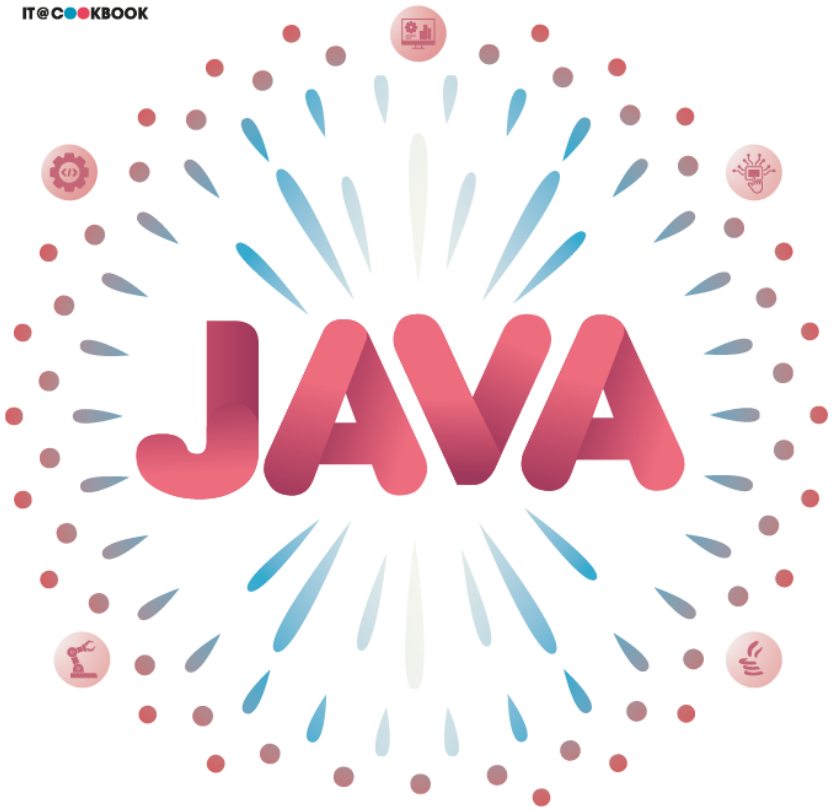


Chapter 16. 유틸리티 패키지와 제네릭

학습목표

- 유틸리티 패키지의 개요에 관해 학습합니다.
- Random 클래스의 사용방법에 관해 학습합니다.
- StringTokenizer 클래스의 사용방법과 응용에 관해 학습합니다.
- Scanner 클래스의 사용방법에 관해 학습합니다.
- 날짜와 관련된 Calendar와 GregorianCalendar 클래스에 관해 학습합니다.
- 날짜와 관련된 새로운 패키지인 java.time 패키지에 관해 학습합니다.
- 제네릭 클래스의 개요와 사용방법에 관해 학습합니다.

