

# 1

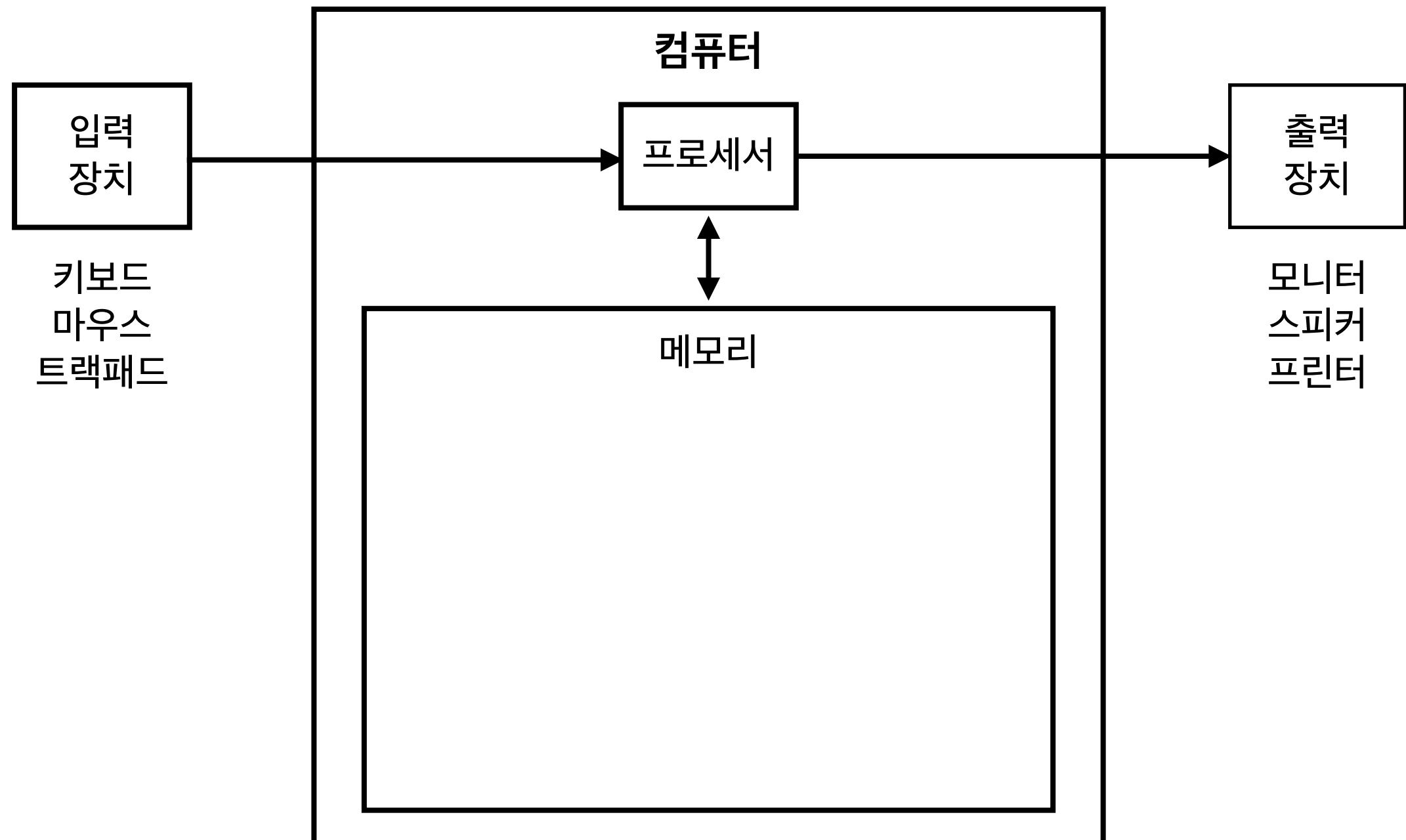
## 첫 애플리케이션 만들기





## 컴퓨터

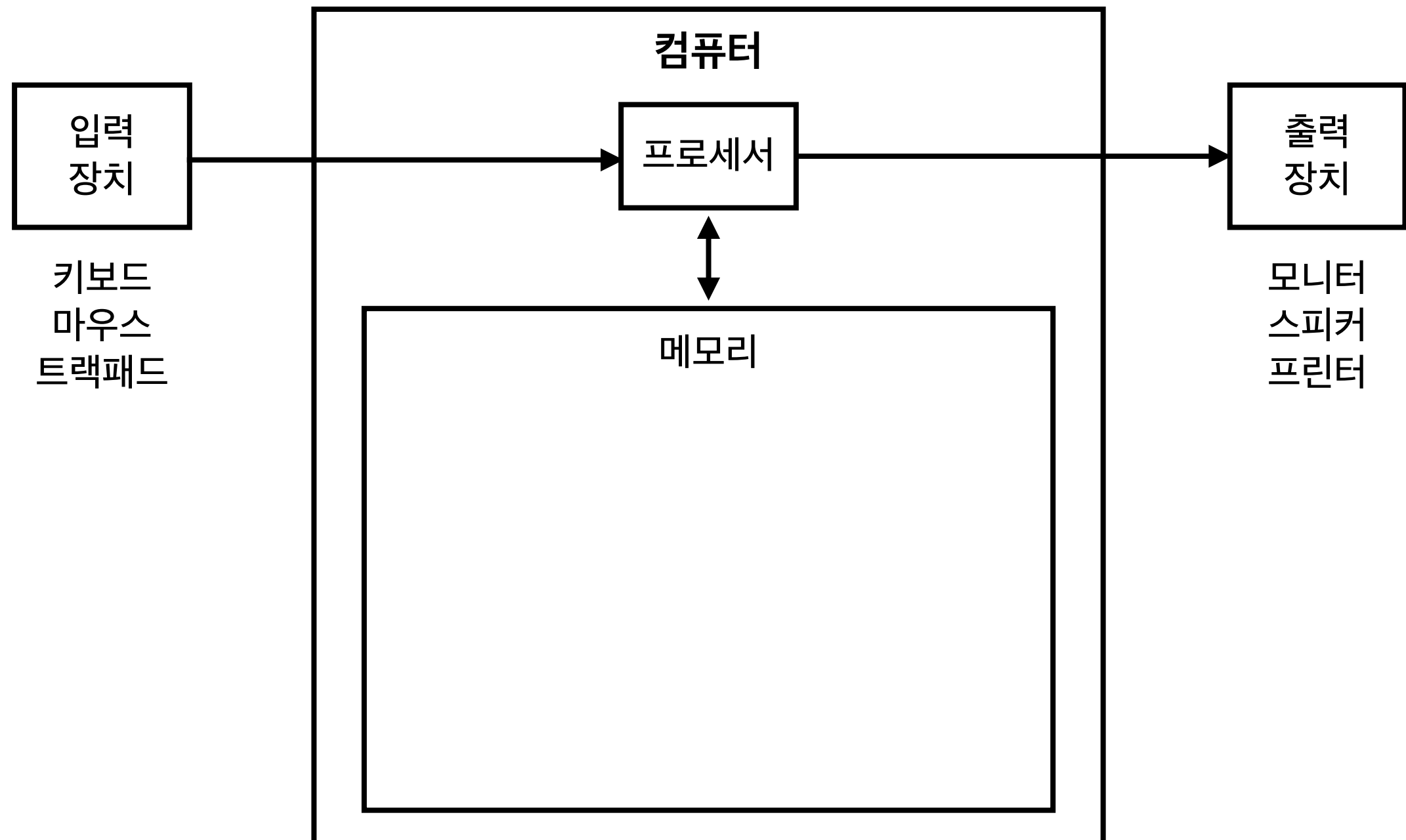
지시/명령에 따라 계산을 수행하는 기계 = 하드웨어





## 프로그램

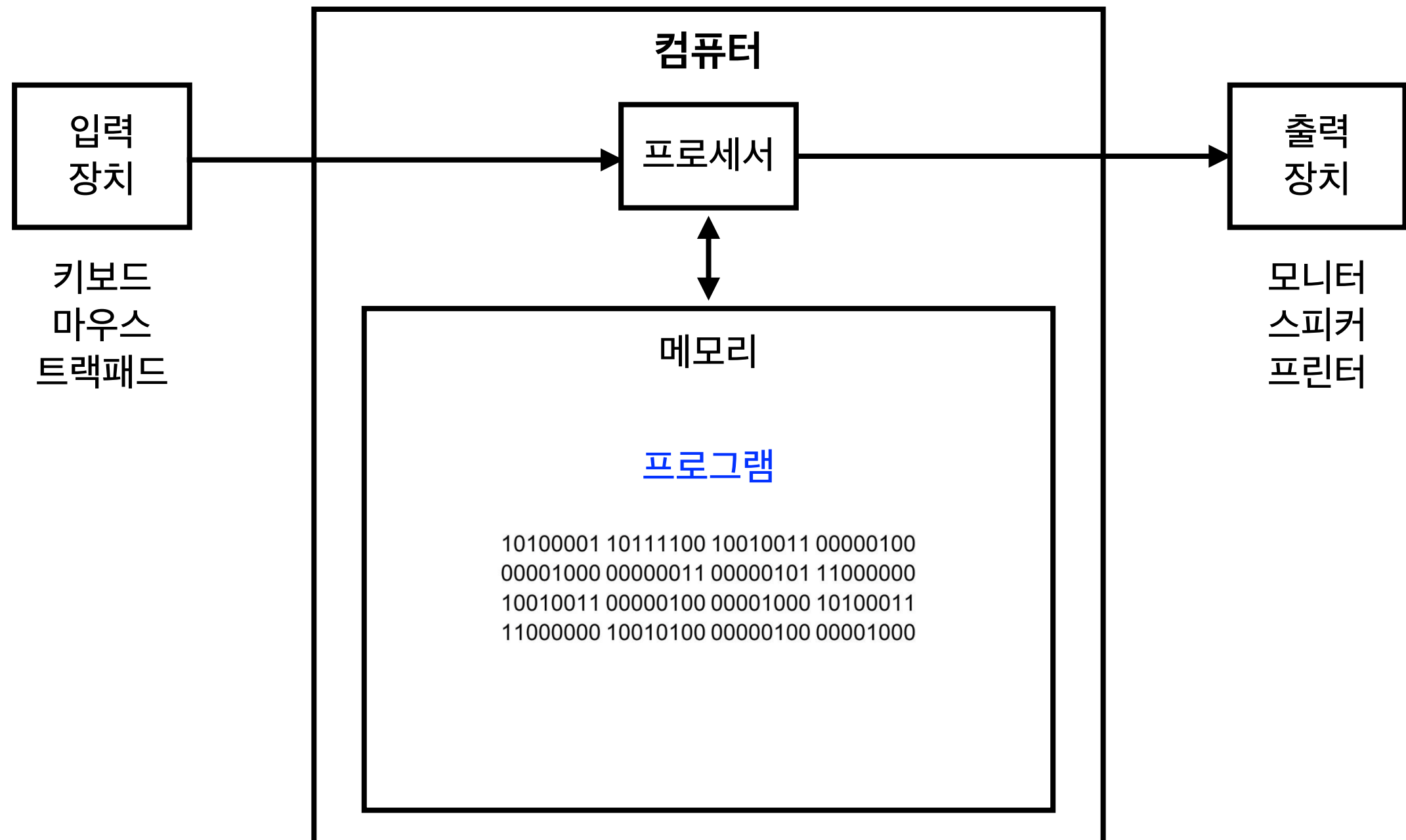
컴퓨터로 계산하기 위해서 작성한 지시 또는 명령 = 소프트웨어





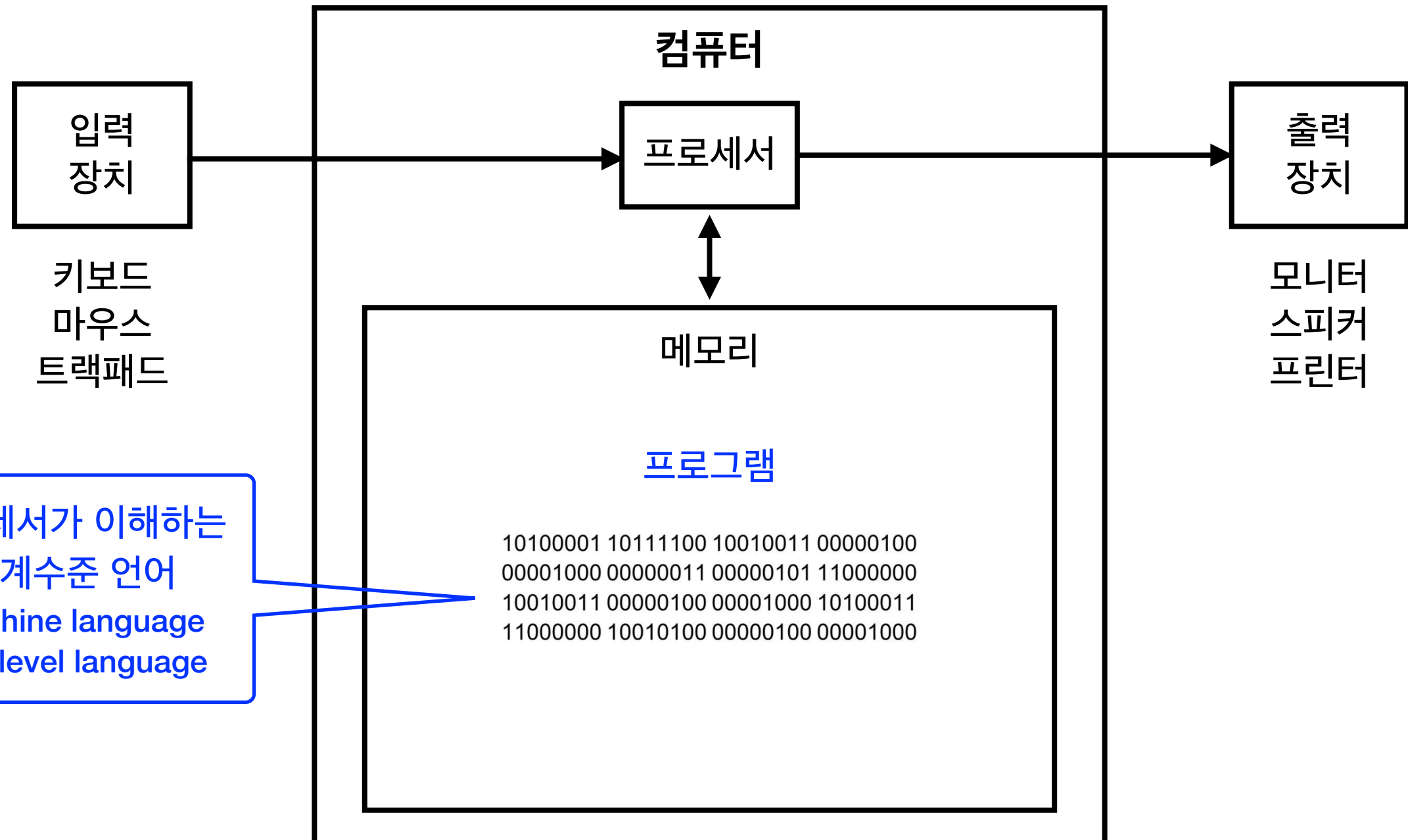
## 프로그램

컴퓨터로 계산하기 위해서 작성한 지시 또는 명령 = 소프트웨어





프로그램  
소프트웨어



프로세서가 이해하는  
기계수준 언어  
machine language  
low-level language

프로그램  
10100001 10111100 10010011 00000100  
00001000 00000011 00000101 11000000  
