CSE2016

프로그램설계방법론

수업 소개

도경구

한양대학교 ERICA 소프트웨어학부

수업 목표

- 목표 : **MVC 아키텍처 기반 객체지향 프로그램 설계의 이해 및 숙달**
  - 제어구조 control structure (CSE1017 프로그래밍기초)
  - 자료구조 data structure (CSE2010 자료구조론)
  - **부품구조 component structure** (**CSE2016 프로그램설계방법론**)

수업 홈페이지

http://doggzone.github.io/cse2016/

CSE2016

프로그램설계방법론

꼬마 애플리케이션 만들기

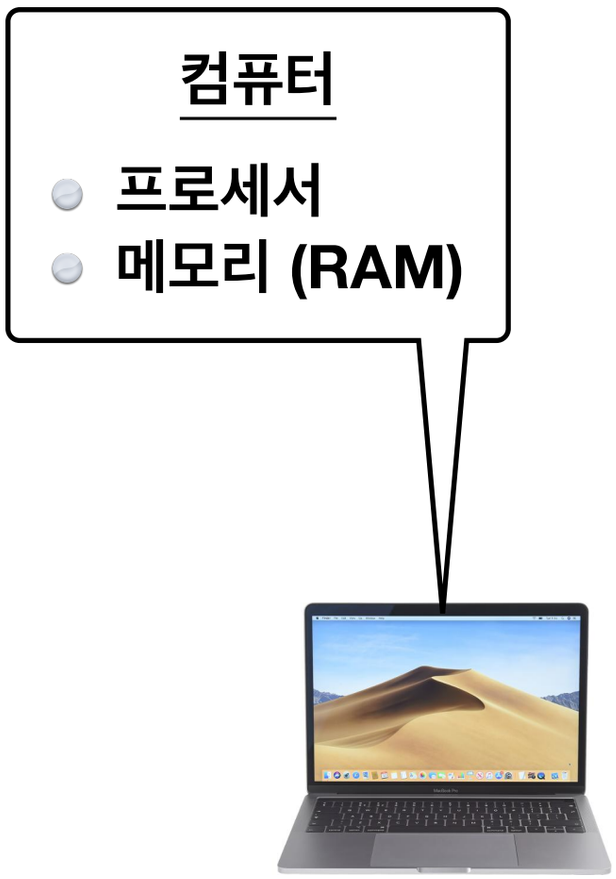
도경구

한양대학교 ERICA 소프트웨어학부

컴퓨터와 프로그래밍

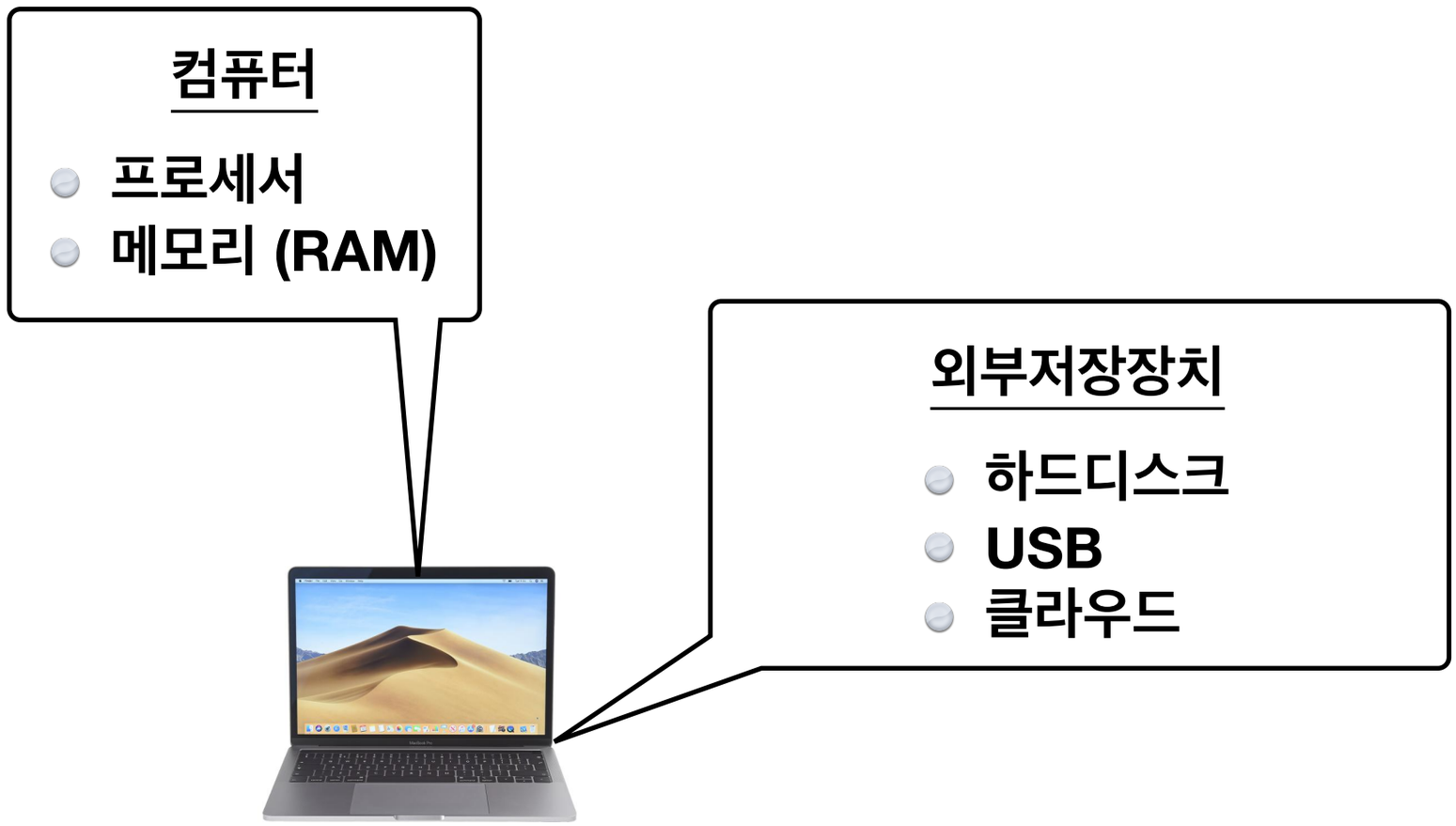
컴퓨터
Computer

지시(프로그램)에 따라 논리적인 계산을 수행하는 기계



컴퓨터
Computer

지시(프로그램)에 따라 논리적인 계산을 수행하는 기계



컴퓨터
Computer

지시(프로그램)에 따라 논리적인 계산을 수행하는 기계

컴퓨터

- 프로세서
- 메모리 (RAM)

