

# 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

Goodbye	cruel	world
0	1	2

# 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");
```

Goodbye	world
0	1



Goodbye	cruel	world
0	1	2



# ArrayList Notation

---

- size() 메소드는 ArrayList에 저장된 element의 수를 반환한다.

Goodbye	cruel	world
0	1	2

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



3
---

# ArrayList Class

Method	Description
<code>ArrayList&lt;Base_Type&gt; (int initialCapacity)</code>	입력된 정수 크기의 비어있는 Base_Type ArrayList 생성
<code>ArrayList&lt;Base_Type&gt;()</code>	10 크기의 비어있는 Base_Type ArrayList 생성
<code>set(int index, Base_Type newElement)</code>	index 위치에 있는 element를 newElement로 변경
<code>get(int index)</code>	index 위치에 있는 element를 반환
<code>add(Base_Type newElement)</code>	newElement를 ArrayList에 추가
<code>add(int index, Base_Type newElement)</code>	index 위치에 newElement를 추가
<code>remove(int index)</code>	index 위치에 있는 element를 리스트에서 삭제하고, 그 element를 반환
<code>removeRange(int fromIndex, int toIndex)</code>	$\text{fromIndex} \leq \text{index} < \text{toIndex}$ 에 해당하는 index의 element들을 삭제
<code>clear()</code>	element를 전부 삭제하고 ArrayList의 size를 0으로 만듦
<code>contains(Object target)</code>	ArrayList에 target이 존재하면 true를 반환
<code>indexOf(Object target)</code>	target과 동일한 첫번째 element의 index를 반환
<code>isEmpty()</code>	ArrayList가 비어있다면 true를 반환
<code>size()</code>	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

---

```
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
5     public void setData(T newData)
6     {
7         data = newData;
8
9     public T getData()
10    {
11        return data;
12    }
```

*T is a parameter for a type.*

# Generic Class

---

```
1  class Test<T>{  
2      private T t;  
3  
4      public void set(T t) {  
5          this.t = t;  
6      }  
7  
8      public T get() {  
9          return t;  
10     }  
11 }
```

```
1  public static void main(String[] args) {  
2  
3      Test<String> test = new Test();  
4      test.set("test");  
5      System.out.println(test.get());  
6  }
```

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*  $\langle T \rangle$  *T* *genMethod*(*T* *a*)

*public*  $\langle T \rangle$  *int* *genMethod2*(*T*[ ] *b*)

-> T 배열을 parameter로 받고 T 타입을 반환

- 호출 시

*String* *s* = *Class*. $\langle$ *String* $\rangle$ *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

---

- Hub, Gyeonggi, Jeju, ServiceManagement, Logistics 클래스를 생성한다.
- Hub 클래스
  - Gyeonggi와 Jeju 클래스는 Hub 클래스를 부모 클래스로 둔다.
  - Hub 의 모든 instance 변수들은 private로 선언하고 모두 setter/getter를 생성한다.
  - 아래 4개의 instance variables를 생성한다.
    - int number(박스 시리얼 번호), String description(박스 객체의 설명), String area(배송 허브 위치), double price\_per\_box(박스 개당 배송 단가)
  - toString 메소드를 아래의 형식으로 String 형을 반환하게끔 overriding 한다.

Box Number : 1005  
Description : GaPyeong#1  
Area : Gyeonggi  
Price per box : 3150.0

# 실습

---

## - Gyeonggi 클래스

```
public static int init_num = 1000;  
public static String init_area = "Gyeonggi";  
public static int init_per_box = 3000;
```

- 3개의 static 변수를 가진다.
- 생성자는 description 값만 받고, area와 price\_per\_box 각각 init\_area 과 init\_per\_box를 그대로 사용하며 number 는 init\_num 을 1씩 증가시킨다.
  - 가장 처음에 추가되는 번호는 1001부터 시작한다.

## - Jeju 클래스

```
public static int init_num = 9000;  
public static String init_area = "Jeju";  
public static int init_per_box = 6000;
```

- 3개의 static 변수를 가진다.
- 생성자는 description 값만 받고, area와 price\_per\_box 각각 init\_area 과 init\_per\_box를 그대로 사용하며 number 는 init\_num 을 1씩 증가시킨다.
  - 가장 처음에 추가되는 번호는 9001부터 시작한다.

# 실습

---

## - ServiceManagement 클래스

- 4개의 static 선언된 generic method를 생성한다.