10010011 00000100 00001000 10100011  
11000000 10010100 00000100 00001000



프로그램

사람이 이해하기 쉬운 고수준 언어로 작성  
high-level language



컴퓨터

프로세서



메모리



프로그램

사람이 이해하기 쉬운 고수준 언어로 작성  
high-level language

인터프리터 실행 방식

Python, ...

```
1 def selection_sort(xs):  
2     def loop(xs,ss):  
3         if xs != []:  
4             smallest = min(xs)  
5             xs.remove(smallest)  
6             ss.append(smallest)  
7             return loop(xs,ss)  
8         else:  
9             return ss  
10    return loop(xs,[])
```



컴퓨터

프로세서



메모리

가상머신 프로세서



가상머신 메모리

```
1 def selection_sort(xs):  
2     def loop(xs,ss):  
3         if xs != []:  
4             smallest = min(xs)  
5             xs.remove(smallest)  
6             ss.append(smallest)  
7             return loop(xs,ss)  
8         else:  
9             return ss  
10    return loop(xs,[])
```



## 프로그램

사람이 이해하기 쉬운 고수준 언어로 작성  
high-level language

## 컴파일러 실행 방식

C, ...

```
47 int main(void){  
48     srand ( time(NULL) );  
49     initWeights();  
50     initData();  
51     for(int j = 0; j <= numEpochs; j++){  
52         for(int i = 0; i < numPatterns; i++){  
53             patNum = rand() % numPatterns;  
54             calcNet();  
55             WeightChangesHO();  
56             WeightChangesIH();  
57         }  
58         calcOverallError();  
59     }  
60     displayResults();  
61     return 0;  
62 }
```



