# Object – Oriented Programming

## LAB #11. Generics and the ArrayList

# ArrayList Class

- Java의 standard library class ( java.util.ArrayList )

- 동적 데이터 구조
  - 아이템이 리스트에서 추가되고 삭제될 수 있음
  - 아이템을 추가하거나 제거할 때 길이가 늘어나거나 줄어든다

# ArrayList vs Array

- Java의 일반적인 array는 초기 크기가 고정되어 있기 때문에 정적 데이터 구조이다.

- ArrayList는 프로그램이 실행되는 동안 Array의 길이를 변경할 수 있다는 점을 제외하면, 배열과 동일한 용도로 사용됨.

# ArrayList Class

- ArrayList 클래스는 array를 private instance variable로 사용하여 구현됨

```
public class ArrayList<E> extends AbstractList<E>
        implements List<E>, RandomAccess, Cloneable, java.io.Serializable
{
    private static final long serialVersionUID = 8683452581122892189L;

    /**
     * The array buffer into which the elements of the ArrayList are stored.
     * The capacity of the ArrayList is the length of this array buffer.
     */
    private transient Object[] elementData;

    /**
```

- Array가 꽉 차면 더 큰 배열을 만들고, 새로운 배열에 데이터를 옮긴다.

# Using ArrayList

- ArrayList의 base type은 class type ( 또는 다른 reference type )이어야 한다.
- Primitive type ( int, char, double, ... ) 은 저장될 수 없다.
- Primitive type을 사용하려면 wrapper class로 boxing되어야 한다.

- ArrayList를 설정하려면
  - ArrayList 패키지를 import 하고
    - import java.util.ArrayList;
  - 기본 베이스 타입을 지정한다.
    - ArrayList<BaseType> aList = new ArrayList<BaseType>();
    - 초기 크기를 지정할 수도 있다.

      ex) ArrayList<String> aList = new ArrayList<String>(20);

# ArrayList Notation

- 일반적인 Array의 경우 [ ] 을 사용하여 Array의 특정 인덱스에 있는 항목에 쉽게 접근 가능
  - String str = myArray[i];
  - myArray[i] = “Hello”;
- ArrayList는 [ ] 표기법을 제공하지 않음

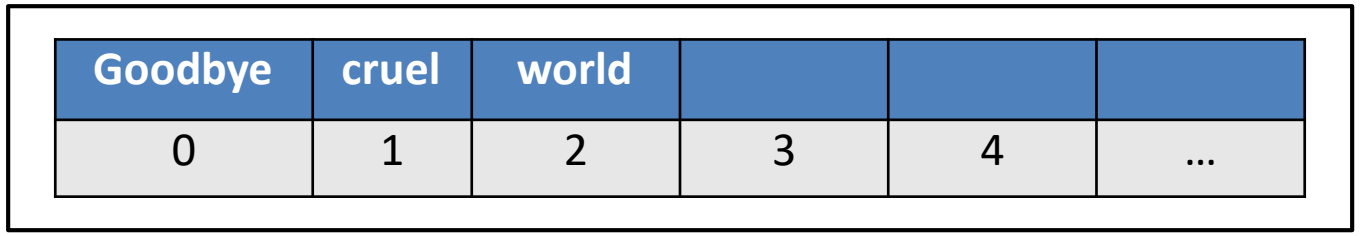
  -> 대신 다음 메소드들을 사용
  - get(index)
  - add(object)
  - set(index, object) -> 초기화는 할 수 없음 ( 기존에 존재하는 항목만 바꿀 수 있음 )

# ArrayList Notation

- ArrayList에 처음 항목을 추가하려면 add()를 사용한다.

