20171622 박건후 pa#2

과제의 내용

```
지난 과제에서 직원관리를 위해 Array 라는 Java 가 제공하는 툴을 사용했다.
이번에는 ADT Bag 를 활용하고자 한다.
BagInterface 를 사용하고 2 가지 구현, array 를 사용한 경우와 linked-node 를 사용한 경
우를 사용해보라. 또 이렇게 Bag 라는 자료구조를 사용하는 이유를 설명하라.
```

1. 소스 코드 구성

1.1 BagInterface.java

```java
package project;

/**
 * BagInterface 인터페이스
 * @author 20171622 박건후
 * @version BagInterface.java 1.0
 */
public interface BagInterface<T>{

    public int getCurrentSize();

    public boolean add(T newEntry);

    public T remove();

    public boolean remove(T anEntry);

    public boolean isEmpty();

    public void clear();

    public int getFrequencyOf(T anEntry);

    public boolean contains(T anEntry);

    public T[] toArray();

}
```

1.2 ArrayBag.java

```java
package project;
```

```java
/**
 * ArrayBag<T> 클래스
 * @author 20171622 박건후
 * @version ArrayBag.java 1.0
 */
public final class ArrayBag<T> implements BagInterface<T>{


    private final T[] bag;
    private int numberOfEntries;
    private boolean initialized=false;
    private static final int DEFAULT_CAPACITY=25;
    private static final int MAX_CAPACITY=10000;

    public ArrayBag(){
        this(DEFAULT_CAPACITY);
    }

    public ArrayBag(int desiredcapacity){
        if(desiredcapacity<=MAX_CAPACITY){
            @SuppressWarnings("unchecked")
            T[] tempBag = (T[])new Object[desiredcapacity];
            bag = tempBag;
            numberOfEntries=0;
            initialized=true;
        }
        else
            throw new IllegalStateException("Attempt to create a bag whose
capacity exceeds allowed maximum.");
    }



    private boolean isArrayFull(){
        return numberOfEntries>=bag.length;
    }

    private void checkInitialization(){
        if(!initialized){
            throw new SecurityException("ArrayBag object is not initialized
properly");
        }
    }

    public boolean add(T newEntry){
        checkInitialization();
        boolean result=true;
        if(isArrayFull()){
            result=false;
        }
        else{
            bag[numberOfEntries]=newEntry;
            numberOfEntries++;
        }

        return result;
    }
    public T remove(){
        checkInitialization();
        T result=null;
        if(numberOfEntries>0){
            result=bag[numberOfEntries-1];
            bag[numberOfEntries-1]=null;
            numberOfEntries--;
        }
        return result;
    }

    public boolean remove(T anEntry){
        checkInitialization();
        int index=getIndexOf(anEntry);
        T result=removeEntry(index);
        return anEntry.equals(result);

    }

    private T removeEntry(int givenIndex){
        T result=null;

        //givenindex 자리를 bag 마지막 원소로 채우고, 마지막 원소는 null로 설정
        if(!isEmpty() && (givenIndex>=0)){
            result=bag[givenIndex];
            bag[givenIndex] = bag[numberOfEntries-1];
            bag[numberOfEntries-1]=null;
        }
        return result;
    }

    //index가 -1이면 못 찾은 것을 의미함
    private int getIndexOf(T anEntry){
        int where=-1;
        boolean found = false;
        int index=0;
        while(!found && (index<numberOfEntries)){
            if(anEntry.equals(bag[index])){
                found=true;
                where=index;
            }
            index++;
        }
        return where;
    }


    public boolean isEmpty(){
        return numberOfEntries == 0;
    }

    public int getCurrentSize(){
        return numberOfEntries;
    }

    public void clear(){
        while(!isEmpty())
            remove();
    }

    public int getFrequencyOf(T anEntry){
        checkInitialization();
        int counter=0;
        for(int index=0;index<numberOfEntries;index++){
            if(anEntry.equals(bag[index])){
                counter++;
            }
        }
        return counter;
    }

    public boolean contains(T anEntry){
        checkInitialization();
        return getIndexOf(anEntry)>-1;
    }

    public T[] toArray(){
        @SuppressWarnings("unchecked")
        T[] result = (T[])new Object[numberOfEntries];

        for(int index=0;index<numberOfEntries;index++){
            result[index]=bag[index];
        }
        return result;
    }
}
```

1.3 LinkedBag.java

```
package project;

/**
 * LinkedBag<T> 클래스
 * @author 20171622 박건후
 * @version LinkedBag.java 1.0
 */
public final class LinkedBag<T> implements BagInterface<T>{
    private Node firstNode;
    private int numberOfEntries;

    public LinkedBag(){
        firstNode=null;
        numberOfEntries=0;
    }

    public boolean add(T newEntry){
        Node newNode=new Node(newEntry);
        newNode.next=firstNode;

        firstNode=newNode;
        numberOfEntries++;
        return true;
    }

    public T[] toArray(){
        @SuppressWarnings("unchecked")
        T[] result=(T[])new Object[numberOfEntries];
        int index=0;

        Node currentNode=firstNode;
        while((index<numberOfEntries) && (currentNode!=null)){
            result[index]=currentNode.data;
            index++;
            currentNode=currentNode.next;
        }
        return result;
    }

    public int getFrequencyOf(T anEntry){
        int frequency=0;
        int loopCounter=0;
        Node currentNode=firstNode;
        while((loopCounter<numberOfEntries) && (currentNode!=null)){
            if(anEntry.equals(currentNode.data))
                frequency++;
            loopCounter++;
            currentNode=currentNode.next;
        }
        return frequency;
    }

    public boolean contains(T anEntry){
        boolean found = false;
        Node currentNode=firstNode;
        while(!found && (currentNode!=null)){
            if(anEntry.equals(currentNode.data))
                found=true;
            else
                currentNode=currentNode.next;
        }
        return found;
    }

    private Node getReferenceTo(T anEntry){
        boolean found=false;
        Node currentNode=firstNode;
        while(!found && (currentNode!=null)){
            if(anEntry.equals(currentNode.data))
                found=true;
            else
                currentNode=currentNode.next;
        }
        return currentNode;
    }

    public T remove(){
        T result=null;
        if(numberOfEntries>0){
            result=firstNode.getData();
            firstNode=firstNode.getNextNode();
```

```java
            numberOfEntries--;
        }
        return result;
    }

    public boolean remove(T anEntry){
        boolean result=false;
        Node nodeN=getReferenceTo(anEntry);
        if(nodeN!=null){
            nodeN.data=firstNode.data;
            firstNode=firstNode.next;
            numberOfEntries--;
            result=true;
        }
        return result;
    }

    public boolean isEmpty(){
        return numberOfEntries == 0;
    }

    public int getCurrentSize(){
        return numberOfEntries;
    }

    public void clear(){
        while(!isEmpty())
            remove();
    }



    private class Node{
        private T data;
        private Node next;

        private Node(T DataPortion){
            this(DataPortion, null);
        }

        private Node(T DataPortion, Node nextNode){
            data=DataPortion;
            next=nextNode;
        }

        private T getData(){
            return data;
        }

        private void setData(T newData){
            data=newData;
        }

        private Node getNextNode(){
            return next;
        }

        private void setNextNode(Node nextNode){
            next=nextNode;
        }

    }
}
```

