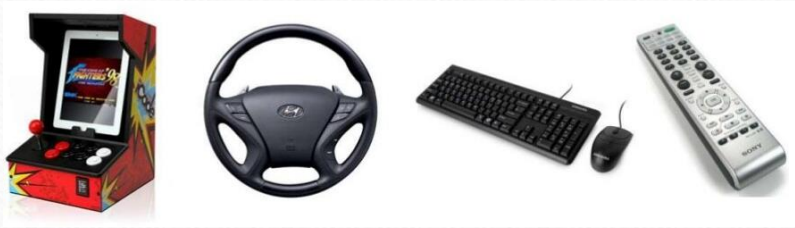
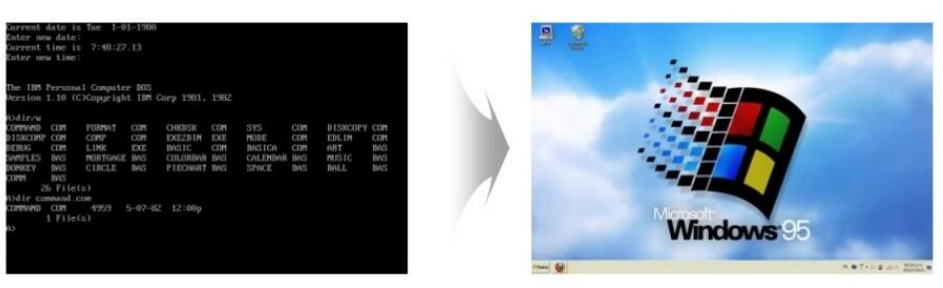
인터페이스

인터페이스(interface)

- 사용자가 이용할 수 있도록 드러낸 부분!



- GUI: 사용자 ⟷ 그래픽 ⟷ 애플리케이션



인터페이스(interface)

- 추상클래스의 추상메서드 처럼 추상화 정도가 높아져
  100% **추상 메서드와 상수만**을 멤버로 가질 수 있다.
- 인터페이스는 객체로 생성할 수 없음 (생성자 X)

```
interface MyInterface {

    // Only 상수

    // Only 추상 메서드

}
```

인터페이스 선언(상수)

- 데이터를 저장할 **인스턴스 또는 정적 멤버 선언 불가!**
- 즉, 오직 **상수**만 선언 가능

```
interface RemoteControl {

    int MIN_VOLUMN = 0;
    public int MAX_VOLUMN = 10;
    public static int a = 20;
    public static final int b = 30;

    // 모두 public static final 임!!!
}
```

인터페이스 선언(추상 메서드)

- 인터페이스의 메서드는 실행블록이 없는 **추상 메서드**로 선언

```
interface RemoteControl {

    void turnOn();
    public void turnOff();
    public abstract void setVolumn(int volumn);

    // 모두 public abstract 인 추상메서드!!!
}
```

인터페이스 구현

- 인터페이스에서 정의된 추상 메서드를 재정의(Override)해 실행내용을 가지고 있는 구현(implement) 클래스
- Implements 키워드를 추가하고 인터페이스 이름 기술

```
class Television implements RemoteControl {

    @Override
    public void turnOn() {
        System.out.println("TV를 켭니다.");
    }

    @Override
    public void turnOff() {
        System.out.println("TV를 끕니다.");
    }

}
```

인터페이스의 사용

- 인터페이스(interface)는

멤버변수, 매개변수, 로컬변수

타입으로 선언 가능

```
class MyClass {

    // 멤버변수
    RemoteControl rc;

    // 생성자의 매개변수로 활용
    MyClass(RemoteControl rc) {}

    // 메서드의 매개변수로 활용
    void methodA(RemoteControl rc) {}

    // 메서드의 로컬변수로 활용
    void methodB() {
        RemoteControl rc = new Television();
    }
}
```

인터페이스 특징

- 순도 100% 상수와 추상 메서드로 구성되어 있다.
- 인터페이스는 생성자를 가질 수 없다.
- 인터페이스간에 다중 상속(extends)이 가능하다.
- 구현 클래스에서 다중 구현(implements)이 가능하다.

```
// 인터페이스 간에 다중 상속
interface A {}
interface B {}
interface C extends A, B {}

// 구현클래스에서 다중 구현
class MyClass implements A, B, C {}
```

인터페이스의 필요성

- 구현의 **강제**로 표준화 가능 (구현의 강제성)
  - ex) 메서드명 통일
- 인터페이스를 통한 **간접적인 클래스 사용** (손쉬운 모듈 교체)
  - ex) 프린터교체, DB교체
- 관계가 없는 클래스들에게 **인터페이스를 통한 관계 부여**(다형성)
  - ex) 전자기기 충전
- 모듈 간 **독립적** 프로그래밍 (개발 시간 단축)
  - ex) UI개발 <-> 서버 로직 개발