프로그래밍 언어론
Lecture Note #09-1

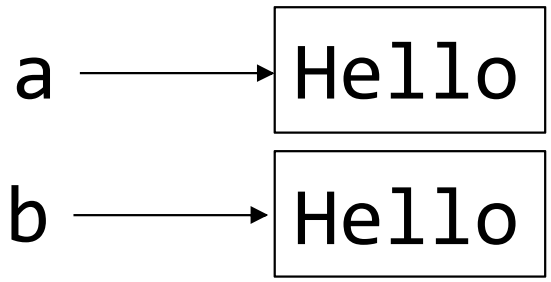
조용주
ycho@smu.ac.kr

Shallow/Deep Comparison

- Shallow Comparison
  - 객체를 참조하는 값을 비교
    - 예: 포인터 값을 비교 (C/C++) 또는 자바에서 참조값을 비교

```
String a = new String("Hello");
String b = new String("Hello");
if (a == b) { // shallow comparison
    System.out.println("a와 b가 같습니다.");
}
else {
    System.out.println("a와 b가 다릅니다.");
}
```

    - 이 코드를 실행시키면 a와 b가 다른 것으로 나타남
    - a와 b는 실제 다른 객체로 되어 있음



Shallow/Deep Comparison

- Shallow Comparison

```
char* s1 = (char*) malloc(MAX_LEN);
strcpy(s1, "Hello");
char* s2 = (char*)malloc(MAX_LEN);
strcpy(s2, "Hello");

if (s1 == s2) { // 주소값 비교 (shallow)
    printf("s1과 s2가 같습니다.");
}
else {
    printf("s1과 s2가 다릅니다.");
}
```

Shallow/Deep Comparison

- Deep Comparison
  - 객체 내용을 모두 비교
  - 예: 문자열은 문자들을 모두 비교

```
String a = new String("Hello");
String b = new String("Hello");
if (a.equals(b)) { // deep comparison
    System.out.println("a와 b가 같습니다.");
}
else {
    System.out.println("a와 b가 다릅니다.");
}
```

Shallow/Deep Comparison

- Deep Comparison

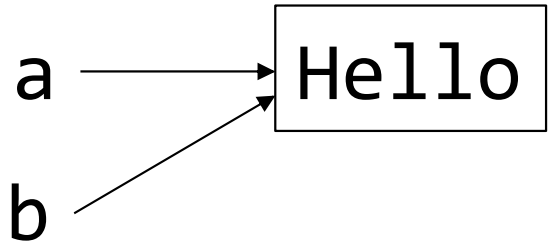
```
char* s1 = (char*) malloc(MAX_LEN);
strcpy(s1, "Hello");
char* s2 = (char*)malloc(MAX_LEN);
strcpy(s2, "Hello");

if (strcmp(s1, s2)) { // 문자열 내용 비교
    printf("s1과 s2가 같습니다.");
}
else {
    printf("s1과 s2가 다릅니다.");
}
```

Shallow/Deep Assignment

- Shallow Assignment (shallow copy라고도 함)
  - 객체의 참조 값 또는 포인터 값 저장(복사)
    - 예: 자바의 참조 값 또는 C/C++의 포인터 값을 저장

```
String a = new String("Hello");
String b = a; // 참조값만 복사
```

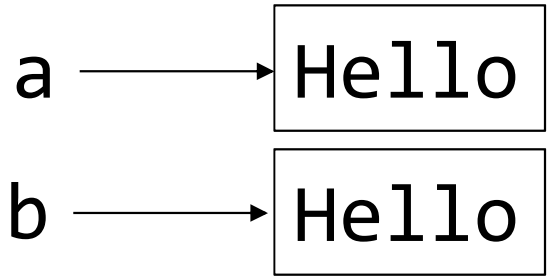



```
char* s1 = (char*) malloc(MAX_LEN);
strcpy(s1, "Hello");
char* s2 = s1; // 포인터 값만 복사
```

Shallow/Deep Assignment

- Deep Assignment (deep copy라고도 함)
  - 객체 내용을 전부 새로 복사
  - 문자열을 통째로 복사

```
String a = new String("Hello");
String b = new String(a); // 문자열을 모두 복사
String c = a.clone(); // 문자열을 새로 복사해서 생성
```




```
char* s1 = (char*) malloc(MAX_LEN);
strcpy(s1, "Hello");
char* s2 = (char*)malloc(MAX_LEN);
strcpy(s2, s1); // 문자열 복사
```

7.4 Equality Testing and Assignment

- C++ 버전

```
class A {
public:
    A(int x, int y) { a = x; b = y; }
    bool operator==(const A& c) {
        return (a == c.a && b == c.b);
    }

    A& operator=(const A& c) {
        a = c.a;
        b = c.b;
    }

private:
    int a;
    int b;
};
```

7.4 Equality Testing and Assignment

```
A a(3, 5);
A b(a); // deep assignment
if (a == b) { … } // deep comparison

A* p1 = new A(3, 5);
A* p2 = p1; // shallow assignment
if (p1 == p2) { … } // shallow comparison
```