2. ArrayBag을 사용한 경우

```
package project;
// Exercise 10.8 Solution: PayrollSystemTest.java
// Employee hierarchy test program.
import java.util.Scanner; // program uses Scanner to obtain user input

import javax.swing.plaf.basic.BasicInternalFrameTitlePane.SystemMenuBar;

import java.util.Calendar;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.text.ParseException;

/**
 * ArrayBag을 사용한 PayrollSystemTest입니다.
 * 데이터를 Employee [] employees 에 임시로 저장한 뒤, ArrayBag에 add를 해주었습니다.
 * @author 20171622 박건후
 * @version PayrollSystemTestArrayBag.java 1.0
 */
public class PayrollSystemTestArrayBag {
    public static String date2String(int value){
        if(value<10){
            return "0"+String.valueOf(value);
        }
        else{
            return String.valueOf(value);
        }
    }


    public static boolean isTenYears(Employee employee, int currentMonth){
        Calendar cal = Calendar.getInstance();

        int currentYear = cal.get ( cal.YEAR );

        int employeeJoinYear=employee.getJoinDate().getYear();
        int employeeJoinMonth=employee.getJoinDate().getMonth();
        int employeeJoinDay=employee.getJoinDate().getDay();
        String
employeeJoinDate=String.valueOf(employeeJoinYear)+date2String(employeeJoinMonth)
+date2String(employeeJoinDay);

        String
currentDate=String.valueOf(currentYear)+date2String(currentMonth)+"01";
```

```
        String strFormat = "yyyyMMdd";
        SimpleDateFormat sdf = new SimpleDateFormat(strFormat);
        try{
            Date startDate = sdf.parse(employeeJoinDate);
            Date endDate = sdf.parse(currentDate);

            long diffDay = (endDate.getTime() - startDate.getTime()) /
(24*60*60*1000);
            System.out.println("근속일수 : "+diffDay+"일");

            if(diffDay>3650){
                return true;
            }


        }catch(ParseException e){
            e.printStackTrace();
        }

        return false;
    }

    public static void paySalary(BagInterface<Employee> employeeBag, int
currentMonth){

        Object[] bagArray=employeeBag.toArray();
        for(int index=0;index<employeeBag.getCurrentSize();index++){
             System.out.println( ((Employee)bagArray[index]) ); // invokes
toString

             // determine whether element is a BasePlusCommissionEmployee
             if ( ((Employee)bagArray[index]) instanceof
BasePlusCommissionEmployee ) {
                    // downcast Employee reference to
                    // BasePlusCommissionEmployee reference
                BasePlusCommissionEmployee employee = (
BasePlusCommissionEmployee ) ((Employee)bagArray[index]);

                double oldBaseSalary = employee.getBaseSalary();
                employee.setBaseSalary( 1.10 * oldBaseSalary );
                System.out.printf("new base salary with 10%% increase is:
$%,.2f\n",employee.getBaseSalary() );
             } // end if

             double tempSalary=((Employee)bagArray[index]).earnings();

             // if month of employee's birthday, add $100 to salary
             if ( currentMonth ==
((Employee)bagArray[index]).getBirthDate().getMonth() ){
                 tempSalary+=100.0;
                 System.out.printf("birthday bonus : +$100.00 \n");
             }


              // if month of employee's work 10 years, multiply 1.1 to salary
              if (isTenYears(((Employee)bagArray[index]),currentMonth)==true){
                  tempSalary*=1.10;
                  System.out.print("(multiply 1.1 to salary / 10years) " );
              }
              //print final salary
              System.out.printf("earned $ %.2f\n\n", tempSalary );

        }

    }

    private static void displayInfoBag(BagInterface<Employee> employeeBag){
        Object[] bagArray=employeeBag.toArray();
        for(int index=0;index<employeeBag.getCurrentSize();index++){
            System.out.printf( "%s\n%s: $%,.2f\n\n", bagArray[index], "earned",
((Employee)bagArray[index]).earnings() );
        }
    }

    private static void displayClassBag(BagInterface<Employee> employeeBag){
        Object[] bagArray=employeeBag.toArray();
        for(int index=0;index<employeeBag.getCurrentSize();index++){
            System.out.printf( "Employee %d is a %s\n", index,
bagArray[index].getClass().getName() );
        }
    }

   public static void main( String[] args )
   {
      // create subclass objects
      SalariedEmployee salariedEmployee =
         new SalariedEmployee(
         "John", "Smith", "111-11-1111", 6, 15, 1944, 800.00 );
      HourlyEmployee hourlyEmployee =
         new HourlyEmployee(
         "Karen", "Price", "222-22-2222", 12, 29, 1960, 16.75, 40 );
      CommissionEmployee commissionEmployee =
         new CommissionEmployee(
         "Sue", "Jones", "333-33-3333", 9, 8, 1954, 10000, .06 );
      BasePlusCommissionEmployee basePlusCommissionEmployee =
         new BasePlusCommissionEmployee(
         "Bob", "Lewis", "444-44-4444", 3, 2, 1965, 5000, .04, 300 );

      SalariedEmployee salariedEmployee1 =
         new SalariedEmployee(
         "박", "건후", "555-55-5555", 2, 9, 1998, 800.00 );
      HourlyEmployee hourlyEmployee1 =
         new HourlyEmployee(
         "맹", "산하", "666-66-6666", 12, 29, 1996, 16.75, 40 );
      CommissionEmployee commissionEmployee1 =
         new CommissionEmployee(
         "민", "대인", "777-77-7777", 9, 8, 1998, 10000, .06 );
      BasePlusCommissionEmployee basePlusCommissionEmployee1 =
         new BasePlusCommissionEmployee(
         "문", "성찬", "888-88-8888", 3, 2, 1998, 5000, .04, 300 );

      SalariedEmployee salariedEmployee2 =
         new SalariedEmployee(
         "김", "학균", "999-99-9999", 6, 15, 1998, 800.00 );
      HourlyEmployee hourlyEmployee2 =
         new HourlyEmployee(
         "노", "성환", "1010-10-1010", 12, 29, 1998, 16.75, 40 );


       //setting join date
       salariedEmployee.setJoinDate(3, 2, 2019);
       hourlyEmployee.setJoinDate(3, 2, 2019);
       commissionEmployee.setJoinDate(3, 2, 2019);
       basePlusCommissionEmployee.setJoinDate(3, 2, 2019);
       salariedEmployee1.setJoinDate(3, 2, 1990);
       hourlyEmployee1.setJoinDate(3, 2, 1993);
       commissionEmployee1.setJoinDate(3, 2, 1999);
       basePlusCommissionEmployee1.setJoinDate(3, 2, 2000);
       salariedEmployee2.setJoinDate(3, 2, 2014);
       hourlyEmployee2.setJoinDate(3, 2, 2012);


      System.out.println( "Employees processed individually:\n" );


      // create four-element Employee array
      Employee[] employees = new Employee[ 10 ];

      // initialize array with Employees
      employees[ 0 ] = salariedEmployee;
      employees[ 1 ] = hourlyEmployee;
      employees[ 2 ] = commissionEmployee;
      employees[ 3 ] = basePlusCommissionEmployee;
      employees[ 4 ] = salariedEmployee1;
      employees[ 5 ] = hourlyEmployee1;
      employees[ 6 ] = commissionEmployee1;
      employees[ 7 ] = basePlusCommissionEmployee1;
      employees[ 8 ] = salariedEmployee2;
      employees[ 9 ] = hourlyEmployee2;


       BagInterface<Employee> employeeBag= new ArrayBag<>(10);
       for(int i=0;i<10;i++){
           employeeBag.add(employees[i]);
       }

       displayInfoBag(employeeBag);



      Scanner input = new Scanner( System.in ); // to get current month
      int currentMonth;

      // get and validate current month
      do
      {
         System.out.print( "Enter the current month (1 - 12): " );
         currentMonth = input.nextInt();
         System.out.println();
      } while ( ( currentMonth < 1 ) || ( currentMonth > 12 ) );

      System.out.println( "Employees processed polymorphically:\n" );

       paySalary(employeeBag, currentMonth);
```