- *myArrayList.add(“Goodbye”);*
  *myArrayList.add(“cruel”);*
  *myArrayList.add(“world.”);*

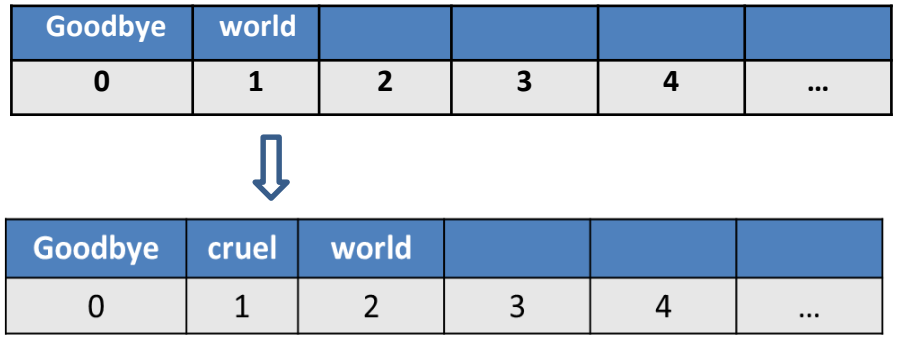
  -> 순차적으로 항목이 추가됨



# ArrayList Notation

- add() 는 Overload되어 다른 매개 변수를 받을 수 있다.
  - ex) add(index, object);

*myArrayList.add("Goodbye");*
*myArrayList.add("world");*
*myArrayList.**add**(1,"cruel.");*



# ArrayList Notation

- size() 메소드는 ArrayList에 저장된 element의 수를 반환한다.

| Goodbye | cruel | world | | | |
|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | … |

- System.out.println( myArrayList.size() );

3

# ArrayList Class

| Method | Description |
|---|---|
| ArrayList<*Base_Type*> (int initialCapacity) | 입력된 정수 크기의 비어있는 Base_Type ArrayList 생성 |
| ArrayList<*Base_Type*>() | 10 크기의 비어있는 Base_Type ArrayList 생성 |
| set(int index, *Base_Type* newElement) | index 위치에 있는 element를 newElement로 변경 |
| get(int index) | index 위치에 있는 element를 반환 |
| add(*Base_Type* newElement) | newElement를 ArrayList에 추가 |
| add(int index, *Base_Type* newElement) | index 위치에 newElement를 추가 |
| remove(int index) | index 위치에 있는 element를 리스트에서 삭제하고, 그 element를 반환 |
| removeRange(int fromIndex, int toIndex) | fromIndex ≤ index < toIndex 에 해당하는 index의 element들을 삭제 |
| clear() | element를 전부 삭제하고 ArrayList의 size를 0으로 만듬 |
| contains(Object target) | ArrayList에 target이 존재하면 true를 반환 |
| indexOf(Object target) | target과 동일한 첫번째 element의 index를 반환 |
| isEmpty() | ArrayList가 비어있다면 true를 반환 |
| size() | ArrayList의 element 개수를 반환 |

# For each loop

- Array와 마찬가지로, for-each loop를 사용하여 collection ( ex) ArrayList )의 모든 요소에 접근할 수 있음

```
for(variable : collection) {
        Statements;
}
```

# For each loop

```java
import java.util.ArrayList;

public class ArrayListTest {
    public static void main(String[] args){
        //create the ArrayList
        ArrayList<Integer> aList = new ArrayList<Integer>();

        //Load objects into the list
        for(int i = 0; i < 20; i++){
            aList.add(new Integer(i));
        }
        System.out.println("Loaded integers into list.");

        //using the the for-each loop data can be retrieved
        for (Integer itger:aList){
            System.out.println(itger.intValue());
        }
    }
}
```

```
Loaded integers into list.
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
```

# Generics

- 일반적인 코드를 작성하고, 이 코드를 다양한 타입의 객체에 대하여 재사용하는 기법

- 클래스에서 사용할 타입을 클래스 외부에서 설정하는 타입

# Generic Class

- Type parameter가 있는 클래스 정의
  - Type parameter는 클래스 선언부의 클래스 이름 뒤 "< >" 안에 작성됨
    - *Public class myClass<T> { }*
    - *Public class myClass<T, U, ...> { }*
- Type parameter의 type은 항상 reference 형식이어야 한다.
  - Primitive type ( int, double, char, ... ) 은 불가능

# Defining Generic Class

**Display 14.4 A Class Definition with a Type Parameter**

```
1  public class Sample<T>
2  {
3      private T data;

4      public void setData(T newData)
5      {
6          data = newData;
7      }

8      public T getData()
9      {
10         return data;
11     }
12 }
```

static 선언 불가능.

