제 24 강 : Math, Calendar, Random

※ 학습목표

✓ Math 클래스를 활용할 수 있다.

✓ Calendar 클래스를 활용할 수 있다.

✓ Singleton 패턴을 설명할 수 있다.

✓ Random 클래스를 활용할 수 있다.

1. Math Class

✓ 수학계산을 위한 상수 값과 수학관련 메소드

✓ Math class 는 인스턴스(객체) 를 생성하지 않고 사용

✓ 사용방법 : Math.PI , Math.메소드()

[실습] Math 클래스 사용예제

```
 1  package tommy.java.exam01;
 2
 3  public class MathEx {
 4          public static void main(String ar[]) {
 5                  double da;
 6                  da = Math.pow(2, 3);
 7                  System.out.println("2 의 3 승 = Math.pow(2, 3) =" + da);
 8                  da = Math.max(300f, 301.0);
 9                  System.out.println("Math.max( 300f , 301.0  )=" + da);
10                  da = Math.min(300f, 301.0);
11                  System.out.println("Math.min( 300f , 301.0  )=" + da);
12                  da = Math.abs(-1234.1);
13                  System.out.println("Math.abs( -1234.1 )=" + da);
14                  da = Math.abs(+1234.1);
15                  System.out.println("Math.abs( +1234.1 )=" + da);
16                  da = Math.random();
17                  System.out.println("Math.random()=" + da);
18                  da = Math.random();
19                  System.out.println("Math.random()=" + da);
20                  da = Math.sqrt(90000);
```

```
21                      System.out.println("Math.sqrt( 90000 )=" + da);
22                      long dda = Math.round(123.56);
23                      System.out.println("Math.round(123.56)=" + dda);
24                      dda = Math.round(123.46);
25                      System.out.println("Math.round(123.46)=" + dda);
26                      da = Math.ceil(123.56);
27                      System.out.println("Math.ceil(123.56)=" + da);
28                      da = Math.ceil(123.46);
29                      System.out.println("Math.ceil(123.46)=" + da);
30                      da = Math.floor(123.56);
31                      System.out.println("Math.floor(123.56)=" + da);
32                      da = Math.floor(123.46);
33                      System.out.println("Math.floor(123.46)=" + da);
34                      da = Math.rint(123.56);
35                      System.out.println("Math.rint(123.56)=" + da);
36                      da = Math.rint(123.46);
37                      System.out.println("Math.rint(123.46)=" + da);
38                      da = Math.sin(45);
39                      System.out.println("Math.sin(45)=" + da);
40                      int degrees = 60;
41                      da = Math.toRadians(degrees);
42                      System.out.println("Math.toRadians(60)=" + da);
43                      double radians = ((double) degrees / 180.0) * Math.PI;
44                      System.out.println("radians (60)=" + radians);
45                      da = Math.toDegrees(radians);
46                      System.out.println("Math.toDegrees(radians)=" + da);
47                      da = Math.PI;
48                      System.out.println("Math.PI=" + da); // PI 라는 원주율
49                      da = Math.E;
50                      System.out.println("Math.E=" + da); // e 라는 자연로그상수
51              }
52  }
```

2. Calendar Class

✓ 날짜 관련 함수

✓ java.util 패키지

✓ 일반적인 달력, 시간의 기능을 선언, 추상클래스 1970년 1월 1일 기준

① 주요 메소드

반환형	메서드명	설 명
boolean	ater(Object when)	현재 Calendar 객체가 인자로 전달된 when 객체의 날짜보다 후의 시각이라면 true, 그렇지 않으면 false를 반환한다.
	before(Object when)	현재 Calendar 객체가 인자로 전달된 when 객체의 날짜보다 전의 시각이라면 true, 그렇지 않으면 false를 반환한다.
int	get(int field)	인자로 전달된 field(년, 월, 일, 시, 분, 초)의 값을 반환한다.
static Calendar	getInstance()	default TimeZone과 Locale을 사용해 Calendar 객체를 반환한다.
void	set(int year, int month, int date)	현 Calendar 객체의 필드 중 년도와 월 그리고 일(DAY_OF_MONTH)를 인자로 전달된 값을 설정한다
	setTimeMillis(long millis)	Calendar의 현재 시각을 인자로 전달된 long형의 값으로 설정한다.

