**Test #12. 2013.5.27 프로그래밍언어론 01 분반**

학번:

이름:

1. 다음 각 항목이 참이면 O, 아니면 X 를 하시오.
2. C++에서 Student \* s = new Person(); 와 같은 assignment는 타입 오류가 발생하지 않는다.
3. 다음 Smalltalk 코드가 수행되면 count의 값은 21이 된다.

count <- 0.

sum <- 20.

addIndex <- [count <= 20]

whileTrue: [sum <- sum + count.

count <- count + 1]

addIndex value

1. 정적 바인딩은 동적 바인딩에 비해 실행 비용이 많이 든다.
2. C++에서 기존 클래스에서 private으로 상속 받아 새로운 클래스를 만들 때는 상위 클래스의 어떤 데이터 구조도 접근이 불가능하다.
3. 상하위 타입은 항상 상하위 클래스와 일치한다.
4. Java에서 인터페이스 I1과 클래스 A1 가 아래와 같이 주어졌을 때,

interface I1 {

int nextTo(I1 arg1) ;

}

class A1 implements I1 {

public A1(){}

public int nextTo (I1 arg1){…}

}

메소드 nextTo () 에 전달할 객체를 생성해서 호출하도록 아래 코드의 빈칸 (1)을 채우시오.

I1 i0 = \_(1)\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

A1 a0 = new A1();

a0.nextTo(i0);

1. Java에서 상위 클래스 Super과 하위 클래스 Sub 모두 void print() 메소드의 구현을 가지고 있다고 가정하자. Super 타입 변수 y에 대해 다음과 같은 호출을 할 때

y.print();

어떤 경우 어느 클래스의 print()가 불리게 되는지 적당한 조건들을 도입해서 설명하시오.

답.

1. 다음 각 항목이 참이면 O, 아니면 X 를 하시오.
2. C++에서 Student \* s = new Person()와 같은 assignment는 타입 오류가 발생하지 않는다. X
3. 다음 Smalltalk 코드가 수행되면 count의 값은 21이 된다. O

count <- 0.

sum <- 20.

addIndex <- [count <= 20]

whileTrue: [sum <- sum + count.

count <- count + 1]

addIndex value

1. 정적 바인딩은 동적 바인딩에 비해 실행 비용이 많이 든다. X
2. C++에서 기존 클래스에서 private으로 상속 받아 새로운 클래스를 만들 때는 상위 클래스의 어떤 데이터 구조도 접근이 불가능하다. O
3. 상하위 타입은 항상 상하위 클래스와 일치한다. X
4. Java에서 클래스 A1 가 아래와 같이 주어졌을 때,

interface I1 {

int nextTo(I1 arg1) ;

}

class A1 implements I1 {

A1(){}

int nextTo (I1 arg1){…}

}

메소드 nextTo () 에 전달할 객체를 생성해서 호출하도록 아래 코드의 빈칸 (1)을 채우시오.

I1 i0 = \_(1)\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

A1 a0 = new A1();

a0. nextTo (i0);

답: new A1();

1. Java에서 상위 클래스 Super과 하위 클래스 Sub 모두 void print() 메소드의 구현을 가지고 있다고 가정하자. Super 타입 변수 y에 대해 다음과 같은 호출을 할 때

y.print();

어느 클래스의 print()가 불리게 되는지, 적당한 조건들을 도입하고 각 조건마다 설명하시오.

답: y가 Sub 객체를 의미한다면( “new Sub(..)” 로 생성된 객체) Sub의 print(), y가 Super 객체를 의미한다면 Super의 print()이다.