Examples

Week 14

Upcasting (상위클래스 변수 ← 하위클래스 변수 or 객체)

```
class Person {
    String name;
    String id;

    public Person(String name) {
        this.name = name;
    }
}

class Student extends Person {
    String grade;
    String department;

    public Student(String name) {
        super(name);
    }
}
```

```
public class UpcastingEx {
  public static void main(String[] args) {
        Person p;
        Student s = new Student("James");
        p = s;

        System.out.println(p.name);
        p.grade = "A";          // error
        p.department = "SW";    // error
  }
}
```

```
James
```

컴파일할 때는 변수의 타입을 본다.

Downcasting (하위클래스 변수 ← 상위클래스 변수)

```
class Person {
    String name;
    String id;

    public Person(String name) {
        this.name = name;
    }
}

class Student extends Person {
    String grade;
    String department;

    public Student(String name) {
        super(name);
    }
}
```

```
public class DowncastingEx {
  public static void main(String[] args) {
        Person p = new Student("James");
        Student s;

        s = (Student) p;

        System.out.println(s.name);
        s.grade = "A";
  }
}
```

```
James
```

1. 강제형변환 필요
2. 실제로는 하위클래스 객체

A instanceof B (A가 B 타입의 객체이면 true, 아니면 false)

```
class Person { }
class Student extends Person { }
class Researcher extends Person { }
class Professor extends Researcher { }

public class InstanceOfEx {
    static void print(Person p) {
        if ( p instanceof Person )  System.out.print("Person ");
        if ( p instanceof Student )  System.out.print("Student ");
        if ( p instanceof Researcher )  System.out.print("Researcher ");
        if ( p instanceof Professor )  System.out.print("Professor ");
        System.out.println();
    }
    public static void main(String[] args) {
        System.out.print("new Student() ->\t"); print( new Student() );
        System.out.print("new Researcher() ->\t"); print( new Researcher() );
        System.out.print("new Professor() ->\t"); print( new Professor() );
    }
}
```

하위클래스 객체는
하위클래스 타입이기도 하고
상위클래스 타입이기도 하다.

```
new Student() ->        Person Student
new Researcher() ->     Person Researcher
new Professor() ->      Person Researcher Professor
```

Method Overriding (상속받은 메소드를 다시 구현)

```
class Shape {
    public Shape next;
    public Shape() { next = null; }
    public void draw() {
        System.out.println("Shape");
    }
}

class Line extends Shape {
    public void draw() {
        System.out.println("Line");
    }
}

class Rect extends Shape {
    public void draw() {
        System.out.println("Rect");
    }
}
```

```
class Circle extends Shape {
    public void draw() {
        System.out.println("Circle");
    }
}
```

실행할 때는 객체의 타입을 본다.

```
public class MethodOverridingEx
{
   static void paint(Shape p) {
        p.draw();
   }
   public static void main(String[] args) {
        Line line = new Line();
        paint( line );
        paint( new Shape() );
        paint( new Line() );
        paint( new Rect() );
        paint( new Circle() );
   }
}
```

```
Line
Shape
Line
Rect
Circle
```

Method Overriding 활용(객체들의 연결리스트)

```java
// Shape, Line, Rect, Circle 클래스는 앞의 예제와 동일
class Shape { ... }
class Line extends Shape { ... }
class Rect extends Shape { ... }
class Circle extends Shape { ... }

public class UsingOverride {
    public static void main(String[] args) {
        Shape start, last, obj;
        obj = new Line(); start = obj; last = obj;
        obj = new Rect(); last.next = obj; last = obj;
        obj = new Line(); last.next = obj; last = obj;
        obj = new Circle(); last.next = obj; last = obj;

        Shape p = start;
        while ( p!= null ) {
            p.draw();
            p = p.next;
        }
    }
}
```

```
Line
Rect
Line
Circle
```

동적바인딩 (실제로 실행할 메소드를 실행할 때 결정)

```
class Shape
{
    protected String name;

    public void paint() {
        draw();
    }

    public void draw() {
        System.out.println("Shape");
    }
}
```

1. 정적바인딩: 실행할 메소드를 컴파일 할 때 결정함
2. 동적바인딩: 실행할 메소드를 실행할 때 결정함
3. 인스턴스메소드(non-static method) 호출은 동적바인딩을 함