## 컴퓨터

프로세서



메모리

프로그램

컴파일  
compile

기계어로 번역

```
10100001 10111100 10010011 00000100  
00001000 00000011 00000101 11000000  
10010011 00000100 00001000 10100011  
11000000 10010100 00000100 00001000
```





## 프로그램

사람이 이해하기 쉬운 고수준 언어로 작성  
high-level language

## 컴파일러 실행 방식

Java

```
1 public class Database {
2
3     private Record[] base;
4     private int NOT_FOUND = -1;
5
6     /** Constructor - Database 초기화
7      * @param initial_size - 데이터베이스 규모 */
8     public Database(int initial_size) {
9         if (initial_size > 0)
10             base = new Record[initial_size];
11         else
12             base = new Record[1];
13     }
14
15     /** findLocation - 데이터베이스에서 키가 k인 레코드의 인덱스 검색
16      * @param k - 검색할 레코드의 키
17      * @return - 찾으면 해당 레코드의 인덱스
18      * return - 찾지 못하면 NOT_FOUND */
19     private int findLocation(Key k) {
20         for (int i = 0; i < base.length; i++)
21             if (base[i] != null && base[i].getKey().equals(k))
22                 return i;
23         return NOT_FOUND;
24     }
25
26     /** findEmpty - 데이터베이스에서 빈 자리 검색
27      * @return - 찾으면 빈 자리 인덱스
28      * return - 찾지 못하면 NOT_FOUND */
29     private int findEmpty() {
30         for (int i = 0; i < base.length; i++)
31             if (base[i] == null)
32                 return i;
33         return NOT_FOUND;
34     }
35 }
```

컴파일  
compile

Java Bytecode 로  
번역



컴퓨터

프로세서



메모리

JVM 프로세서



JVM 메모리

```
6a61 7661 782f 7377 696e 672f 4a50 616e
656c 0100 084e 4554 5f53 495a 4501 0001
4901 0005 5749 4445 5201 0006 5441 4c4c
4552 0100 0577 6964 7468 0100 0668 6569
6768 7401 0006 3c69 6e69 743e 0100 0528
4949 2956 0100 0443 6f64 650a 0003 000f
0c00 0b00 1001 0003 2829 5609 0001 0012
```

JVM = Java Virtual Machine



## 프로그램

사람이 이해하기 쉬운 고수준 언어로 작성  
high-level language

## 컴파일러 실행 방식

Java

```
1 public class Database {
2
3     private Record[] base;
4     private int NOT_FOUND = -1;
5
6     /** Constructor - Database 초기화
7      * @param initial_size - 데이터베이스 규모 */
8     public Database(int initial_size) {
9         if (initial_size > 0)
10             base = new Record[initial_size];
11         else
12             base = new Record[1];
13     }
14
15     /** findLocation - 데이터베이스에서 키가 k인 레코드의 인덱스 검색
16      * @param k - 검색할 레코드의 키
17      * @return - 찾으면 해당 레코드의 인덱스
18      * return - 찾지 못하면 NOT_FOUND */
19     private int findLocation(Key k) {
20         for (int i = 0; i < base.length; i++)
21             if (base[i] != null && base[i].getKey().equals(k))
22                 return i;
23         return NOT_FOUND;
24     }
25
26     /** findEmpty - 데이터베이스에서 빈 자리 검색
27      * @return - 찾으면 빈 자리 인덱스
28      * return - 찾지 못하면 NOT_FOUND */
29     private int findEmpty() {
30         for (int i = 0; i < base.length; i++)
31             if (base[i] == null)
32                 return i;
33         return NOT_FOUND;
34     }
35 }
```

컴파일  
compile

Java Bytecode 로  
번역

왜???



컴퓨터

프로세서



메모리

JVM 프로세서



JVM 메모리

```
6a61 7661 782f 7377 696e 672f 4a50 616e
656c 0100 084e 4554 5f53 495a 4501 0001
4901 0005 5749 4445 5201 0006 5441 4c4c
4552 0100 0577 6964 7468 0100 0668 6569
6768 7401 0006 3c69 6e69 743e 0100 0528
4949 2956 0100 0443 6f64 650a 0003 000f
0c00 0b00 1001 0003 2829 5609 0001 0012
```

JVM = Java Virtual Machine

# 프로그래밍

설계



구현

설계도 작성

코딩

MVC 아키텍처

객체지향 프로그래밍

Object-Oriented Programming

Model

View

Controller

Java

# 객체지향 프로그래밍

객체  
Object

String

상태 Field

"ERICA"

기능 Method

int length()

·  
·  
·

메소드 호출 메시지

length()

5

응답

통합개발환경

**IDE**

Integrated  
Development  
Environment



Hello, World!