입출력장치

- 입력 : 키보드, 마우스
- 출력 : 모니터, 프린터

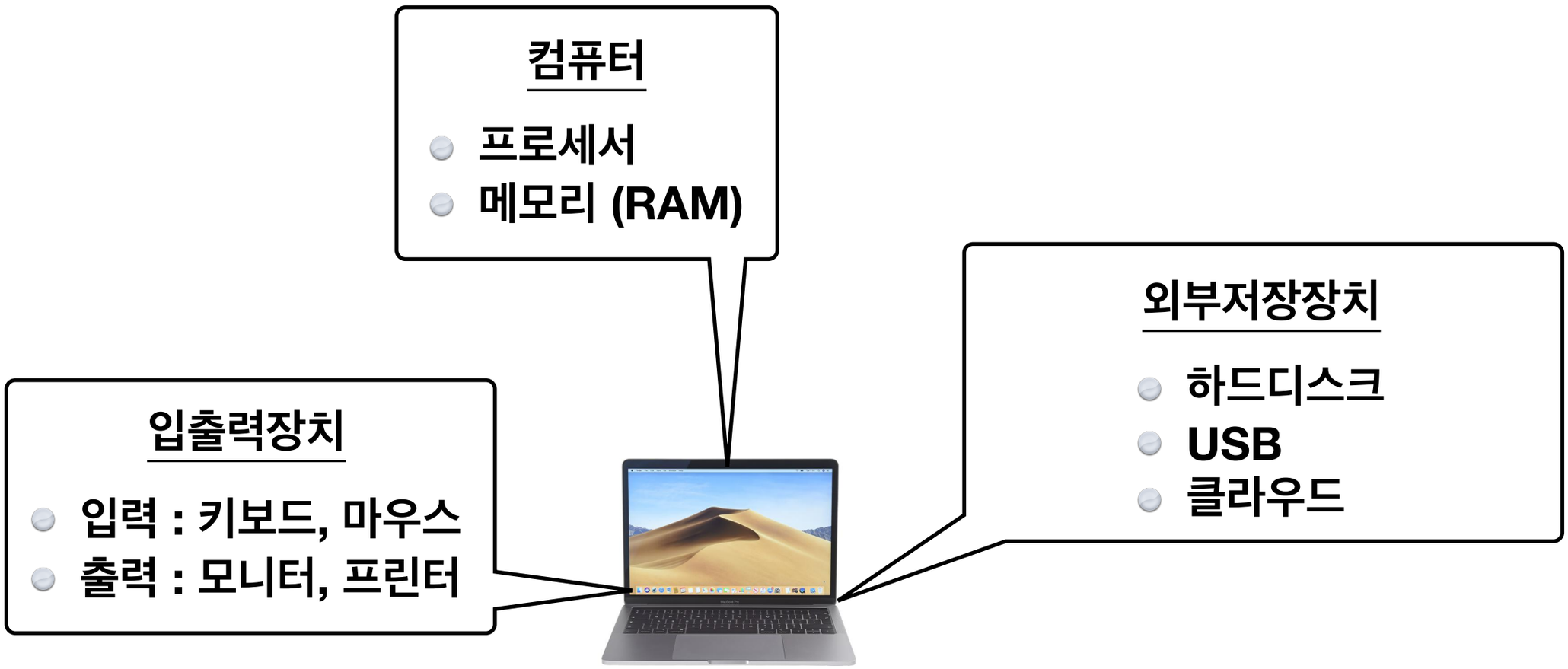
외부저장장치

- 하드디스크
- USB
- 클라우드

프로그래밍

Programming = Coding

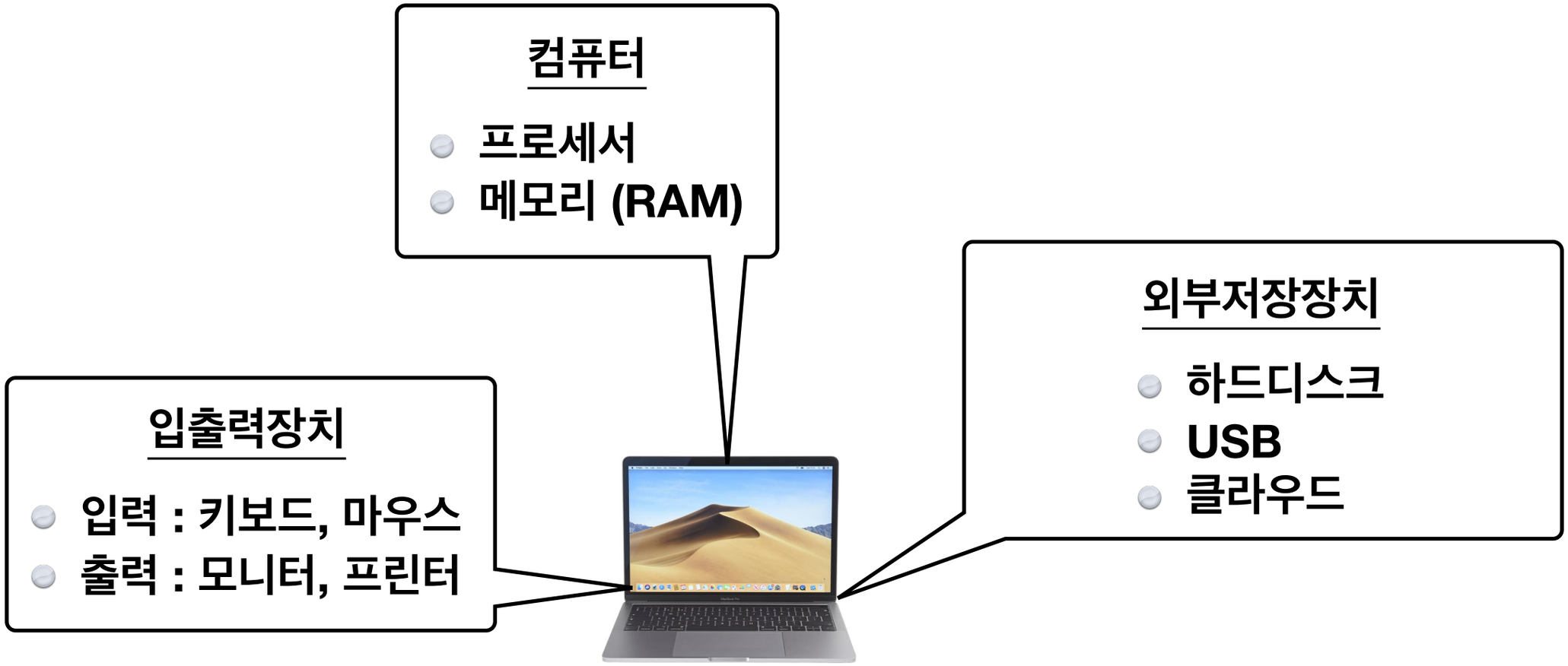
컴퓨터가 실행할 프로그램(지시,명령)을 작성하는 행위



프로그래밍

