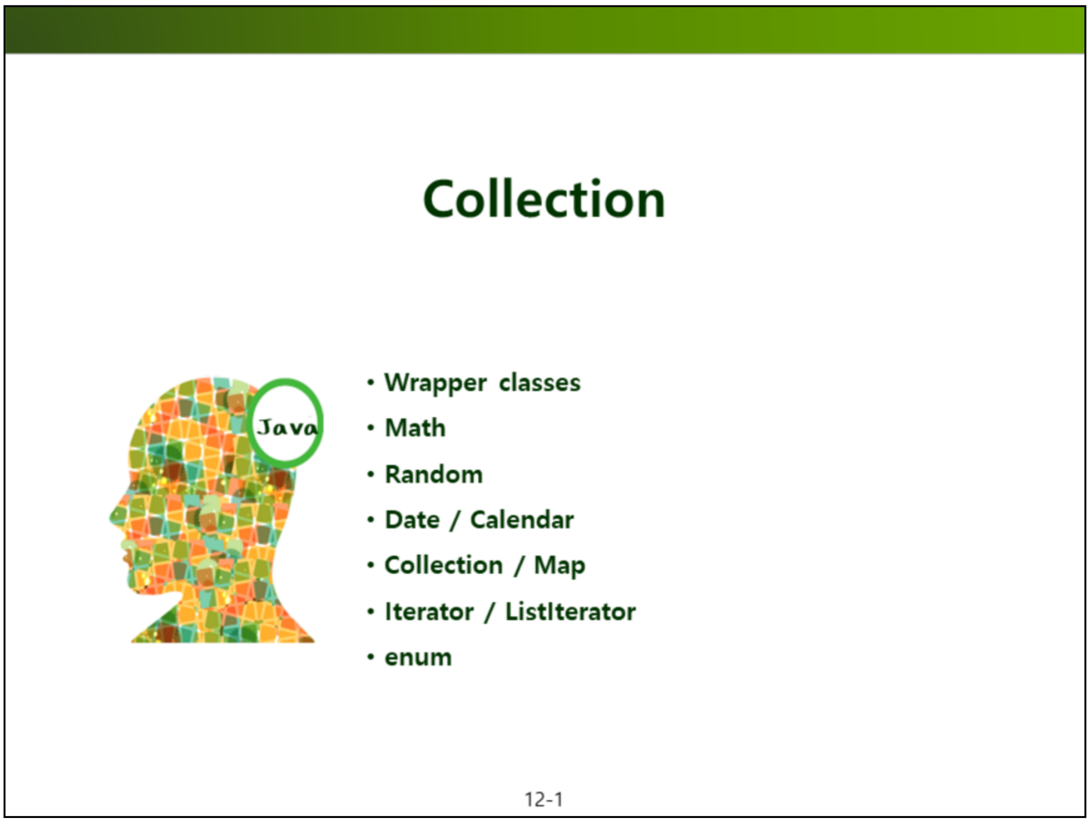
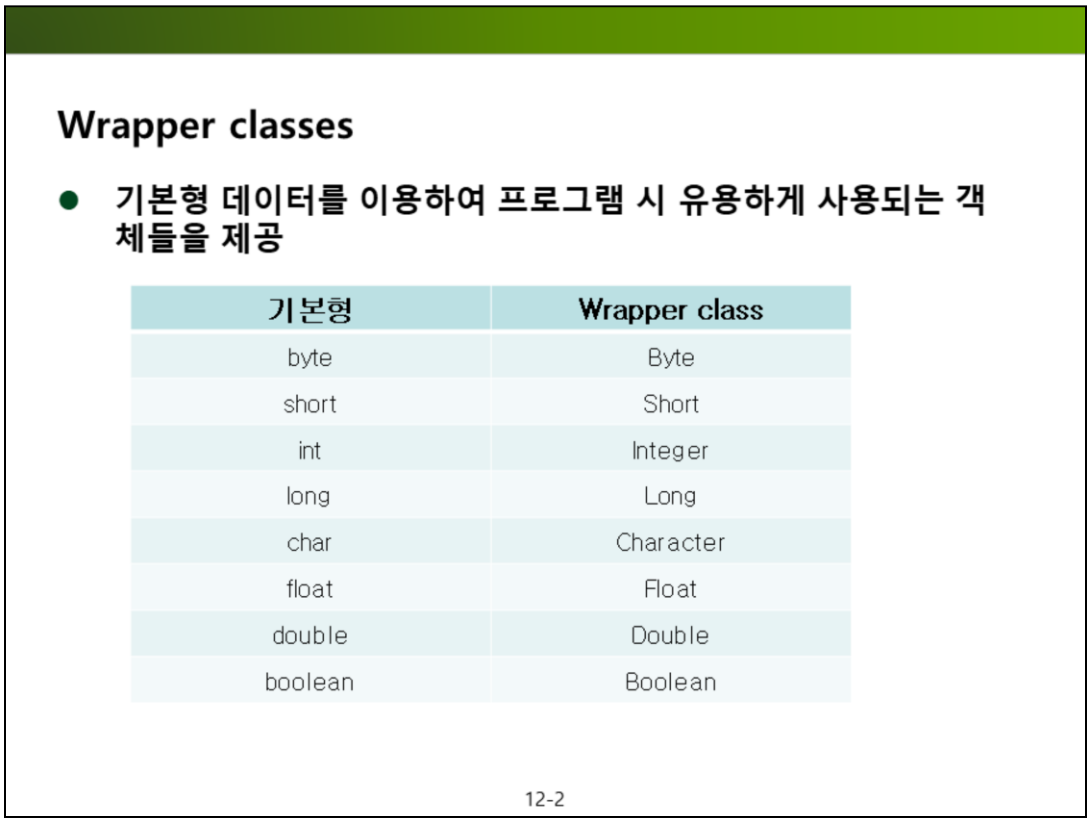


Collection

- Wrapper classes
- Math
- Random
- Date / Calendar
- Collection / Map
- Iterator / ListIterator
- enum

