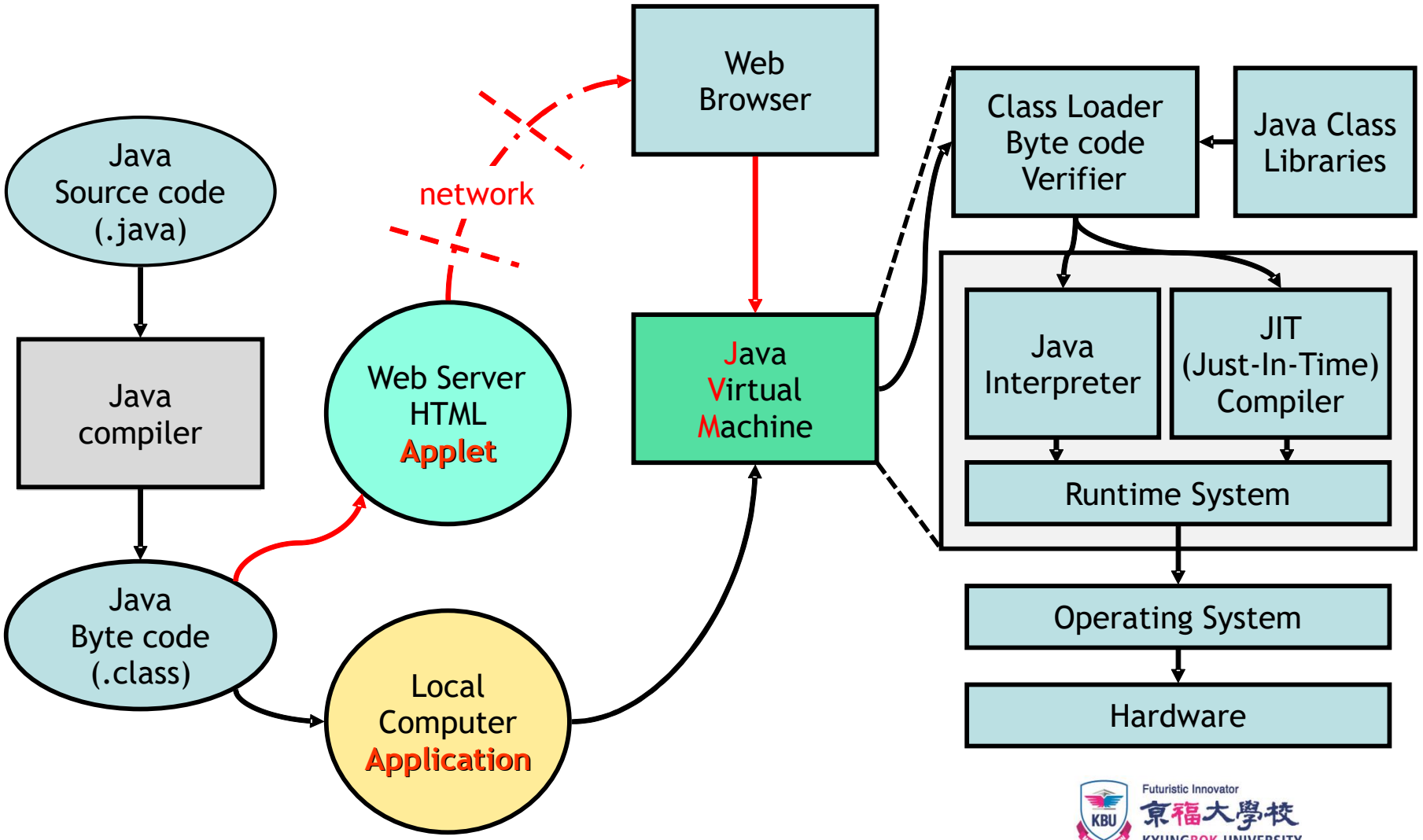
- Application과 Applet
  - Application
    - 일반 Application Program과 유사하게 동작
    - Program Code가 저장되어 있는 PC에서 실행
    - 모든 자원을 사용할 수 있으므로 실행 당사자가 유의함
    - C나 C++ Program과 같은 일반적인 응용 Program
    - Byte Code로 번역된 후에 검색기나 애플릿 뷰어 (appletviewer)를 이용하지 않고 바로 실행

JAVA Program

- Application과 Applet
  - Applet
    - Web Browser에 동작하는 Program
    - Program Code는 Web Server에 저장
    - 실행은 Web Server로 부터 HTML 문서와 함께 전달된 Applet은 PC에서 실행
    - HTML 문서에 의해서 자동 실행
    - 악의적인 Code로부터 보안 장치 있음
    - WWW 검색기나 AppletViewer에서 실행되는 Program
    - 작고 간단한 응용에 많이 사용
    - Network을 통하여 실행될 수 있으므로 개발 환경과 실행 환경의 독립성이 보장