```
public static <T extends Hub> T moveLocation(T t, String area) {  
    // 택배 도착 area 변경  
    return t;  
}
```

```
public static <T extends Hub> T raisePerBox(T t, double rate) {  
    // 택배 개당 요금 변경, e.g. 1.05 = 5% 상승  
    return t;  
}
```

```
public static <T extends Hub> int findIndexByNum(ArrayList<T> tList, int num) {  
    // 택배 번호로 찾기  
    // 찾으면 해당 인덱스, 반환하지 못하면 -1 반환  
    return -1;  
}
```

```
public static <T extends Hub> ArrayList<T> raiseAll(ArrayList<T> tList, double rate) {  
    // 한 지역의 모든 택배 요금 변경, e.g. 1.05 = 5% 상승  
    return tList;  
}
```

# 실습

---

- Logistics 클래스
  - main method를 가진 클래스이다.
  - 박스를 각 배송지로 배분하는 `deliverBoxes` generic 메소드를 만든다.

```
@SuppressWarnings("unchecked")
/*
 * Comments for Warning
 * 1. area 로 어느 지역인지 판단한 뒤, 해당 지역의 인스턴스만 담는다. 따라서 타입 안전성을 보장한다.
 * 2. ArrayList 에 String[] 배열에 있는 Description 값을 이용해 객체 생성해서 하나씩 add 아래 예시처럼 타입 캐스팅
 *     e.g., ((ArrayList<?>tList).add(new ?(desc));
 *     ref. https://jjjwodls.github.io/java/2020/02/02/02-Java-Generic-Exam.html
 */
public static <T extends Hub> void deliverBoxes(ArrayList<T> tList, String[] description, String area) {

}
```

# 실습

---

```
public static void main(String[] args) {
    String[] new_gyeonggi_boxes = {"SuWon#1", "SeongNam#1", "YongIn#1", "HwaSeong#1", "GaPyeong#1"};
    String[] new_jeju_boxes = {"JeJu-si#1", "SeoGuiPo-si#1"};
    ArrayList<Gyeonggi> gList = new ArrayList<>();
    ArrayList<Jeju> jList = new ArrayList<>();

    Scanner scan = new Scanner(System.in);

    deliverBoxes(gList, new_gyeonggi_boxes, "GyeongGi-dou");
    deliverBoxes(jList, new_jeju_boxes, "JeJu-dou");

    System.out.println("*** Oh, delivery price has been increased!! ***");
    ServiceManagement.raiseAll(gList, 1.05);
    ServiceManagement.raiseAll(jList, 1.05);

    System.out.println("Which box is important in Jeju-area?");
    int selected = scan.nextInt();
    int index = ServiceManagement.findIndexByNum(jList, selected);
    if(index == -1) {
        System.out.println("nothing");
    } else {
        ServiceManagement.raisePerBox(jList.get(index), 1.20); // payment for risk
        System.out.println("The box to "+jList.get(index).getDescription()+" is important! be careful!\n");
    }
}
```



# 실습

---

```
System.out.println("Which box has the wrong area in Gyeonggi-area?");
selected = scan.nextInt();

index = ServiceManagement.findIndexByNum(gList, selected);
if(index == -1) {
    System.out.println("nothing");
} else {
    ServiceManagement.moveArea(gList.get(index), "GangWon"); // 박스의 이름과 무관, 목적 허브(area 변경)
    ServiceManagement.raisePerBox(gList.get(index), 0.9); // discount for late delivery
    System.out.println("The box to "+gList.get(index).getDescription()+" is actually has to go to "
        +gList.get(index).getArea()+"! late! hurry up!\n");
}

for(Gyeonggi g : gList) {
    System.out.println(g+"\n");
}
for(Jeju j : jList) {
    System.out.println(j+"\n");
}
scan.close();
}
```

# 실습

## — 실행화면

다음주 실습 시간 퀴즈 :

{ Generics  
File IO  
Exception  
Interface }

이 중에서 볼 예정

```
*** Oh, delivery price has been increased!! ***
```

```
Which box is important in Jeju-area?
```

```
9001
```

```
The box "JeJu-si#1" is important! be careful!
```

```
Which box has the wrong location in Gyeonggi-area?
```

```
1004
```

```
The box "HwaSeong#1" is actually has to go to GangWon! late! hurry up!
```

```
----- Delivery List -----
```

```
Box Number : 1001
```

```
Description : SuWon#1
```

```
Area : Gyeonggi
```

```
Price per box : 3150.0
```

```
Box Number : 1002
```

```
Description : SeongNam#1
```

```
Area : Gyeonggi
```

```
Price per box : 3150.0
```

```
Box Number : 1003
```

```
Description : YongIn#1
```

```
Area : Gyeonggi
```

```
Price per box : 3150.0
```

```
Box Number : 1004
```

```
Description : HwaSeong#1
```

```
Area : GangWon
```

```
Price per box : 2835.0
```

```
Box Number : 1005
```

```
Description : GaPyeong#1
```

```
Area : Gyeonggi
```

```
Price per box : 3150.0
```

```
Box Number : 9001
```

```
Description : JeJu-si#1
```

```
Area : Jeju
```

```
Price per box : 7560.0
```

```
Box Number : 9002
```

```
Description : SeoGuiPo-si#1
```

```
Area : Jeju
```

```
Price per box : 6300.0
```