자바 애플리케이션

# 구현

## 자바 애플리케이션 Java Application (표준 출력 버전)

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

# 실행 추적 Execution Trace

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

애플리케이션 시동

**HelloWorld**

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



# 실행 추적 Execution Trace

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

애플리케이션 시동

①

↓ **main** 메소드 호출

**HelloWorld**

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

# 실행 추적 Execution Trace

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

애플리케이션 시동

1

↓ main 메소드 호출

**HelloWorld**

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

**PrintStream**

```
void println(String x)
```

# 실행 추적 Execution Trace

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

애플리케이션 시동

1

↓ main 메소드 호출

**HelloWorld**

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

2

**println("Hello, World!")**

**PrintStream**

```
void println(String x)
```

# 실행 추적 Execution Trace

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

애플리케이션 시동

①

↓  
main 메소드 호출

**HelloWorld**

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

②

println("Hello, World!")

**PrintStream**

```
void println(String x)
```

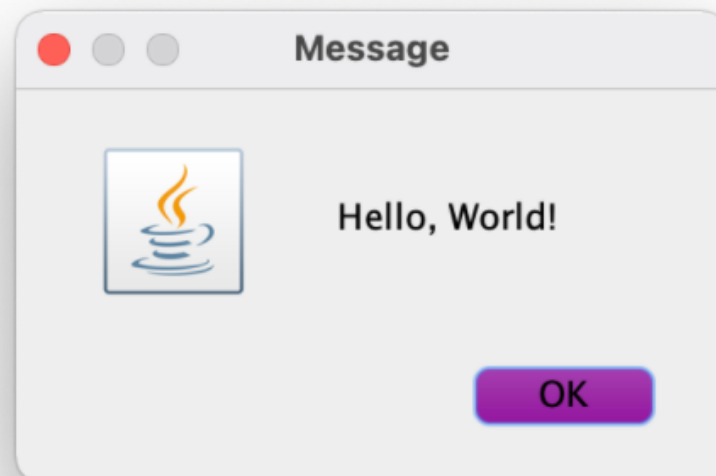
③ 실행

# 구현

## 자바 애플리케이션 Java Application (Swing 패키지 활용 버전)

```
import javax.swing.*;

public class HelloWorldGUI {
    public static void main(String[] args) {
        JOptionPane.showMessageDialog(null, "Hello, World!");
    }
}
```



# 실행 추적 Execution Trace

```
import javax.swing.*;

public class HelloWorldGUI {
    public static void main(String[] args) {
        JOptionPane.showMessageDialog(null, "Hello, World!");
    }
}
```

## 애플리케이션 시동

### HelloWorldGUI

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

# 실행 추적 Execution Trace

```
import javax.swing.*;

public class HelloWorldGUI {
    public static void main(String[] args) {
        JOptionPane.showMessageDialog(null, "Hello, World!");
    }
}
```

애플리케이션 시동

1

↓ **main** 메소드 호출

**HelloWorldGUI**

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

# 실행 추적 Execution Trace

```
import javax.swing.*;

public class HelloWorldGUI {
    public static void main(String[] args) {
        JOptionPane.showMessageDialog(null, "Hello, World!");
    }
}
```

애플리케이션 시동

1

↓  
main 메소드 호출

**HelloWorldGUI**

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

**JOptionPane**

```
... void showMessageDialog(...)
```



# 실행 추적 Execution Trace

```
import javax.swing.*;

public class HelloWorldGUI {
    public static void main(String[] args) {
        JOptionPane.showMessageDialog(null, "Hello, World!");
    }
}
```

애플리케이션 시동

1

main 메소드 호출

**HelloWorldGUI**

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

2

**showMessageDialog**  
**(null, "Hello, World!")**

**JOptionPane**

```
... void showMessageDialog(...)
```

# 실행 추적 Execution Trace

```
import javax.swing.*;

public class HelloWorldGUI {
    public static void main(String[] args) {
        JOptionPane.showMessageDialog(null, "Hello, World!");
    }
}
```

애플리케이션 시동

1

main 메소드 호출

**HelloWorldGUI**

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

2

showMessageDialog  
(null, "Hello, World!")