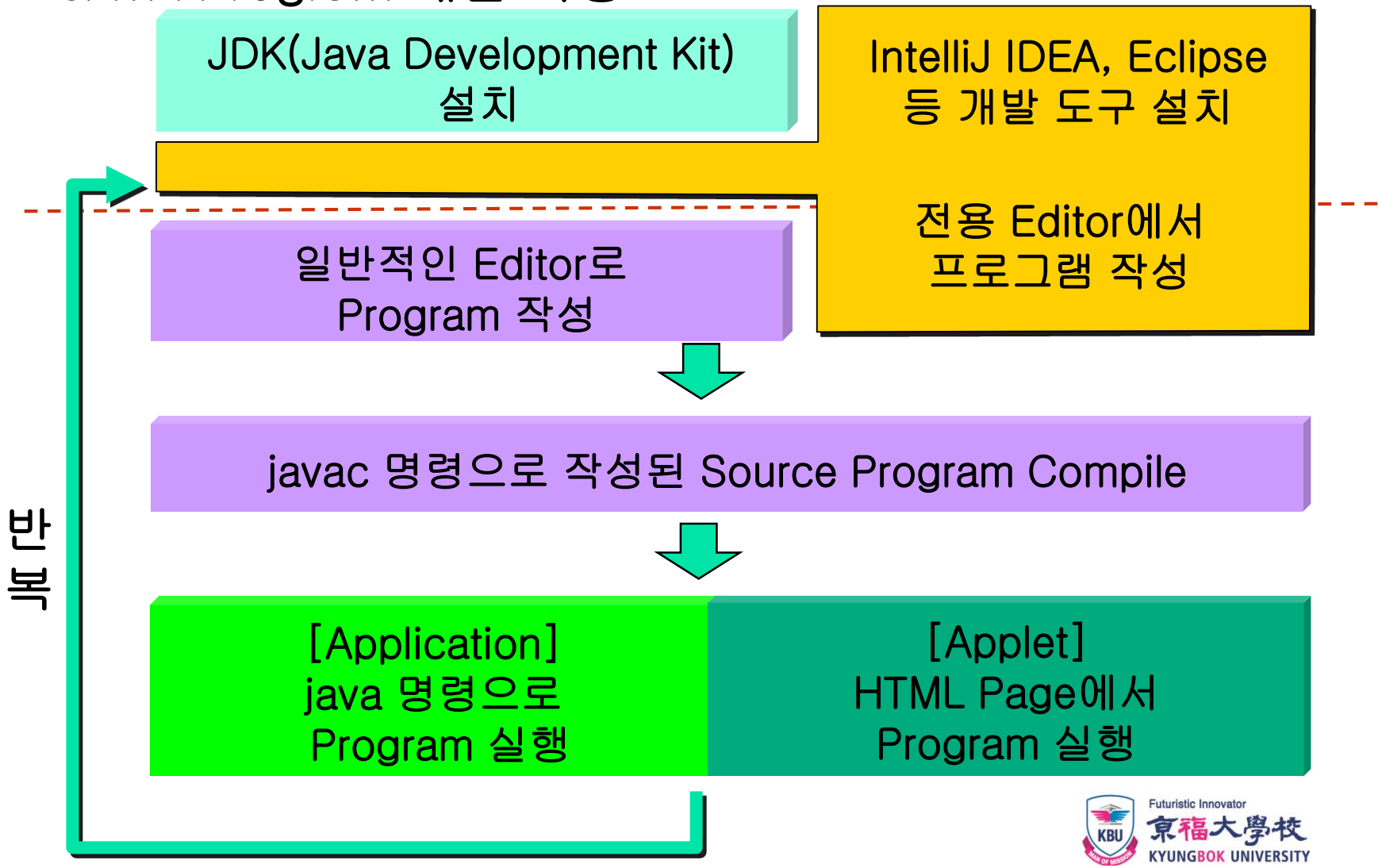
JAVA Program

- Application과 Applet의 동작



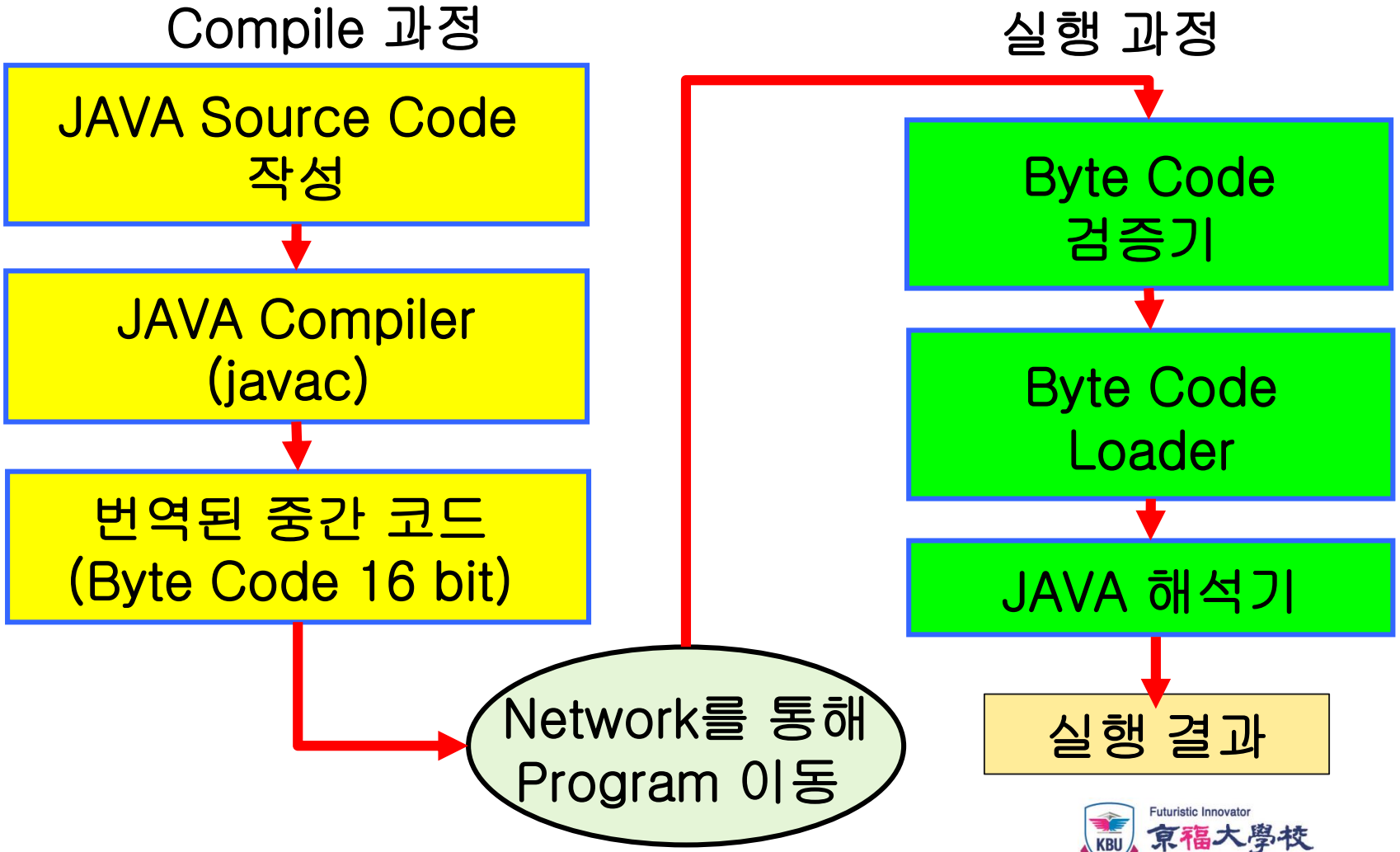
JAVA Program

- JAVA Program 개발 과정



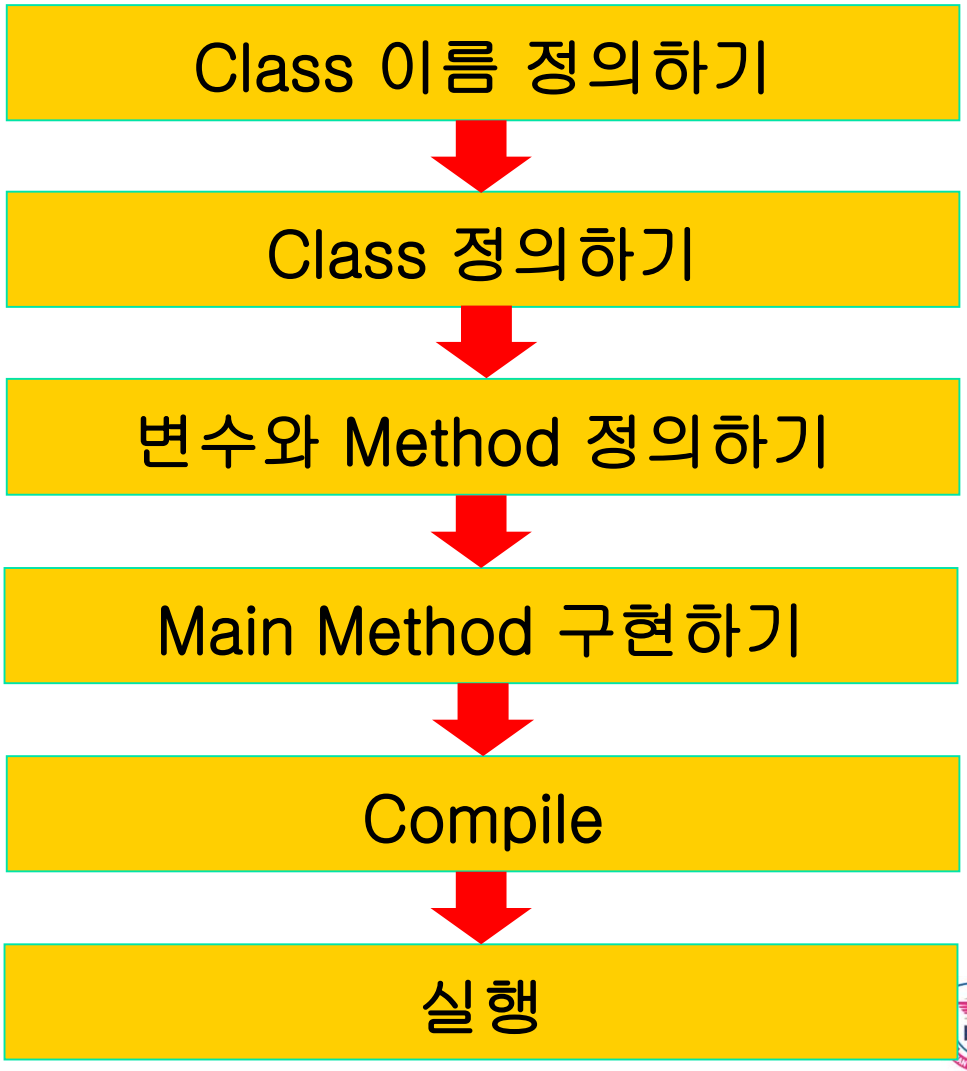
JAVA Program

- JAVA Program의 실행 과정



JAVA Program

- JAVA Programming 순서



JAVA Program

- JAVA Programming 순서
  - [1단계] Class 이름 정하기
    - Class 이름은 반드시 대문자로 시작해야 함
    - Class 이름과 Program Source File 이름은 반드시 같아야 함
    - File 확장자는 소문자 *.java
    - JAVA는 대소문자를 엄격히 구분

FirstProgram.java

Class 이름 = Source File 이름

JAVA Program

- JAVA Programming 순서
  - Class 생성
    - JAVA에서는 Program의 최소 단위는 Class 임
    - Class는 하나의 “Program”, “기능”이라 생각하면 됨
    - Program이라 불리우는 Class를 생성하기 위해서 Project를 생성 했던 것

JAVA Program

- JAVA Programming 순서
  - Source File과 Class 이름
    - JAVA에서 Source File 이름과 Class 이름은 상당한 관련이 있음
      - Source File안에 public class가 있다면 반드시 Source File 이름은 public class 이름과 일치하여야 함
      - 만약 하나의 Source File 안에 public class가 없다면, Source File 안에 포함된 어떤 Class 이름으로 하여도 상관 없음
      - 하나의 Source File안에 public class가 2개 이상 있으면 Compile 오류가 발생

JAVA Program

- JAVA Programming 순서
  - [2단계] Class 정의하기
    - JAVA에서 class의 첫 번째 글자는 모두 대문자임
    - Source File 안에는 Class 정의
    - Class나 Method는 {}로 실행 범위를 지정

```
public class FirstProgram {



}
```

JAVA Program

- JAVA Programming 순서
  - [3단계] 변수와 Method 정의하기

```
public class FirstProgram {
    String hello = "안녕하세요. 첫 프로그래밍입니다.";
    public static void print( ) {
        System.out.println(hello);
    }
}
```

변수 → `String hello = "안녕하세요. 첫 프로그래밍입니다.";`

메소드 → `public static void print( ) { System.out.println(hello); }`

- 문장의 마지막에 세미콜론(;)을 붙여 줌
- JAVA는 띄어쓰기 등에 영향 받지 않음

JAVA Program

- JAVA Programming 순서
  - [4단계]: Program이 시작되는 main Method 구현

```
public class FirstProgram {
        String hello = "안녕하세요. 첫 프로그래밍입니다.";
        public void print( ) {
                System.out.println(hello);
        }
        public static void main( String[ ] args) {
                FirstProgram fp = new FirstProgram( );
                fp.print( );
        }
}
```

main 메소드

JAVA Program

- JAVA Programming 순서
  - [5 단계]: Compile
    - Source Directory에서 명령 Prompt 실행 후 javac 실행
    - 정상적으로 Compile이 완료되면 “*.class” 파일 생성

```
C:/>javac FirstProgram.java
```

  - [6 단계]: 실행
    - Compile 완료 후 java로 실행

```
C:/>java FirstProgram.class
```

  - [7 단계]: 실행결과 확인

```
안녕하세요. 첫 프로그래밍입니다.
```

JAVA Program

- JAVA Compiler

```
public class HelloWorld {
        public static void main(String [ ] args) {
                System.out.println("Hello World");
        }
}
```