12-1



Wrapper classes

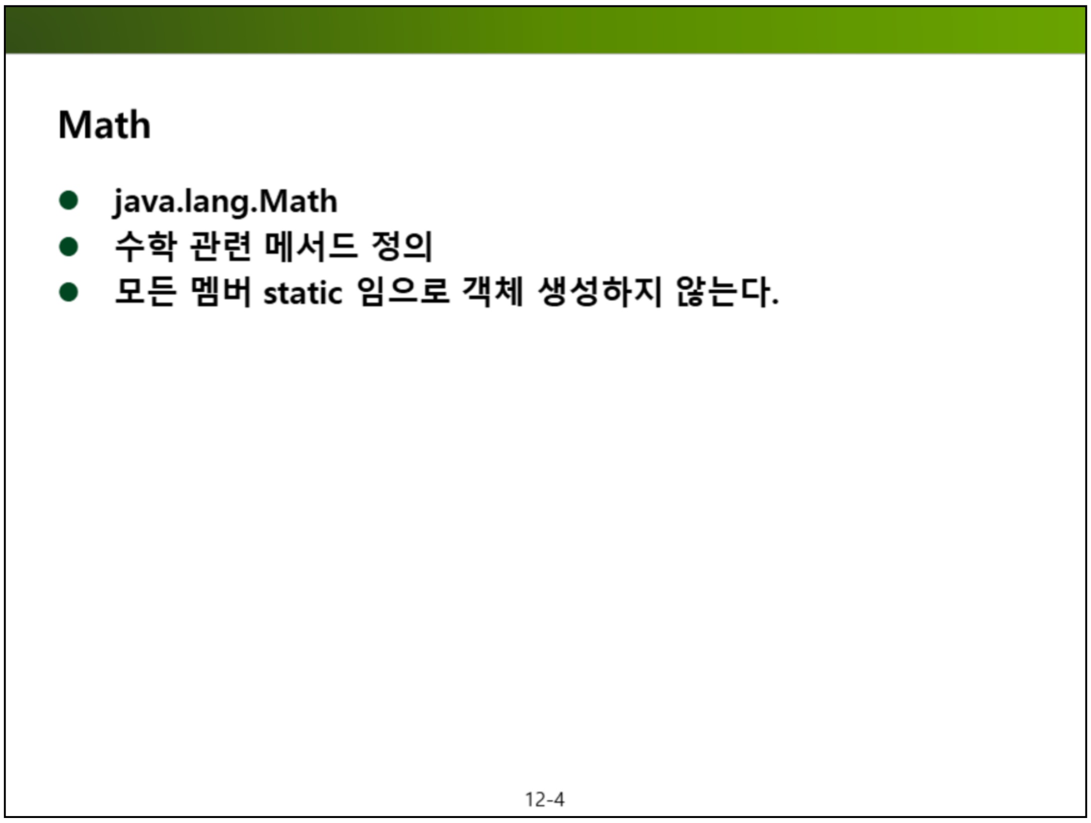
- 기본형 데이터를 이용하여 프로그램 시 유용하게 사용되는 객체들을 제공

기본형	Wrapper class
byte	Byte
short	Short
int	Integer
long	Long
char	Character
float	Float
double	Double
boolean	Boolean

12-2

• Wrapper class

```
public class WrapperTest {
    public static void main(String[] args) {
        Integer i1=new Integer("1234");
        Integer i2=new Integer(1234);
        Boolean b1=new Boolean("True");
        Boolean b2=new Boolean("TRUE");

        System.out.println(i1.equals(i2));
        System.out.println(b1.equals(b2));

        System.out.println(i1);
        System.out.println(Integer.toBinaryString(i1));
        System.out.println(Integer.toOctalString(i1));
        System.out.println(Integer.toHexString(i1));

        System.out.println(Character.isUpperCase('g'));
        System.out.println(Character.isUpperCase('G'));

        System.out.println(Byte.SIZE);
        System.out.println(Byte.MAX_VALUE);
        System.out.println(Byte.MIN_VALUE);

        System.out.println(Float.SIZE);
        System.out.println(Float.MAX_VALUE);
        System.out.println(Float.MIN_VALUE);

        System.out.println(Double.SIZE);
        System.out.println(Double.MAX_VALUE);
        System.out.println(Double.MIN_VALUE);

        System.out.println(Integer.parseInt("123"));
        System.out.println(Float.parseFloat("21.6"));
        System.out.println(Double.parseDouble("234.6"));

    }
}
```

