**Test #12. 2012.5.31 프로그래밍언어론 01 분반- 개인 문제**

학번:

이름:

조번호:

1. 다음 각 항목이 참이면 O, 아니면 X 를 하시오.
   1. 상위클래스의 객체는 하위클래스에도 속하는 것으로 간주한다.

※ (2~7)상위클래스 Person의 하위클래스가 Student 일때,

* 1. 클래스 Person에서 제공하는 메소드 void hello()의 구현을 그대로 상속받지 않고 클래스 Student에서 자신에게 맞게 변경해서 void hello()를 정의하는 것을 오버라이드라고 한다.
  2. C++에서 Person \* x = new Student()와 같은 assignment는 타입 오류가 발생하지 않는다.
  3. C++에서 Student \* s = new Person()와 같은 assignment는 타입 오류가 발생하지 않는다.
  4. 어떤 변수 x가 Student 타입이라면, x가 가리키는 것은 Student 클래스 객체일 수도 있고, 그의 상위 클래스Person의 객체일 수도 있다.
  5. 상위클래스 Person과 하위 클래스 Student 모두 void hello() 메소드의 구현을 가지고 있는 경우, Person 타입 변수 y에 대해 다음과 같은 호출을 할 때

y.hello();

C++ 에서는 별도의 조치가 없으면 y의 값에 상관없이 Person에 정의된 hello()가 불린다.

* 1. 위 (6)번의 경우 Java에서는 y의 값에 상관없이 Student에 정의된 hello()가 불린다.
  2. 다음 Smalltalk 코드가 수행되면 sum의 값은 count의 값은 21이 된다.

count <- 0.

sum <- 20.

addIndex <- [count <= 20]

whileTrue: [sum <- sum + count.

count <- count + 1]

addIndex value

* 1. C++에서 기존 클래스에서 private으로 상속 받아 새로운 클래스를 만들 때는 상위 클래스의 어떤 데이터 구조도 접근이 불가능하다.
  2. 정적 바인딩은 동적 바인딩에 비해 실행 비용이 많이 든다.

1. 상속할 때 하위 클래스가 상위클래스의 구현 내용을 열람하도록 허용하는 경우, 그렇지 않은 경우에 비해 가지는 장단점을 적으시오.
2. 강의자료에서 언급된Java의 interface의 특징이나 장점을 생각나는대로 세가지만 적으시오.

**Test #12. 2012.5.31 프로그래밍언어론 01 분반 조별 문제**

조번호:

참여자 (학번, 이름)

-

-

-

-

-

1. 연상
2. 다음 Java 프로그램에서 main의 수행 결과를 적으시오.

**import** java.util.\*;

**public** **class** M {

List s;

**public** M() { s = **new** ArrayList();}

**public** Object get (**int** i) {**return** s.get(i);}

**public** **void** add(Object e) { s.add(e);}

**public** **void** addAll(M c) {

**for** (**int** i=0; i < c.s.size(); i++){

add(c.get(i));

}

}

}

**public** **class** CountedM **extends** M{

**private** **int** addCount= 0;

**public** CountedM() { }

**public** **void** add(Object e) {

addCount++;

**super**.add(e);

}

**public** **void** addAll(M c) {

addCount += c.s.size();

**super**.addAll(c);

}

**public** **int** size() { **return** addCount;}

**public** **static** **void** main(String[] args){

CountedM m1 = **new** CountedM();

m1.add("a"); m1.add("b"); m1.add("c"); m1.add("d");

System.*out*.println("m1=" + m1.size());

CountedM m2 = **new** CountedM();

m2.addAll(m1);

System.*out*.println("m2=" + m2.size());

}

}