HelloWorld.java
소스 코드

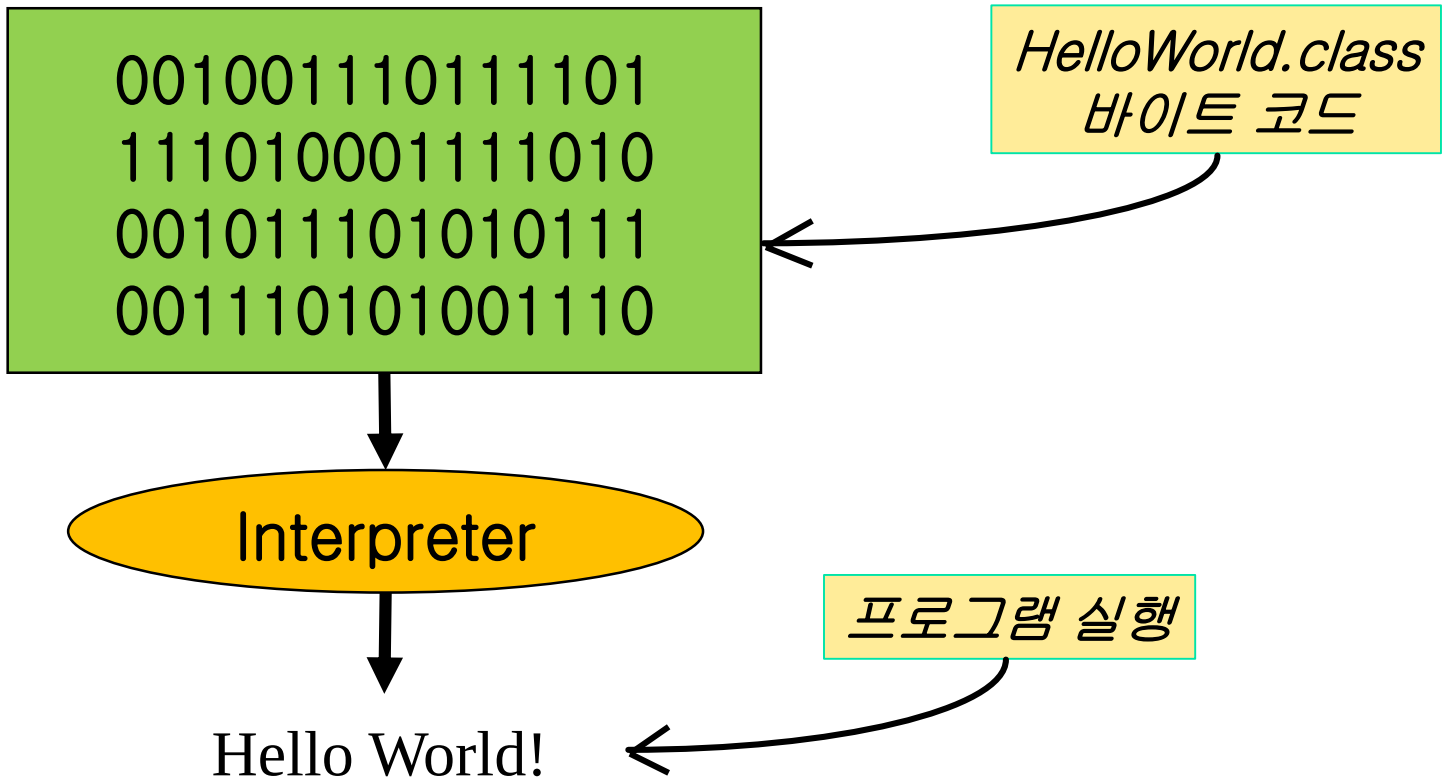
Compiler

```
001001110111101
111010001111010
001011101010111
001110101001110
```

HelloWorld.class
바이트 코드

JAVA Program

- JAVA Interpreter



JAVA Program의 구조


```
public class Dog {
    void bark() {
        statement1;
        statement2;

    }
}
```

문장

JAVA Program의 구조

- Hello.java Program

Hello.java

```
01 public class Hello
02
03     public static void main(String[] args)
04         System.out.println("Hello World!");
05     }
06 }
```

클래스를 정의하는 문장이다.

메소드를 정의하는 문장이다.


```
Hello World!
```

JAVA Program의 구조

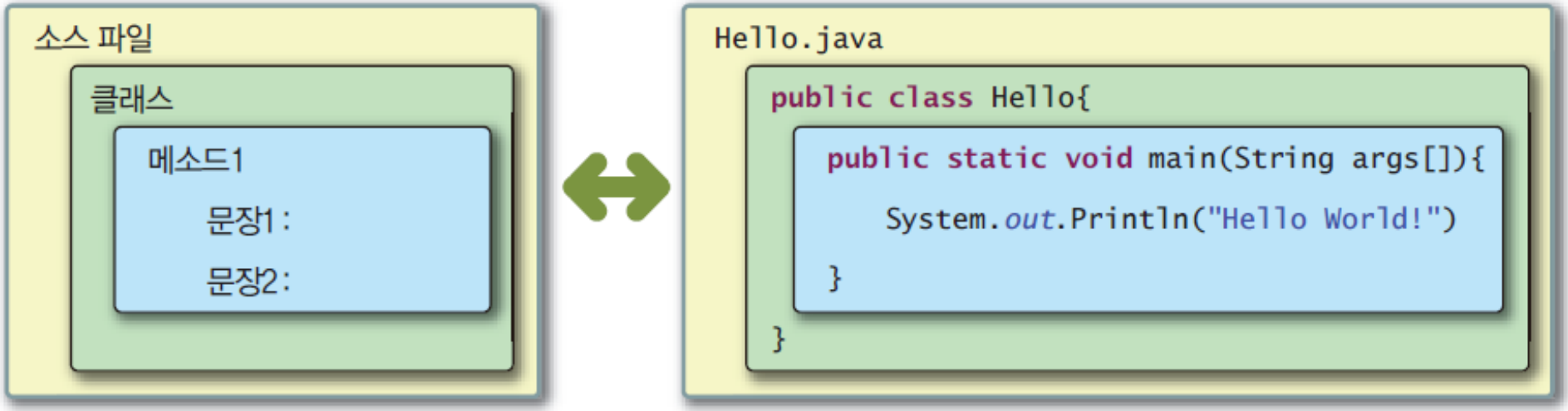
- Hello.java를 구성하는 요소
  - main() Method
  - String 배열 String[]
  - main() Method의 매개변수 args
  - static 키워드
  - System.out.println();
  - “Hello World!” 문자열

JAVA Program의 구조

- Class 정의
  - Class는 JAVA와 같은 객체 지향 언어의 기본적인 빌딩 블록
  - 필요한 Class를 하나씩 만들어 감으로써 전체 Program이 완성

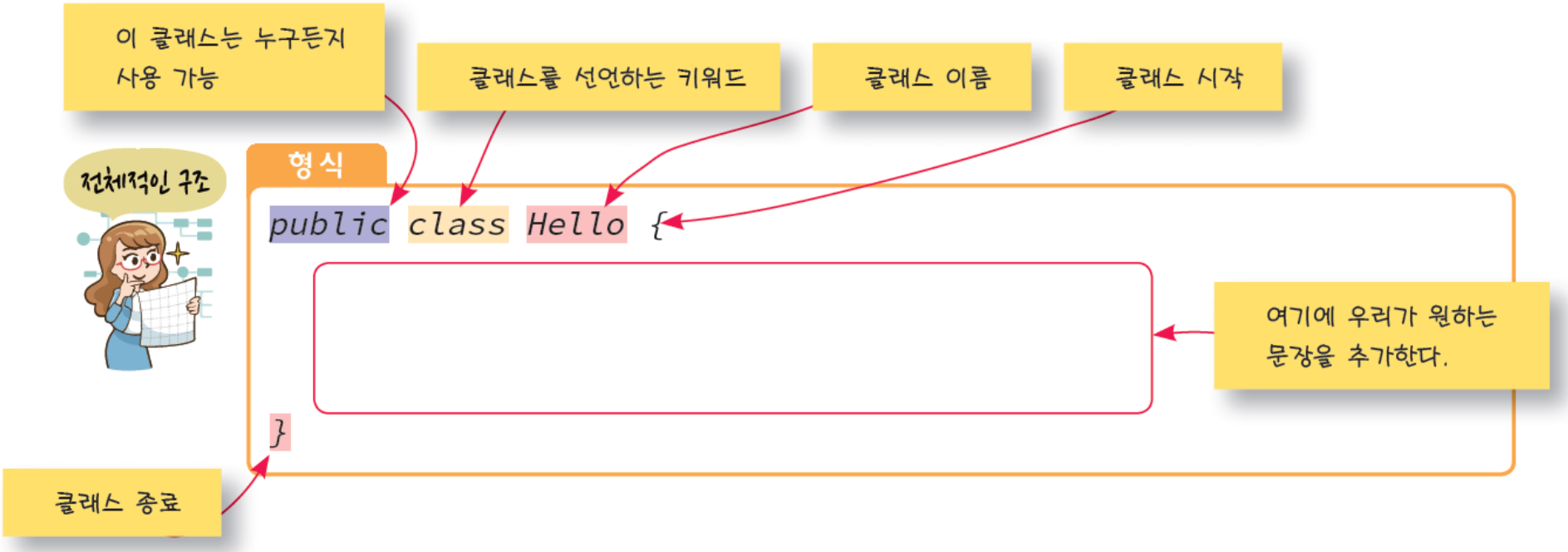
JAVA Program의 구조

- Class 정의
  - 객체를 만드는 설계도(추후에 학습)
  - JAVA Program은 Class들로 구성
  - JAVA Program에는 적어도 하나의 Class는 반드시 필요