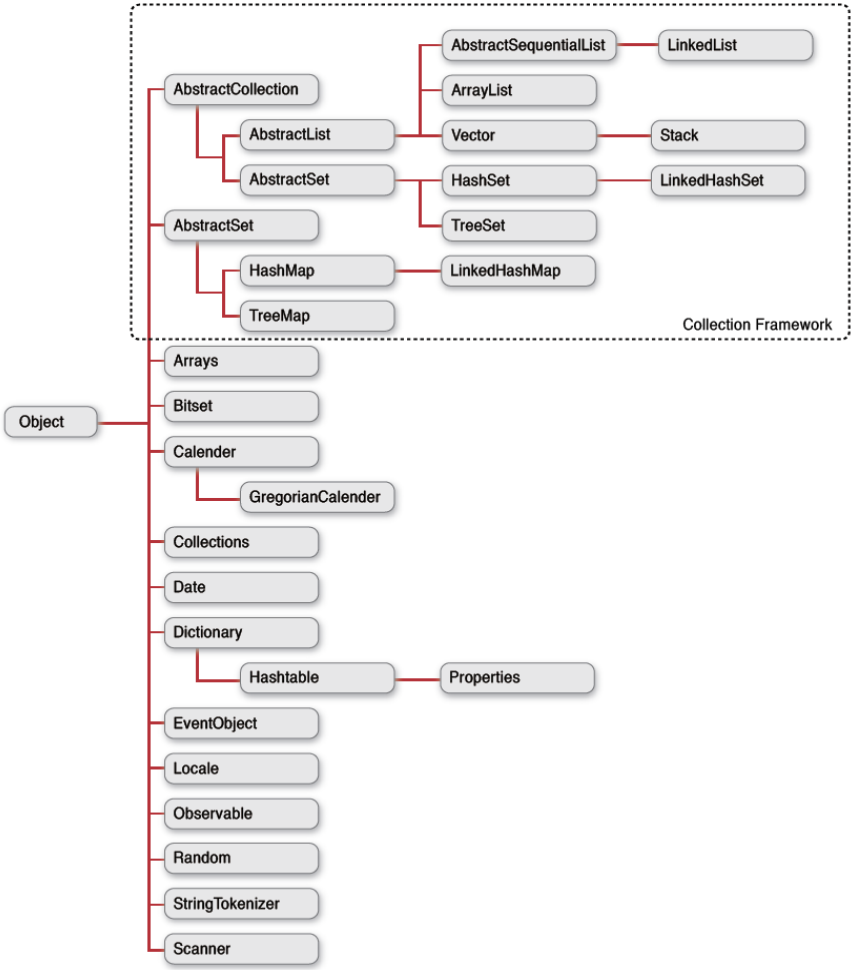
목차

Section 1.

유틸리티 패키지 개요

1 유틸리티 패키지 개요

- 사용자 프로그램에서 필요로 되는 다양하고 편리한 기능들을 모아 java.util 패키지로 제공



Section 2.

Random 클래스

2 Random 클래스

- **난수를 발생시키는 기능을 제공**
  - double, float, int, long 등 다양한 형태의 난수를 발생

[형식] Random 클래스의 생성자

```
Random()
Random(long seed)
```

Random() : 현재의 시간을 초기 값으로 하는 난수 객체를 생성

Random(long seed) : 주어진 seed 값을 초기 값으로 하는 난수 객체를 생성. 특정 초기 값에 대해서는 항상 같은 난수가 생성됨.

2 Random 클래스

메소드 이름	설명
void nextBytes(byte buffer[])	지정된 buffer 배열에 바이트 난수를 채워서 반환
boolean nextBoolean()	boolean형의 난수를 반환
float nextFloat()	float형의 난수를 반환
int nextInt()	int형의 난수를 반환
int nextInt(int n)	0~(n-1) 사이의 정수 난수를 반환
long nextLong()	long형의 난수를 반환
double nextDouble()	double형의 난수를 반환
double nextGaussian()	Gaussian형의 난수를 double 값으로 반환
void setSeed(long newseed)	난수 발생기의 seed 값을 newseed 값으로 설정

2 Random 클래스

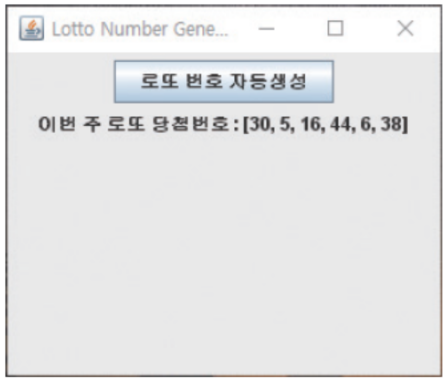
예제 16.1 LottoNumGUI.java

```
01: import java.util.Arrays;
02: import java.util.Random;
03: import javax.swing.*;
04: import java.awt.*;
05: import java.awt.event.*;
06:
07: class Lotto extends JFrame implements ActionListener {
08:   private JLabel lotto_num = new JLabel();
09:   public Lotto() {
10:     JButton lotto = new JButton("로또 번호 자동생성");
11:     Container ct = getContentPane();
12:     ct.setLayout(new FlowLayout());
13:     ct.add(lotto);
14:     ct.add(lotto_num);
15:     lotto.addActionListener(this);
16:     setTitle("Lotto Number Generate");
```

2 Random 클래스

```
17:     setSize(300,250);
18:     setVisible(true);
19:   }
20:   public void actionPerformed(ActionEvent ae) {
21:     Random r = new Random();
22:     int[] lnum = new int[6];
23:     int tmp;
24:     int i = 0;
25:     a : while(i < 6) {
26:             tmp = r.nextInt(45)+1;
27:             for(int j = 0 ; j <= i ; j++) {
28:                     if (tmp == lnum[j])
29:                             continue a;
30:             }
31:     lnum[i]=tmp;
32:     i++;
33:     }
34:     lotto_num.setText(" 이번 주 로또 당첨번호 : "+Arrays.toString(lnum));
35:   }
36: }
37:
38: public class LottoNumGUI {
39:   public static void main(String[] args) {
40:     new Lotto();
41:   }
42: }
```

실행 결과



Section 3.