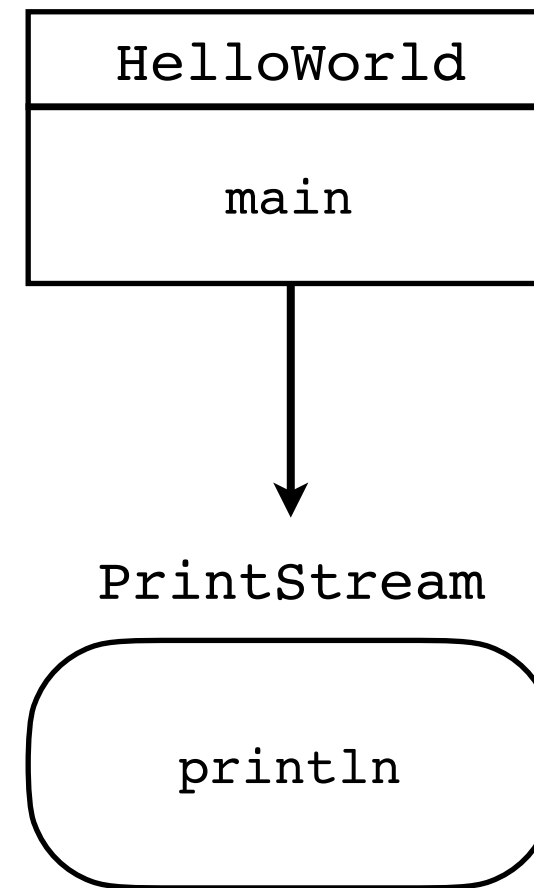
**JOptionPane**

```
... void showMessageDialog(...)
```

3 실행

## 설계

### 클래스 다이어그램 Class Diagram



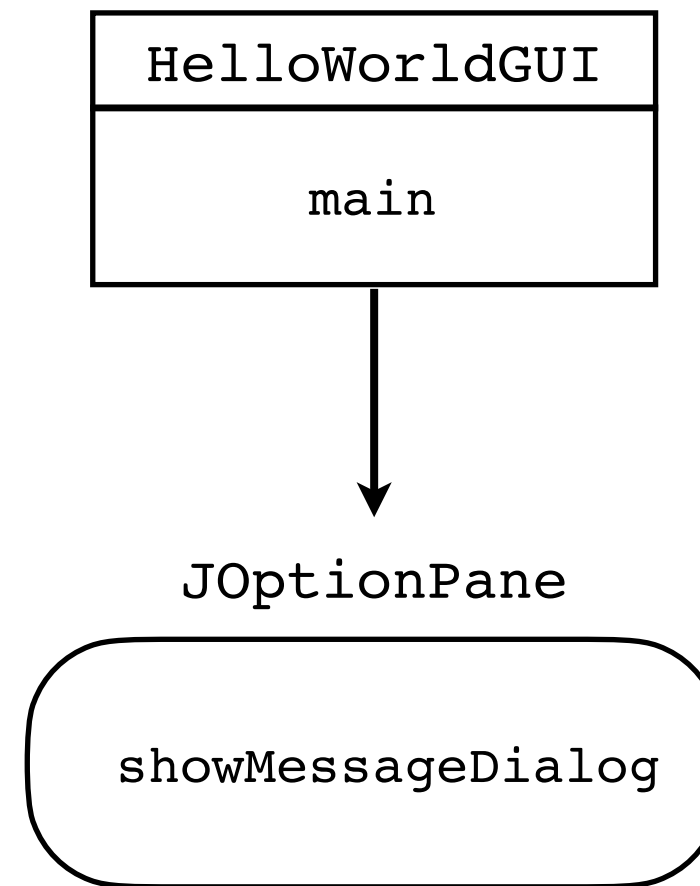
## 구현

### 자바 애플리케이션 Java Application

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

## 설계

### 클래스 다이어그램 Class Diagram



## 구현

### 자바 애플리케이션 Java Application

```
import javax.swing.*;

public class HelloWorldGUI {
    public static void main(String[] args) {
        JOptionPane.showMessageDialog(null, "Hello, World!");
    }
}
```