② 사용방법

✓ Calendar now = Calendar.getInstance();

✓ value = now.get(Calendar 클래스의 상수값);

③ Calendar 클래스의 상수값

✓ 년도를 얻기위한 상수 = Calendar.YEAR

✓ 월를 얻기위한 상수 = Calendar.MONTH

✓ 일를 얻기위한 상수 = Calendar.DATE or Calendar.DAY_OF_MONTH

✓ 시를 얻기위한 상수 = Calendar.HOUR

✓ 분를 얻기위한 상수 = Calendar.MINUTE

✓ 초를 얻기위한 상수 = Calendar.SECOND

[실습]

```
1  package tommy.java.exam02;
2
3  import java.util.Calendar;
4
5  public class CalendarEx {
6          public static void main(String[] args) {
7                  StringBuffer sb = new StringBuffer("1년 중 ");
8                  Calendar now = Calendar.getInstance();
9                  int week_yy = now.get(Calendar.WEEK_OF_YEAR);
10                 int yy = now.get(Calendar.YEAR);
11                 int mm = now.get(Calendar.MONTH) + 1; // 1월이 0을 기억한다.
12                 int dd = now.get(Calendar.DAY_OF_MONTH);
13                 sb.append(week_yy);
14                 sb.append("주째인 ");
15                 sb.append(yy);
16                 sb.append("년 ");
17                 sb.append(mm);
18                 sb.append("월 ");
19                 sb.append(dd);
20                 sb.append("일");
21                 System.out.println(sb.toString());
22         }
23 }
```

3. Singleton 패턴

✓ 패턴이란?

✓ 소프트웨어 공학에서 패턴이란 특정 상황에 문제를 해결하기 위한 솔루션

① 디자인 패턴

✓ 소프트웨어 설계 시 특정 상황에서 자주 만나는 문제를 해결하기 위해 사용할 수 있는 재사용 가능한 솔루션을 말한다.

✓ GoF의 디자인 패턴

✓ Erich Gamma, Richard Helm, Ralph Johnson, John Vlissdes

✓ 생성패턴 5가지, 구조패턴 7가지, 행위패턴 11가지 = 23가지 형태

② 싱글톤 패턴

✓ 키보드 리더, 프린터 스풀러, 점수기록표 등 클래스의 객체를 하나만 만들어야 하는 경우 사용한다.

✓ 클래스 내에서 인스턴스가 단 하나뿐임을 보장하므로, 프로그램 전역에서 해당 클래스의 인스턴스를 바로 얻을 수 있고, 불필요한 메모리 낭비를 최소화한다.

✓ 이 패턴에서는 생성자를 클래스 자체만 사용할 수 있도록 private 등의 접근제한자를 통하여 제한하여야 한다. (생성자를 다른 곳에서도 사용할 수 있으면 그곳에서도 인스턴스를 만들 수 있기 때문.)

✓ 싱글톤 패턴을 사용하기 위해서는 반드시 접근제한자를 이용하여 외부의 접근을 막거나, final로 reference를 변경 불가능하게 설정하여야 한다. 물론 생성자에 접근제한자를 사용하면 최소한 다른 인스턴스로 레퍼런스시키지는 못하겠지만, ClassName.singleton = null;처럼 레퍼런스 자체를 지워버릴 수 있기 때문.

✓ 구현 방법에는 사전 초기화, 사후 초기화 등이 있다.

② Eager initialization(사전 초기화)

✓ 클래스 로딩 시에 인스턴스를 생성하는 방법이다. 멀티스레드 환경에서의 이중 객체 생성 문제가 없지만, 인스턴스를 호출하지 않아도 무조건 클래스를 초기화하기에 메모리 효율이나 연산 효율은 낮다.