StringTokenizer 클래스

3 StringTokenizer 클래스

- **문자열을 파싱parsing하여 다양한 형태의 토큰token으로 분리하여 사용할 수 있게 해주는 편리한 기능을 제공**
  - 일정한 형태의 텍스트 데이터를 입력받아 원하는 형태로 분리하여 처리하는 응용 프로그램에서 유용하게 사용

[형식] StringTokenizer 클래스의 생성자

```
StringTokenizer(String str)
StringTokenizer(String str, String delimiters)
StringTokenizer(String str, String delimiters, boolean delimitersAsToken)
```

str : 파싱할 문자열. 분리자가 지정되지 않은 경우 공백space을 분리자로 사용

delimeters : 토큰으로 분리할 분리자를 지정

delimitersAsToken : 토큰이 분리되어 반환될 때 분리자를 포함하여 반환할 것인지를 지정

3 StringTokenizer 클래스

메소드 이름	설명
int countTokens()	문자열에 있는 토큰의 개수를 반환
boolean hasMoreTokens()	토큰이 있으면 true, 없으면 false를 반환
String nextToken()	다음 토큰을 문자열로 반환
String nextToken(String delimiters)	지정된 분리자에 따라 다음 토큰을 문자열로 반환

3 StringTokenizer 클래스

예제 1.6.2 StringTokenizerExam.java

```
01: import java.util.*;
02:
03: class StringTokenizerExam {
04:   public static void main(String args[]) {
05:     String s1 = "국적 대한민국 성명 홍길동 성별 남 주소 서울시";
06:     String s2 = "국적,대한민국,성명,홍길동,성별,남,주소,서울시";
07:     StringTokenizer st1 = new StringTokenizer(s1);
08:     while(st1.hasMoreTokens()) {
09:             String first = st1.nextToken();
10:             String second = st1.nextToken();
11:             System.out.println(first + "\t" + second);
12:     }
13:     System.out.println("==========");
14:     StringTokenizer st2 = new StringTokenizer(s2,",");
15:     while(st2.hasMoreTokens()) {
16:             String first = st2.nextToken();
17:             String second = st2.nextToken();
18:             System.out.println(first + "\t" + second);
19:     }
20:     System.out.println("==========");
21:     st2 = new StringTokenizer(s2);
22:     while(st2.hasMoreTokens()) {
23:             String first = st2.nextToken(",");
24:             String second = st2.nextToken(",");
25:             System.out.println(first + "\t" + second);
26:     }
27:   }
28: }
```

문자열을 공백으로 구분하여 생성

문자열을 콤마로 구분하여 생성

문자열만 지정하여 객체 생성

공백으로 구분된 토큰을 생성하여 출력

문자열과 분리자를 지정하여 객체 생성

분리자를 기준으로 토큰을 생성하여 출력

문자열만 지정하여 객체 생성

토큰을 생생할 때 분리자를 지정하여 출력

실행 결과

```
국적    대한민국
성명    홍길동
성별    남
주소    서울시
==========
국적    대한민국
성명    홍길동
성별    남
주소    서울시
==========
국적    대한민국
성명    홍길동
성별    남
주소    서울시
```

Section 4.

Scanner 클래스

4 Scanner 클래스

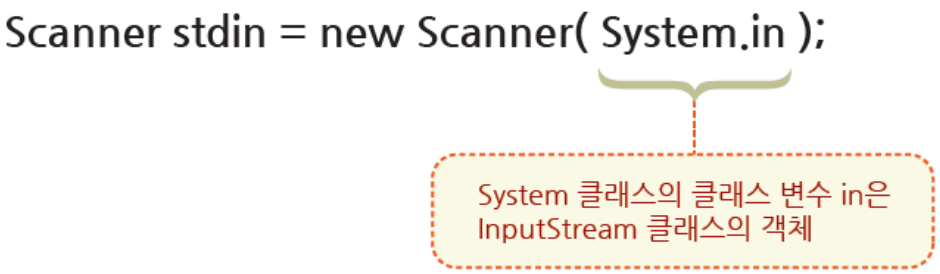


그림 14-2 Scanner 클래스 사용 시 System.in의 의미

- 표준입력(키보드), 파일, 문자열과 같은 다양한 입력으로부터 데이터를 읽어오기 위해 사용

4 Scanner 클래스

[형식] Scanner 클래스의 생성자

```
Scanner(String str)
Scanner(File filename)
Scanner(File filename, String charsetName)
Scanner(InputStream source)
Scanner(InputStream source, String charsetName)
```

