객체지향프로그래밍 – 9주차 실습활동지

성명: _____ 학과: _____ 학번: _____ 실습실명: _____호

※ 본 실습활동지를 작성함에 있어 다른 학생의 문서로부터 일부 또는 전체를 복사하였습니까?
예( ) 아니오( ) (복사 하였다면 예에 체크하고 아니라면 아니오에 체크하시오)

제출방식: 별도의 답안지 문서에 각 문제에 대하여 작성한 답안이나 작성한 코드 및 그 실행 결과를 복사해 넣는다. 답안지 문서명은 "report_주차_학번" (예, report_9_201811111)로 만든다. 답안지는 pdf형식으로 저장하여 과제게시판(Bb)에 차주 월요일 23시59분까지 제출한다.

I. Objectives

1. Interface 및 abstract class의 차이를 이해한다.
2. Interface와 polymorphism의 관계를 이해하고 활용할 수 있다.
3. Object class의 개념을 이해하고 이를 활용할 수 있다.

II. Exercises (15점)

1. Employee.java와 EmployeeSortTest.java(첨부파일 제공)를 수행해 보고 다음 물음에 답하시오.

(가) Employee 객체를 이름순으로 정렬하도록 Employee class를 수정해보시오. EmployeeSortTest class는 그대로 이용한다. (수정된 부분 코드 포함, 수행결과 포함) (2점)
힌트) compareTo method만 변경하여 보여주면 됨.

(나) Employee.java를 수정하지 않고 Comparator<Employee> 인터페이스를 이용하여 Employee객체를 이름순으로 정렬하고자 한다. 단, EmployeeSortTest 클래스는 정렬부분을 다음과 같이 수정한다.

```
...
Arrays.sort(staff, new NameComparator);
...
```

아래와 같이 NameComparator 클래스를 정의하여 Employee.java에 추가한 후, EmployeeSortTest2를 실행해보시오.

```
class NameComparator implements Comparator<Employee>
{

}
```

(NameComparator<Employee> 코드 포함, 수행결과 포함) (3점)

2. 다음은 abstract class를 이용한 프로그램이다. 지문을 읽고 코드를 실행해본 후 아래 물음에 답하시

오.

Dog, Cat, Duck과 같이 동물들은 각각 고유한 울음소리를 가지고 있다. 각 동물의 고유한 울음소리를 내는 메소드로 cry()를 정의하고 있다. 울음 소리를 내는 작업은 울음소리에 해당하는 문자열을 출력하도록 한다.

```
// Animal.java
public abstrct class Animal
 {
      public abstract void cry();
 }
class Dog extends Animal
{
     public void cry() {
          System.out.println("Bow Wow!");
     }
}
class Cat  extends Animal
{
     public void cry() {
          System.out.println(Meow!");
     }
}
 class Duck  extends Animal
{
     public void cry() {
          System.out.println("Quack Quack!");
     }
}

// AnimalTest.java
public class AnimalTest
{
       public static void main(String[] args) {
            Animal[] animals = new Animal[3];
            anmials[0] = new Dog();
            anmials[1] = new Cat();
            anmials[2] = new Duck();

            for(int i = 0; i < 3; i++) {
                 animals[i].cry();
            }
```

```
        }
    }
```

(가) Spider class를 동물의 subclass로 추가하고자 한다. 그런데 Spider는 일반적으로 소리를 내지 않는다고 가정하자. 어떤 문제가 있는가? (1점) (설명포함)

(나) (가) 번의 문제를 해결하기 위하여 '소리를 낼 수 있는 것'들을 AbleToCry interface로 정의하고 Dog, Cat, Duck 클래스들을 Animal class를 상속을 받으면서 AbleToCry interface를 구현하여 다시 정의하고자 한다. AbleToCry interface와 각 클래스를 정의하여 아래 AbleToCryTest.java를 이용하여 실행해 보시오. Dog, Cat, Duck 외에도 Siren 클래스도 subtype으로 추가해보시오. 단, Siren 소리는 'wee-oww wee-oww"라고 가정하자 (3점) (코드 및 수행결과 포함)

```
// AbleToCryTest.java
public class AbleToCryTest
{
        public static void main(String[] args) {
            AbleToCry[] cryings = new AbleToCry[4];
            cryings[0] = new Dog();
            cryings[1] = new Cat();
            cryings[2] = new Duck();
            cryings[3] = new Siren();

            for(int i = 0; i < 4; i++) {
                cryings[i].cry();
            }
        }
    }
```

3. 강의노트 Chapter 5 part 3(Abstract Class)에서 사용한 Employee class 계층도 (첨부파일 제공)에 대한 ObjectClassTest 클래스를 실행하여 Employee class 및 SalariedEmployee class의 equals() 및 toString() method의 용도 및 의미를 파악한 후 다음 물음에 답하시오.

(가) Employee class의 각 subclass에 toString() method를 추가한 후 ObjectClassTest class를 수정하여 이를 테스트하시오. 힌트) SalariedEmployee class에 정의된 toString() method를 참조할 것. (3점)

(나) Employee class의 각 subclass에 equals() method를 추가한 후 bjectClassTest class를 수정하여 이를 테스트하시오. 힌트) SalariedEmployee class에 정의된 equals() method를 참조할 것. (3점)