Data Structures

Advanced Java

Ja-Hee Kim@seoultech

Contents

Data Structures . Advanced Java

Model

Model Our Lives

How to manage animals using a computer

You should model what you want to manage using a computer.
What do you need to consider to model it?

Abstraction

How to model them in detail?

Encapsulation

What do you want to let externals to know and not know about it?

Inheritance

Commons with and differences from others, Scientific classification

Abstraction

Concrete ⟷ abstract

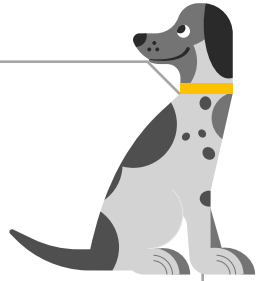
Instance | class | abstract class | interface | UML model | ADT

Developing process

ADT

- Abstract Data Type
- What do all have in common?
- Computer stores/organizes items in similar manners as the exam ples
- An ADT specifies
  - description of a target object
  - data that is stored
  - operations that can be done on the data
    - Parameter: we should know for doing
    - pre-condition: should be satisfied before the operation
    - post-condition: become satisfied after the operation
    - Return: return information

Example of ADT



Dog

Responsibilities

Eat: It can eat given food when it is hungry

Run: It can run fast

Bark: It can make the sound “bowwow”

Collaborations

The owner can make the dog eat, run and bark.

Example of ADT

	Eat	Run	Bark
Pre-condition	hungry	Full	
Post-condition	Full	Hungry	
Parameter	Feed		
Return			sound

Example of ADT

- A Dog is a species of animal.

- **void eat(Object feed)**

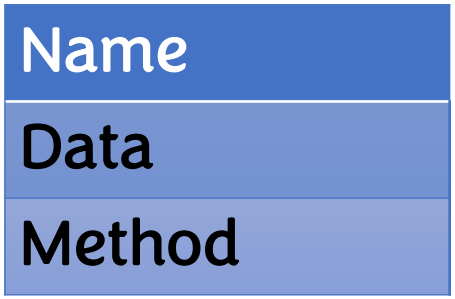
  It can eat given food when it is hungry
  - Precondition: it is hungry
  - Post-condition: it is full
- **void run()**

  It can run fast
  - Precondition: it is full
  - Post-condition: it is hungry
- **Sound bark()**

  It can make the sound “bowwow”
  - Return: return “bowwow”

UML

- Unified Modeling Language
- graphical language used for designing and documenting OOP software
  - UML Class Diagrams
    - The top section contains the class name
    - The middle section contains the data specification for the class
    - The bottom section contains the actions or methods of the class



UML

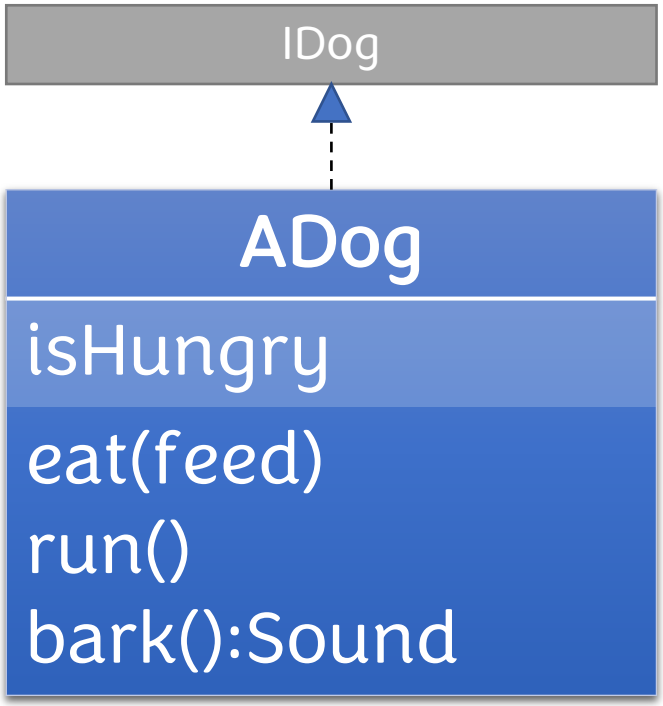
- Provides a class diagram
  - Class name: dog
  - Attributes: breed, isHungry
  - Operations: eat, run, bark