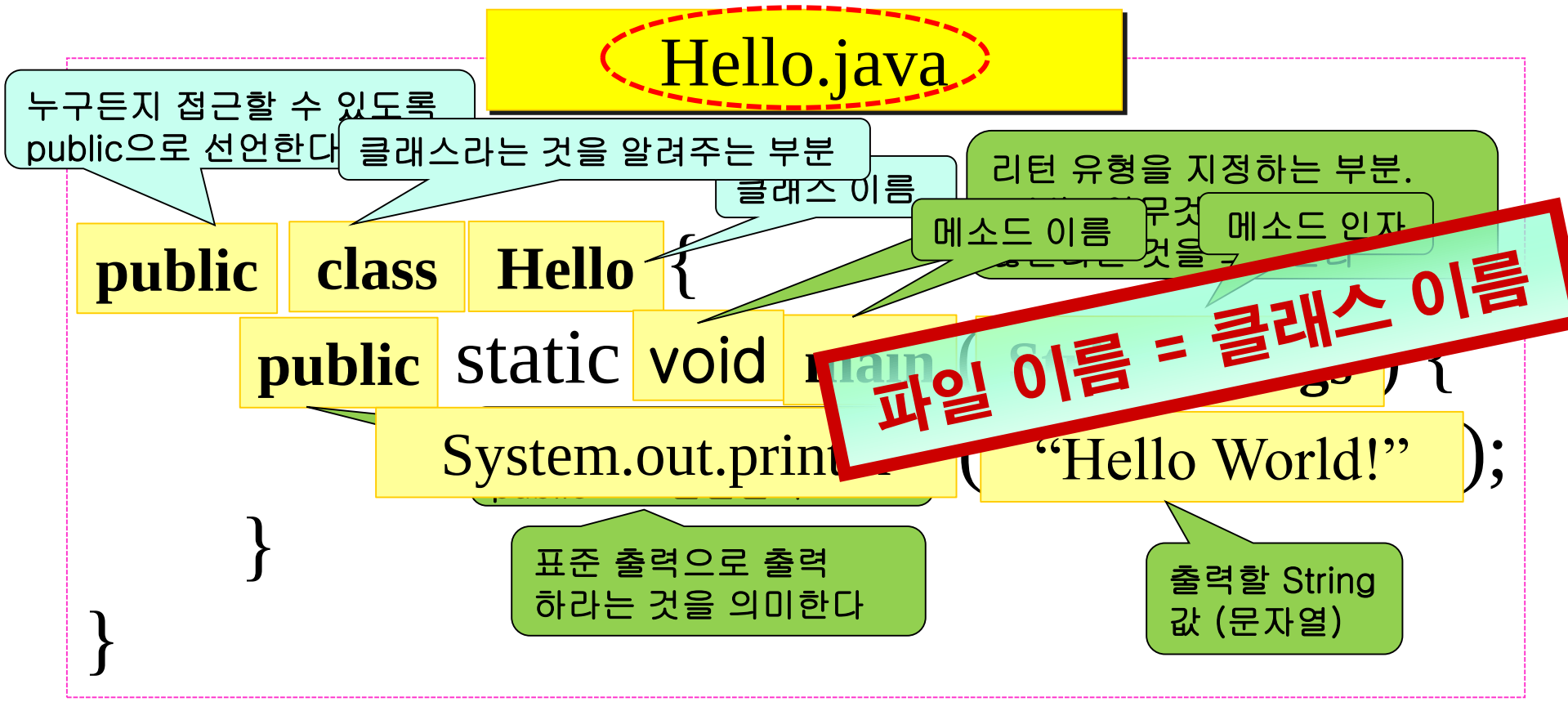
JAVA Program의 구조

- Class 정의
  - Source 안에 public class가 있다면 반드시 Source File의 이름은 public class의 이름과 일치하여야 함
  - 하나의 Source File 안에 public class가 2개 이상 있으면 Compile 오류가 발생



JAVA Program의 구조

- Class 파헤쳐 보기



JAVA Program의 구조

- Class 파헤쳐 보기
  - JVM이 실행되면 가장 먼저 하는 일
    - Class에서 다음과 같은 부분을 찾음

```
public static void main (String[] args) {
                        // 코드가 들어갈 자리
}
```

    - 이 main Method를 실행

  - 모든 JAVA Application Program에는 한 개 이상의 Class가 있어야 하며, 한 개의 main Method가 있어야 함

JAVA Program의 구조

- Class 파헤쳐 보기
  - public class Hello {
    - Hello라는 Class를 정의함
    - Class body는 '{'로 시작해서 '}'로 끝냄
  - public static void main(String args[]) {
    - public은 다른 Class에 공개되었음을 나타냄
    - static은 main( ) Method가 static Method 임을 나타냄
    - void는 main( )의 반환 값이 없음을 나타냄
    - main( ) Method는 Hello Class의 주 메소드임을 표시
    - String형 배열인 args[]는 명령행 인자를 보관하는 문자열 배열임
      - 메소드를 호출할 때 같이 넘겨 주겠다는 의미

JAVA Program의 구조

- 왜 main Method의 인자(parameter)는 2개인가?
  - main() Method는 Program을 실행하기 운영체제가 호출하는 첫번째 Method
  - 따라서, 운영체제는 프로그램을 실행할 때 사용자가 입력하는 부가적인 옵션(option 혹은 parameter)을 main() Method에 전달할 수 있어야 함
  - Program의 인자는 명령행(command line)에서 사용자가 Keyboard를 이용해 Typing하는 것이 일반적이고 그 형태는 복수의 문자열을 공백(space)으로 구분해서 입력
  - 즉, 0(zero)개 이상의 문자열 배열 형태라고 표현할 수 있다. 인자의 개수가 일정하지 않기 때문에 인자의 개수와 인자 값(문자열)의 목록(혹은 배열)을 함께 전달하며, 인자의 개수와 인자 배열을 순서대로 전달하기 위해 정수형의 argc와 문자열 포인터 배열 형태의 argv 변수를 사용

JAVA Program의 구조

- main() Method의 argc 그리고 argv
  - argc 변수의 공식적인 명칭(full name)은 'argument counter'이고, argv 변수는 argument variable 혹은 argument vector
  - ac, av 라던가 argumentcounter, argumentvariable 이라는 극단적으로 길거나 짧지 않은 변수 명칭을 사용하는 이유는 너무 짧아서 모호한 변수 명칭도 배제하고, 너무 길어서 코딩하는데 걸리는 시간을 허비하거나 오타 때문에 고생하지 않도록 실용적인 접근을 시도할 것
  - 의미를 담을 수 있고 최대한 유니크한 짧은 약어(abbreviation)를 이용해 변수 명칭을 부여하는 것

JAVA Program의 구조

- Method의 정의
  - public: 누구나 이용할 수 있음
  - static: 정적 메소드(9장 부근에서 학습)
  - void: 반환값이 없음
  - main: 메소드 이름
  - String args[]: 매개 변수(메소드가 외부에서 받는 데이터)

```
/**
* 표준 출력으로 "Hello World!"를 표시하는 간단한 자바 애플리케이션의 구현이다.
*/
public class Hello {
    public static void main(String args[]) {
        System.out.println("Hello World!"); // 문자열 출력
    }
}
```

메소드 정의

JAVA Program의 구조

- main() Method
  - JAVA 응용 프로그램에 반드시 하나 만 있어야 하는 특수한 Method
  - C나 C++ 언어의 main() 함수와 같은 역할
  - Program 실행 시 자동으로 실행되는 Method
  - 일반적으로 JAVA 응용 Program은 main() Method 내에서 다른 Class의 객체를 생성한 다음 그 객체에 메시지를 보내어 원하는 결과를 얻음

```
public static void main(String args[]) {

                ……

}
```

JAVA Program의 구조

- main() Method
  - Program의 첫 시작점
  - 실행이 끝나면 Program도 종료
  - JAVA의 main() Method는 형식화 되어있음
  - 매개변수의 이름이외에는 바꿀 수 있는 것이 없음
  - main() Method를 포함하는 Class를 실행 Class라고 함
- main( ) Method에서 할 수 있는 일
  - 뭔가를 준비시킴 (선언문)
  - 뭔가를 반복해서 수행 (순환문)
  - 조건에 따라 뭔가를 수행 (분기문)

JVM

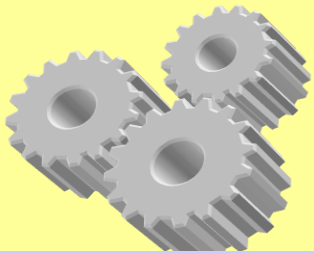
나는 main() method를
제일 먼저 실행

JAVA Program의 구조

MainTest.java **(main메서드의 특징을 테스트한 예제)**

```
public class MainTest {
3   public void sayHello() {
        System.out.println("Hello World!");
    }
1   public static void main(String[] args) {
2       MainTest my = new MainTest();
        my.sayHello();
    }
}
```

1 자바 가상 머신



MainTest.java를 실행하면 자바 가상 머신은 main() 메소드를 실행

2 main()메소드

```
MainTest my=new MainTest();
my.sayHello();
```

데이터 타입이 MainTest인 my객체를 생성
생성된 객체로부터 sayHello()메소드 호출

3 MainTest

sayHello()

sayHello()메소드의 System.out.Println()메소드 실행
도스창에 Hello World! 출력

JAVA Program의 구조

```
public static void main(String[] args) {
    ...
}
```

- String[] args
  - main() Method의 실행 시 매개변수 지정 형식
  - String[] 까지가 데이터 타입
  - args[0], args[1] 형식으로 매개변수를 입력 받음
- String args[]
  - 사용가능하나 자바의 기본은 아님
- 참고
  - args : Argument String의 약자
  - argv : Argument Value의 약자

