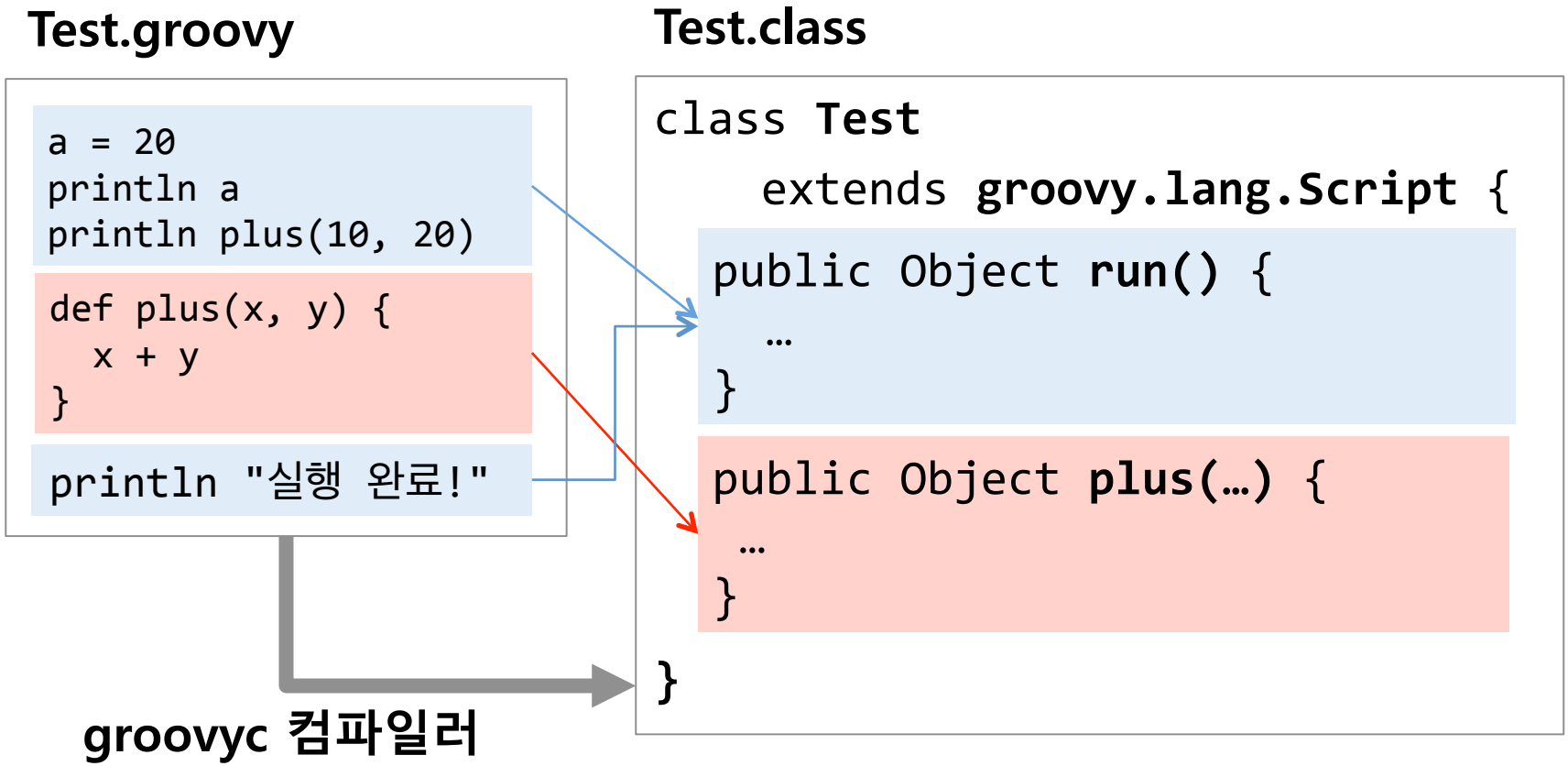
Seminar 08
Gradle과 Groovy

Groovy 소개

- **역사**
  - 2003년 8월, 제임스 스트라칸(James Strachan)이 블로그에서 Groovy 개발을 처음으로 언급
  - 2004년 3월, JSP(Java Community Process)에 자바 스펙 요구서 "JSR241"를 제출하고 승인 받음
  - 1.0(2007년 1월) →2.0(2012년 7월) → 2.2(2013년 11월)
  - 2.3(2014년 5월 예정) → 3.0(2014년 4분기 예정)
- **주요 특징**
  - JVM에서 실행됨
  - 파이썬(Python), 루비(Ruby), 스몰토크(Smalltalk)의 특징 포함
  - 자바와 유사한 문법. 자바보다 더 간결함. 자바 코드를 그대로 사용 가능
  - 정적 및 동적 타입 모두 지원
  - 클로저 및 연산자 오버로딩 지원
  - Groovy 2.0부터 정적 컴파일 및 타입 체킹 지원

Groovy 실행 원리



- **groovy로 만든 자바 클래스를 실행하기**

```
프롬프트> java –classpath groovy.jar:asm.jar Test
```

변수선언과 값 할당

- **동적 타입 바인딩**

```
def a = 20        // 값을 할당할 때 a 변수의 타입이 결정됨
a = “문자열”      // 다른 값을 할당하면 변수의 타입이 바뀜
b = “문자열”      // 변수 선언 시 def를 생략해도 됨
```