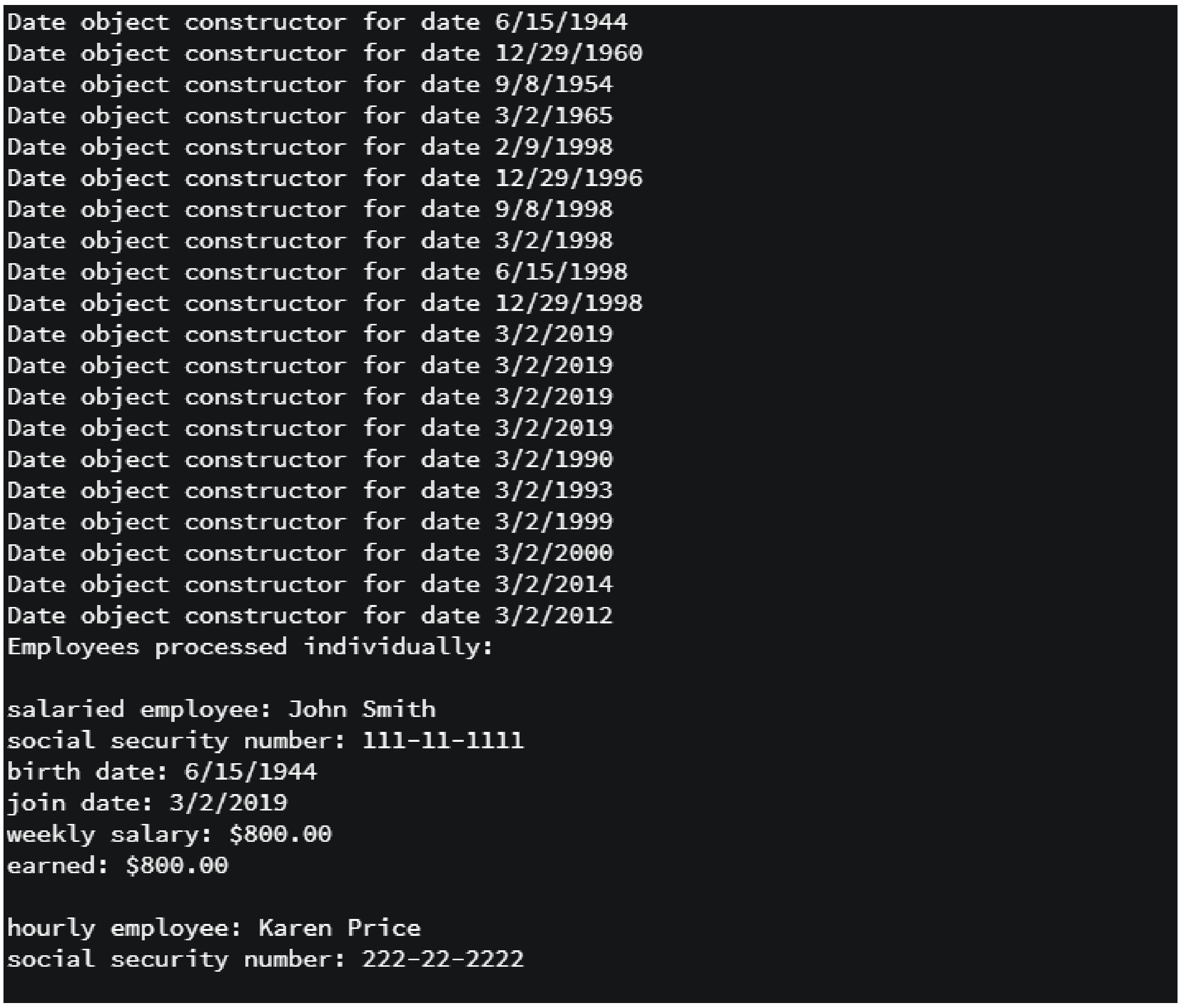
```
        displayClassBag(employeeBag);
      // // get type name of each object in employees array
      // for ( int j = 0; j < employeeBag.getCurrentSize(); j++ )
      //    System.out.printf( "Employee %d is a %s\n", j, employees[ j
].getClass().getName() );
   } // end main
} // end class PayrollSystemTest

/**************************************************************************
 * (C) Copyright 1992-2010 by Deitel & Associates, Inc. and               *
 * Pearson Education, Inc. All Rights Reserved.                           *
 *                                                                        *
 * DISCLAIMER: The authors and publisher of this book have used their     *
 * best efforts in preparing the book. These efforts include the          *
 * development, research, and testing of the theories and programs        *
 * to determine their effectiveness. The authors and publisher make       *
 * no warranty of any kind, expressed or implied, with regard to these    *
 * programs or to the documentation contained in these books. The authors *
 * and publisher shall not be liable in any event for incidental or       *
 * consequential damages in connection with, or arising out of, the       *
 * furnishing, performance, or use of these programs.                     *
 *************************************************************************/
```

