객체지향프로그래밍 – 7주차 실습 활동지

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_7_201811111)로 만든다. 답안지는 pdf형식으로 저장하여 과제게시판(Bb)에 차주 월요일 23시59분까지 제출한다.

I. Objectives

1. Inheritance의 기본 구조 (superclass와 subclass의 관계)를 이해하고 subclass를 작성할 수 있다.
2. Public 및 private access modifier의 차이를 이해한다.
3. Override의 개념을 이해하고 polymorphism을 활용할 수 있다.
4. Abstract class와 polymorphism의 관계를 이해하고 활용할 수 있다.

II. Review (15점)

강의노트의 payroll system (첨부파일 제공, Employee, SalariedEmployee, HourlyEmployee, CommissionEmployee, Manager, PayrollSystemTest 클래스)를 수행해 보고 다음 물음에 답하시오. 코드 속에 아직 다루지 않은 exception은 무시해도 좋다.

1. 위 프로그램을 수행해보시오. 각 결과 값이 나온 근거를 설명하시오. **(결과값 및 설명 포함) (1점)**

2. Manager class의 getEarnings 메소드 본문을 아래와 같이 변경하여 컴파일해보시오. 어떤 결과가 나오는가? 그 이유는 무엇인가? **(컴파일 결과 및 설명 포함) (1점)**

```
return weeklySalary + bonus;
```

3. Manager class의 getEarnings 메소드의 이름을 getearnings으로 변경한 후 실행해 보시오. 어떤 결과가 나오는가? 왜 그런지 이유를 설명하시오. **(수행결과 및 설명 포함) (2점)**

4. 3번에서 수정한 getearnings 메소드에서 @Override를 삭제한 후 실행해 보시오. 1번의 결과값과 비교해보시오? 왜 차이가 나는지 설명하고 @Override의 역할에 대해 설명하시오. **(실행 결과 및 설명 포함) (2점)**

5. 다음은 employee[3]에 저장된 Manager object의 bonus를 증가시킨 후 정보를 출력하는 코드이다. PayrollSystemTest class의 main함수의 끝부분에 다음과 같은 코드를 넣은 후 컴파일시켜 보시오. 어떤 문제가 발생하며 왜 그런지 설명하시오. **(컴파일 결과 및 설명 포함) (2점)**

```
employees[3].setBonus(1000.0);
System.out.printf("%s %s: $%,.2f%n%n",
        employees[3].getName(), "earned", employees[3].getEarnings());
```

6. 5번 문제의 문제를 해결하여 bonus가 반영된 급여가 출력될 수 있도록 코드를 수정해보시오. **(해결 부분 코드 포함) (2점)**

7. Employee class modifier에서 다음과 같이 abstract를 없애고 또한 getEarnings 함수를 다음과 같이 변경한다.

```
public double getEarnings() { return 0.0; }
```

   이 프로그램을 수행하면 결과가 어떻게 나오나? 기존 프로그램과 비교하시오. **(결과 및 설명 포함) (1점)**

8. 7번처럼 변경하기 전과 변경한 후에 아래 코드를 main()함수에 추가하여 실행해보시오. 어떤 차이가 나며 왜 그런지 설명하시오. **(설명만 포함) (1점)**

```
Employee e = new Employee("Kildong", "000-00-0000");
```

9. 각 constructor에 하나의 출력문을 추가하여 수행해본 후 상속을 하는 경우 constructor들이 일반적으로 어떤 순서로 실행되는지 설명하시오. **(실행결과 화면 및 설명포함) (1점)**

10. CommissionEmployee class를 상속받아 BasePlusCommissionEmployee class를 정의하고자 한다. BasePlusCommissionEmployee는 판매액에 따른 커미션 외에 기본 base salary가 소득에 추가된다. 아래 getEarnings method 코드를 채우시오. **(해당 코드 부분 포함) (2점)**

```
public class BasePlusCommissionEmployee extends CommissionEmployee
{
   private double baseSalary; // base salary per week

   // constructor
   public BasePlusCommissionEmployee(String name,
      String socialSecurityNumber, double grossSales,
      double commissionRate, double baseSalary)
   {
      super(name, socialSecurityNumber, grossSales, commissionRate);

      if (baseSalary < 0.0) // validate baseSalary
         throw new IllegalArgumentException("Base salary must be >= 0.0");

      this.baseSalary = baseSalary;
   }
```

```
    // set base salary
    public void setBaseSalary(double baseSalary)
    {
        if (baseSalary < 0.0) // validate baseSalary
            throw new IllegalArgumentException("Base salary must be >= 0.0");

        this.baseSalary = baseSalary;
    }

    // return base salary
    public double getBaseSalary()
    {
        return baseSalary;
    }

    // calculate earnings; override method earnings in CommissionEmployee
    @Override
    public double getEarnings()
    {

    }

} // end class BasePlusCommissionEmployee
```