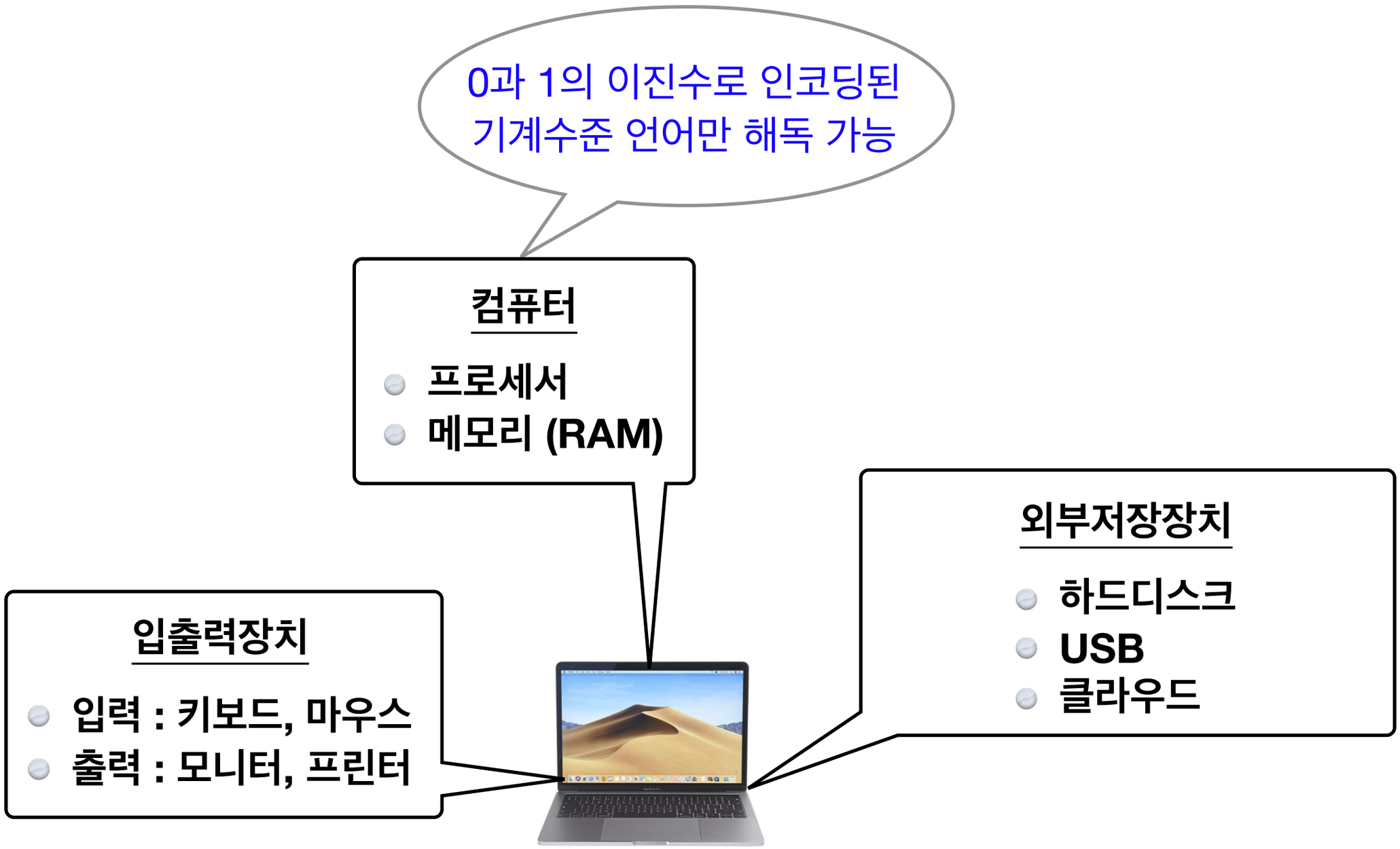
Programming = Coding

- 작성한 프로그램은 외부저장장치에 파일로 저장
- 프로그램을 메모리에 로딩
- 프로세서가 로딩된 프로그램을 실행



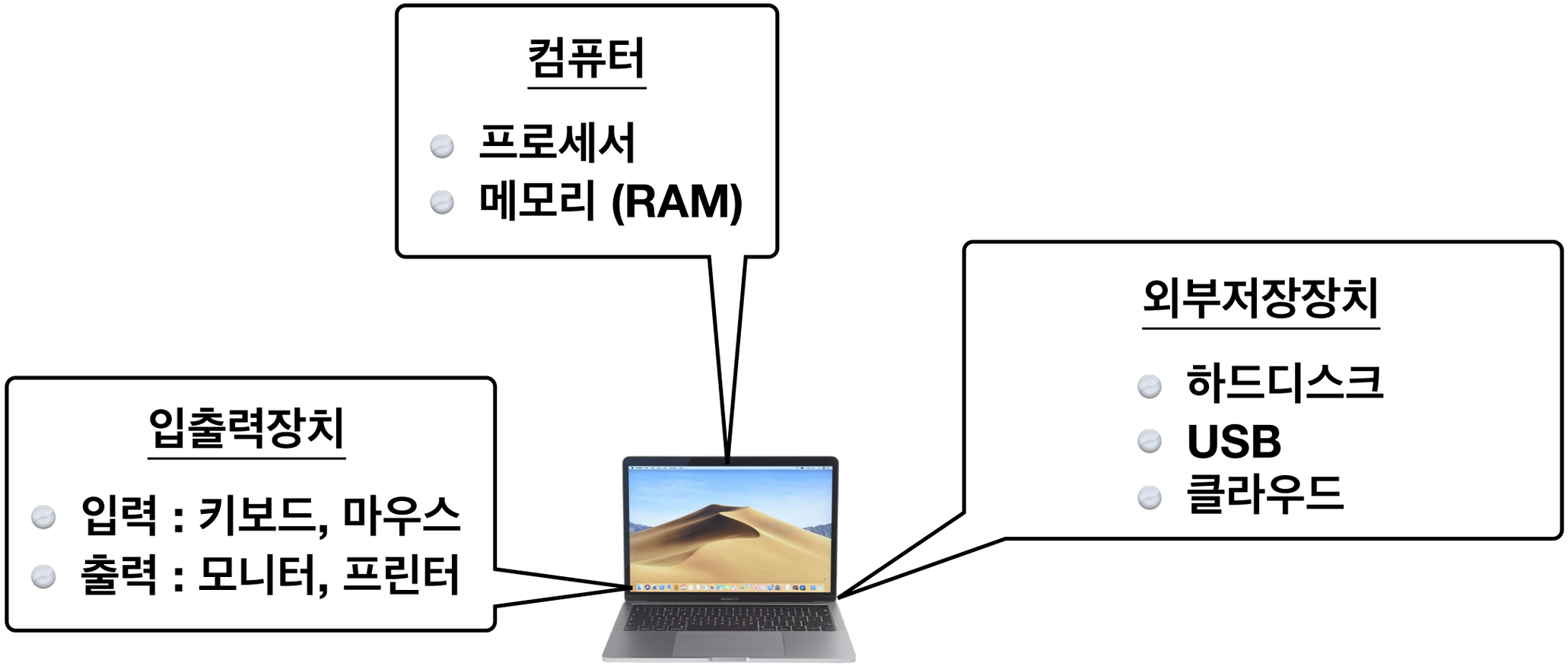
프로그래밍

Programming = Coding



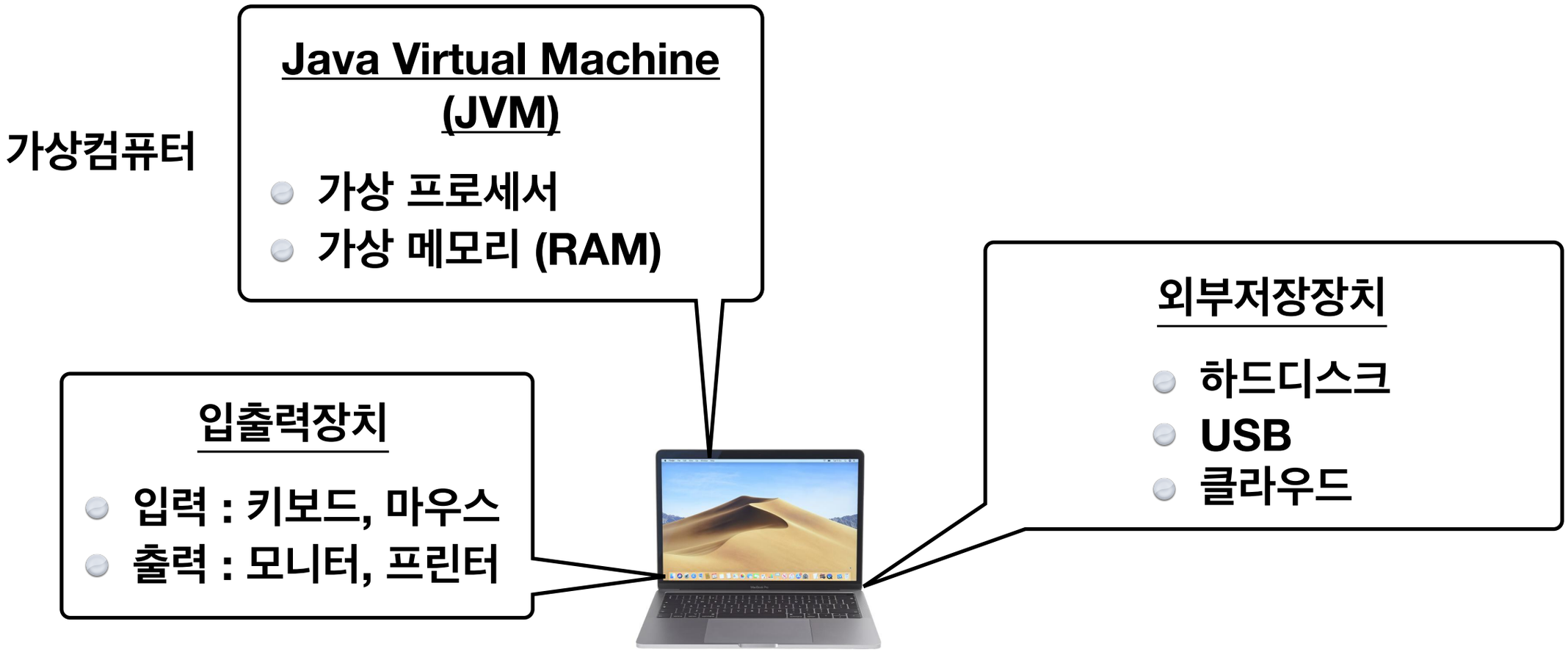
프로그래밍
Programming = Coding

- 프로그램을 사람수준 언어high-level language로 작성
- 기계수준 언어로 번역(컴파일compile)하여 메모리에 로딩
- 프로세서가 기계수준 프로그램을 실행



자바 프로그래밍
Java Programming

- 프로그램을 Java로 작성하여 파일에 저장
- 프로그램을 Java byte code로 컴파일하여 가상 메모리에 로딩
- 가상 프로세서가 Java byte code를 실행



프로그래밍

Programming = Coding

- 프로그램을 Java로 작성하여 외부저장장치에 파일로 저장
- 프로그램을 Java byte code로 컴파일하여 가상 메모리에 카피
- 가상 프로세서가 Java byte code를 실행

왜???

가상컴퓨터

Java Virtual Machine (JVM)

- 가상 프로세서
- 가상 메모리 (RAM)