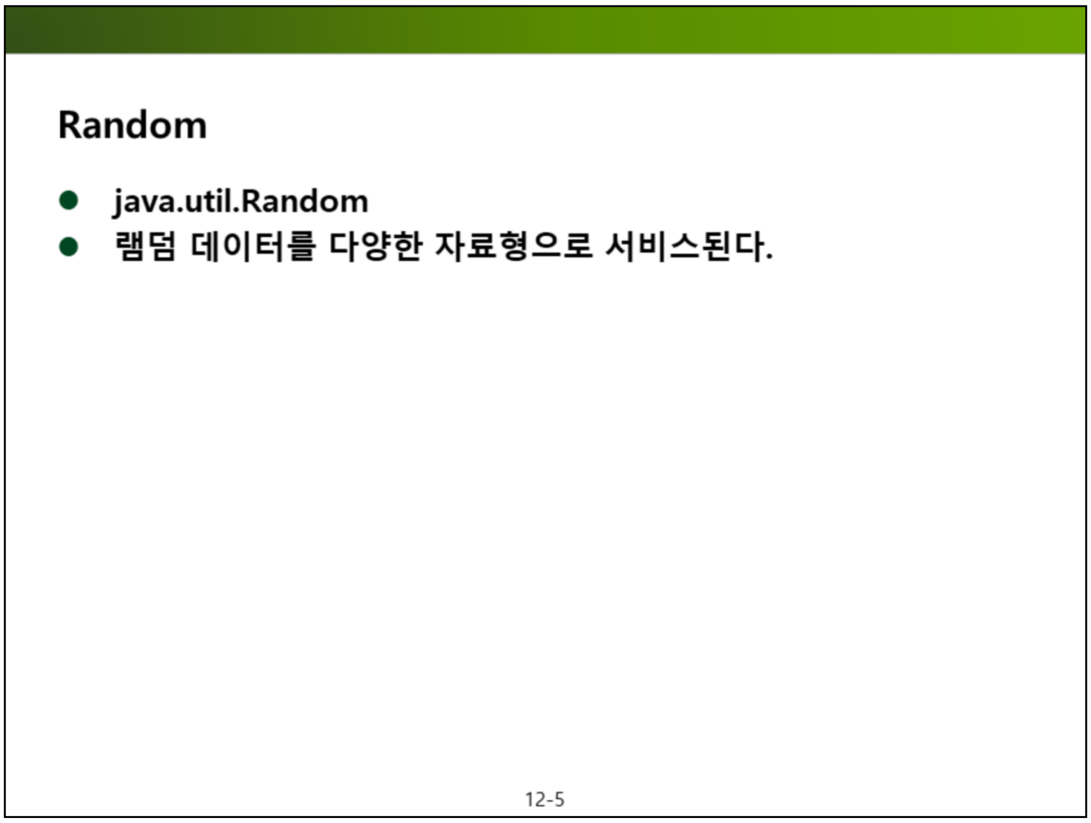


Math

- java.lang.Math
- 수학 관련 메서드 정의
- 모든 멤버 static 임으로 객체 생성하지 않는다.

12-4

```
public class MathTest {
    public static void main(String[] args) {
        System.out.println(Math.PI);
        System.out.println(Math.E);

        System.out.println(Math.random());
        System.out.println(Math.abs(-23.45));
        System.out.println(Math.abs(23.45));
        System.out.println(Math.max(12, 3));
        System.out.println(Math.min(12, 3));

    }
}
```

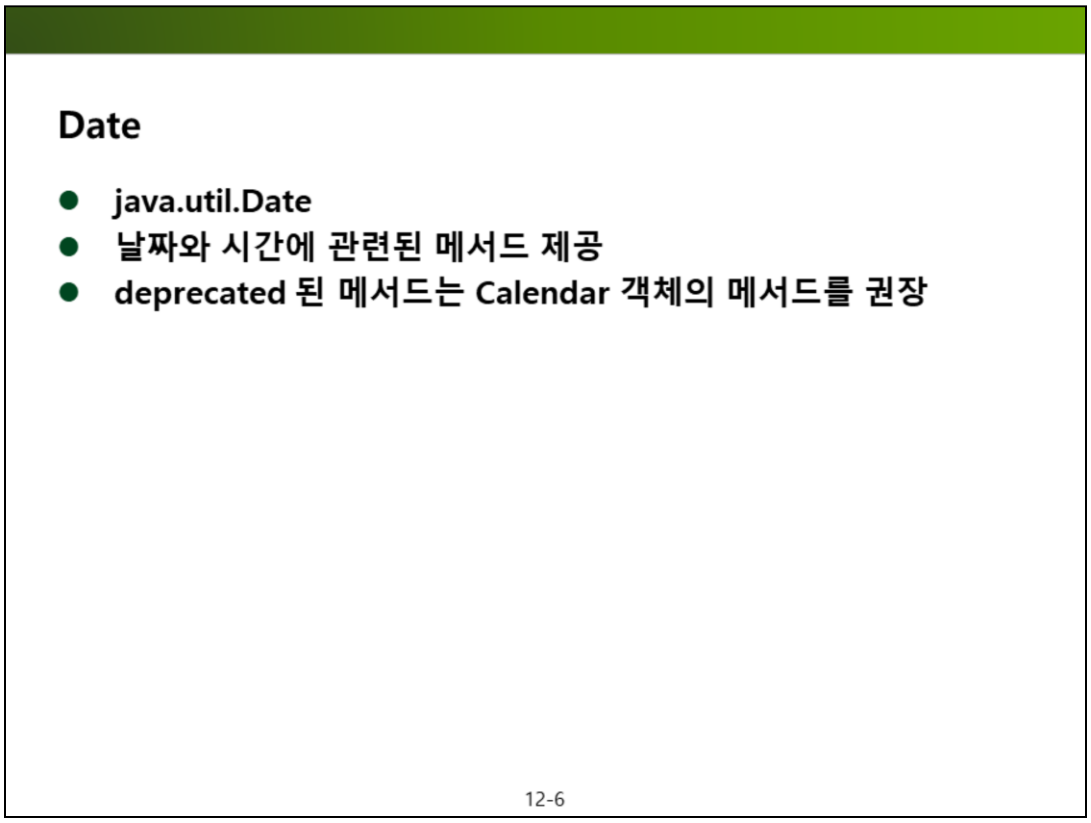


Random

- java.util.Random
- 램덤 데이터를 다양한 자료형으로 서비스된다.

12-5

```
import java.util.Random;

public class RandomTest {
    public static void main(String[] args) {
        Random random=new Random();

        System.out.println(random.nextInt());
        System.out.println(random.nextInt(2));
        System.out.println(random.nextLong());
        System.out.println(random.nextBoolean());
        System.out.println(random.nextFloat());
        System.out.println(random.nextDouble());

    }
}
```