T is a parameter for a type.

# Generic Class

```java
class Test<T>{
    private T t;

    public void set(T t) {
        this.t = t;
    }

    public T get() {
        return t;
    }
}
```

```java
public static void main(String[] args) {

    Test<String> test = new Test();
    test.set("test");
    System.out.println(test.get());
}
```

실행 결과 : test

# Generic Methods

- generic 클래스가 정의되면 해당 generic 클래스의 메소드 정의에 type parameter를 사용할 수 있다.

- generic 메소드는 일반 클래스에서도 선언 가능하다.

- generic 메소드의 type parameter는 해당 메소드에만 적용된다. ( not class )

# Generic Methods

- Example
  - 선언 시

    *public static <T> T genMethod(T a)*

    *public <T> int genMethod2(T[ ] b)*

    -> T 배열을 parameter로 받고 T 타입을 반환

  - 호출 시

    *String s = Class.<String>genMethod(c);*

    *int i = object.genMethod2(d);*

# Generic Methods

```
public class Utility{
    //…

    public static <T> T getMidpoint(T[] a){
        return a[a.length/2];
    }

    public static <T> T getFirst(T[] a){
        return a[0];
    }
    //…
}

String midString = Utility.<String>getMidpoint(b);
double firstNumber = Utility.<Double>getFirst(c);
```

# 실습

- Employee, Manager, Engineer, Management, Company 클래스를 생성한다.
- Employee 클래스
  - Manager와 Engineer 클래스는 Employee 클래스를 부모 클래스로 둔다.
  - Employee의 모든 instance 변수들은 private로 선언하고 모두 setter/getter를 생성한다.
  - 아래 4개의 instance variables를 생성한다.
    - String name, int employeeNum, String department, double salary
  - toString 메소드를 아래의 형식으로 String 형을 반환하게끔 overriding 한다.

```
Name : Sally
Employee Number : 1001
Department : Section of Personnel
Salary : 3276.0
```

# 실습

- Manager 클래스
  - 3개의 static 변수를 가진다.

```
public static int initial_manager_number = 1000;
public static String initial_manager_dept = "Sales Management";
public static int initial_manager_salary = 3000;
```

  - 생성자는 name 값만 받고, department와 salary는 각각 initial_manager_dept와 initial_manager_salary를 그대로 사용하며 employeeNum은 initial_manager_number를 1씩 증가시킨다.
    - 가장 처음에 추가되는 manager 번호는 1001부터 시작한다.

- Engineer 클래스
  - 3개의 static 변수를 가진다.

```
public static int initial_engineer_number = 2000;
public static String initial_engineer_dept = "Computational Management";
public static int initial_engineer_salary = 3300;
```

  - 생성자는 name 값만 받고, department와 salary는 각각 initial_engineer_dept와 initial_engineer_salar를 그대로 사용하며 employeeNum은 initial_engineer_number를 1씩 증가시킨다.
    - 가장 처음에 추가되는 engineer 번호는 2001부터 시작한다.

# 실습

- Management 클래스
  - 4개의 static 선언된 generic method를 생성한다.

```
public class Management {
    public static <T extends Employee> T moveDepartment(T t, String department) {
        // 특정 employee의 department를 매개변수로 들어온 String department 값으로 바꾸는 메소드
    }

    public static <T extends Employee> T raiseSalary(T t, double rate) {
        // 특정 employee의 salary를 매개변수로 들어온 rate만큼 곱해주는 메소드
        // rate는 1.00 이상의 값만 들어온다고 가정한다.
    }

    public static <T extends Employee> int findIndexByEmpNum(ArrayList<T> tList, int employeeNum) {
        // 어떤 직군에서 특정 employee를 employeeNum으로 찾는 메소드
        // 매개변수로 들어온 employeeNum으로 특정 employee를 찾지 못하면 -1를 반환한다(별도의 예외처리 필요 X)
    }

    public static <T extends Employee> ArrayList<T> raiseAllSalary(ArrayList<T> tList, double rate) {
        // 한 직군의 모든 employee들의 salary를 rate만큼 곱하여 증가시킨다.
        // rate는 1.00 이상의 값만 들어온다고 가정한다.
    }
}
```