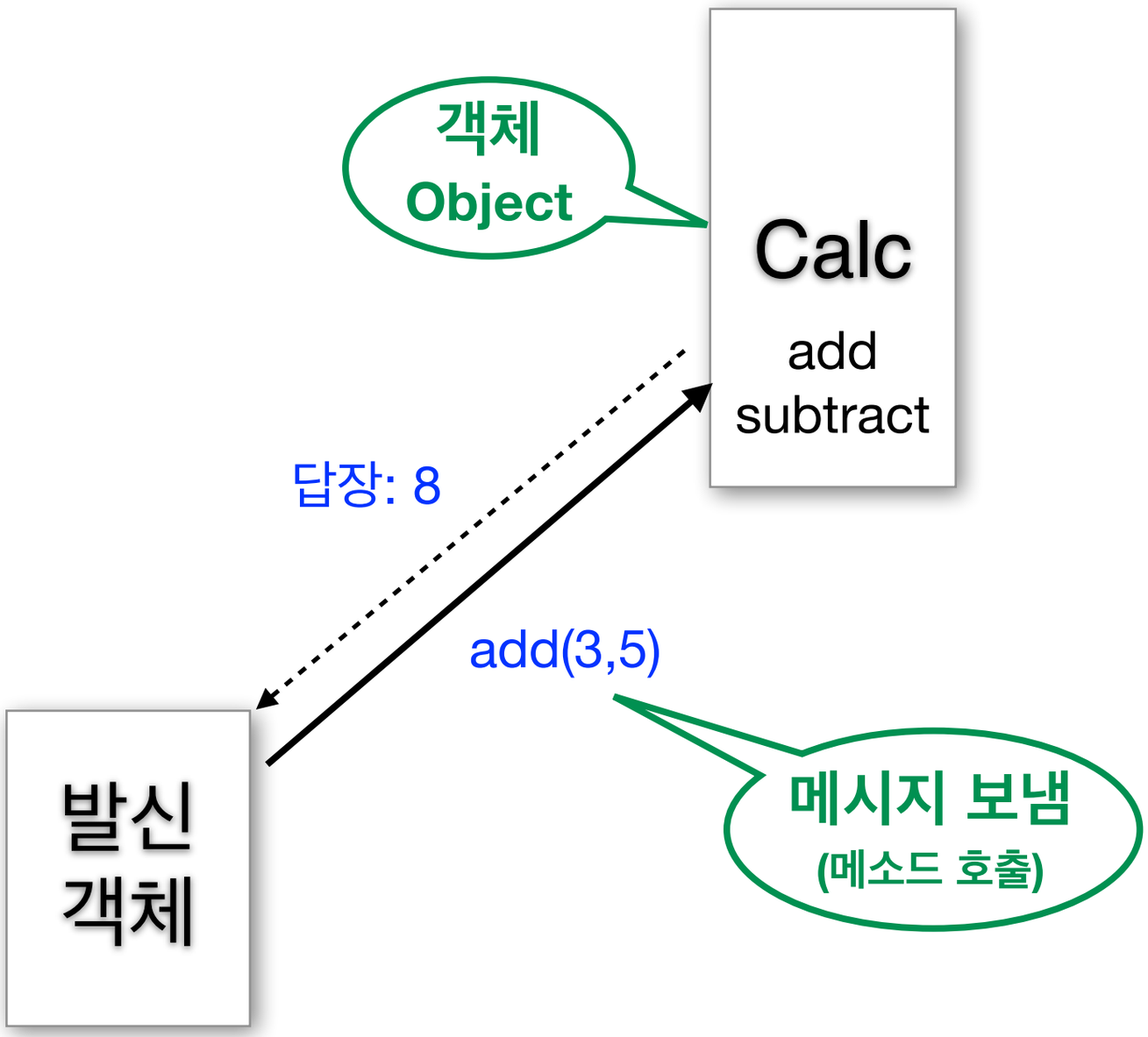
입출력장치

- 입력 : 키보드, 마우스
- 출력 : 모니터, 프린터

외부저장장치

- 하드디스크
- USB
- 클라우드

자바 프로그램의 작동 개념



계산

=

등장 객체들끼리

메시지 전달을 통한

소통의 연속

통합개발환경
IDE
(Integrated Development Environment)

Hello, World!

자바 애플리케이션

구현

자바 애플리케이션
Java Application
(표준 출력 버전)

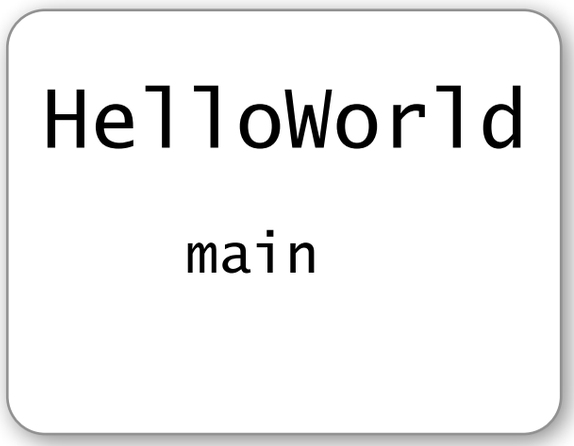
```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```

실행 추적
Execution Trace

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

애플리케이션 시동

❶ main 메소드 호출



실행 추적
Execution Trace