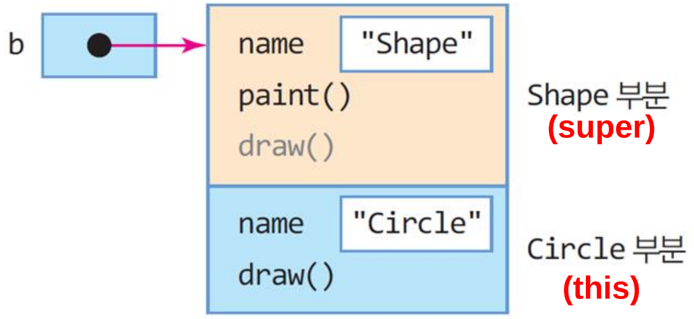
```
public class Circle extends Shape
{
   @Override
   public void draw() {
        System.out.println("Circle");
   }

   public static void main( String[] args )
   {
        Shape b = new Circle();
        b.paint();
   }
}
```

```
Circle
```

super 레퍼런스 (상위클래스 멤버 접근할 때 사용)

```
class Shape {
    protected String name;
    public void paint() {
        draw();
    }
    public void draw() {
        System.out.println(name);
    }
}
```



```
public class Circle extends Shape {
    protected String name;
    @Override
    public void draw() {
        name = "circle";
        super.name = "Shape";
        super.draw();
        System.out.println(name);
    }
    public static void main(String[] args)
    {
        Shape b = new Circle();
        b.paint();
    }
}
```

```
Shape
Circle
```

EmployeeTest.java

- 어느 회사의 직원은 다음 4가지 타입 중 하나이다.
  - salaried employee: 매달 일정한 임금을 받음.
  - hourly employee: 근무한 시간만큼 시급을 받음.
  - commission employee: 매출의 일정 비율을 받음.
  - base+commission employee: 기본급+(매출의 일정 비율)을 받음

- 다형성을 이용하여 직원정보를 출력하고, 이번 달 총 임금을 계산하여 출력한다.

EmployeeTest.java

```java
class Employee
{
   private String name;
   private String id;
   static private int count = 0;

   public Employee(String name, String id)
   {
      this.name = name;
      this.id = id;
      count++;
   }

   public double earnings()
   {
      return 0;
   }

   public String toString()
   {
      return name + "(" + id + ")";
   }

   public static int getCount()
   {
      return count;
   }
}
```

```java
class SalariedEmployee extends Employee
{
   private double monthlySalary;

   public SalariedEmployee(String name,
                    String id, double salary)
   {
      super(name, id);
      monthlySalary = salary;
   }

   public double earnings()
   {
       return 0;
   }

   public String toString()
   {
       return "??";
   }
}
```

EmployeeTest.java

```java
class HourlyEmployee extends Employee
{
   private double wage;
   private double hours;

   public HourlyEmployee(String name,
     String id, double wage, double hours)
   {
      super(name, id);
      this.wage = wage;
      this.hours = hours;
   }

   public double earnings()
   {
      return 0;
   }

   public String toString()
   {
      return "??";
   }
}
```

```java
class CommissionEmployee extends Employee
{
   private double grossSales;
   private double commissionRate;

   public CommissionEmployee(String name,
       String id, double sales, double rate)
   {
      super(name, id);
      grossSales = sales;
      commissionRate = rate;
   }

   public double earnings()
   {
      return 0;
   }

   public String toString()
   {
      return "??";
   }
}
```

EmployeeTest.java

```java
class BasePlusCommissionEmployee extends CommissionEmployee
{
   private double baseSalary;

   public BasePlusCommissionEmployee(String name, String id, double sales,
                                                              double rate, double salary)
   {
      super(name, id, sales, rate);
      baseSalary = salary;
   }

   public double earnings()
   {
      return 0;
   }

   public String toString()
   {
      return "??";
   }
}
```

EmployeeTest.java

```java
public class EmployeeTest
{
    public static void main(String[] args)
    {
        Employee[] arr = new Employee[4];
        arr[0] = new SalariedEmployee("Smith", "s1111", 300);
        arr[1] = new HourlyEmployee("Karen", "h2222", 1, 160);
        arr[2] = new CommissionEmployee("Jones", "c3333", 2000, 0.1);
        arr[3] = new BasePlusCommissionEmployee("Lewis", "b4444", 2000, 0.06, 100);

        double sum = 0.0;
        for( Employee e : arr )
        {
            System.out.println( e );
            System.out.println( "payment: " + e.earnings() );
            System.out.println();
            sum += e.earnings();
        }
        System.out.println("Total employees: " + Employee.getCount() );
        System.out.println("Total payment: " + sum );
    }
}
```

문제) EmployeeTest.java

- earnings()와 toString() 메소드를 완성하시오.
  - 출력결과는 아래와 같아야 한다.

```
Smith(s1111)
monthly salary: 300.0
payment: 300.0

Karen(h2222)
wage: 1.0
hours: 160.0
payment: 160.0

Jones(c3333)
gross sales: 2000.0
commission rate: 0.1
payment: 200.0
```

```
Lewis(b4444)
gross sales: 2000.0
commission rate: 0.06
base salary: 100.0
payment: 220.0

Total employees: 4
Total payment: 880.0
```