- **정적 타입 바인딩**

```
int a = 20        // 변수 a는 int로 데이터형이 고정됨
a = “문자열”      // 문자열 값을 할당하면 오류 발생!
```

- **문자열과 자동 형변환**

```
String a = “문자열”        // 변수 a는 String으로 데이터 형이 고정됨
a = 20  // 정수 값 20이 문자열 “20”으로 자동 변환된 후 변수 a에 할당됨
```

- **자바 변수 선언**

```
java.util.Date a = new java.util.Date()  // 자바 문법 사용 가능
Date b = new Date() // 기본으로 java.util 패키지 임포트됨
```

리스트와 맵 데이터 다루기

- **리스트 데이터 할당과 사용**

```
scoreList = [90, 80, 100, 87]    // ArrayList 객체가 생성됨
println scoreList[2]     // 배열처럼 사용할 수 있음
println scoreList.get(2) // 원래대로 ArrayList의 get() 호출 가능
println scoreList.size() // 결과는 4
nameList = ["홍길동", "임꺽정", "장보고"]
println nameList[1]
emptyList = []  // 빈 리스트 생성
```

- **맵 데이터 할당과 사용**

```
scoreMap = ["국어":100,"영어":90,"수학":100,"사회":89]
println scoreMap["수학"] // 변수.[키] 형태로 값 추출
println scoreMap.수학    // 변수.키 형태로 값 추출
println scoreMap.getClass()  // java.util.LinkedHashMap 출력
scoreMap.수학 = 93       // 맵의 값 변경
println scoreMap.수학    // 결과는 93
emptyMap = [:]           // 빈 맵 생성
```

분기문

- **if – else 문**

```
if (true 또는 false 조건) {
  …
} else if ( … ) {
  …
} else {
  …
}
```

- **조건 연산자**

```
age = 17
title = (age < 19) ?
        "청소년" : "성인"
```

그루비의 switch 문은 자바와 달리 숫자나 문자열 외에 실수, 불린, 객체도 다룰 수 있음

- **switch 문**

```
x = "aaa"
result = ""
switch(x) {
case "aaa": result = "aaa"
case "123": result += "123"
case [1,20,3.14, true]:
  result = "숫자,문자열,참거짓 값"
  break
case 100..200:
  result = "100 ~ 200 값"; break
case Number:
  result = "기타 숫자 값"; break
default:
  result = "기타"
}
println result
```

반복문

- **while 문**

```
while (조건) { … }
```

- **for 문**

```
for (int i = 0; i < 5; i++) {}
for (i in 0..9) {}
for (i in [100, 90, 95, 80]) {}
for (i in 배열) {}
for (entry in 맵) {}
for (c in 문자열) {}
```

- **eachXXX()**

```
목록.each({ … })
맵.each({key, value ->  … })
목록.eachWithIndex({obj,i -> … })
맵.eachWithIndex({obj, i -> … })
```

- **for 문 예제**

```
맵 = ['kor':100,'eng':90, …]
for (e in 맵) {
  println e.key + "=" + e.value
}

for (v in 맵.values()) {
  println v
}

for (c in "abcdef") {
  print c + ","
}
```

메서드와 클로저

- **메서드 정의**

```
def plus(a, b) {
  a + b
}

int minus(int a, int b) {
  return a - b
}
```

- **클로저 정의**

```
plus = {a, b ->
  a + b
}
println plus(10, 20)

plus = {int a, int b ->
  return a + b
}
```

- **클로저 Free 변수**

```
pi = 3.14159
getCircleArea = {radius ->
  //pi는 'free' 변수
  pi * radius * radius
}
```

- **클로저 it 파라미터**

```
plus = {
  it[0] + it[1]
}
println plus([10, 20])
```

- **클로저 리턴 값**

```
plus = { a, b ->
  a + b // 마지막 문장의 실행 값
}
// return 을 사용해도 됨
// 마지막 문장의 실행 값이 없으면 null
```

클래스

- **기본으로 임포트(import)하는 패키지**

java.lang, java.io, java.math, java.net, java.util, groovy.lang, groovy.util

- **클래스 정의와 사용**

```
class Test {
  void hello() {
    println "안녕하세요!"
  }
}

t = new Test()
t.hello()
```

- **프로퍼티 선언과 사용**

```
class Student {
  Integer no
  String name
  Date regDate
}

s = new Student(no:1,
  name:"홍길동",
  regDate:new Date())

println s.no + "," + s.name +
  "," + s.regDate
```

String과 GString

- **문자열 표현 " 과 ', /**

```
println "I like 'groovy'."
println 'I like "groovy”.'
println (/I like "groovy"./)
```

- **멀티 라인 문자열**

```
intro = """ Hi, My name is J.
 I am teaching Groovy script.
 Let's go Groovy programming."""
println intro
```

- **GString**

```
name = "Jinyoung"
println "Hi, $name."
println "Hi, \$name."
```

- **GString과 String**

```
str1 = "Jinyoung"
str2 = "My name is $str1"
str3 = str1 + str2
assert str1 instanceof String
assert str2 instanceof GString
assert str3 instanceof String
```

- **GString과 Closure**

```
now = new Date()
str1 = "일반객체: $now 입니다."
str2 = "파라미터 없는 클로저: ${new Date()}"
str3 = "파라미터 있는 클로저: ${writer -> writer << new Date()}"
```