3. LinkedBag을 사용한 경우

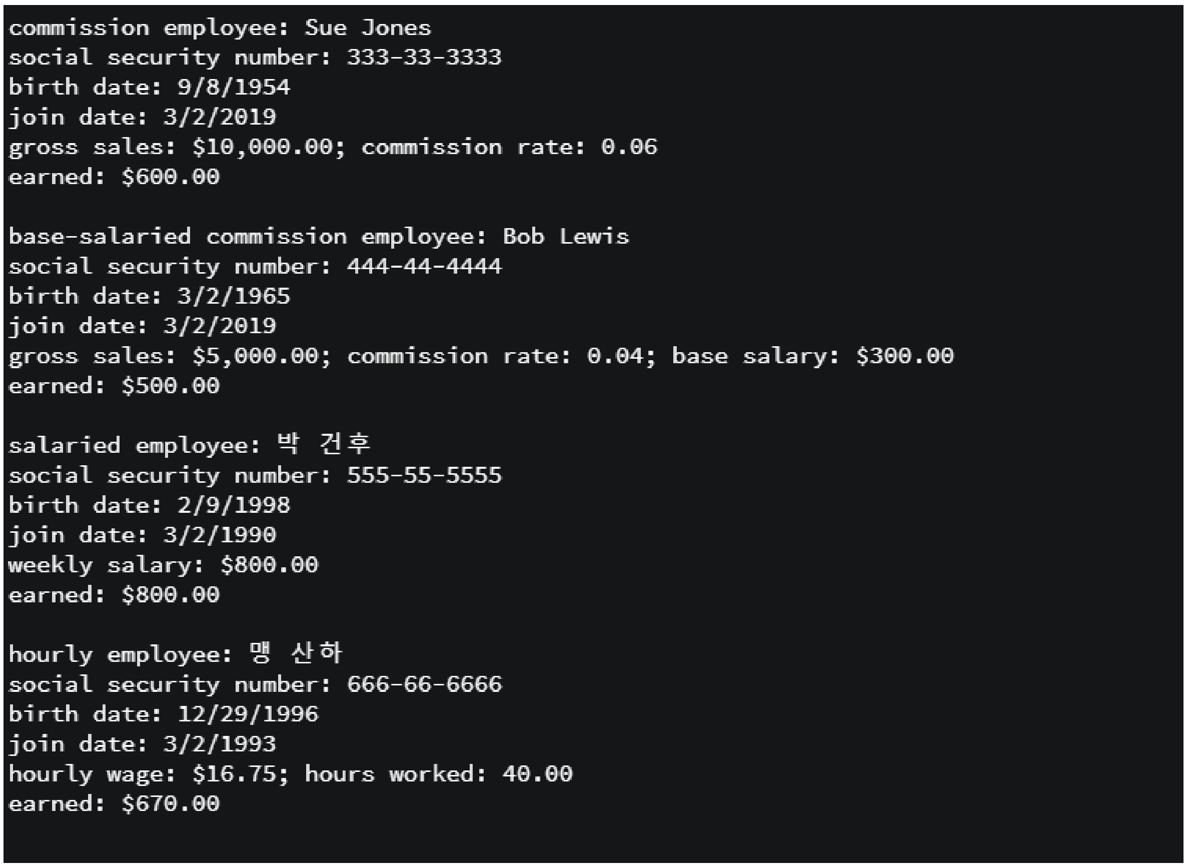
```


package project;
// Exercise 10.8 Solution: PayrollSystemTest.java
// Employee hierarchy test program.
import java.util.Scanner; // program uses Scanner to obtain user input

import javax.swing.plaf.basic.BasicInternalFrameTitlePane.SystemMenuBar;

import java.util.Calendar;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.text.ParseException;

/**
 * LinkedBag을 사용한 PayrollSystemTest입니다.
 * 데이터를 Employee [] employees 에 임시로 저장한 뒤, LinkedBag에 add를 해주었습니다.
 * @author 20171622 박건후
 * @version PayrollSystemTestLinkedBag.java 1.0
 *
 */

public class PayrollSystemTestLinkedBag {
    public static String date2String(int value){
        if(value<10){
            return "0"+String.valueOf(value);
        }
        else{
            return String.valueOf(value);
        }
    }


    public static boolean isTenYears(Employee employee, int currentMonth){
        Calendar cal = Calendar.getInstance();

        int currentYear = cal.get ( cal.YEAR );

        int employeeJoinYear=employee.getJoinDate().getYear();
        int employeeJoinMonth=employee.getJoinDate().getMonth();
        int employeeJoinDay=employee.getJoinDate().getDay();
        String
employeeJoinDate=String.valueOf(employeeJoinYear)+date2String(employeeJoinMonth)
+date2String(employeeJoinDay);

        String
currentDate=String.valueOf(currentYear)+date2String(currentMonth)+"01";
        String strFormat = "yyyyMMdd";
        SimpleDateFormat sdf = new SimpleDateFormat(strFormat);
        try{
            Date startDate = sdf.parse(employeeJoinDate);
            Date endDate = sdf.parse(currentDate);

            long diffDay = (endDate.getTime() - startDate.getTime()) /
(24*60*60*1000);
            System.out.println("근속일수 : "+diffDay+"일");

            if(diffDay>3650){
                return true;
            }


        }catch(ParseException e){
            e.printStackTrace();
        }

        return false;
    }

    public static void paySalary(BagInterface<Employee> employeeBag, int
currentMonth){

        Object[] bagArray=employeeBag.toArray();
        for(int index=0;index<employeeBag.getCurrentSize();index++){
             System.out.println( ((Employee)bagArray[index]) ); // invokes
toString

             // determine whether element is a BasePlusCommissionEmployee
             if ( ((Employee)bagArray[index]) instanceof
BasePlusCommissionEmployee ) {
                    // downcast Employee reference to
                    // BasePlusCommissionEmployee reference
                BasePlusCommissionEmployee employee = (
BasePlusCommissionEmployee ) ((Employee)bagArray[index]);
```

```java
                double oldBaseSalary = employee.getBaseSalary();
                employee.setBaseSalary( 1.10 * oldBaseSalary );
                System.out.printf("new base salary with 10%% increase is:
$%,.2f\n",employee.getBaseSalary() );
             } // end if

             double tempSalary=((Employee)bagArray[index]).earnings();

             // if month of employee's birthday, add $100 to salary
             if ( currentMonth ==
((Employee)bagArray[index]).getBirthDate().getMonth() ){
                 tempSalary+=100.0;
                 System.out.printf("birthday bonus : +$100.00 \n");
             }


              // if month of employee's work 10 years, multiply 1.1 to salary
              if (isTenYears(((Employee)bagArray[index]),currentMonth)==true){
                  tempSalary*=1.10;
                  System.out.print("(multiply 1.1 to salary / 10years) " );
              }
              //print final salary
              System.out.printf("earned $ %.2f\n\n", tempSalary );

        }

    }


    private static void displayInfoBag(BagInterface<Employee> employeeBag){
        Object[] bagArray=employeeBag.toArray();
        for(int index=0;index<employeeBag.getCurrentSize();index++){
            System.out.printf( "%s\n%s: $%,.2f\n\n", bagArray[index], "earned",
((Employee)bagArray[index]).earnings() );
        }
    }

    private static void displayClassBag(BagInterface<Employee> employeeBag){
        Object[] bagArray=employeeBag.toArray();
        for(int index=0;index<employeeBag.getCurrentSize();index++){
            System.out.printf( "Employee %d is a %s\n", index,
bagArray[index].getClass().getName() );
        }
    }

   public static void main( String[] args )
   {
      // create subclass objects
      SalariedEmployee salariedEmployee =
         new SalariedEmployee(
         "John", "Smith", "111-11-1111", 6, 15, 1944, 800.00 );
      HourlyEmployee hourlyEmployee =
         new HourlyEmployee(
         "Karen", "Price", "222-22-2222", 12, 29, 1960, 16.75, 40 );
      CommissionEmployee commissionEmployee =
         new CommissionEmployee(
         "Sue", "Jones", "333-33-3333", 9, 8, 1954, 10000, .06 );
      BasePlusCommissionEmployee basePlusCommissionEmployee =
         new BasePlusCommissionEmployee(
         "Bob", "Lewis", "444-44-4444", 3, 2, 1965, 5000, .04, 300 );

      SalariedEmployee salariedEmployee1 =
         new SalariedEmployee(
         "박", "건후", "555-55-5555", 2, 9, 1998, 800.00 );
      HourlyEmployee hourlyEmployee1 =
         new HourlyEmployee(
         "맹", "산하", "666-66-6666", 12, 29, 1996, 16.75, 40 );
      CommissionEmployee commissionEmployee1 =
         new CommissionEmployee(
         "민", "대인", "777-77-7777", 9, 8, 1998, 10000, .06 );
      BasePlusCommissionEmployee basePlusCommissionEmployee1 =
         new BasePlusCommissionEmployee(
         "문", "성찬", "888-88-8888", 3, 2, 1998, 5000, .04, 300 );

      SalariedEmployee salariedEmployee2 =
         new SalariedEmployee(
         "김", "학균", "999-99-9999", 6, 15, 1998, 800.00 );
      HourlyEmployee hourlyEmployee2 =
         new HourlyEmployee(
         "노", "성환", "1010-10-1010", 12, 29, 1998, 16.75, 40 );


       //setting join date
       salariedEmployee.setJoinDate(3, 2, 2019);
       hourlyEmployee.setJoinDate(3, 2, 2019);
       commissionEmployee.setJoinDate(3, 2, 2019);
       basePlusCommissionEmployee.setJoinDate(3, 2, 2019);
       salariedEmployee1.setJoinDate(3, 2, 1990);
       hourlyEmployee1.setJoinDate(3, 2, 1993);
       commissionEmployee1.setJoinDate(3, 2, 1999);
       basePlusCommissionEmployee1.setJoinDate(3, 2, 2000);
       salariedEmployee2.setJoinDate(3, 2, 2014);
       hourlyEmployee2.setJoinDate(3, 2, 2012);


      System.out.println( "Employees processed individually:\n" );


      // create four-element Employee array
      Employee[] employees = new Employee[ 10 ];

      // initialize array with Employees
      employees[ 0 ] = salariedEmployee;
      employees[ 1 ] = hourlyEmployee;
      employees[ 2 ] = commissionEmployee;
      employees[ 3 ] = basePlusCommissionEmployee;
      employees[ 4 ] = salariedEmployee1;
      employees[ 5 ] = hourlyEmployee1;
      employees[ 6 ] = commissionEmployee1;
      employees[ 7 ] = basePlusCommissionEmployee1;
      employees[ 8 ] = salariedEmployee2;
      employees[ 9 ] = hourlyEmployee2;


       BagInterface<Employee> employeeBag= new LinkedBag<>();
       for(int i=0;i<10;i++){
           employeeBag.add(employees[i]);
       }

       displayInfoBag(employeeBag);



      Scanner input = new Scanner( System.in ); // to get current month
      int currentMonth;

      // get and validate current month
      do
      {
         System.out.print( "Enter the current month (1 - 12): " );
         currentMonth = input.nextInt();
         System.out.println();
      } while ( ( currentMonth < 1 ) || ( currentMonth > 12 ) );

      System.out.println( "Employees processed polymorphically:\n" );

       paySalary(employeeBag, currentMonth);

       displayClassBag(employeeBag);
      // // get type name of each object in employees array
      // for ( int j = 0; j < employeeBag.getCurrentSize(); j++ )
      //    System.out.printf( "Employee %d is a %s\n", j, employees[ j
].getClass().getName() );
   } // end main
} // end class PayrollSystemTest

/**************************************************************************
 * (C) Copyright 1992-2010 by Deitel & Associates, Inc. and               *
 * Pearson Education, Inc. All Rights Reserved.                           *
 *                                                                        *
 * DISCLAIMER: The authors and publisher of this book have used their     *
 * best efforts in preparing the book. These efforts include the          *
 * development, research, and testing of the theories and programs        *
 * to determine their effectiveness. The authors and publisher make       *
 * no warranty of any kind, expressed or implied, with regard to these    *
 * programs or to the documentation contained in these books. The authors *
 * and publisher shall not be liable in any event for incidental or       *
 * consequential damages in connection with, or arising out of, the       *
 * furnishing, performance, or use of these programs.                     *
 *************************************************************************/
```

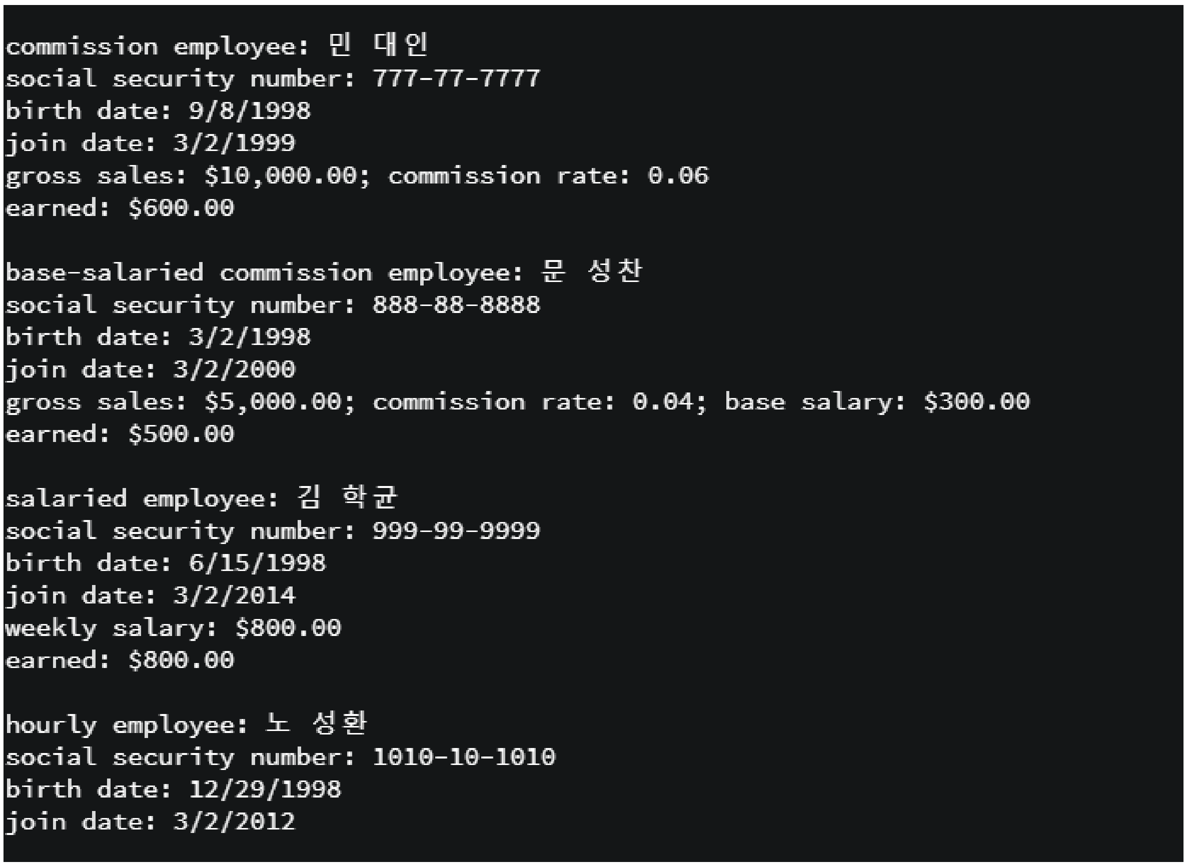
4. 실행 결과 (screen dump)