JAVA Program의 구조

MainParam.java **(main메서드의 매개변수를 테스트한 예제)**

```
public class MainParam {
    public static void main(String[] args) {
        System.out.println(args[0]);
        System.out.println(args[1]);
    }
}
```

```
C:□>javac MainParam.java
C:□>java MainParam Hello World!
Hello
World!
```

1 자바 파일 실행

java MainParam Hello World!

java.exe를 실행시키면 자바 가상 머신은 main()메소드를 찾아서 실행

2 main()메소드

args[0]

args[1]

이 때 Hello와 World을 main()의 매개 변수로 받음

3 main() 실행

```
System.out.println(args[0]);
System.out.println(args[1]);
```

main()메소드의 System.out.Println()을 실행하여 args[0], args[1]의 내용을 출력

JAVA Program의 구조

```
public static void main(String[] args) {
    ...
}
```

- static 멤버 변수
  - static으로 정의된 멤버 변수는 공유의 의미를 갖음
  - 모든 객체에서 공통으로 사용하는 메모리
  - 메소드내에는 static변수를 선언할 수 없음
- static 멤버 메소드
  - 자동으로 final 메소드가 됨(overriding 불가능)
- static 블록
  - 객체가 생성되기 전에 static 영역의 메모리를 제어하기 위한 블록
- static 메모리는 객체가 생성되기 전에 클래스명으로 접근이 가능

JAVA Program의 구조

StaticTest.java

```
public class StaticTest {
        private static int sint = 0;
        private int nint = 0;
        public StaticTest() {
            sint = sint +1;
            nint = nint +1;
        }
        public void sayMember() {
           System.out.println("sint:" + sint + "  nint:" + nint);
        }
        public static void main(String[] args) {
           for(int i=0; i<10; i++) {
                StaticTest test = new StaticTest();
                test.sayMember();
           }
       }
}
```

결과화면>

```
C:₩>javac StaticTest.java
C:₩>java StaticTest
sint:1  nint:1
sint:2  nint:1
sint:3  nint:1
sint:4  nint:1
sint:5  nint:1
sint:6  nint:1
sint:7  nint:1
sint:8  nint:1
sint:9  nint:1
sint:10  nint:1
```

JAVA Program의 구조

- main() 메소드가 실행되기 전에 static 메모리 영역을 먼저 작업
- StaticTest 객체를 생성하는 순간 생성자 메서드 호출
- sint는 처음에 한번 생성된 것을 계속 사용하므로 새로 생성되지 않음
- 하지만 nint는 객체가 생성될 때마다 새로이 생성됨

JAVA Program의 구조

StaticTimeMain.java (static 멤버필드를 초기화하는 예제)

```
class StaticTime {
        private static int sint = 0;
        static {
              sint = 100;
              System.out.println("sint:" + sint);
        }
}
public class StaticTimeMain {
    public static void main(String[] args) {
          StaticTime s = null;
    }
}
```

결과화면>
C:₩04>javac StaticTimeMain.java
C:₩04>java StaticTimeMain
sint:100

- main() 메소드 실행
- StaticTime 클래스명이 사용되는 순간 static 블록이 실행
  - 객체가 생성여부와 상관없이 static 메모리는 생성되고 static 블록도 실행됨
  - static블록의 실행 결과 sint를 초기화하고 그 값을 출력함

JAVA Program의 구조

StaticAccess.java (static 멤버필드로의 접근 예제)

```
public class StaticAccess {
   public static int sint = 0;
   public int nint = 0;
   public static void main(String[] args) {
      StaticAccess.sint =  3333;
      System.out.println("static 직접 접근:" + StaticAccess.sint);
   }
}
```

결과화면>
C:₩>javac StaticAccess.java
C:₩>java StaticAccess
Static 직접 접근:3333

- main()메소드 실행
- 객체를 생성하지않고 StaticAccess.sint로 값을 할당
  - 객체가 생성여부와 상관없이 static 메모리에 접근도 가능

JAVA Program의 구조

StaticMethodAccess.java (static멤버 메서드 사용 예제)

```
public class StaticMethodAccess {
    private static int sint =100;
    public int nint =0;
    public static void setStaticInt(int x) {
        sint = x;
    }
    public static int getStaticInt(){
       return sint;
    }
    public static void main(String[] args) {
        StaticMethodAccess.setStaticInt(33333);
        int s = StaticMethodAccess.getStaticInt();
        System.out.println("static값은:" + s);
   }
}
```

결과화면>
C:₩>javac StaticMethodAccess.java
C:₩>java StaticMethodAccess
static값은:33333

JAVA Program의 구조

- main() 메소드 실행
- static 메소드도 객체생성 이전에 접근이 가능
- static 메소드에는 일반변수를 사용할 수 없음
  - 일반 멤버변수는 객체가 생성되면서 메모리를 할당 받음
  - static 메소드는 객체가 생성되기 전에 사용
  - 따라서 static 메소드에서는 일반 멤버변수를 사용할 수 없음

JAVA Program의 구조

```
System.out.println("Hello World");
```

- System.out.println()
  - 모든 클래스는 java.lang.* 을 자동으로 import함
    - System 클래스는 그 안에 포함된 클래스
    - System 클래스는 io 클래스를 import하고 있음
  - out은 printStream 타입의 static 변수
    - println()메소드는 printStream 클래스의 멤버 메소드
  - 따라서 객체 생성 이전에 static 변수 out로 println()메소드 사용이 가능
- "Hello World!"
  - 자바에서 문자열을 이중따옴표로 표시
  - println()은 표준출력메서드로 문자열을 매개변수로주면 도스창에 출력

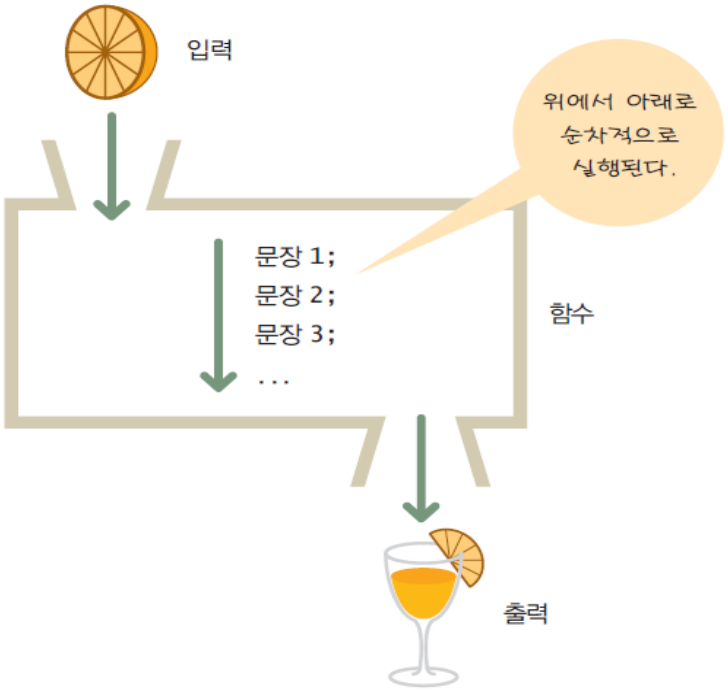
JAVA Program의 구조

- System.out.println()은 데이터를 출력한 후 자동으로 다음 줄로 넘어간다
  - 즉 "개행문자(줄바꿈 문자)"가 붙는다
  - 엔터키(Enter Key)가 자동으로 쳐지는 것이다

- System.out.print()는 줄 바꿈을 하지 않는다
- 대부분의 경우 println()을 쓰고, 줄바꿈을 하지 말아야 하는 특수한 경우에만 print()를 사용