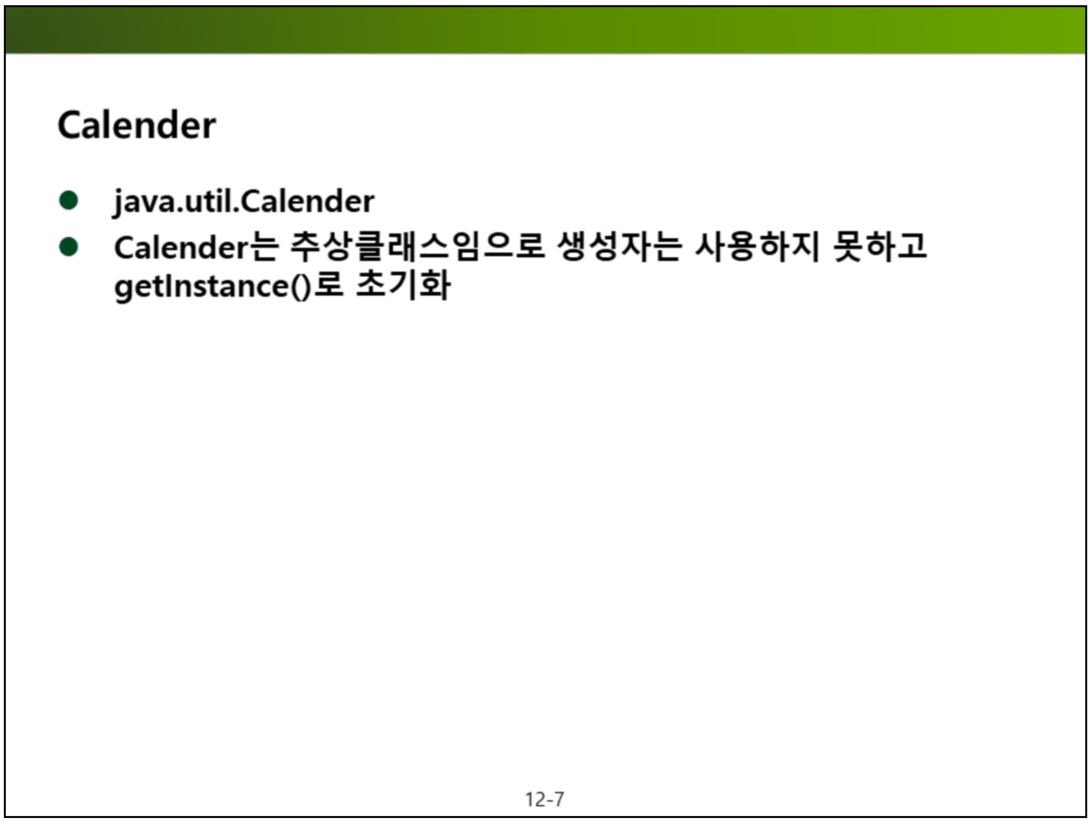


Date

- java.util.Date
- 날짜와 시간에 관련된 메서드 제공
- deprecated 된 메서드는 Calendar 객체의 메서드를 권장

12-6

```
import java.util.Date;

public class DateTest {
    public static void main(String[] args) {
        Date date=new Date();

        System.out.println(date);
        System.out.println(date.getYear());
        System.out.println(date.getMonth());
        System.out.println(date.getDay());

        date=new Date(12355422345L);
        System.out.println(date);
    }
}
```

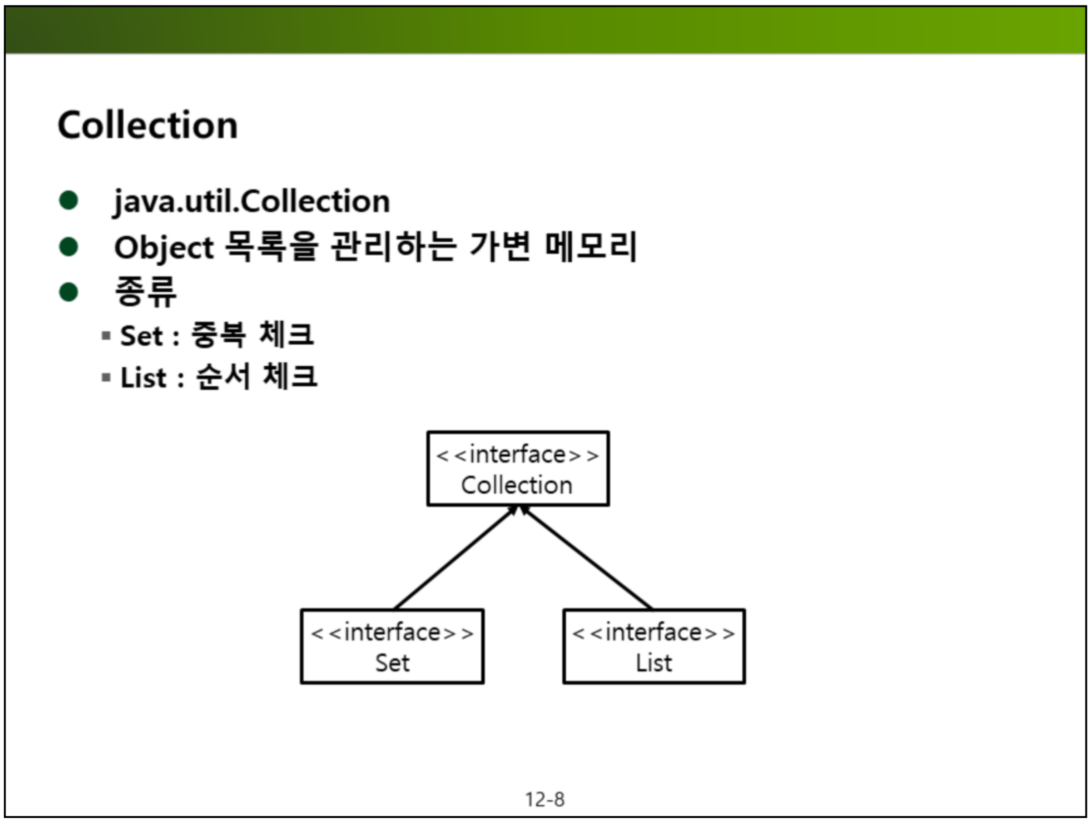


Calender

- java.util.Calender
- Calender는 추상클래스임으로 생성자는 사용하지 못하고 getInstance()로 초기화

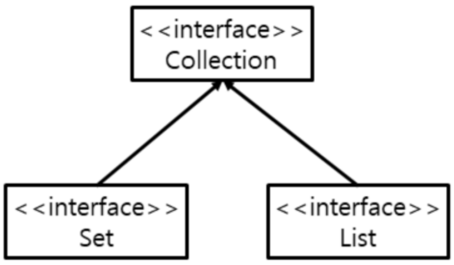
12-7

```
import java.util.Calendar;

public class CalenderTest {
    public static void main(String[] args) {
        Calendar calender=Calendar.getInstance();

        System.out.println(calender.get(Calendar.YEAR));
        System.out.println(calender.get(Calendar.MONTH));
        System.out.println(calender.get(Calendar.DAY_OF_WEEK));
        System.out.println(calender.get(Calendar.DAY_OF_WEEK_IN_MONTH));

        System.out.println(calender.get(Calendar.HOUR_OF_DAY));
        System.out.println(calender.get(Calendar.MINUTE));
        System.out.println(calender.get(Calendar.SECOND));

    }

}
```