str : str은 읽어들일 문자열

filename : File 형의 객체(13장 5절 참조)

source : InputStream 형의 객체

charsetName : 읽어들인 문자의 엔코딩 방법

4 Scanner 클래스

예제 16.3 ScannerTest1.java

```
01: import java.util.*;
02:
03: class ScannerTest1 {
04:   public static void main(String args[]) {
05:     String s = "생각하는 자바, 재미있는 자바, 즐거운 자바, 신나는 자바";
06:     Scanner scanner = new Scanner(s);
07:     scanner.useDelimiter("자바, ");
08:     while (scanner.hasNext())
09:             System.out.println(scanner.next());
10:     s = "1  fish  2  fish  red  fish  blue  fish";
11:     scanner = new Scanner(s).useDelimiter("\\s*fish\\s*");
12:     // "\\s*"는 정규 표현식으로서 한 개 이상의 스페이스를 의미
13:     System.out.println(scanner.nextInt());
14:     System.out.println(scanner.nextInt());
15:     System.out.println(scanner.next());
16:     System.out.println(scanner.next());
17:     scanner.close();
18:   }
19: }
```

06: 문자열을 지정하여 Scanner 객체 생성
07: 분리자를 지정
08~09: 다음 토큰이 있을 때까지 읽어 들여 출력
10: 다수개의 공백이 포함된 문자열 생성
11~12: 한 개 이상의 공백과 fish를 구분자로 하여 객체 생성
13~14: 정수 토큰을 생성하여 출력
15~16: 문자열 토큰을 생성하여 출력

실행 결과

```
생각하는
재미있는
즐거운
신나는 자바
1
2
red
blue
```

4 Scanner 클래스

예제 16.4 ScannerTest2.java

```
01: import java.util.Scanner;
02: import java.io.File;
03:
04: public class ScannerTest2 {
05:   public static void main(String[] args) throws Exception{
06:     Scanner p = new Scanner(System.in);   <-- 표준 입력을 받기 위한 Scanner 객체 생성
07:     System.out.print("검색을 원하는 학생의 학번을 입력하세요 : ");
08:     int id = p.nextInt();
09:     Scanner s = new Scanner(new File("phone.txt"));
10:     while (s.hasNext()) {                 <-- File 객체를 이용하여 Scanner 객체를 생성
11:             if (id == s.nextInt()) {      <-- 학번과 전화번호부의 내용을 비교
12:                     System.out.println(id+"학생의 전화번호 : " + s.next());
13:                     return;               <-- 같으면 출력하고 종료
14:             }
15:             else
16:                     s.next();             <-- 같지 않은 경우 전화 번호를 읽어 스킵
17:     }
18:     System.out.println("학생의 번호가 저장되어 있지 않습니다");
19:   }                                       <-- 입력한 학번이 없는 경우 메시지 출력
20: }
```

```
201795001 101-7777-7777
201895003 101-9999-9999
201895008 101-1234-5678
201997890 101-9876-5432
```

실행 결과

```
검색을 원하는 학생의 학번을 입력하세요 : 201895008
201895008학생의 전화번호 : 101-1234-5678
```

Section 5.

java.time 패키지

JDK 1.8부터 제공되는 패키지

패키지 이름	설명
java.time	국제표준(ISO 8601)에 기반하여 날짜와 시간을 나타내는 핵심 클래스들로 구성된 패키지
java.time.chrono	국제 표준이 아닌 달력 시스템을 사용하기 위한 클래스들을 제공하는 패키지
java.time.format	날짜와 시간을 파싱하고 포맷팅하는 클래스들을 제공하는 패키지
java.time.temporal	날짜와 시간을 연산하기 위한 클래스들을 제공하는 패키지
java.time.zone	타임존time-zone을 지원하는 클래스들을 제공하는 패키지

* ISO 8601 : 날짜와 시간과 관련된 데이터 교환을 다루는 국제 표준. 국제 표준화 기구(ISO)에 의해 1988년에 제정.