Dog
isHungry
eat(feed) run() bark():Sound

interface

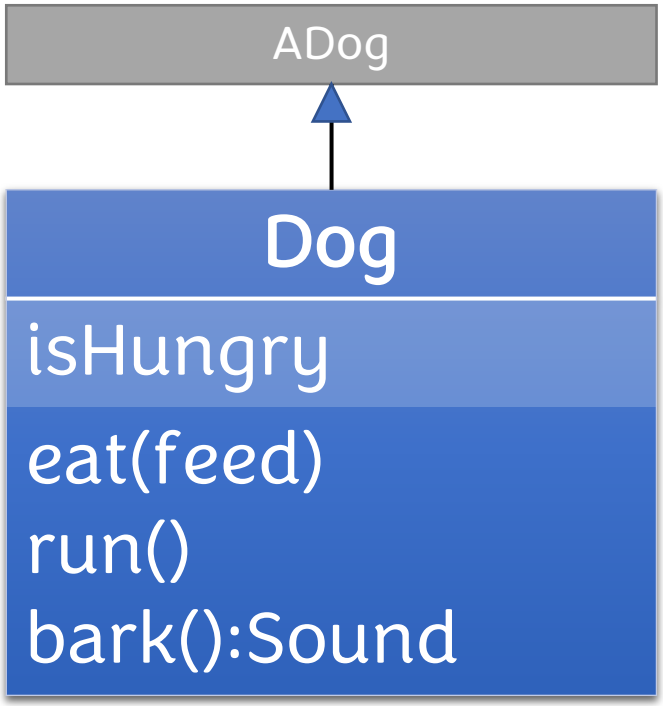

```
interface IDog {

        public void eat(String feed);
        public void run();
        public Sound bark();
}
```

Abstract class




```
abstract class ADog implements IDog{
    boolean isHungry;
    public void eat(String feed) {
        isHungry = false;
    }
    public void run() {
        isHungry = true;
    }
    public abstract Sound bark();
}
```

class




```
class Dog extends ADog {

        public void eat(String feed){/* how to eat */}
        public void run() {/* how to run  */}
        public Sound bark() {
            /* how to bark */
            return sound;
        }
}
```

instance

Dog
isHungry
eat(feed) run() bark():Sound

```
Dog ddangchil    = new Dog();
ddangchil.run();
```

Data Structures . Advanced Java

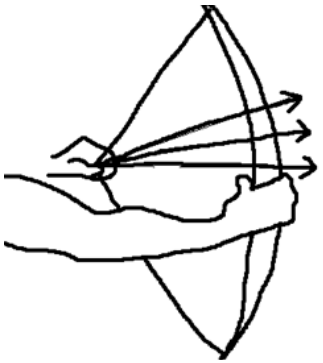
Multiple Classes Interfaces

Agenda

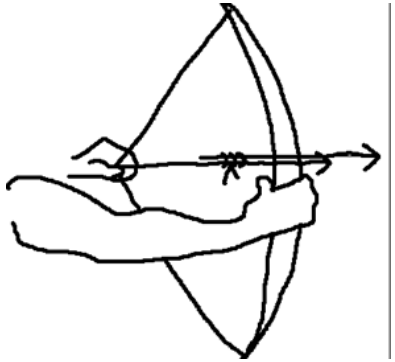
- Overriding vs Overloading
- Polymorphism
- Inner class
- Static
  - Variable
  - Methods