Collection

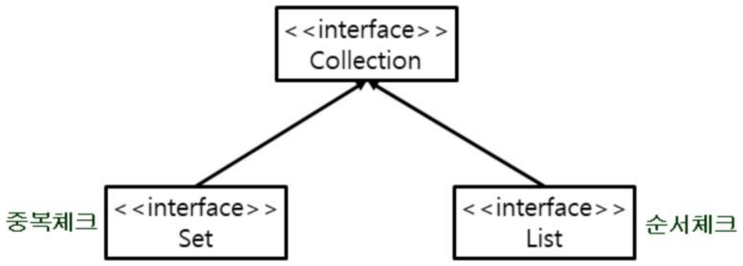
- java.util.Collection
- Object 목록을 관리하는 가변 메모리
- 종류
  - Set : 중복 체크
  - List : 순서 체크



12-8

Collection

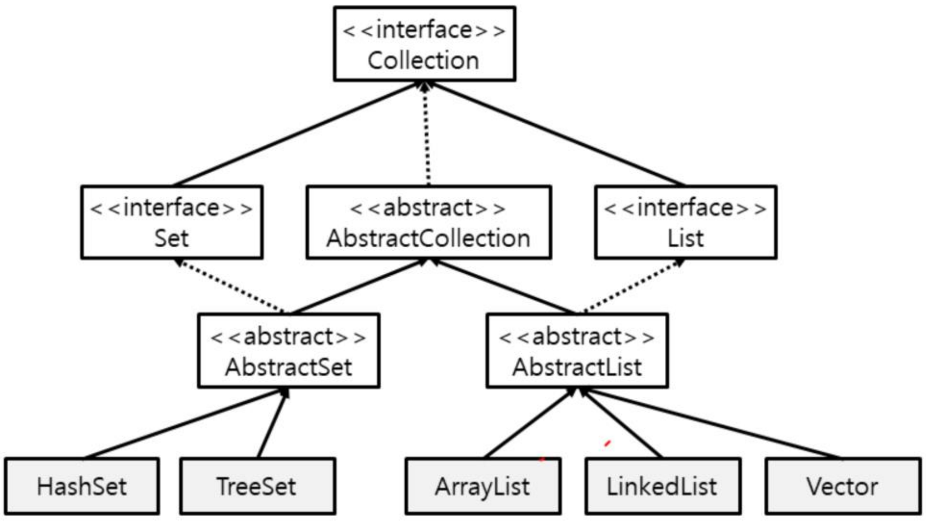
- java.util.Collection
- Object 목록을 관리하는 가변 메모리



12-9

Collection

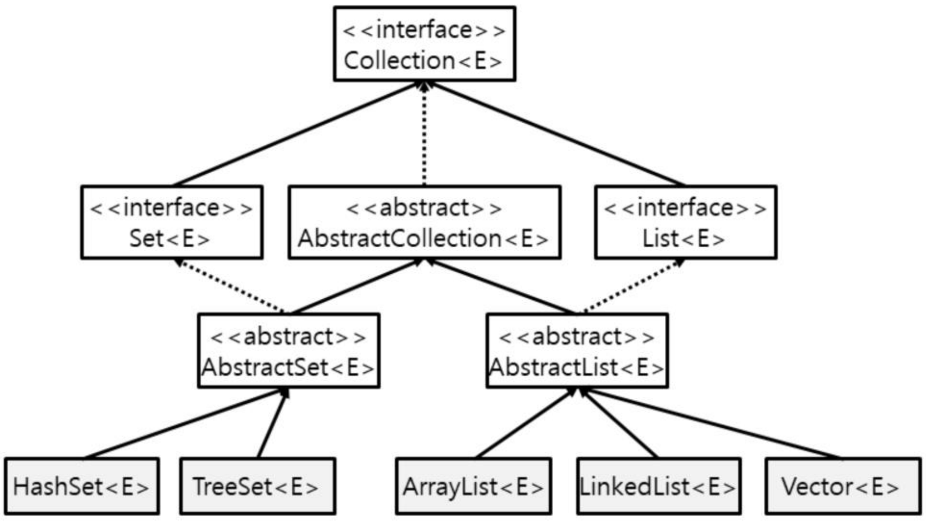
- **Collection API**



12-10

Collection

- **Collection API 5.0**



<<interface>> Collection<E>
<<interface>> Set<E>
<<abstract>> AbstractCollection<E>
<<interface>> List<E>
<<abstract>> AbstractSet<E>
<<abstract>> AbstractList<E>
HashSet<E>
TreeSet<E>
ArrayList<E>
LinkedList<E>
Vector<E>

12-11

• **Employee.java**

```
public class Employee {
    private int number;
    private String name;
    public Employee(int number, String name) {
        this.number = number;
        this.name = name;
    }
    public Employee() {        super();    }
    public int getNumber() {        return number;    }
    public void setNumber(int number) {        this.number = number;}
    public String getName() {        return name;    }
    public void setName(String name) {        this.name = name;    }

   @Override
   public int hashCode() {
        final int prime = 31;
        int result = 1;
        result = prime*result + ((name == null) ? 0 : name.hashCode());
        result = prime * result + number;
        return result;
    }
    public boolean equals(Object obj) {
        if (this == obj)return true;
        if (obj == null)return false;
        if (getClass() != obj.getClass()) return false;
        Employee other = (Employee) obj;
        if (name == null) {
            if (other.name != null)           return false;
        } else if (!name.equals(other.name))
            return false;
        if (number != other.number) return false;
        return true;
    }
    public String toString() {
        return " number=" + number + ", name=" + name ;
    }
}
```