5.1 LocalDate, LocalTime, LocalDateTime 클래스

- **java.time 패키지는 날짜와 시간관련 클래스를 3개의 클래스로 구분하여 제공**
  - 날짜와 관련된 클래스 : LocalDate
  - 시간과 관련된 클래스 : LocalTime
  - 날짜와 시간을 모두 가지는 클래스 : LocalDateTime

5.1 LocalDate, LocalTime, LocalDateTime 클래스

메소드 이름	설명
int getDayOfMonth()	월의 1일을 기준으로 객체의 날짜가 몇 번째 일인지를 정수로 반환
DayOfWeek getDayOfWeek()	객체의 요일에 해당되는 DayOfWeek 열거형 (SUNDAY ~ SATURDAY) 문자열로 반환
int getDayOfYear()	1월 1일을 기준으로 객체의 날짜가 몇 번째 일인지를 정수로 반환
Month getMonth()	객체의 월에 해당되는 Month 열거형 (JANUARY ~ DECEMBER) 문자열로 반환
int getMonthValue()	객체의 월에 해당되는 숫자(1~12)를 반환
int getYear()	객체의 년에 해당되는 값을 정수로 반환
boolean isAfter(ChronoLocalDate d)	이 객체보다 이후 인지를 이진값으로 반환
boolean isBefore(ChronoLocalDate d)	이 객체보다 이전 인지를 이진값으로 반환
boolean isLeapYear()	객체의 년이 윤년인지를 이진값으로 반환
LocalDate minusDays(long d)	일을 d 만큼 감소시켜 객체 반환
LocalDate minusMonths(long m)	월을 m 만큼을 감소시켜 객체 반환
LocalDate minusWeeks(long w)	주를 w 만큼 감소시켜 객체 반환
LocalDate minusYears(long y)	년을 y 만큼 감소시켜 객체 반환
static LocalDate now()	현재의 날짜를 가진 LocalDate 객체를 반환하는 클래스 메소드
static LocalDate of(int y, int m, int d)	연월일을 y,m,d로 지정하여 생성된 LocalDate 객체를 반환하는 클래스 메소드
LocalDate plusDays(long d)	일을 d 만큼 증가시켜 객체 반환
LocalDate plusMonths(long m)	년을 m 만큼 증가시켜 객체 반환
LocalDate plusWeeks(long w)	주를 w 만큼 증가시켜 객체 반환
LocalDate plusYears(long y)	년을 y 만큼 증가시켜 객체 반환
long until(Temporal end, TemporalUnit unit)	현재의 객체의 값으로부터 end로 지정된 날짜까지의 정보를 반환
LocalDate withDayofMonth(int d)	일을 d로 변경하여 객체 반환

메소드 이름	설명
LocalDate withDayofYear(int d)	일을 년의 시작일로부터 d 만큼 증가시켜 객체 반환
LocalDate withMonth(int m)	월을 m으로 변경하여 객체 반환
LocalDate withYear(int y)	년을 y로 변경하여 객체 반환

5.1 LocalDate, LocalTime, LocalDateTime 클래스

메소드 이름	설명
int getHour()	시간을 정수로 반환
int getMinute()	분을 정수로 반환
int getNano()	나노초를 정수로 반환
int getSecond()	초를 정수로 반환
boolean isAfter(LocalTime t)	이 객체보다 이후 인지를 이진값으로 반환
boolean isBefore(LocalTime t)	이 객체보다 이전 인지를 이진값으로 반환
LocalTime minusHours(long h)	시를 h 만큼 감소시켜 객체 반환
LocalTime minusMinutes(long m)	분을 m 만큼을 감소시켜 객체 반환
LocalTime minusNanos(long n)	나노초를 n 만큼 감소시켜 객체 반환
LocalTime minusSeconds(long s)	초를 s 만큼 감소시켜 객체 반환
static LocalTime now()	현재의 시간을 가진 LocalTime 객체를 반환하는 클래스 메소드
static LocalTime of(int h, int m)	시와 분을 h,m으로 지정하여 생성된 LocalTime 객체를 반환하는 클래스 메소드
static LocalTime of(int h, int m, int s)	시,분,초를 h,m,s로 지정하여 생성된 LocalTime 객체를 반환하는 클래스 메소드
LocalTime plusHours(long h)	시를 h 만큼 증가시켜 객체 반환
LocalTime plusMinutes(long m)	분을 m 만큼을 증가시켜 객체 반환
LocalTime plusNanos(long n)	나노초를 n 만큼 증가시켜 객체 반환
LocalTime plusSeconds(long s)	초를 s 만큼 증가시켜 객체 반환
long until(Temporal end, TemporalUnit unit)	현재의 객체의 값으로부터 end로 지정된 시간까지의 정보를 반환
LocalTime withHour(int h)	시를 h로 변경하여 객체 반환