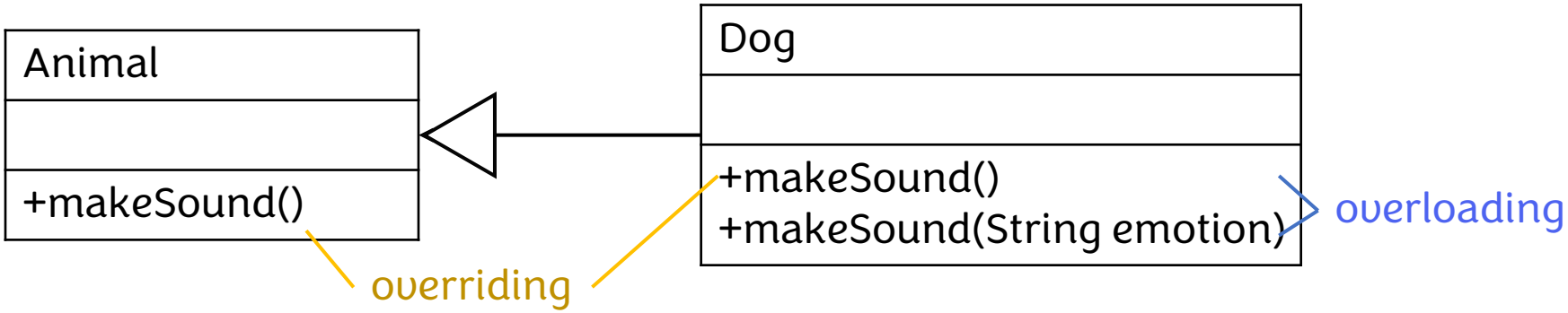
Overriding vs overloading

- Overloading
  - The same name of method in one class with different parameter



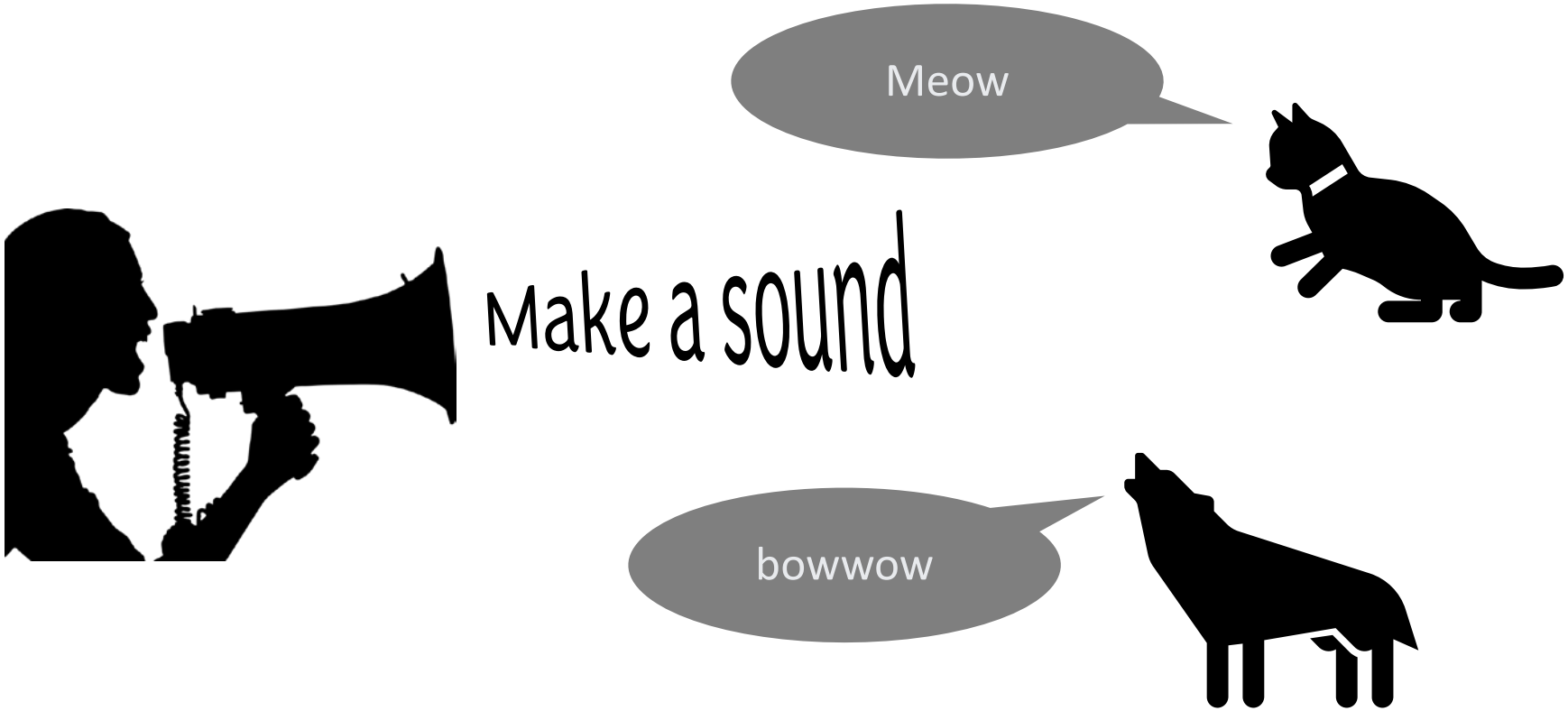
- Overriding
  - The same name of method in a parent class and its child class





polymorphism

- The ability to associate many meanings to one method name
- It does this through a special mechanism known as late binding or dynamic binding