- **ListTest.java**

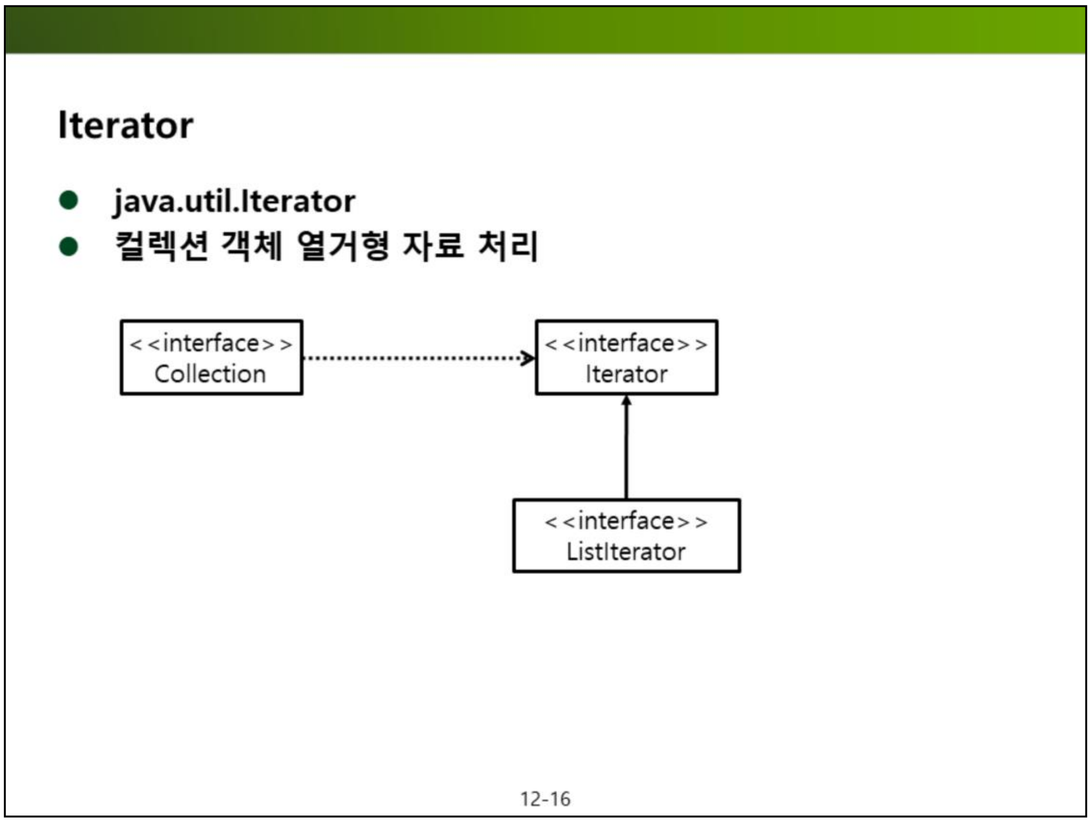
```
import java.util.ArrayList;

/**
List --> 가변메모리 객체를 저장, 순서 체크, 중복 허용,   ArrayList,
Vector
 */
public class ListTest {
    public static void main(String[] args) {

        //int ==>Integer 자동 처리 (JDK5 이상)
        ArrayList list=new ArrayList();
        System.out.println(list.add(new Integer(1234)));
        System.out.println(list.add(1234));
        System.out.println(list.add(new String("hi")));
        System.out.println(list.add("hi"));
        System.out.println(list.add(new Employee(1234, "홍길동")));

        System.out.println(list.add(new Integer(1234)));
        System.out.println(list.add(1234));
        System.out.println(list.add(new String("hi")));
        System.out.println(list.add("hi"));
        System.out.println(list.add(new Employee(1234, "홍길동")));

        System.out.println(list);
    }
}
```

• **SetTest.java**

```
package kr.zeroand.java.collection;
import java.util.HashSet;
/**
Set --> 가변메모리 객체를 저장, 중복허용 X,  HashSe
t*/
public class SetTest {
    public static void main(String[] args) {

        HashSet set=new HashSet();
        System.out.println(set.add(new Employee(1234,"홍길동")));
        System.out.println(set.add(new Integer(1234)));
        System.out.println(set.add(1234));
        System.out.println(set.add(new String("hi")));
        System.out.println(set.add("hi"));

        System.out.println(set.add(new Employee(1234,"홍길동")));
        System.out.println(set.add(new Integer(1234)));
        System.out.println(set.add(1234));
        System.out.println(set.add(new String("hi")));
        System.out.println(set.add("hi"));

        System.out.println(set);
    }
}
```

• CollectionTest.java

```
import java.util.ArrayList;
import java.util.Collection;
import java.util.HashSet;

/**Collection ==> List, Set*/
public class CollectionTest {

    public static Collection methodA(Collection coll){
        System.out.println(coll.add(new Employee(1234,"홍길동")));
        System.out.println(coll.add(new Integer(1234)));
        System.out.println(coll.add(1234));
        System.out.println(coll.add(new String("hi")));
        System.out.println(coll.add("hi"));

        System.out.println(coll.add(new Employee(1234, "홍길동")));
        System.out.println(coll.add(new Integer(1234)));
        System.out.println(coll.add(1234));
        System.out.println(coll.add(new String("hi")));
        System.out.println(coll.add("hi"));

        return collection;
    }

    public static void main(String[] args) {
        System.out.println(methodA(new HashSet()));
        System.out.println(methodA(new ArrayList()));
    }
}
```