메소드 이름	설명
LocalTime withMinute(int m)	분을 m으로 변경하여 객체 반환
LocalTime withNano(int n)	나노초를 n으로 변경하여 객체 반환
LocalTime withSecond(int s)	초를 s로 변경하여 객체 반환

5.1 LocalDate, LocalTime, LocalDateTime 클래스

열거형 ChronoUnit에서 제공되는 상수

상수 이름	의미
ChronoUnit.YEARS	년
ChronoUnit.MONTHS	월
ChronoUnit.WEEKS	주
ChronoUnit.DAYS	일
ChronoUnit.HOURS	시간
ChronoUnit.SECONDS	초
ChronoUnit.MILLIS	밀리초
ChronoUnit.NANOS	나노초

5.1 LocalDate, LocalTime, LocalDateTime 클래스

LocalDateTime 클래스의 주요 메소드

메소드 이름	설명
static LocalDateTime now()	현재의 시간을 가진 LocalDateTime 객체를 반환하는 클래스 메소드
static LocalDateTime of(int y, int m, int d, int h, int mi)	지정된 날짜와 시간을 가지는 LocalDateTime 객체를 반환하는 클래스 메소드
static LocalDateTime of(int y, int m, int d, int h, int mi, int s)	지정된 날짜와 시간(초 포함)을 가지는 LocalDateTime 객체를 반환하는 클래스 메소드

5.1 LocalDate, LocalTime, LocalDateTime 클래스

예제 1.6.8 TimeTest1.java

```
01: import java.time.LocalDate;
02: import java.time.LocalDateTime;
03: import java.time.LocalTime;
04:
05: public class TimeTest1 {
06:   public static void main(String args[]){
07:     LocalDate date = LocalDate.now();
08:     LocalTime time = LocalTime.now();
09:     LocalDateTime dt = LocalDateTime.now();
10:     System.out.println("오늘의 날짜 : " + date);
11:     System.out.println("현재의 시간 : " + time);
12:     System.out.println("현재의 날짜와 시간 : " + dt);
13:     String s = dt.getYear()+"년 ";
14:     s += dt.getMonthValue()+"월 ";
15:     s += dt.getDayOfMonth()+"일 ";
16:     s += dt.getDayOfWeek()+" ";
17:     s += dt.getHour()+"시";
18:     s += dt.getMinute()+"분";
19:     s += dt.getSecond()+"초";
20:     System.out.println("현재의 날짜와 시간 : " + s);
21:
22:     System.out.println("오늘부터 100일 기념일 : " + date.plusDays(100));
23:     System.out.println("오늘부터 10주 후의 날짜 : " + date.plusWeeks(10));
24:   }
25: }
```

현재의 날짜와 시간으로 객체를 생성

객체를 직접 출력

메소드를 이용하여 객체의 요소를 조합하여 출력

100일 후의 날짜를 출력

10주 후의 날짜를 출력

실행 결과

```
오늘의 날짜 : 2017-01-04
현재의 시간 : 11:34:40.183
현재의 날짜와 시간 : 2017-01-04T11:34:40.183
현재의 날짜와 시간 : 2017년 1월 4일 WEDNESDAY 11시34분40초
오늘부터 100일 기념일 : 2017-04-14
오늘부터 10주 후의 날짜 : 2017-03-15
```

5.1 LocalDate, LocalTime, LocalDateTime 클래스

예제 16.9 TimeTest2.java

```
01: import java.time.LocalDate;
02: import java.time.LocalDateTime;
03: import java.time.LocalTime;
04: import java.time.temporal.ChronoUnit;
05: import java.util.Scanner;
06:
07: public class TimeTest2 {
08:   public static void main(String args[]){
09:     Scanner s = new Scanner(System.in);
10:     System.out.print("생일 입력(년월일을 공백으로 구분하여 입력) : ");
11:     int year = s.nextInt();
12:     int month = s.nextInt();
13:     int day = s.nextInt();
14:     LocalDate birth = LocalDate.of(year, month, day); <---------- 입력받은 데이터로 객체 생성
15:     LocalDate today = LocalDate.now();
```