Example

```java
interface Animal{
    public void makeSound();
}
class Dog implements Animal {
    public void makeSound() {System.out.println("Bowwow");}
}
class Cat implements Animal {
    public void makeSound() {System.out.println("Meow");}
}
public class Polymorphism {
    public static void main(String[] args) {
        Animal[] pets = new Animal[3];
        pets[0]=new Cat();
        pets[1]=new Dog();
        pets[2]=new Cat();
        for(int i = 0 ; i < pets.length ; i++) pets[i].makeSound();
    }

}
```

Problems @ Javadoc Declaration Console

```
<terminated> Polymorphism [Java Application] C:₩Users₩user₩.p2₩pool₩plugins₩org.eclipse.ju
Meow
Bowwow
Meow
```

Inner class

- A nest class (a class within a class)
- The purpose of nested classes is to group classes that belong together, which makes your code more readable and maintainable.

```
public class Polymorphism {
    abstract class Animal{
        public abstract void makeSound();
    }
    class Dog extends Animal {
        public void makeSound() {System.out.println("Bowwow");}
    }
    class Cat extends Animal {
        public void makeSound() {System.out.println("Meow");}
    }
    public static void main(String[] args) {
        Polymorphism p = new Polymorphism();
        Animal[] pets = new Animal[3];
        pets[0]=p.new Cat();
        pets[1]=p.new Dog();
        pets[2]=p.new Cat();
        for(int i = 0 ; i < pets.length ; i++) pets[i].makeSound();
    }
}
```

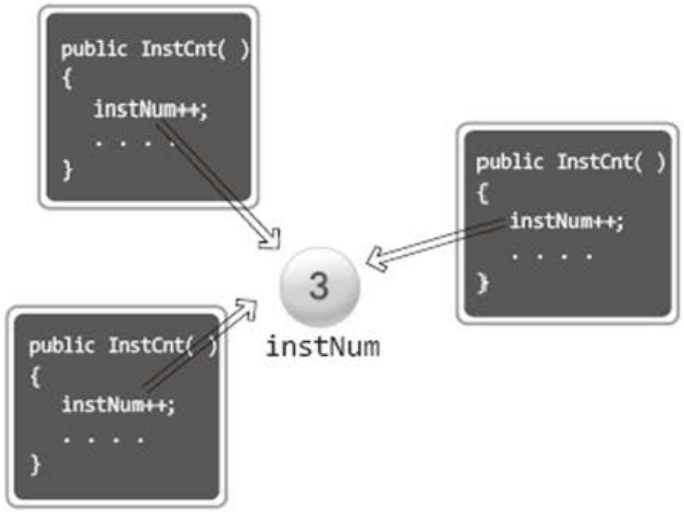
Inner class

Static variable

- belongs to the class as a whole, and not just to one object
- There is only one copy of a static variable per class
- Although a static method cannot access an instance variable, a static method can access a static variable
- A static variable is declared like an instance variable, with the addition of the modifier static
- If not explicitly initialized, a static variable will be automatically initialized to a default value
  - private static int myStaticVariable;
  - private static int myStaticVariable = 0;

Example

```
class InstCnt
{
    static int instNum=0;
    public InstCnt()
    {
        instNum++;
        System.out.println("number of Instance: "+instNum);
    }
}
class ClassVar
{
    public static void main(String[] args)
    {
        InstCnt cnt1=new InstCnt();
        InstCnt cnt2=new InstCnt();
        InstCnt cnt3=new InstCnt();
    }
```



Static methods

- It can be used without a calling object
- A static method still belongs to a class
- When a static method is defined, the keyword static is placed in the method header

```
public static returnedType myMethod(parameters)
                    { . . . }
```

- Static methods are invoked using the class name in place of a calling object

```
returnedValue = MyClass.myMethod(arguments);
```

Data Structures . Advanced Java

Array

Array

- A collection of a same type of objects

- variable

  - code: `int variable = 15;`

- array

  - code: `int[] variables = {1,2,3,4};`

EArray.java

```
1  public class EArray {
2      public static void main(String[] args) {
3          int[] vars = {1, 2, 3, 4};
4      }
5  }
6
```