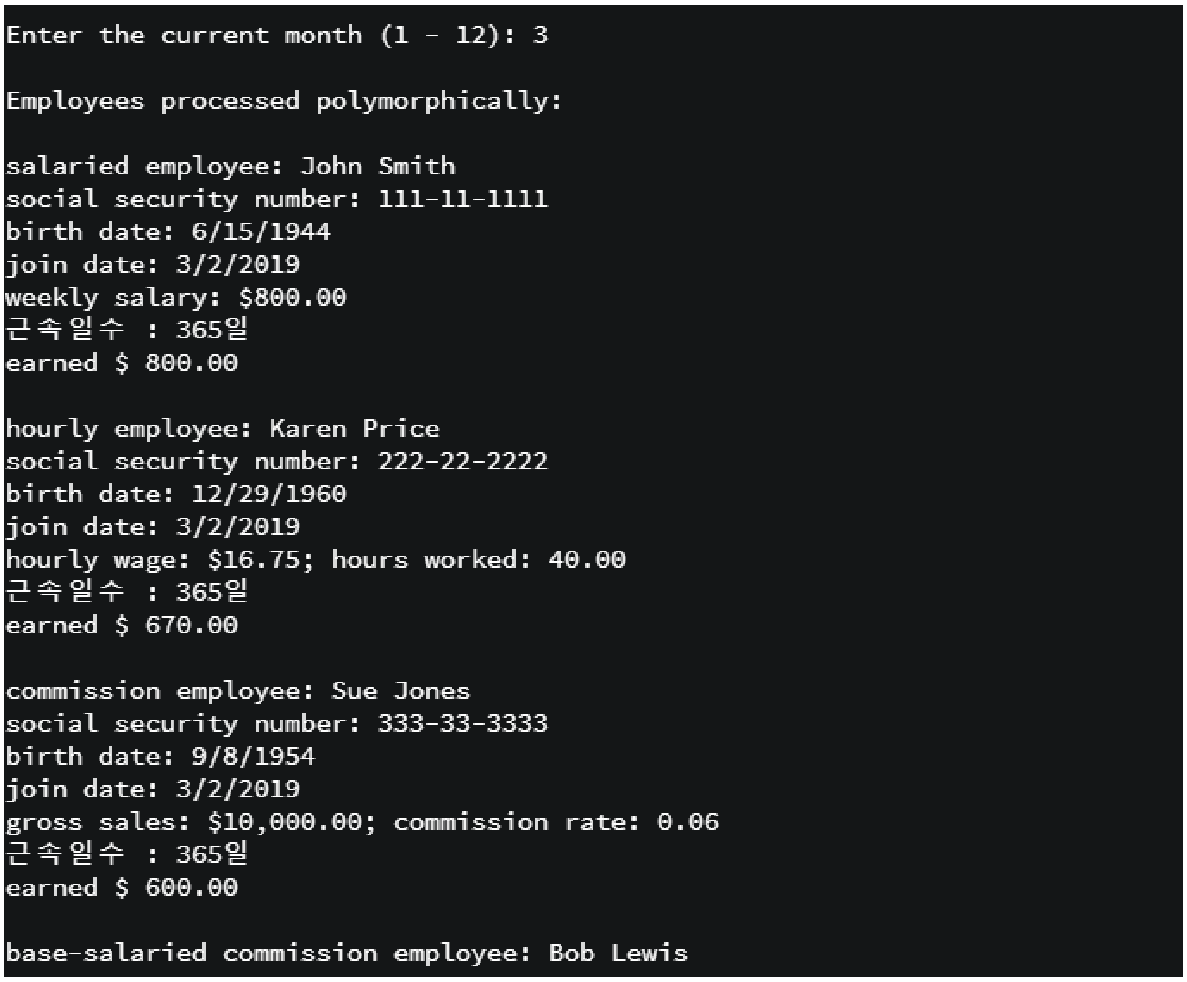
```
Date object constructor for date 6/15/1944
Date object constructor for date 12/29/1960
Date object constructor for date 9/8/1954
Date object constructor for date 3/2/1965
Date object constructor for date 2/9/1998
Date object constructor for date 12/29/1996
Date object constructor for date 9/8/1998
Date object constructor for date 3/2/1998
Date object constructor for date 6/15/1998
Date object constructor for date 12/29/1998
Date object constructor for date 3/2/2019
Date object constructor for date 3/2/2019
Date object constructor for date 3/2/2019
Date object constructor for date 3/2/2019
Date object constructor for date 3/2/1990
Date object constructor for date 3/2/1993
Date object constructor for date 3/2/1999
Date object constructor for date 3/2/2000
Date object constructor for date 3/2/2014
Date object constructor for date 3/2/2012
Employees processed individually:

salaried employee: John Smith
social security number: 111-11-1111
birth date: 6/15/1944
join date: 3/2/2019
weekly salary: $800.00
earned: $800.00

hourly employee: Karen Price
social security number: 222-22-2222
```



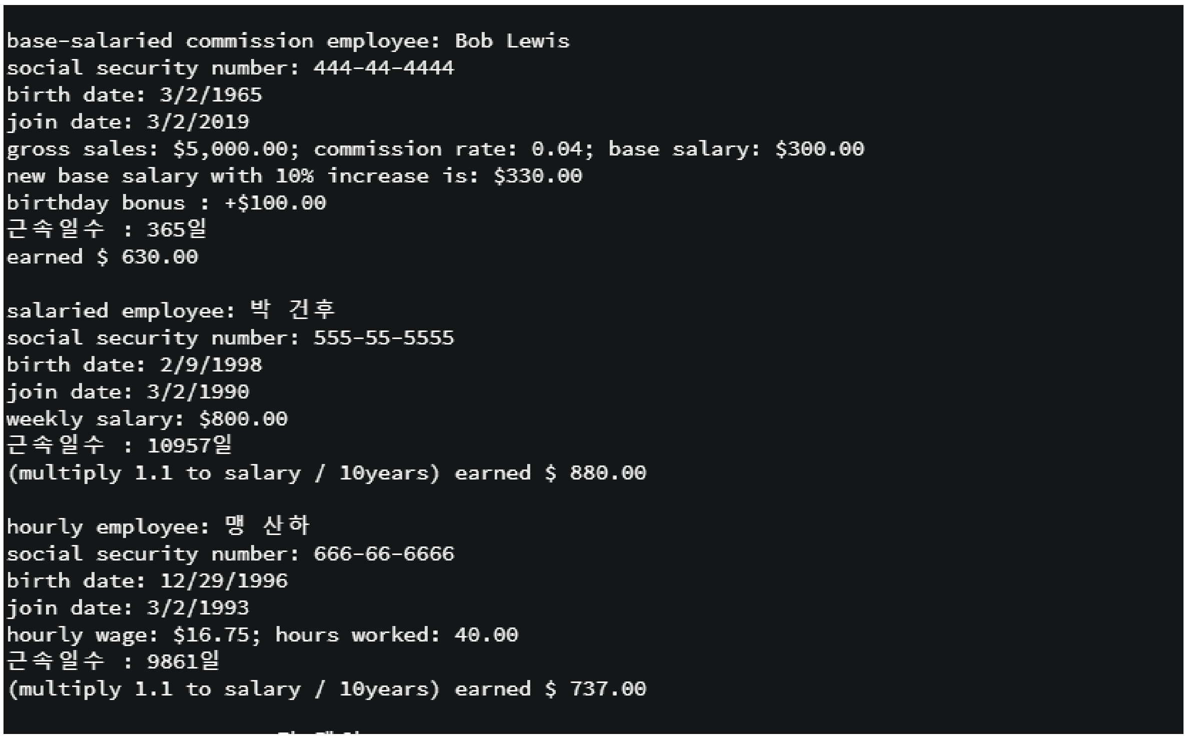
```
commission employee: Sue Jones
social security number: 333-33-3333
birth date: 9/8/1954
join date: 3/2/2019
gross sales: $10,000.00; commission rate: 0.06
earned: $600.00

base-salaried commission employee: Bob Lewis
social security number: 444-44-4444
birth date: 3/2/1965
join date: 3/2/2019
gross sales: $5,000.00; commission rate: 0.04; base salary: $300.00
earned: $500.00

salaried employee: 박 건후
social security number: 555-55-5555
birth date: 2/9/1998
join date: 3/2/1990
weekly salary: $800.00
earned: $800.00

hourly employee: 맹 산하
social security number: 666-66-6666
birth date: 12/29/1996
join date: 3/2/1993
hourly wage: $16.75; hours worked: 40.00
earned: $670.00
```