5.1 LocalDate, LocalTime, LocalDateTime 클래스

```
16:     System.out.println("당신의 생일은 : " + toString(birth));
17:     System.out.println("오늘의 날짜는 : " + toString(today));
18:     System.out.println("생일부터 오늘까지의 일 : " + birth.until(today,ChronoU
    nit.DAYS));
19:     System.out.println("생일부터 오늘까지의 년 : " + birth.until(today,ChronoU
    nit.YEARS));
20:     System.out.println("=====================================");
21:     LocalDate birth_100 = birth.plusYears(100);
22:     System.out.println("당신이 100살이 되는 날은 : " + toString(birth_100));
23:     System.out.println("100살까지 남은 주(week) 수 : " + today.until(birth_100
    ,ChronoUnit.WEEKS));
24:     System.out.println("100살까지 남은 일 수 : " + today.until(birth_100,Chron
    oUnit.DAYS));
25:     System.out.println("=====================================");
26:     LocalDateTime current = LocalDateTime.now();
27:     LocalTime mid = LocalTime.of(0,0);
28:     LocalDateTime midnight = LocalDateTime.of(today,mid);
29:     System.out.println("현재의 시간은 : " + current);
30:     System.out.println("오늘 자정까지 남은 시간(초) : " + midnight.until(curre
    nt,ChronoUnit.SECONDS));
31:   }
32:   public static String toString(LocalDate d) {
33:     return d.getYear()+"년 " + d.getMonthValue()+"월 "+d.getDayOfMonth()+"일";
34:   }
35: }
```

생일부터 오늘까지의 년과 일을 출력

생일의 연도에 100을 더하여 객체 생성

100살까지 남은 주와 일을 출력

날짜와 시간 객체로 새로운 객체 생성

자정까지 남은 초를 출력

연월일을 문자열로 출력하는 메소드

실행 결과

```
생일 입력(년월일을 공백으로 구분하여 입력) : 1998 7 26
당신의 생일은 : 1998년 7월 26일
오늘의 날짜는 : 2017년 1월 4일
생일부터 오늘까지의 일 : 6737
생일부터 오늘까지의 년 : 18
=====================================
당신이 100살이 되는 날은 : 2098년 7월 26일
100살까지 남은 주(week) 수 : 4255
100살까지 남은 일 수 : 29788
=====================================
현재의 시간은 : 2017-01-04T14:53:34.060
오늘 자정까지 남은 시간(초) : 53614
```

Section 6.

제네릭

● 객체의 타입을 컴파일 시간에 확정시켜 주는 기능

```
01: .............
02: String allname="";
03: Vector v1 = new Vector();
04: v1.add("kim ");
05: v1.add("park ");
06: v1.add("lee ");
07: v1.add("jung ");
08: v1.add(123);
09: for(int i=0 ; i < v1.size() ; i ++) {
10:   allname = allname + (String)v1.get(i);
11: }
12: System.out.println(allname);
13: .............
```

벡터 객체 생성

벡터 요소 추가

숫자도 추가 가능

모든 요소에 대해 반복 처리

get()메소드는 Object 형을 반환하므로 문자열로 사용하기 위해서는 형 변환이 필요

이러한 **제네릭의 사용은 프로그램에서 다음과 같은 장점**을 제공합니다.

① 컴파일러가 미리 형을 검사할 수 있게 하여 실행시간 오류를 줄일 수 있다.

② 요소 처리를 위한 형 변환casting이 생략되어 효율성이 증대됩니다.

```
01: ......................
02: String allname="";
03: Vector<String> v2 = new Vector<String>();
04: v2.add("kim ");
05: v2.add("park ");
06: v2.add("lee ");
07: v2.add("jung ");
08: //v2.add(123);
09: for(int i=0 ; i < v2.size() ; i ++) {
10:     allname = allname + v2.get(i);
11: }
12: System.out.println(allname);
13: ......................
```