Problems | Javadoc | Declaration | Console

The value of ...s is not used | Writable | Smart Insert | 3 : 35

EArray.java

```java
1  public class EArray {
2      public static void main(String[] args) {
3          int[] vars = {1, 2, 3, 4};
4          System.out.println("The size of the array is "+ vars.length);
5          System.out.println("0th element of the array is "+vars[0]);
6          System.out.println("1th element of the array is "+vars[1]);
7          System.out.println("2th element of the array is "+vars[2]);
8          System.out.println("3th element of the array is "+vars[3]);
9          System.out.println("4th element of the array is "+vars[4]);
10     }
11 }
```

Problems @ Javadoc Declaration Console

<terminated> EArray [Java Application] C:₩Users₩user₩.p2₩pool₩plugins₩org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.2.v20220201-1

```
The size of the array is 4
0th element of the array is 1
1th element of the array is 2
2th element of the array is 3
3th element of the array is 4
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: Index 4 out of bounds for length 4
        at EArray.main(EArray.java:9)
```

Writable | Smart Insert | 9 : 30

EArray.java

```java
1 public class EArray {
2     public static void main(String[] args) {
3         int[] vars = {1, 2, 3, 4};
4         System.out.println("The size of the array is "+ vars.length);
5         for(int i = 0 ; i < vars.length ; i++)
6             System.out.println(i+"th element of the array is "+vars[i]);
7     }
8 }
```

```java
System.out.println("0th element of the array is "+vars[0]);
System.out.println("1th element of the array is "+vars[1]);
System.out.println("2th element of the array is "+vars[2]);
System.out.println("3th element of the array is "+vars[3]);
```

Problems @ Javadoc

<terminated> EArray [Java Applica ...pool#plugins#org.eclipse.justj.openjdk.hotspot.j

```
The size of the array is 4
0th element of the array is 1
1th element of the array is 2
2th element of the array is 3
3th element of the array is 4
```

Writable | Smart Insert | 6 : 73

EArray.java

```java
public class EArray {
    public static void main(String[] args) {
        int[] vars = {1, 2, 3, 4};
        System.out.println("The size of the array is "+ vars.length);
        for(int i = 0 ; i < vars.length ; i++)
            vars[i] = 2*i+1;
        for(int var:vars)
            System.out.println(var);
    }
}
```

Problems | @ Javadoc | Declaration | Console

<terminated> EArray [Java Application] C:₩Users₩user₩.p2₩pool₩plugins₩org.eclipse.justj.openjdk.hotspot.j

```
The size of the array is 4
1
3
5
7
```

Writable | Smart Insert | 11 : 1

Problem of an array

What happens If you want to store more numbers

- 

Predicting the future
is not easy.

ArrayList

Java - 2017ds/src/eArrays.java - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

Quick Access | Java | Debug

eArrays.java

```java
import java.util.*; // ArrayList를 사용하려면 꼭 java.util package가 필요하다.
public class eArrays {
    public static void main(String[] args){
        /*
         * ArrayList는 class를 만드는 것과 유사하게 만든다.
         * 단 무엇(base type)에 대한 집합체인지를 다음과 같이 표시한다.
         * ArrayList<BaseType> aList = new ArrayList<BaseType>();
         * 아래 예제는 int의 wrapper class인 Integer가 base type이다.
         */
        ArrayList<Integer> va
        for(int i = 0; i < 4
            vars.add(i);
        System.out.println("ArrayList의 크기는: " + vars.size()); //int size() : 몇 개의 요소를 저장했는지 반환

        for(int i = 0; i < 4 ; i++)
            System.out.println(vars.get(i));
    }
}
```

vars.add

vars.size();

vars.get(i);

Problems | Javadoc | Console

<terminated> eArrays [Java Application] C:₩Program Files₩Java₩jdk1.8.0_73₩bin₩javaw.exe (2017. 7. 27. 오전 12:38:33)

```
ArrayList의 크기는: 4
0
1
2
3
```

Writable | Smart Insert | 19 : 1

Java - 2017ds/src/eArrays.java - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