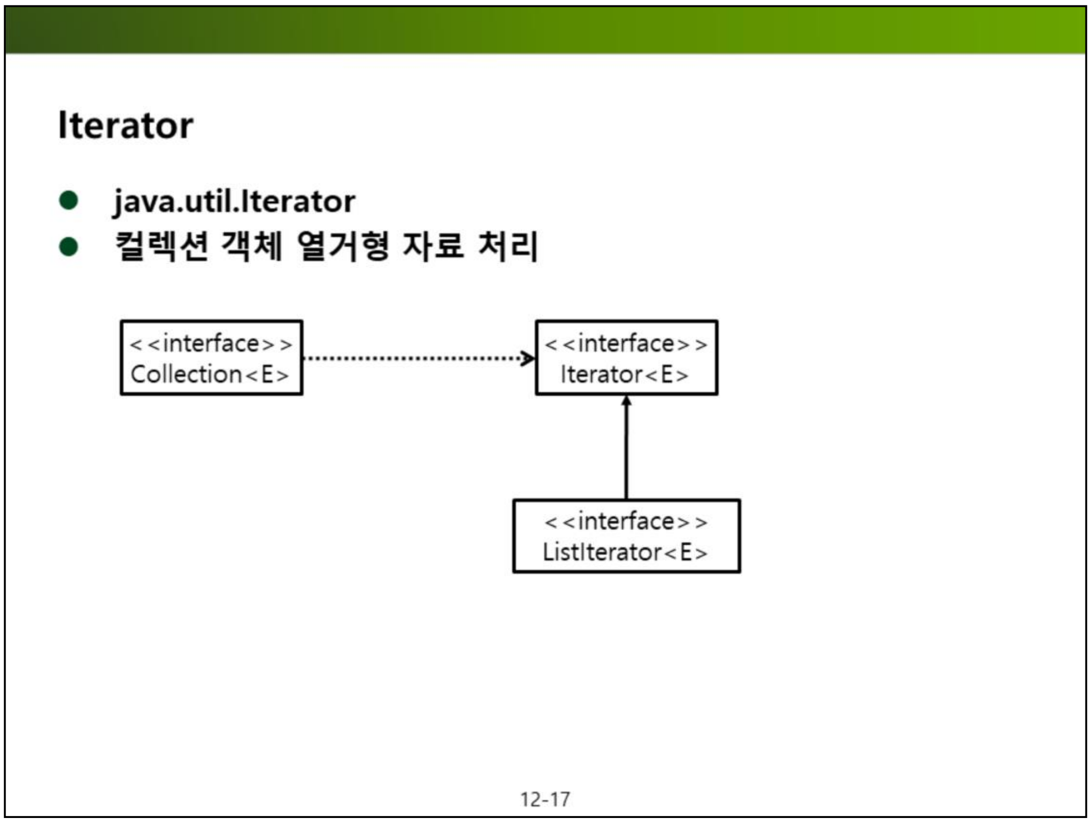
Iterator

- java.util.Iterator
- 컬렉션 객체 열거형 자료 처리

$$\begin{tikzcd}
\text{<<interface>> Collection} \arrow[r, dotted] & \text{<<interface>> Iterator} \\
& \text{<<interface>> ListIterator} \arrow[u]
\end{tikzcd}$$

12-16

```
public class EmployeeIterator_1 {
    public static void main(String[] args) {

        Collection list=new ArrayList();
        list.add(new Employee(123, "홍길동"));
        list.add(new Employee(456, "전혜영"));
        list.add(new String("홍길동"));

        Iterator iter=list.iterator();
        while(iter.hasNext()){
          Object obj=iter.next();
            if(obj instanceof Employee){
                Employee emp=(Employee)obj;
                System.out.println(emp.getNumber()+" = "+ emp.getName());
            }
        }
    }
}
```



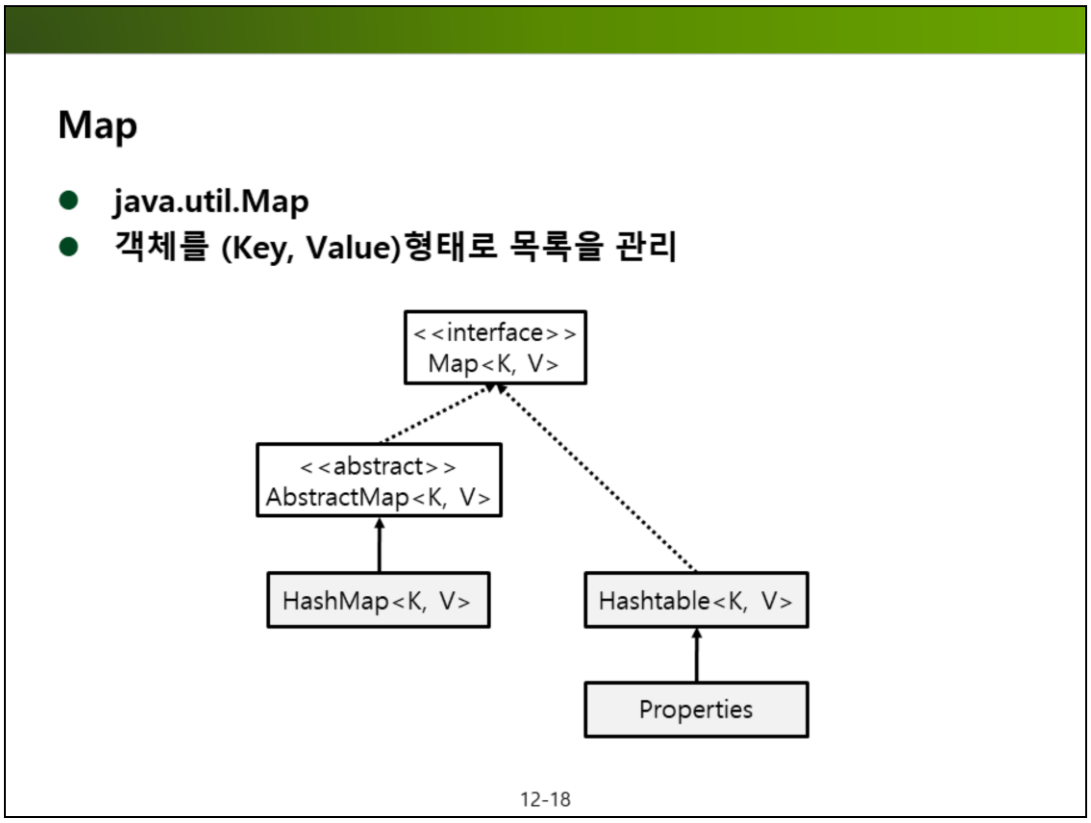
Iterator

- java.util.Iterator
- 컬렉션 객체 열거형 자료 처리

$$\begin{tikzcd} \text{<<interface>> Collection<E>} \arrow[r, dotted] & \text{<<interface>> Iterator<E>} \\ & \text{<<interface>> ListIterator<E>} \arrow[u] \end{tikzcd}$$