객체를 생성할 때 <String>과 같이 형을 지정

벡터 요소 추가

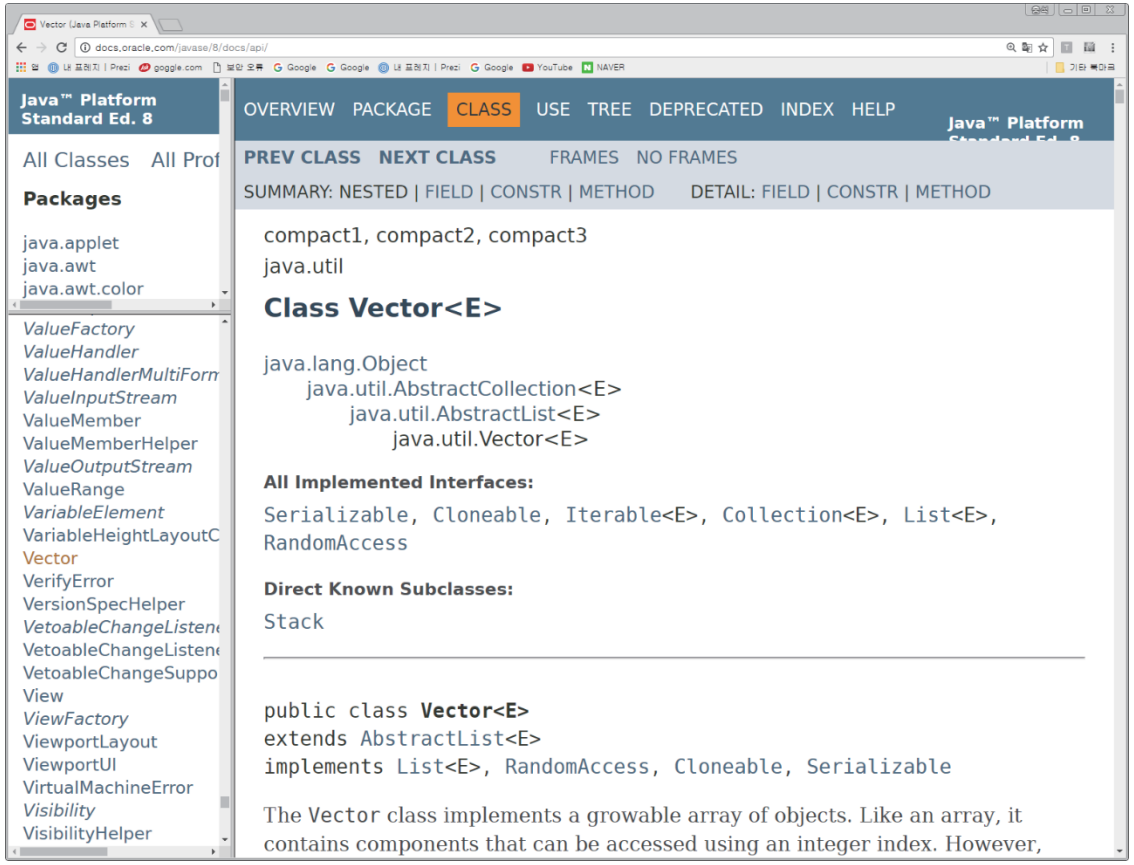
오류발생. 문자열만 가능

모든 요소에 대해 반복 처리

벡터의 요소를 문자열로 정의하였기 때문에 형 변환이 불필요

6.1 제네릭의 선언과 사용

- 자바에서 제네릭은 클래스와 인터페이스에 모두 적용 가능



6.1 제네릭의 선언과 사용

제네릭 클래스와 제네릭 인터페이스를 선언하기 위해서는 "<"와 ">" 사이에 형을 지정하여 선언합니다.

```
01: public class MyClass<T> {...}
02: public interface MyInterface<T> {...}
```

형을 지정하는 매개변수는 임의의 문자를 사용할 수 있으나, 관례적으로 다음과 같은 형태로 사용됩니다.

- E : 요소element를 의미하며 컬렉션의 객체를 나타낼 때 많이 사용
- T : 형type을 의미
- V : 값value을 의미
- K : 키key를 의미

6.1 제네릭의 선언과 사용

예제 16.10 GenericsTest1.java

```
01: class Box<T> {                          ← 제네릭 클래스 선언
02:    T vol;                               ← 변수를 제네릭으로 선언
03:    void setVolume(T v){                 ← 메소드 매개변수를 제네릭으로 선언
04:       vol=v;
05:    }
06:    T getVolume(){                       ← 반환 값의 형을 제네릭으로 선언
07:       return vol;
08:    }
09: }
10:
11: public class GenericsTest1 {
12:    public static void main(String args[]) {
13:       Box<Integer> ibox = new Box<Integer>();   ← 제네릭 형을 지정하여 객체 생성
14:       ibox.setVolume(200);
15:       //ibox.setVolume(32.3);                   ← 형이 다르므로 오류 발생
16:       System.out.println("<Integer>박스의 부피는 : " + ibox.getVolume());
17:       Box<Double> dbox = new Box<Double>();     ← 제네릭 형을 지정하여 객체 생성
18:       dbox.setVolume(123.456);
19:       //dbox.setVolume(300);                    ← 형이 다르므로 오류 발생
20:       System.out.println("<Double>박스의 부피는 : " + dbox.getVolume());
21:    }
22: }
```

실행 결과

```
<Integer>박스의 부피는 : 200
<Double>박스의 부피는 : 123.456
```

Thank You!