✓ Java에서는 static block initialization이라는 변종도 있다. 클래스가 로딩될 때 최초 1회만 실행되는 static block을 통해 싱글톤 인스턴스를 초기화하는 방법인데, 구조적으로는 크게 다르지 않다.

[사전 초기화 방식의 샘플코드]

```
1  public class Singleton {
2      static final Singleton instance = new Singleton();
3
4      private Singleton() {
5
6      }
7
8      public Singleton getInstance() {
9          return instance;
10     }
11 }
```

③ Lazy initialization(사후 초기화)

✓ 인스턴스를 실제로 사용할 시점에서 인스턴스를 생성하는 방법이다. 세심한 방법을 쓰지 않으면 위에 언급한 이중 객체 생성 문제가 발생할 가능성이 높으나, 인스턴스를 실제로 사용하지 않는다면 메모리와 연산량을 아낄 수 있다는 장점이 있다.

[실습] 일반 객체와 Singleton 객체 생성 비교예제

✓ 같은 패키지에 아래의 예제 클래스를 작성하고 실행한다.

✓ 일반 객체 파일 MyClass.java를 작성한다.

```
1  package tommy.java.exam03;
2
3  public class MyClass {
4
5          public MyClass() {
6                  System.out.println("MyClass 객체 생성됨");
7          }
8
9  }
```

✓ 싱글톤 객체 파일 MySingleton.java를 작성한다.

```
1  package tommy.java.exam03;
2
3  public class MySingleton {
4          private static MySingleton instance;
5
6          private MySingleton() {
7                  System.out.println("MySingleton 객체 생성됨");
8          }
9
10         public static MySingleton getInstance() {
11                 if (instance == null) {
12                         synchronized (MySingleton.class) {
13                                 instance = new MySingleton();
14                         }
15                 }
16                 return instance;
17         }
18
19 }
```

✓ 테스트용 메인 클래스 SingletonEx.java를 작성한다.

```
1  package tommy.java.exam03;
2
3  public class SingletonEx {
4          public static void main(String[] args) {
5                  MyClass mc1 = new MyClass();
6                  MyClass mc2 = new MyClass();
7                  MyClass mc3 = new MyClass();
8                  System.out.println();
9                  MySingleton ms1 = MySingleton.getInstance();
10                 MySingleton ms2 = MySingleton.getInstance();
11                 MySingleton ms3 = MySingleton.getInstance();
12         }
13 }
```

4. Random 클래스

✓ Random 객체는 일련의 난수를 생성한다.

✓ 이렇게 생성된 Random 객체는 int형 float형 등의 난수가 발생 가능하며 정수형 난수 발생은 특정 범위가 없다.

✓ 하지만 부동소수점을 가지는 실수 형들의 난수는 0.0에서 1.0사이의 값을 받도록 되어 있다.

① Random Class의 주요 메소드

반환형	메서드명	설 명
double	nextDouble()	double형 자료에 따른 자료를 반환하게 되며 0.0~1.0 사이의 값을 반환한다.
float	nextFloat()	float형 자료에 따른 자료를 반환하게 되며 0.0~1.0 사이의 값을 반환한다.
double	nextGaussian()	평균 0.0 표준편차 1.0의 Gauss 분포의 double형 난수를 발환한다.
int	nextInt()	int형의 범위 전체에서 난수를 발생하여 반환한다.
	nextInt(int n)	0부터 인자로 전달된 값의 전까지를 범위로 하여 난수를 발생하여 반환한다.
long	nextLong()	long형의 범위전체에서 난수를 발생하여 반환한다.
void	setSeed(long seed)	인자로 전달된 long형인 seed를 난수를 발생기의 사작 seed로 재설정한다.

[실습]

```
1  package tommy.java.exam04;
2
3  import java.util.Random;
4
5  public class RandomEx {
6          public static void main(String[] ar) {
7                  String[] lesson = { “Java“, “JSP“, “Database“, “Spring“};
8                  Random r1 = new Random();
9                  int index = r1.nextInt(4);
10                 System.out.println(“선택과목 :“ + lesson[index]);
11         }
12 }
```