12-17

```
public class EmployeeIterator_2 {
    public static void main(String[] args) {
        //Collection은 기본적으로 자료형이 Object
        //그런데 <>으로 지정하면 원하는 자료형의 Collection 지정
        Collection<Employee> list=new ArrayList<Employee>(); //고정
        list.add(new Employee(123, "홍길동"));
        list.add(new Employee(456, "전혜영"));
        //list.add(new String("홍길동")); //컴파일 오류

        Iterator<Employee> iter=list.iterator();
        while(iter.hasNext()){
            Employee emp=iter.next();
            System.out.println(emp.getNumber()+" = "+ emp.getName());
        }
    }
}
```



Map

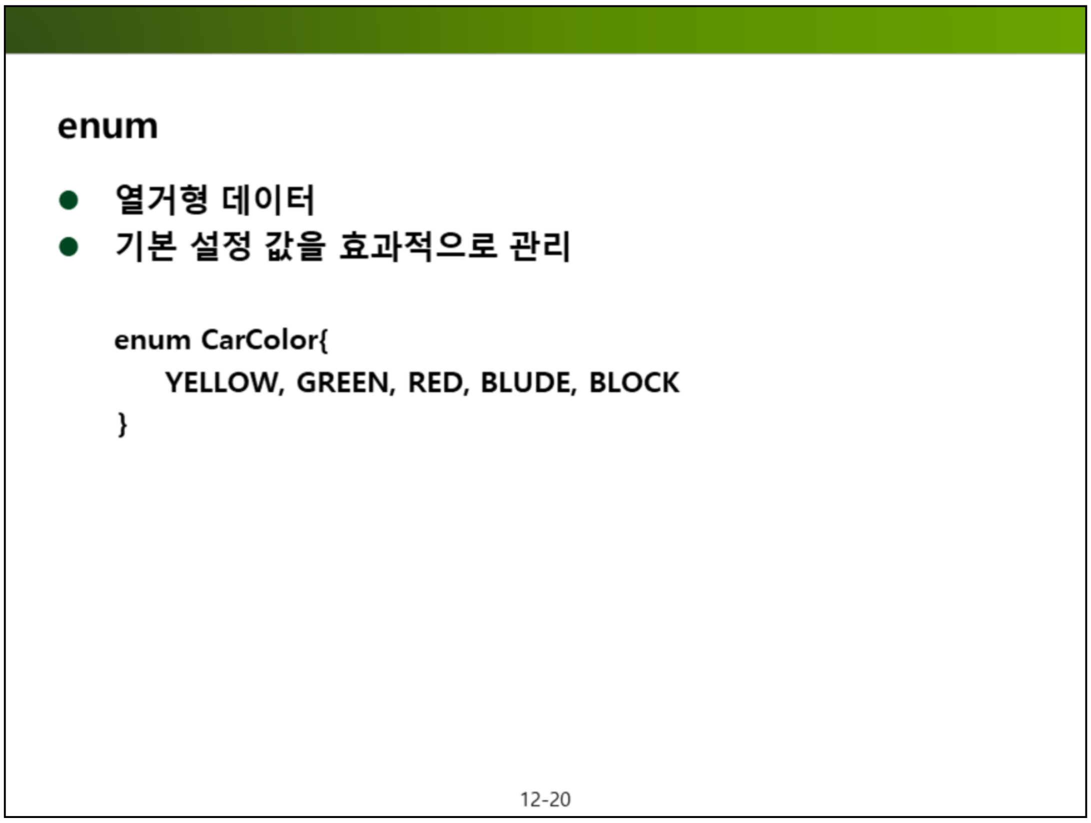
- java.util.Map
- 객체를 (Key, Value)형태로 목록을 관리

<<interface>>
Map<K, V>
<<abstract>>
AbstractMap<K, V>
HashMap<K, V>
Hashtable<K, V>
Properties
12-18

```
public class MapTest {
    public static void main(String[] args) {
        Map map=new HashMap();
        map.put(123, "홍길동");
        map.put(456,"전혜영");
        map.put("emp_1", new Employee(789, "김민성"));

        System.out.println(map);

        Iterator keyNames=map.keySet().iterator();
        while(keyNames.hasNext()){
            Object keyName=keyNames.next();
            Object keyValue=map.get(keyName);
            System.out.println(keyName+"'s value => "+ keyValue);
        }
    }
}
```

• JVM 시스템 정보

```
import java.util.Enumeration;
import java.util.Properties;

public class SystemInfo {
    public static void main(String[] args) {

        Properties pro=System.getProperties();
        Enumeration names=pro.propertyNames();

        while(names.hasMoreElements()){
            String key=names.nextElement().toString();
            String value=System.getProperty(key);
            System.out.println(key+" : " + value);
        }
    }
}
```



enum

- 열거형 데이터
- 기본 설정 값을 효과적으로 관리

```
enum CarColor{
    YELLOW, GREEN, RED, BLUDE, BLOCK
}
```

12-20

• **enum**

```
enum CarColor{
            YELLOW, GREEN, RED, BLUDE, BLOCK
}

public class EnumTest {

  public String carColorPrint(CarColor carColor){
    String message="흰색“;
    switch (carColor) {
      case YELLOW:
          message="노랑";
          break;
      case GREEN:
          message="초록";
          break;
      case RED:
          message="빨강";
          break;
      case BLUDE:
          message="파랑";
          break;
      case BLOCK:
          message="검정";
          break;
    }
      return message;
  }

  public static void main(String[] args) {
    EnumTest t=new EnumTest();
    System.out.println(t.carColorPrint(CarColor.YELLOW));
  }
}
```

• **JVM 시스템 정보**

```
import java.util.Enumeration;
import java.util.Properties;

public class SystemInfo {
    public static void main(String[] args) {

        Properties pro=System.getProperties();
        Enumeration names=pro.propertyNames();

        while(names.hasMoreElements()){
            String key=names.nextElement().toString();
            String value=System.getProperty(key);
            System.out.println(key+" : " + value);
        }
    }
}
```

LAB