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

애플리케이션 시동

❶ main 메소드 호출

실행 추적
Execution Trace

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

실행 추적
Execution Trace

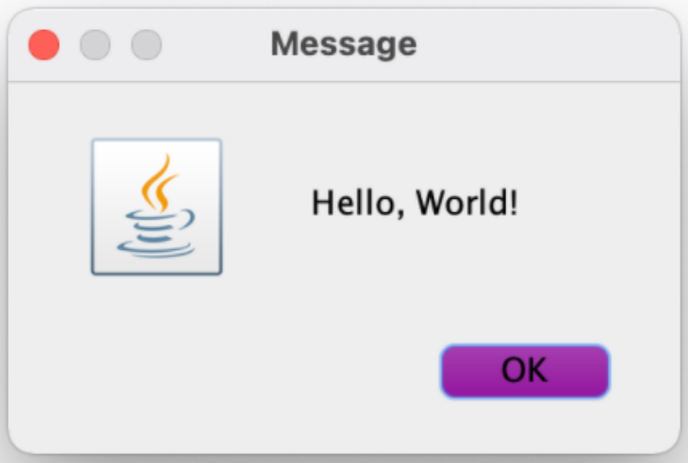
```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```

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

❶ main 메소드 호출

HelloWorldGUI

main

JOptionPane

showMessageDialog

실행 추적
Execution Trace

```
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!");
  }
}
```