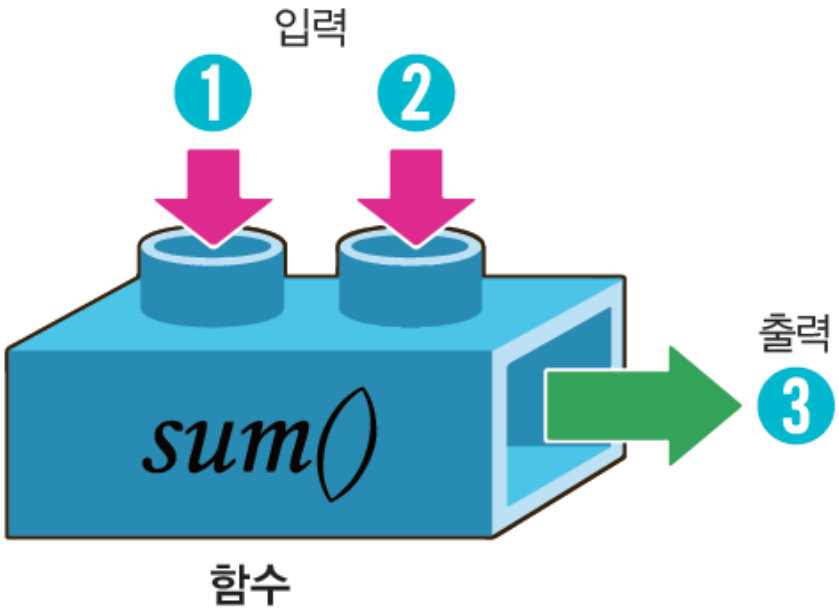
Method

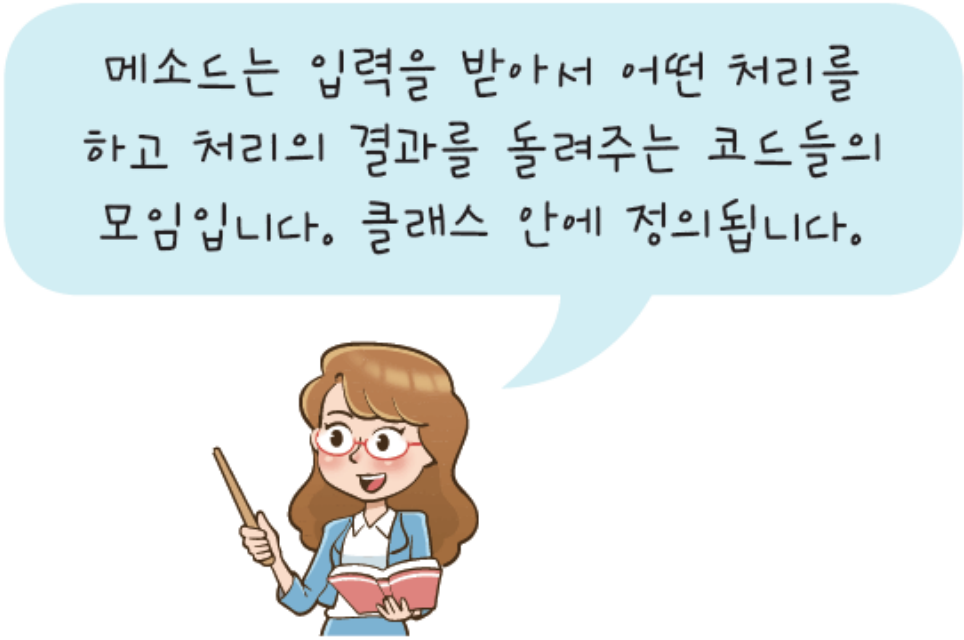
- Class = Field(변수) + Method(함수)
- Method는 Data를 입력 받아서 작업을 수행하고 결과를 내보내는 작은 Module
- Method에는 문장들이 들어 있고 이들 문장들을 위에서 아래로 차례대로 실행한 후에 작업의 결과를 외부로 반환



Method

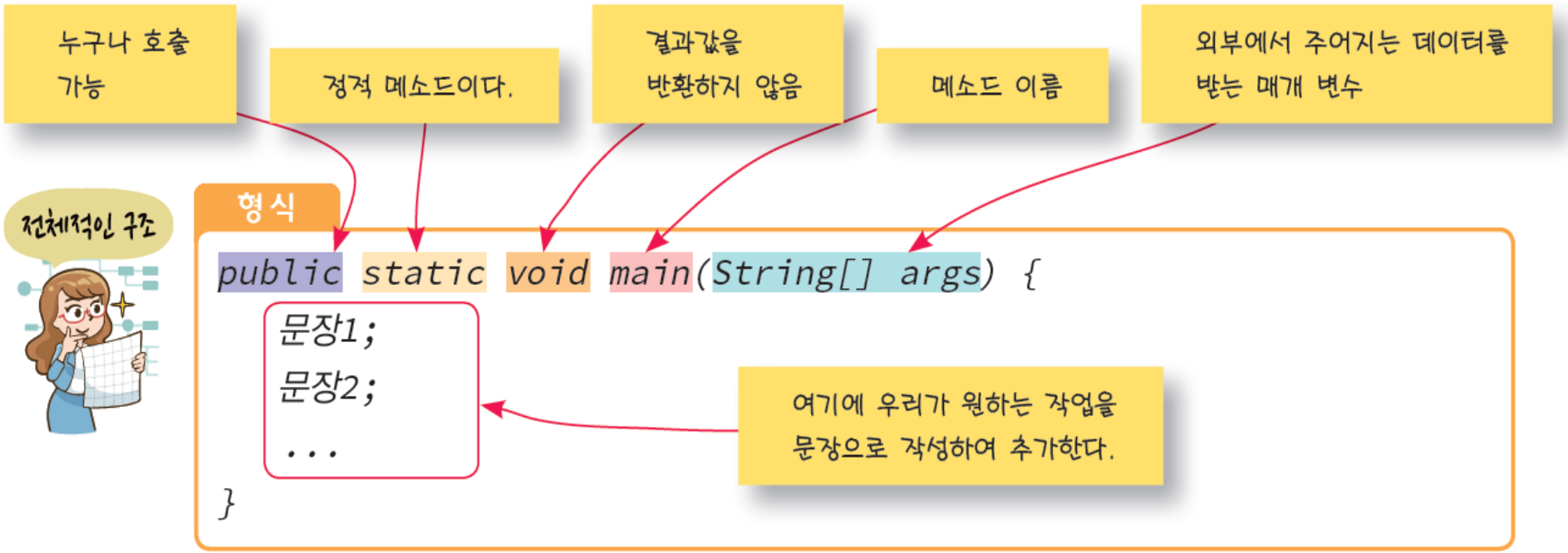
- Method는 특정한 작업을 수행하는 Code의 묶음
- Method는 외부로부터 입력을 받아서 특정한 작업을 수행하고 작업의 결과를 반환하는 Black Box





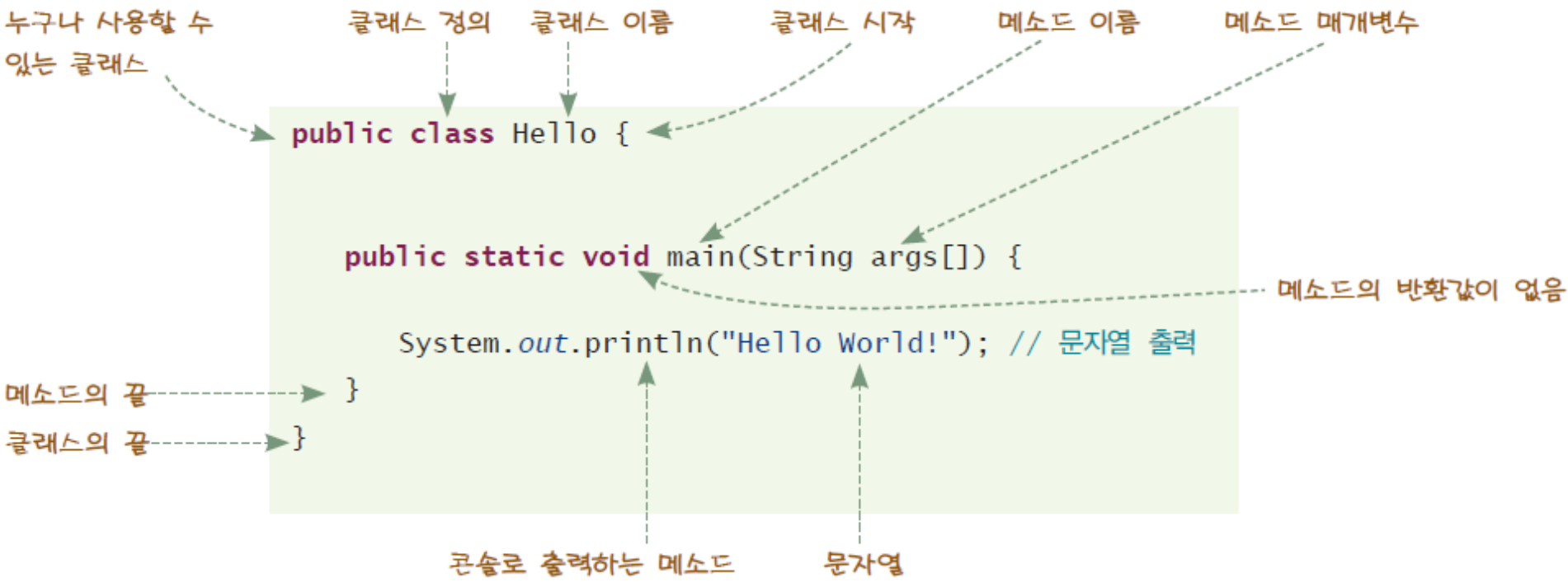
Method

- Method의 구조
  - 하나의 Class 안에는 여러 개의 Method가 포함될 수 있음
  - 하나의 Method 안에는 여러 개의 문장이 포함될 수 있음



Method

용어 설명



Expression

- 계산을 수행 후, 되돌림(Return)값을 되돌려 주는 일련의 작업
  - 수식값이나 논리값 등
  - 예)
    - a++ : a값을 반환한 후 a값을 하나 증가 시킴
    - x * y * Z
    - x + y / 100

Statement

- 사용자가 Computer에게 작업을 지시하는 단위
- 문장들은 Method 내부에 들어 있음
- 보통 Program의 한 줄(Line)이 하나의 문장이 됨
- 문장의 끝은 항상 Semicolon(;)으로 끝남
  - 예) int i = 0;
    System.out.println("Welcome to Java Lecture!");
- Block Statement(복합문)
  - 여러 개의 문장이 모인 복합 문장
  - Brace로 표시 : { …… }

```
… main(…)
{
    문장;
    문장;
    ……
}
```

Statement

- 사용자가 Computer에게 작업을 지시하는 단위
- Statement는 순차적으로 실행

Hello.java

```
01 public class Hello {
02
03     public static void main(String[] args) {
04         System.out.println("Hello World!");
05     }
06 }
```

이것이 작업의 내용을 기술하는 문장이다.

