Examples

Week 15

추상클래스 상속 (상속 받은 추상메소드 구현)

```
abstract class Calculator {
    public abstract int add( int a, int b );
    public abstract int subtract( int a, int b );
    public abstract double average( int[] a );
}

public class GoodCalc extends Calculator {
    @Override
    public int add( int a, int b ) {
        return a + b;
    }
    @Override
    public int subtract( int a, int b ) {
        return a - b;
    }
```

상속받은 모든 추상메소드를 구현해야 일반클래스가 됨

추상클래스 상속 (상속 받은 추상메소드 구현)

```
    @Override
    public double average( int[] a ) {
        double sum = 0;
        for ( int i = 0; i < a.length; i++ )
            sum += a[i];
        return sum / a.length;
    }

    public static void main( String[] args ) {
        GoodCalc c = new GoodCalc();
        System.out.println( c.add(2,3) );
        System.out.println( c.subtract(2,3) );
        System.out.println( c.average( new int[] { 2,3,4 } ) );
    }
}
```

```
5
-1
3.0
```

인터페이스 구현 (물려 받은 추상메소드 구현)

```
interface PhoneInterface {
    final int TIMEOUT = 1000;
    void sendCall();
    void receiveCall();
    default void printLogo() {
        System.out.println("** Phone **");
    };
}

class SamsungPhone implements PhoneInterface {
    @Override
    public void sendCall() {
        System.out.println( "Ring-ring-ring" );
    }
```

인터페이스 구현 (물려 받은 추상메소드 구현)

```
    @Override
    public void receiveCall() {
        System.out.println( "You got a call" );
    }
    public void flash() {
        System.out.println( "Light is on" );
    }
}

public class InterfaceEx {
    public static void main(String[] args) {
        SamsungPhone phone = new SamsungPhone();
        phone.printLogo();
        phone.sendCall();
        phone.receiveCall();
        phone.flash();
    }
}
```

```
** Phone **
Ring-ring-ring
You got a call
Light is on
```

클래스상속, 다중 인터페이스 구현

```
interface PhoneInterface {
   final int TIMEOUT = 1000;
   void sendCall();
   void receiveCall();
   default void printLogo() {
      System.out.println("** Phone **");
   };
}

interface MobilePhoneInterface
                 extends PhoneInterface {
   void sendSMS();
   void receiveSMS();
}
```

```
interface MP3Interface {
   public void play();
   public void stop();
}

class PDA {
    public int calculate(int x, int y) {
        return x + y;
    }
}
```

클래스상속, 다중 인터페이스 구현

```
class SmartPhone extends PDA implements MobilePhoneInterface, MP3Interface {
    @Override
    public void sendCall() {
        System.out.println( "Ring-ring-ring" );
    }
    @Override
    public void receiveCall() {
        System.out.println( "You got a call" );
    }
    @Override
    public void sendSMS() {
        System.out.println("Sending a text");
    }
    @Override
    public void receiveSMS() {
        System.out.println("You got a text");
    }
    @Override
    public void play() {
        System.out.println("Playing music");
    }
```

클래스상속, 다중 인터페이스 구현

```
    @Override
    public void stop() {
        System.out.println("Stop playing");
    }
    public void schedule() {
        System.out.println("Make a schedule");
    }
}

public class InterfaceEx {
    public static void main(String[] args) {
        SmartPhone phone = new SmartPhone();
        phone.printLogo();
        phone.sendCall();
        phone.play();
        System.out.println("3 + 5 = " + phone.calculate(3,5));
        phone.schedule();
    }
}
```

```
** Phone **
Ring-ring-ring
Playing music
3 + 5 = 8
Make a schedule
```

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 (Employee 클래스를 추상클래스로)

```
abstract 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 abstract double earnings();

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

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

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

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

   @Override
   public double earnings()
   {
       return monthlySalary;
   }

   @Override
   public String toString()
   {
       return super.toString() + "\n"
       + "monthly salary: " + monthlySalary;
   }
}
```

EmployeeTest.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;
   }

   @Override
   public double earnings()
   {
      return wage * hours;
   }

   @Override
   public String toString()
   {
      return super.toString() + “\n”
                 + “wage: “ + wage + “\n”
                 + “hours: “ + hours;
   }
}
```

```
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;
   }

   @Override
   public double earnings()
   {
      return commissionRate * grossSales;
   }

   @Override
   public String toString()
   {
      return super.toString() + “\n”
     + "gross sales: " + grossSales + "\n"
     + "commission rate: " + commissionRate;
   }
}
```

EmployeeTest.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;
   }

   @Override
   public double earnings()
   {
      return baseSalary + super.earnings();
   }

   @Override
   public String toString()
   {
      return  super.toString() + "\n"
              + "base salary: " + baseSalary;
   }
}
```

EmployeeTest.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 );
    }
}
```

EmployeeTest.java

- 어느 회사의 지출은 직원의 임금과 물품구매 청구서이다.
  - invoice: 품명, 단가, 수량을 기록

- 다형성을 이용하여 직원정보와 청구서정보를 출력하고, 이번 달 총 지출을 계산하여 출력한다.

EmployeeTest.java (Payable 인터페이스 구현)

- 동일한 인터페이스를 구현한 클래스들의 객체를 다형성을 이용하여 한 번에 처리한다.

```
interface Payable
{
    double getPaymentAmount();
}

class Invoice implements Payable
{
    private String description;
    private int quantity;
    private double price;
    private static int count = 0;

    public Invoice(String description,
              int quantity, double price)
    {
        this.description = description;
        this.quantity = quantity;
        this.price = price;
        count++;
    }
```

```
    @Override
    public double getPaymentAmount()
    {
        return quantity * price;
    }

    @Override
    public String toString()
    {
        return description + "\n"
          + "quantity: " + quantity + "\n"
          + "price: " + price;
    }

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

EmployeeTest.java (Payable 인터페이스 구현)

```
abstract class Employee implements Payable
{
   private String name;
   private String id;
   static private int count = 0;

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

   // getPaymentAmount()를 추상메소드로 가짐

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

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

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

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

   @Override
   public double getPaymentAmount()
   {
       return monthlySalary;
   }

   @Override
   public String toString()
   {
       return super.toString() + “\n”
       + “monthly salary: “ + monthlySalary;
   }
}
```

EmployeeTest.java (Payable 인터페이스 구현)

```
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;
   }

   @Override
   public double getPaymentAmount()
   {
      return wage * hours;
   }

   @Override
   public String toString()
   {
      return super.toString() + “\n”
                 + “wage: “ + wage + “\n”
                 + “hours: “ + hours;
   }
}
```

```
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;
   }

   @Override
   public double getPaymentAmount()
   {
      return commissionRate * grossSales;
   }

   @Override
   public String toString()
   {
      return super.toString() + “\n”
     + "gross sales: " + grossSales + "\n"
     + "commission rate: " + commissionRate;
   }
}
```

EmployeeTest.java (Payable 인터페이스 구현)

```
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;
   }

   @Override
   public double getPaymentAmount()
   {
      return baseSalary + super.getPaymentAmount();
   }

   @Override
   public String toString()
   {
      return  super.toString() + "\n"
              + "base salary: " + baseSalary;
   }
}
```

EmployeeTest.java (Payable 인터페이스 구현)

```
public class EmployeeTest
{
    public static void main(String[] args)
    {
        Payable[] arr = new Payable[6];
        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);
        arr[4] = new Invoice("seat", 2, 30000);
        arr[5] = new Invoice("tire", 4, 80000);

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