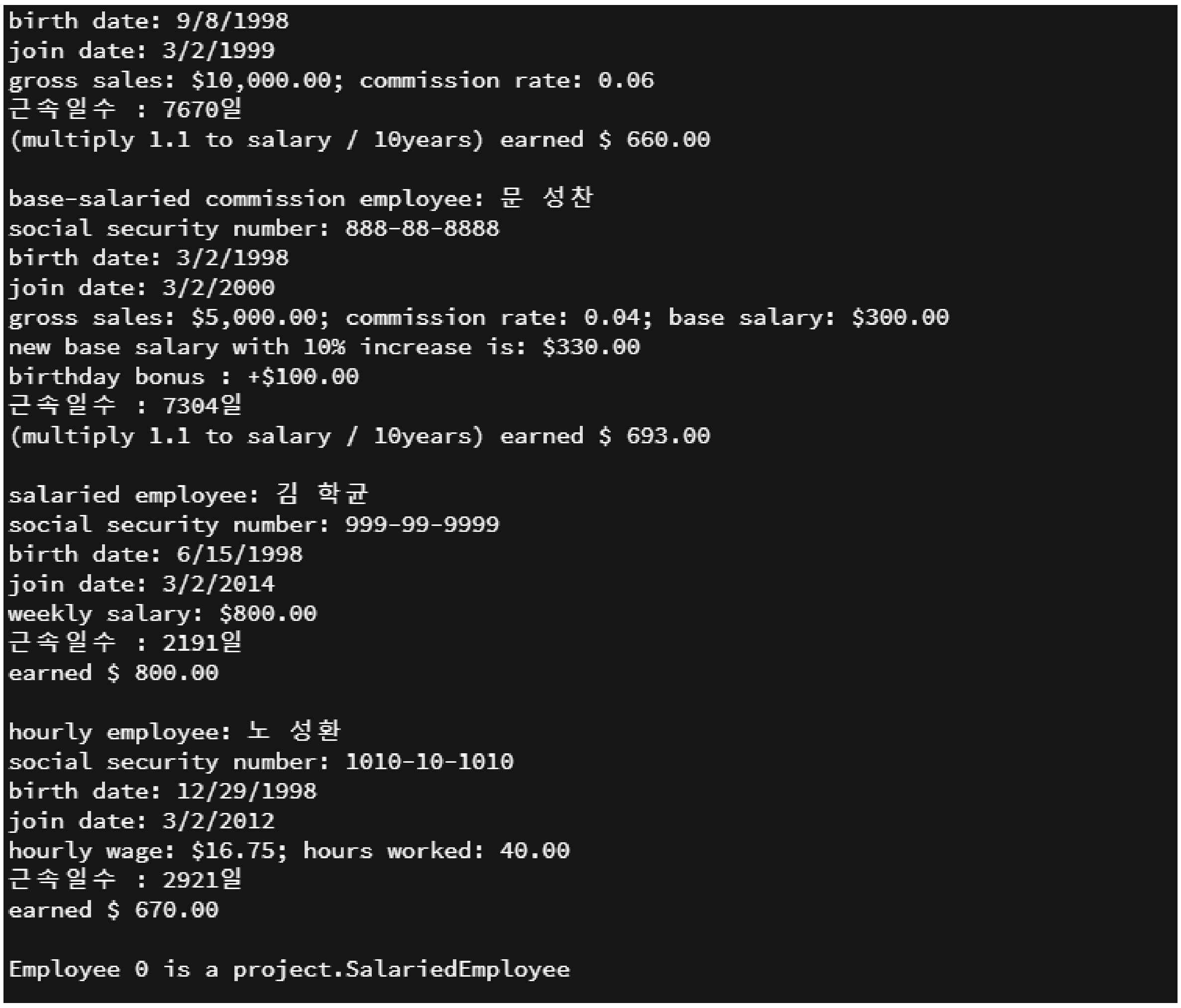
```
commission employee: 민 대인
social security number: 777-77-7777
birth date: 9/8/1998
join date: 3/2/1999
gross sales: $10,000.00; commission rate: 0.06
earned: $600.00

base-salaried commission employee: 문 성찬
social security number: 888-88-8888
birth date: 3/2/1998
join date: 3/2/2000
gross sales: $5,000.00; commission rate: 0.04; base salary: $300.00
earned: $500.00

salaried employee: 김 학균
social security number: 999-99-9999
birth date: 6/15/1998
join date: 3/2/2014
weekly salary: $800.00
earned: $800.00

hourly employee: 노 성환
social security number: 1010-10-1010
birth date: 12/29/1998
join date: 3/2/2012
```