Statement

- System.out.println("Hello World!");
  - "Hello World"라는 문자열을 화면에 출력하라는 명령어
  - System.out는 객체(Object)임
  - 이 객체는 println( ), print( ), … 라는 많은 Method를 가지고 있음

Class 이름과 Method 이름은 도트(.)로 구분

System.out.println에 있는 점(.) : '~의'를 의미 -> System의 out의 println 메소드
System 클래스의 PrintStream 클래스의 println 메소드

Statement

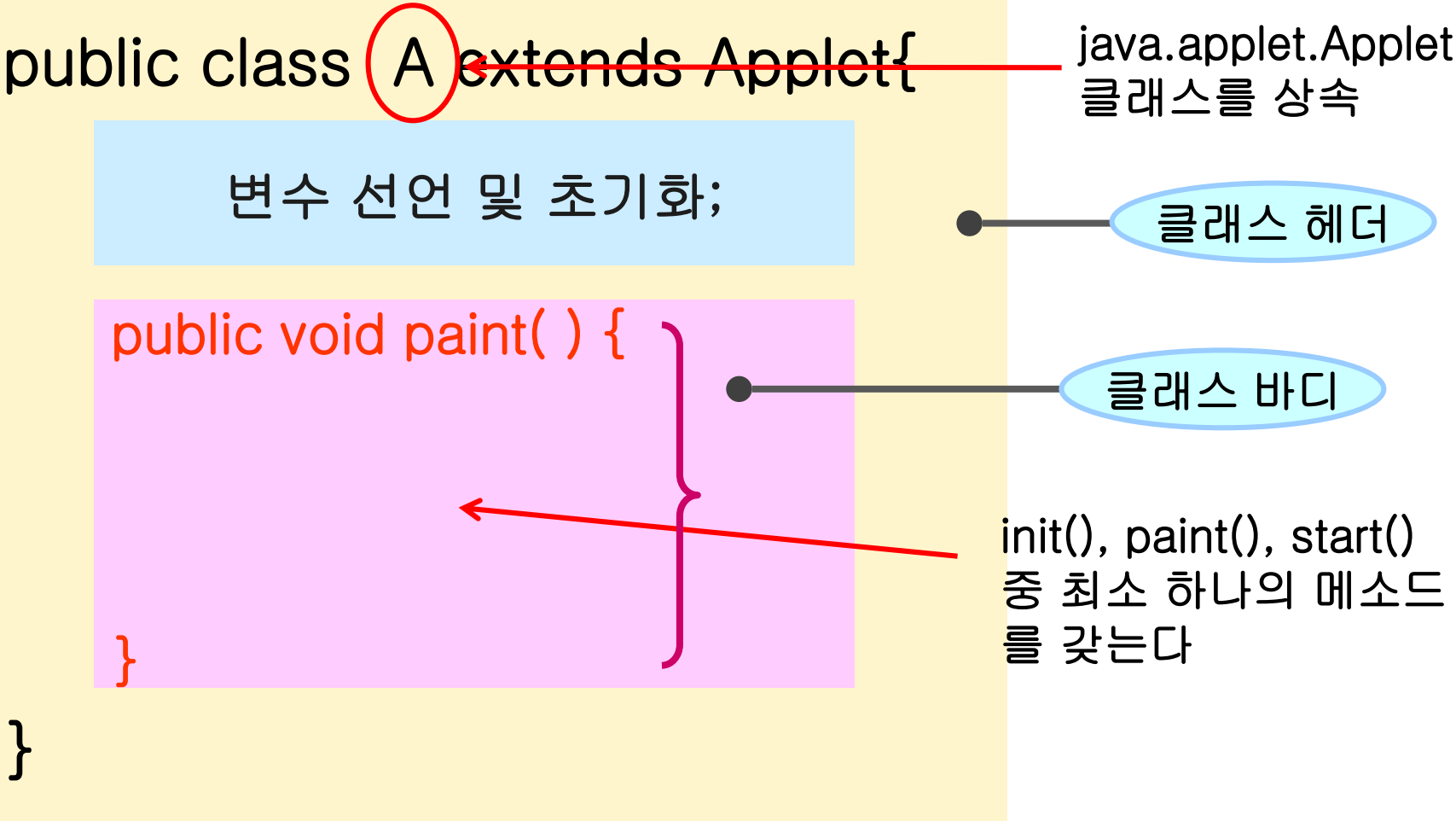
- 문장과 공백
  - 공백(white space; space, tab, blank line)은 JAVA Compiler가 무시함
  - 문자열 내에서는 공백을 구별함
  - 제어 문자열에서 여러 줄의 문자열은 백슬래시(₩)를 이용하여 구분

```
✓ x=2+3;          /* 모두 동일 */
✓ x = 2 + 3 ;
✓ x   =
  2
  + 3;
```

```
✓ System.out.println("Hello, world !");
✓ System.out.println("Hello,
   world !");
```

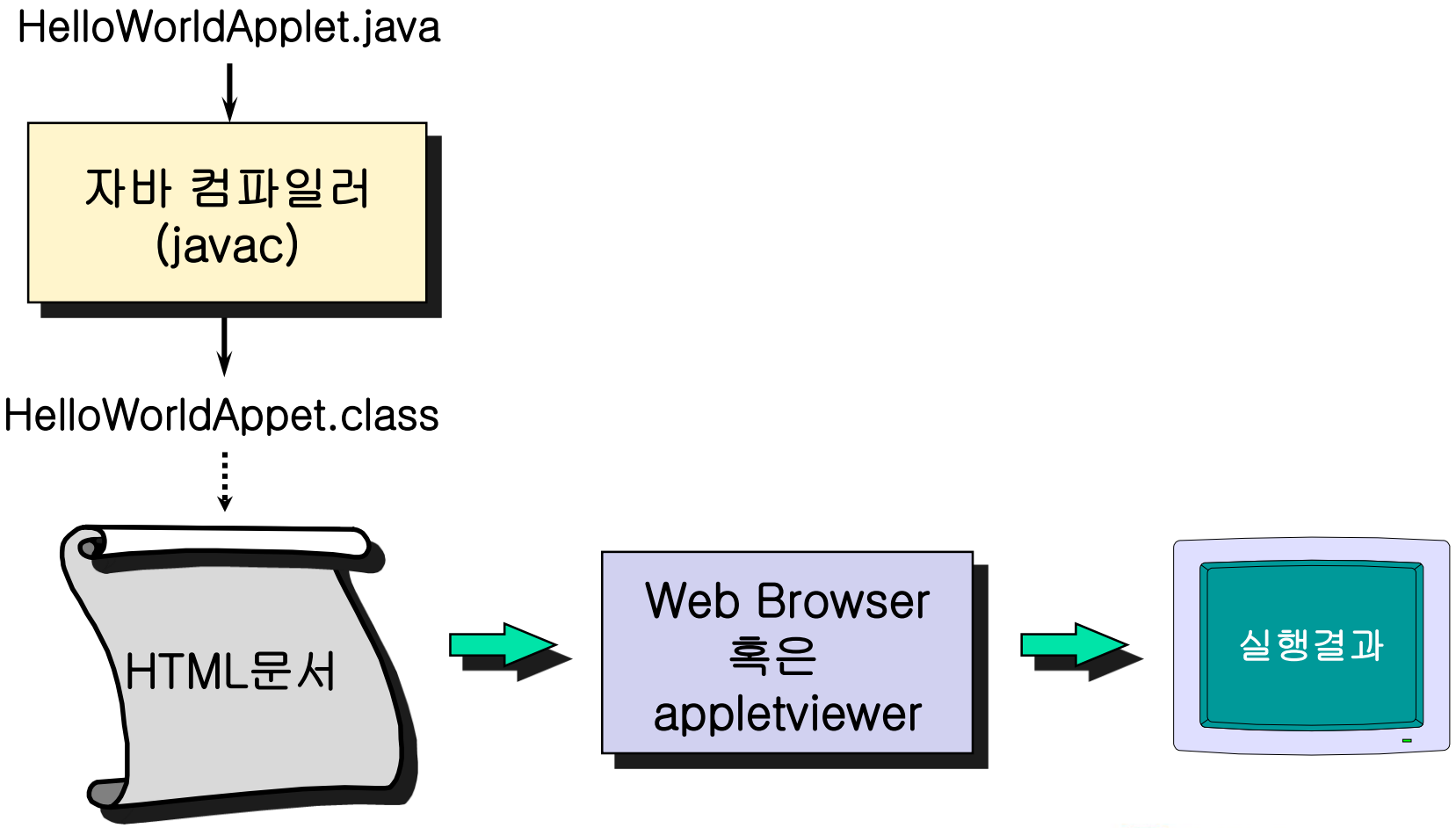
```
✓ System.out.printf("Hello, ₩
  world !");
```

Applet 프로그램



Applet 프로그램

- JAVA Applet 수행 과정



Applet 프로그램

- JAVA Applet 수행 과정
  - 메모장이나 편집기(Editor)를 사용하여, JAVA Program을 입력
    - 입력된 Program을 클래스의 이름과 동일하도록 저장
      예) HelloWorldApplet.java
  - Javac 명령어를 사용하여 Compile
    예) javac HelloApplet.java
  - HelloApplet.class File을 호출하는 HelloApplet.html File을 작성
    - 이때 사용할 HTML tag는
      <applet code= Java_class_name.class
      width=x, height=y>
    - 여기에서 x와 y는 Applet의 크기를 나타냄

Applet 프로그램

- JAVA Applet Program 작성

HelloWorldApplet.java

```
import java.applet.*;
import java.awt.*;
public class HelloWorldApplet extends Applet {
    public void paint(Graphics g) {
        g.drawString((“안녕하세요 !! 애플릿 프로그램 예제입니다”,
        100, 60);
    }
}
```