Quick Access | Java | Debug

eArrays.java

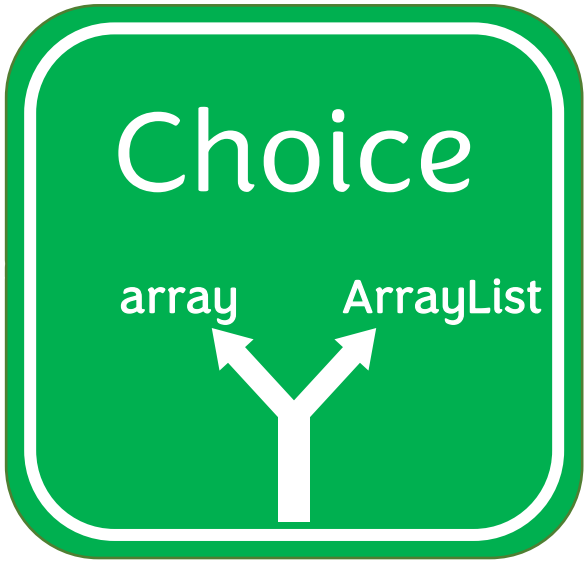
```java
1  import java.util.*; // ArrayList를 사용하려면 꼭 java.util package가 필요하다.
2  public class eArrays {
3      public static void main(String[] args){
4          ArrayList<Integer> vars=new ArrayList<Integer>();
5          for(int i = 0; i < 4 ; i++)
6              vars.add(i); // boolean add(E e) : e를 vars의 끝에 추가
7          System.out.println("ArrayList의 크기는: " + vars.size()); //int size() : 몇 개의 요소를 저장했는지 반환
8          System.out.println(vars);
9          // 2번째 요소 삭제 -      E remove(int i): i번째 요소 삭제
10         vars.remove(2);
11         System.out.println("2번째 요소 삭제 후"+ vars);
12         // 2번째 요소 위치에 삽입 - void add(int i, E e): i번째에 e를 삽입
13         vars.add(2,5);
14         System.out.println("2번째 요소에 5를 삽입 후"+ vars);
15         // 2번째 위치에 내용 바꾸기 -E set(int i, E e): i번째 요소를 e로 변경
16         vars.set(2,7);
17         System.out.println("2번째 요소를 7를 바꾼 후"+ vars);
18     }
19 }
```

Problems | Javadoc | Console

<terminated> eArrays [Java Application] C:₩Program Files₩Java₩jdk1.8.0_73₩bin₩javaw.exe (2017. 7. 27. 오전 12:45:59)

```
ArrayList의 크기는: 4
[0, 1, 2, 3]
2번째 요소 삭제 후[0, 1, 3]
2번째 요소에 5를 삽입 후[0, 1, 5, 3]
2번째 요소를 7를 바꾼 후[0, 1, 7, 3]
```

Writable | Smart Insert | 20 : 1



- size can change
- for only classes

! **use** Wrapping class for primitive type

Data Structures . Advanced Java

Generics

Would you like the generic drug or a name brand?

Generic

- generic

- base type

Generics

- parameterized class

- example
  - ArrayList<BaseType>
  - Glass<Coke> or Glass<Pepsi>

```
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  }
```

parameter

the data type for a member variable

a parameter type of a method

T is a parameter for a type.

a return type of a method

```
public class Sample<T1, T2> {     // 2 parameters: T1 and T2
        private T1 d1; {      // declare d1 whose type is T1
        private T2 d2;        // declare d2 whose type is T2
        public Sample() {
                d1=null;
                d2=null;
        }
        public void setData(T1 d1, T2 d2) {
                this.d1 = d1;
                this.d2 = d2;
        }
        public T1 getData1() {
                return d1;
        }
        public T2 getData2() {
                return d2;
        }
        public String toString() {
                return d1 + ", "+d2;
        }
}
```

```
public class SampleDemo {
        public static void main(String[] args) {
                // determine T1 and T2 as Integer and String, respectively
                Sample<Integer,String> s = new Sample<Integer,String>();
                s.setData(new Integer(3), new String("Hello"));
                System.out.println(s);
        }
}
```

```
1   public class Pair<T extends Comparable>
2   {
3       private T first;
4       private T second;

5       public T max()
6       {
7           if (first.compareTo(second) <= 0)
8               return first;
9           else
10              return second;
11      }
12  }
```

One of T's ancestor or T itself implements Comparable. So, first can use method compareTo

Generics = parameterized class

A box that can hold anything

Thank you!

Questions?

Exit