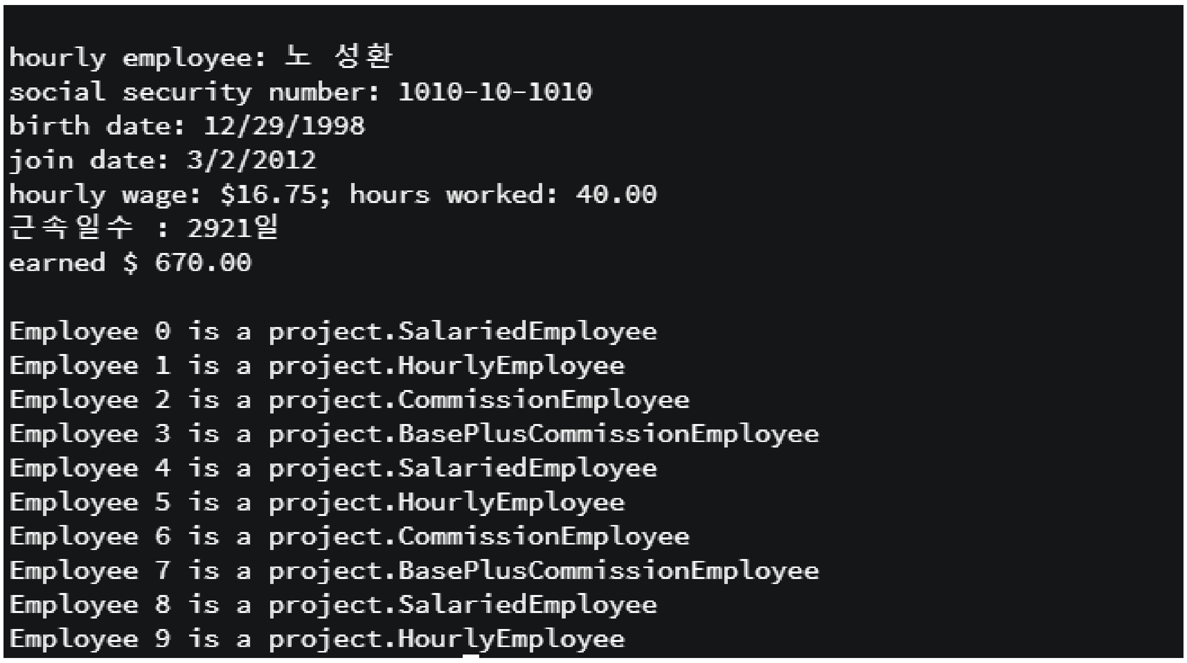
```
Enter the current month (1 - 12): 3

Employees processed polymorphically:

salaried employee: John Smith
social security number: 111-11-1111
birth date: 6/15/1944
join date: 3/2/2019
weekly salary: $800.00
근속일수 : 365일
earned $ 800.00

hourly employee: Karen Price
social security number: 222-22-2222
birth date: 12/29/1960
join date: 3/2/2019
hourly wage: $16.75; hours worked: 40.00
근속일수 : 365일
earned $ 670.00

commission employee: Sue Jones
social security number: 333-33-3333
birth date: 9/8/1954
join date: 3/2/2019
gross sales: $10,000.00; commission rate: 0.06
근속일수 : 365일
earned $ 600.00

base-salaried commission employee: Bob Lewis
```



```
base-salaried commission employee: Bob Lewis
social security number: 444-44-4444
birth date: 3/2/1965
join date: 3/2/2019
gross sales: $5,000.00; commission rate: 0.04; base salary: $300.00
new base salary with 10% increase is: $330.00
birthday bonus : +$100.00
근속일수 : 365일
earned $ 630.00

salaried employee: 박 건후
social security number: 555-55-5555
birth date: 2/9/1998
join date: 3/2/1990
weekly salary: $800.00
근속일수 : 10957일
(multiply 1.1 to salary / 10years) earned $ 880.00

hourly employee: 맹 산하
social security number: 666-66-6666
birth date: 12/29/1996
join date: 3/2/1993
hourly wage: $16.75; hours worked: 40.00
근속일수 : 9861일
(multiply 1.1 to salary / 10years) earned $ 737.00
```



```
birth date: 9/8/1998
join date: 3/2/1999
gross sales: $10,000.00; commission rate: 0.06
근속일수 : 7670일
(multiply 1.1 to salary / 10years) earned $ 660.00

base-salaried commission employee: 문 성찬
social security number: 888-88-8888
birth date: 3/2/1998
join date: 3/2/2000
gross sales: $5,000.00; commission rate: 0.04; base salary: $300.00
new base salary with 10% increase is: $330.00
birthday bonus : +$100.00
근속일수 : 7304일
(multiply 1.1 to salary / 10years) earned $ 693.00

salaried employee: 김 학균
social security number: 999-99-9999
birth date: 6/15/1998
join date: 3/2/2014
weekly salary: $800.00
근속일수 : 2191일
earned $ 800.00

hourly employee: 노 성환
social security number: 1010-10-1010
birth date: 12/29/1998
join date: 3/2/2012
hourly wage: $16.75; hours worked: 40.00
근속일수 : 2921일
earned $ 670.00

Employee 0 is a project.SalariedEmployee
```



```
hourly employee: 노 성환
social security number: 1010-10-1010
birth date: 12/29/1998
join date: 3/2/2012
hourly wage: $16.75; hours worked: 40.00
근속일수 : 2921일
earned $ 670.00

Employee 0 is a project.SalariedEmployee
Employee 1 is a project.HourlyEmployee
Employee 2 is a project.CommissionEmployee
Employee 3 is a project.BasePlusCommissionEmployee
Employee 4 is a project.SalariedEmployee
Employee 5 is a project.HourlyEmployee
Employee 6 is a project.CommissionEmployee
Employee 7 is a project.BasePlusCommissionEmployee
Employee 8 is a project.SalariedEmployee
Employee 9 is a project.HourlyEmployee
```

---

5. 정리

5.1 자료구조 Bag을 사용하는 이유

Bag이란 원소들을 단순히 모아놓은 것을 말합니다. 따라서 원소들 간에 아무 순서도 없고, 중복된 원소가 있을 수 있습니다. list와 array의 경우, 순서가 있고, 중복이 가능합니다. set와 같은 경우, 집합이므로 순서는 없지만 중복이 불가능합니다. 자료구조의 기본 Interface로써 Bag를 꼽을 수 있기 때문에 최상단 추상화를 거친 것이 BagInterface입니다. BagInterface라는 것을 만들어, interface 상속하여 ArrayBag과 LinkedBag을 만들 수 있습니다.

5.2 자료구조 Bag에 사용된 제네릭

추가로 제네릭이란 데이터 타입을 매개변수로 지정하는 것을 의미합니다. BagInterface는 이것을 이용합니다.

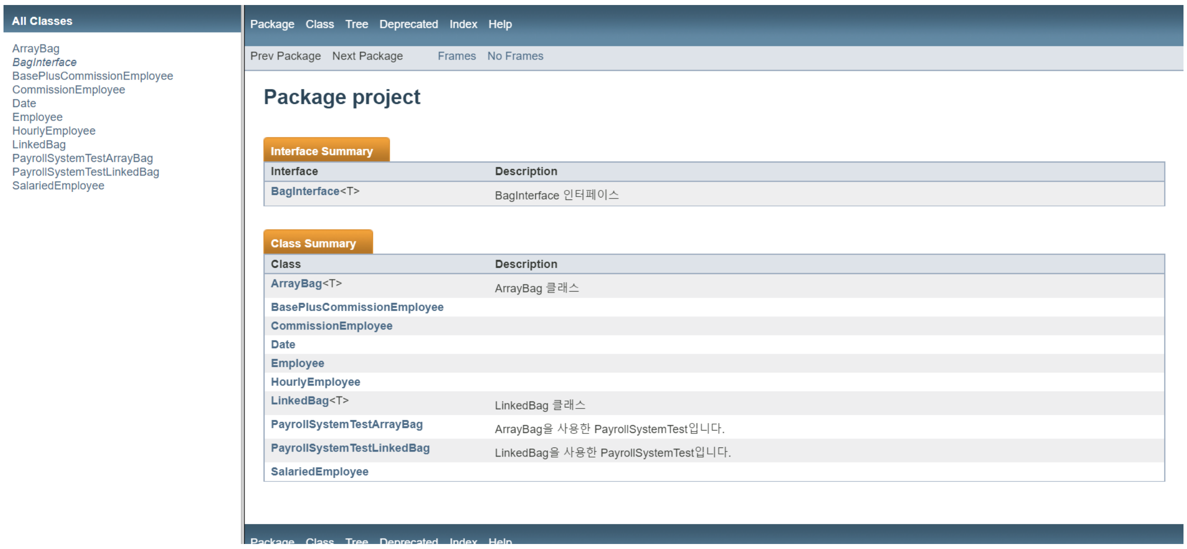
```java
public class Bag<T> {
    T thing;
    public Bag(T thing) {
        this.thing = thing;
    }
}
```

이처럼 제네릭 클래스를 사용하면 인스턴스 생성 시 타입을 지정할 수 있으므로 동적으로 코드를 재사용할 수 있는 장점이 있습니다.

```java
class Book {}
class PencilCase {}
class Notebook {}
public class BagTest {
    public static void main(String[] args) {
        Bag<Book> bag = new Bag<>(new Book());
        Bag<PencilCase> bag2 = new Bag<>(new PencilCase());
        Bag<Notebook> bag3 = new Bag<>(new Notebook());
        bag.showType();
        bag2.showType();
        bag3.showType();
    }
}
```

위와 같이 재사용할 수 있다는 점에서 Bag 인터페이스를 사용해 만든 것입니다.

6. javadoc 실행결과


All Classes

ArrayBag
BagInterface
BasePlusCommissionEmployee
CommissionEmployee
Date
Employee
HourlyEmployee
LinkedBag
PayrollSystemTestArrayBag
PayrollSystemTestLinkedBag
SalariedEmployee