Applet 프로그램

- JAVA Applet Program 작성
  - import java.applet.*;
    - import 문은 C에서의 include 문과 같은 것
    - Program에서 사용할 클래스가 정의되어 있는 패키지(Package)를 포함(import)하기 위해 사용
    - 모든 JAVA Applet Program은 반드시 java.applet.*를 import해야 함
  - import java.awt.*;
    - paint() 메소드의 인자로서 Graphics 객체를 사용하고 있기 때문에, 그래픽스 클래스가 정의된 java.awt.*를 import하여야 함

Applet 프로그램

- JAVA Applet Program 작성
  - public class HelloApplet extends Applet {
    - 정의하는 HelloApplet 클래스가 Applet 클래스에게 상속받고 있다는 것을 나타냄
    - 위의 import java.applet.*과 함께 반드시 명시해야 함
  - public void paint(Graphics g)
    - paint() 메소드는 윈도우 화면에 그림을 그리거나, 원하는 위치에 문자열을 출력시킬 때 사용하는 메소드
    - 애플릿을 실행시키면 자동으로 호출되어 실행됨
  - g.drawString("안녕하세요 !! 애플릿 프로그램 예제입니다",100,60);
    - Graphics 클래스 객체 g의 메소드 drawString을 사용하여 원하는 위치 x 좌표 100, y 좌표 60에 "안녕하세요 !! 애플릿 프로그램 예제입니다"를 출력

Applet 프로그램

- Applet을 보기 위한 HTML 문서작성

HelloWorld.html

```
<HTML>
<HEAD>
<TITLE>Hello World Applet</TITLE>
</HEAD>
<BODY>
아래 화면에 보이는 것이 AppletHelloWorld 애플릿입니다.<BR>
<APPLET CODE="HelloWorldApplet.class" WIDTH=230 HEIGHT=130>
</APPLET>
</BODY>
</HTML>
```

Applet 프로그램

- HTML의 <applet> 태그

```
<APPLET CODE=appletFile ALT=alterernateText WIDTH=pixels HEIGHT=pixels>
</APPLET>
```

- <applet> 태그의 옵션
  - CODE = appletFile
    - Applet의 *.class File의 이름을 지정
      (그 파일 이름은 애플릿의 기저 URL에 대한 상대 URL로 표현되어야 하며, 절대 URL로 표현될 수 없음)
  - ALT = alternateText
    - Applet을 실행하지 못하는 브라우저를 사용하는 사용자에게 보여줄 어떤 Text를 지정

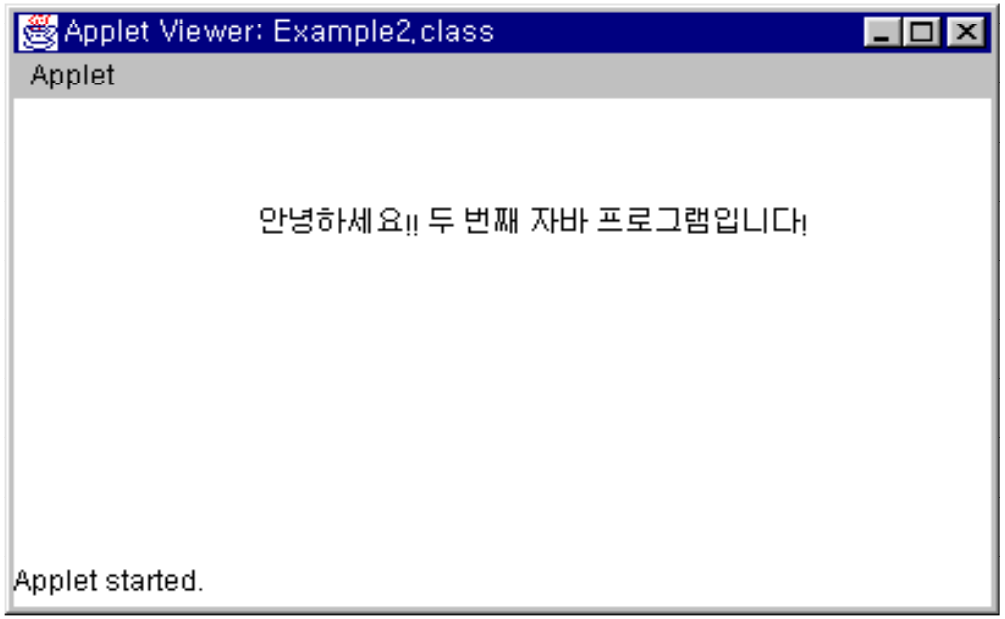
Applet 프로그램

- WIDTH = pixels HEIGHT = pixels
  - Applet이 만들어 내는 어떤 윈도우나 다이얼로그 박스를 포함하지 않은 애플릿 초기 화면의 넓이와 높이를 픽셀 값으로 지정

Applet 프로그램

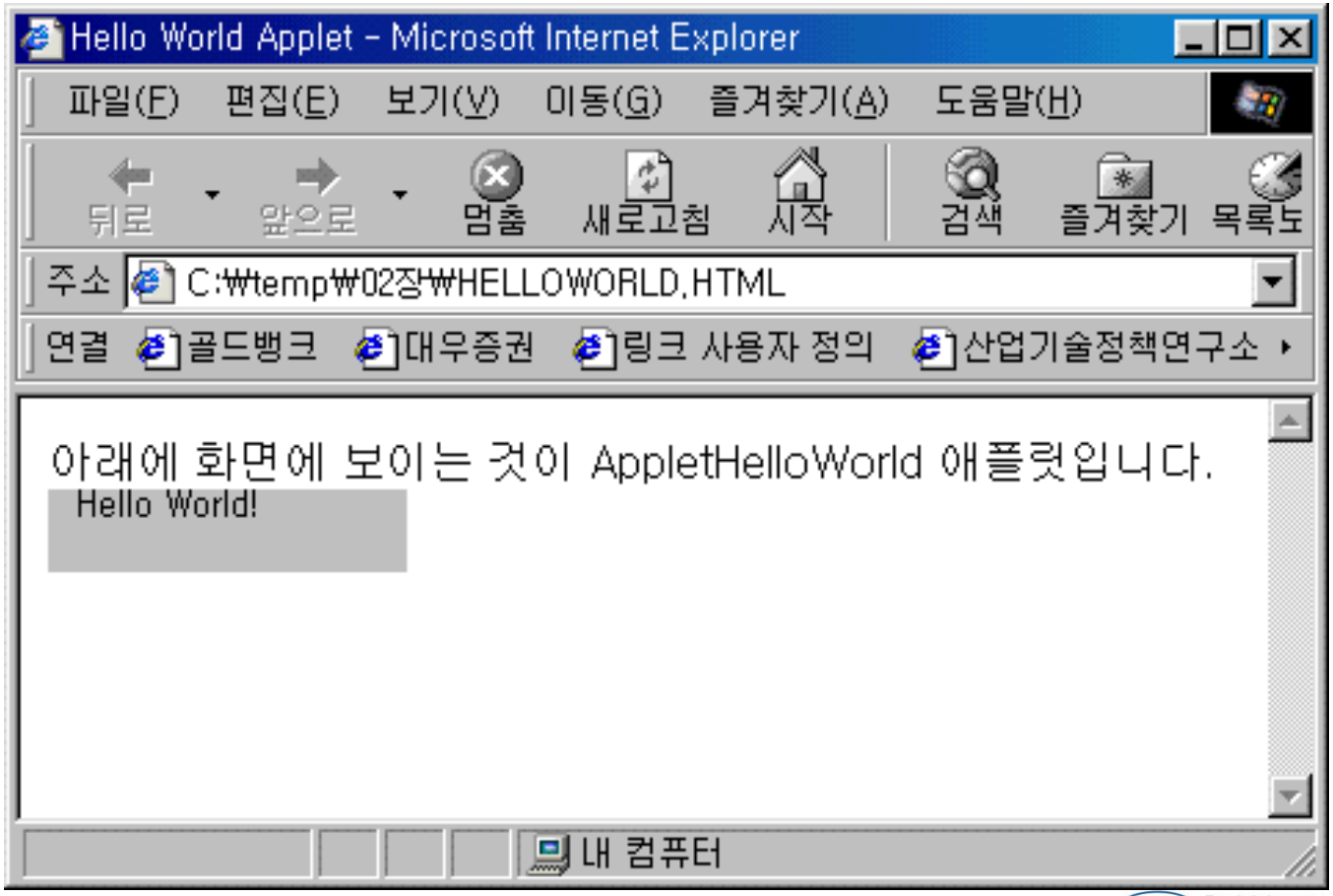
- JAVA Applet 실행하기
  - 작성된 HelloWorld.html파일을 애플릿 뷰어나 웹 브라우저를 사용 오픈 함
    - 애플릿 뷰어를 사용할 경우 명령어

```
c:₩> appletviewer HelloWorld.html
```



Applet 프로그램

- Applet Program의 실행
  - WWW 검색기



C/C++에서 제거된 특성

- typedef문, #define문
  - 클래스와 인터페이스
- 구조체(struct)와 공용체(union)
  - 클래스로 대체 가능
- 함수(function)
  - 모두 메소드로 처리
- 다중상속(multiple inheritance)
  - 인터페이스는 다중상속 지원
- goto 문을 지원하지 않음
  - 다중 레이블 break/continue문

C/C++에서 제거된 특성

- 포인터 연산을 제거
- 강제적인 자동변환
  - 명시적인 cast 연산 필요
  - strongly typed language
- 연산자 중복(operator overloading)을 제거
- 메모리 관리
  - malloc()을 제거