# 실습

- Company 클래스
  - main method를 가진 클래스이다.
  - 한번에 다수를 고용시키는 hireEmployees generic 메소드를 만든다

```
public static <T extends Employee> ArrayList<T> hireEmployees(ArrayList<T> tList, String[] names, String dept) {
    // 1. dept 매개변수를 통해 Manager인지 Engineer인지 판한한다.
    // 2. 이름들이 들어있는 String 배열들을 가지고 ArrayList<Base_Type>에 차례로 add한다.
    // 3. add시 type이 맞지않을 때는 type casting을 이용한다.
    // ex) tList.add((T) new Class(name));
    
    return tList;
}
```

# 실습

- Company 클래스
  - main method

```
public class Company {

    public static void main(String[] args) {
        String[] new_manager_names = {"Sally", "Tammy", "John", "Jessi", "Ariana"};
        String[] new_engineer_names = {"Jenny", "Mason", "Kevin", "Jolly", "Jack", "Ash"};
        ArrayList<Manager> mList = new ArrayList<>();
        ArrayList<Engineer> eList = new ArrayList<>();

        Scanner scan = new Scanner(System.in);

        hireEmployees(eList, new_engineer_names, "engineer");
        hireEmployees(mList, new_manager_names, "manager");

        System.out.println("All employees' salaries were raised by 4%");
        Management.raiseAllSalary(mList, 1.04);
        Management.raiseAllSalary(eList, 1.04);

        System.out.print("enter the number of ENGINEER who will additionally raise his salary : ");
        int selectedEmployee = scan.nextInt();

        int index = Management.findIndexByEmpNum(eList, selectedEmployee);
        if(index != -1) {
            Management.raiseSalary(eList.get(index), 1.20);
            System.out.println(eList.get(index).getName() + "'s salary is raised.\n");
        }
        else
            System.out.println("No one gets chance of raising salary.\n");
```

# 실습

- Company 클래스
  - main method

```
        System.out.print("enter the number of MANAGER who will move his department to Section of Personnel : ");
        selectedEmployee = scan.nextInt();

        index = Management.findIndexByEmpNum(mList, selectedEmployee);
        if(index != -1) {
            Management.moveDepartment(mList.get(index),  "Section of Personnel").toString();
            Management.raiseSalary(mList.get(index), 1.05);
            String selectedName = mList.get(index).getName();
            System.out.println(selectedName + " moves department.\nAnd " + selectedName + "'s salary is raised by 5%.\n");
        }
        else
            System.out.println("No one moves to Section of Personnel");

        for(Manager m : mList)
            System.out.println(m.toString() + "\n");

        for(Engineer e : eList)
            System.out.println(e.toString() + "\n");

    }
```

# 실습

- 실행화면

```
All employees' salaries were raised by 4%
enter the number of ENGINEER who will additionally raise his salary : 2004
Jolly's salary is raised.

enter the number of MANAGER who will move his department to Section of Personnel : 1001
Sally moves department.
AndSally's salary is raised by 5%.

Name : Sally
Employee Number : 1001
Department : Section of Personnel
Salary : 3276.0

Name : Tammy
Employee Number : 1002
Department : Sales Management
Salary : 3120.0

Name : John
Employee Number : 1003
Department : Sales Management
Salary : 3120.0

Name : Jessi
Employee Number : 1004
Department : Sales Management
Salary : 3120.0

Name : Ariana
Employee Number : 1005
Department : Sales Management
Salary : 3120.0

Name : Jenny
Employee Number : 2001
Department : Computational Management
Salary : 3432.0

Name : Mason
Employee Number : 2002
Department : Computational Management
Salary : 3432.0

Name : Kevin
Employee Number : 2003
Department : Computational Management
Salary : 3432.0

Name : Jolly
Employee Number : 2004
Department : Computational Management
Salary : 4118.4

Name : Jack
Employee Number : 2005
Department : Computational Management
Salary : 3432.0

Name : Ash
Employee Number : 2006
Department : Computational Management
Salary : 3432.0
```

10주차 실습 과제 관련 질문: minah741@naver.com