Package Class Tree Deprecated Index Help

Prev Package Next Package Frames No Frames

Package project

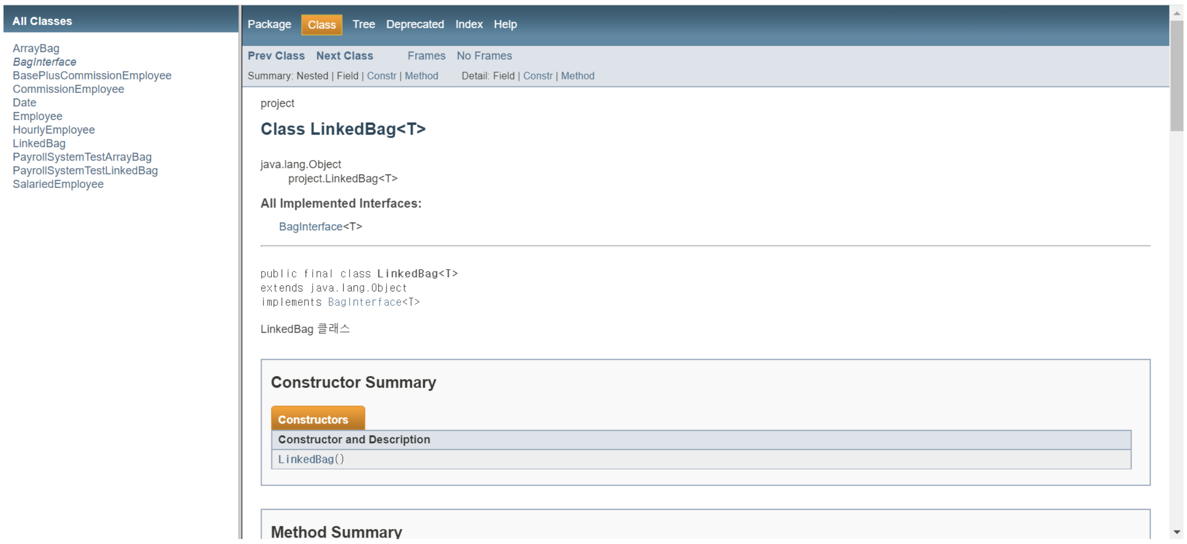
Interface Summary

Interface	Description
BagInterface<T>	BagInterface 인터페이스

Class Summary

Class	Description
ArrayBag<T>	ArrayBag 클래스
BasePlusCommissionEmployee	
CommissionEmployee	
Date	
Employee	
HourlyEmployee	
LinkedBag<T>	LinkedBag 클래스
PayrollSystemTestArrayBag	ArrayBag을 사용한 PayrollSystemTest입니다.
PayrollSystemTestLinkedBag	LinkedBag을 사용한 PayrollSystemTest입니다.
SalariedEmployee	

Package Class Tree Deprecated Index Help


All Classes

ArrayBag
BagInterface
BasePlusCommissionEmployee
CommissionEmployee
Date
Employee
HourlyEmployee
LinkedBag
PayrollSystemTestArrayBag
PayrollSystemTestLinkedBag
SalariedEmployee

Package Class Tree Deprecated Index Help

Prev Class Next Class Frames No Frames
Summary: Nested | Field | Constr | Method Detail: Field | Constr | Method

project

Class LinkedBag<T>

java.lang.Object
    project.LinkedBag<T>

All Implemented Interfaces:

BagInterface<T>

```
public final class LinkedBag<T>
extends java.lang.Object
implements BagInterface<T>
```

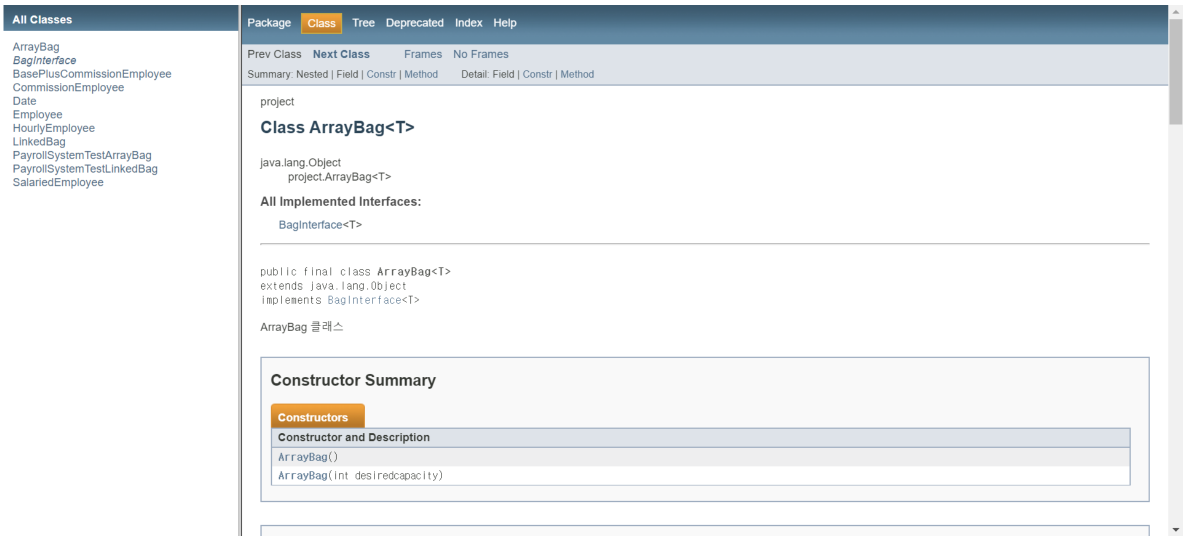
LinkedBag 클래스

Constructor Summary

Constructors

Constructor and Description
LinkedBag()

Method Summary


All Classes
ArrayBag
BagInterface
BasePlusCommissionEmployee
CommissionEmployee
Date
Employee
HourlyEmployee
LinkedBag
PayrollSystemTestArrayBag
PayrollSystemTestLinkedBag
SalariedEmployee
Package Class Tree Deprecated Index Help
Prev Class Next Class Frames No Frames
Summary: Nested | Field | Constr | Method Detail: Field | Constr | Method

project

Class ArrayBag<T>

java.lang.Object
    project.ArrayBag<T>

All Implemented Interfaces:

BagInterface<T>

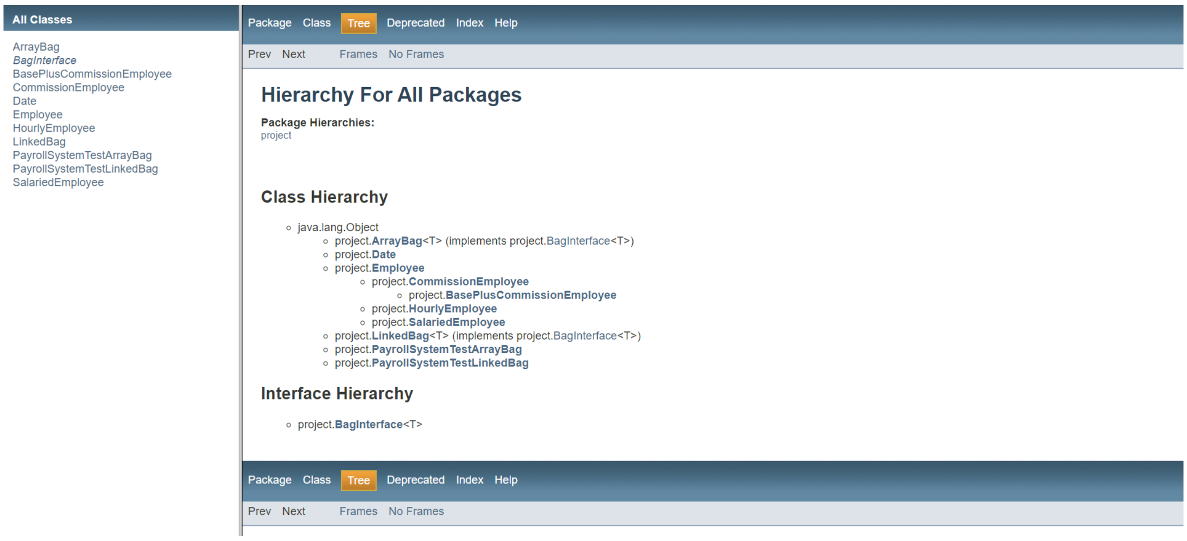
```
public final class ArrayBag<T>
extends java.lang.Object
implements BagInterface<T>
```

ArrayBag 클래스

Constructor Summary

Constructors

Constructor and Description
`ArrayBag()`
`ArrayBag(int desiredcapacity)`


All Classes
ArrayBag
BagInterface
BasePlusCommissionEmployee
CommissionEmployee
Date
Employee
HourlyEmployee
LinkedBag
PayrollSystemTestArrayBag
PayrollSystemTestLinkedBag
SalariedEmployee
Package Class Tree Deprecated Index Help
Prev Next Frames No Frames

Hierarchy For All Packages

Package Hierarchies:
project

Class Hierarchy

- java.lang.Object
  - project.**ArrayBag**<T> (implements project.BagInterface<T>)
  - project.**Date**
  - project.**Employee**
    - project.**CommissionEmployee**
      - project.**BasePlusCommissionEmployee**
    - project.**HourlyEmployee**
    - project.**SalariedEmployee**
  - project.**LinkedBag**<T> (implements project.BagInterface<T>)
  - project.**PayrollSystemTestArrayBag**
  - project.**PayrollSystemTestLinkedBag**

Interface Hierarchy

- project.**BagInterface**<T>

Package Class Tree Deprecated Index Help
Prev Next Frames No Frames