System.currentTimeMillis 와 System.nanoTime의 차이

"시간"의 용도?

시스템에서 "시간"은 크게 두 가지 용도로 사용됩니다. 첫 번째는 말 그대로 시각(시/분/초)을 표현하기 위한 용도이고, 두 번째는 스케줄링이나 이벤트 처리를 위한 타이머 용도입니다. 이 두 가지 시간을 각각 반환하는 메서드가 바로 currentTimeMillis와 nanoTime입니다.

1. CurrentTimeMillis 메서드

```
public class Ex01_CurrentTimeMillis {
    public static void main(String[] args) {
        long currentTimeMillis = System.currentTimeMillis();
        System.out.println(currentTimeMillis);
    }
}
```

결과

```
1633574394206
```

currentTimeMillis 메서드는 UTC 시간 즉, 1970년 1월 1일 자정부터 현재까지 카운트된 시간을 ms(milliseconds) 단위로 표시합니다. 따라서 출력 결과는 1970년 1월 1일 UTC 시작 시점으로부터 현재까지 총 1335895964247(ms) 만큼의 시간이 흘렀음을 말해주는 것입니다. 이 값을 각각 시/분/초로 계산해 보면 현재 시각(GMT+0)을 얻을 수 있습니다.

그럼 currentTimeMillis 메서드로 특정 코드의 수행 시간을 계산해 봅시다.

```
import java.util.Random;

public class Ex02_CurrentTimeMillis2 {
    public static void main(String[] args) {
        // 경과 시간 계산
        long startTime = System.currentTimeMillis();
        doHardWork(); // 작업 수행
        long endTime = System.currentTimeMillis();
        long elapsedTime = endTime - startTime;
        System.out.println("경과시간 : " + elapsedTime);
    }
    // 어느 정도 시간을 요하는 작업
    public static void doHardWork() {
        try {
            Thread.sleep(new Random().nextInt(10)*1000); // 0~9초
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

결과

```
경과시간 : 3014
```

Total elapsed time = 3014 위 코드에서 startTime과 endTime 사이에 어느 정도 계산시간을 요하는 doHardWork 메서드를 실행하고 결과를 확인해 보니 대략 11.844초가 나왔습니다. currentTimeMillis 메서드로 특정 코드의 수행 시간을 측정하는데 아무런 문제가 없어보입니다. 그렇다면 nanoTime 메서드는 왜 제공되는 것일까요?

2. nanoTime 메서드

레퍼런스에 따르면 nanoTime 메서드는 현재 Java 가상머신의 high-resolution 시간값을 ns(nano sec.) 단위로 반환한다고 되어 있습니다. 이 메서드는 오직 경과된 시간을 측정하는데 사용해야하고, 시스템이나 시각(시/분/초)과는 아무런 연관성이 없음을 말해주고 있습니다.

```
public class Ex03_NanoTime {
    public static void main(String[] args) {
        long nanoTime = System.nanoTime();
        System.out.println(nanoTime);
    }
}
```

결과

```
9708963649100
```

그럼 간단한 테스트 코드로 nanoTime 메서드의 반환값과 특정 코드의 수행 시간을 계산해 봅시다.

```
import java.util.Random;

public class Ex04_NanoTime2 {
    public static void main(String[] args) {
        // 경과 시간 계산
        long startTime = System.nanoTime();
        doHardWork(); // 작업 수행
        long endTime = System.nanoTime();
        long elapsedTime = endTime - startTime;
        System.out.println("경과시간 : " + elapsedTime);
    }
    // 어느 정도 시간을 요하는 작업
    public static void doHardWork() {
        try {
            Thread.sleep(new Random().nextInt(10)*1000); // 0~9초
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

결과

```
경과시간 : 9015081000
```

출력 결과를 확인해 보면 대략 9.01초가 경과되었음을 알 수 있습니다.

"시간"의 용도?

시스템에서 "시간"은 크게 두 가지 용도로 사용됩니다. 첫 번째는 말 그대로 시각(시/분/초)을 표현하기 위한 용도이고, 두 번째는 스케줄링이나 이벤트 처리를 위한 타이머 용도입니다. 오라클의 Davidholmes는 각각을 "passive clock"과 "active clock"으로 표현하기도 했습니다. 이 두 가지 시간을 각각 반환하는 메서드가 바로 currentTimeMillis와 nanoTime입니다.

currentTimeMillis 메서드

우선 간단한 코드로 currentTimeMillis 메서드의 결과를 살펴보겠습니다.

```
public class CurrentTimeMillisTest {
        public static void main(String[] args) {
            long currentTime = System.currentTimeMillis();
            System.out.println("current time value = " + currentTime);
            System.exit(0);
        }
    }
```

current time value = 1335895964247

currentTimeMillis 메서드는 UTC 시간 즉, 1970년 1월 1일 자정부터 현재까지 카운트된 시간을 ms(milliseconds) 단위로 표시합니다. 따라서 출력 결과는 1970년 1월 1일 UTC 시작 시점으로부터 현재까지 총 1335895964247(ms) 만큼의 시간이 흘렀음을 말해주는 것입니다. 이 값을 각각 시/분/초로 계산해 보면 현재 시각(GMT+0)을 얻을 수 있습니다.

그럼 currentTimeMillis 메서드로 특정 코드의 수행 시간을 계산해 봅시다.

```
public class CurrentTimeMillisTest {
        public static void doHardWork(int n, int m) {
            int result = 1;
            for(int i=0; i<n; i++)
                for(int j=0; j<m; j++)
                    for(int k=0; k<n+m; k++)
                        for(int l=0; l<n*m; l++)
                            result *= 2;
        }
        public static void main(String[] args) {
            long startTime = System.currentTimeMillis();
            doHardWork(100, 100);
            long endTime = System.currentTimeMillis();

            long elapsedTime = endTime - startTime;
            System.out.println("Total elapsed time = " + elapsedTime);

            System.exit(0);
        }
    }
```

Total elapsed time = 11844 위 코드에서 startTime과 endTime 사이에 어느 정도 계산시간을 요하는 doHardWork 메서드를 실행하고 결과를 확인해 보니 대략 11.844초가 나왔습니다. currentTimeMillis 메서드로 특정 코드의 수행 시간을 측정하는데 아무런 문제가 없어보입니다. 그렇다면 nanoTime 메서드는 왜 제공되는 것일까요?

nanoTime 메서드

레퍼런스에 따르면 nanoTime 메서드는 현재 Java 가상머신의 high-resolution 시간값을 ns(nano sec.) 단위로 반환한다고 되어 있고, 이 메서드는 오직 경과된 시간을 측정하는데 사용해야하고, 시스템이나 시각(시/분/초)과는 아무런 연관성이 없음을 말해주고 있습니다. 그럼 간단한 테스트 코드로 nanoTime 메서드의 반환값과 특정 코드의 수행 시간을 계산해 봅시다.

```
public class NanoTimeTest {
        public static int doHardWork(int n, int m) {
            int result = 1;
            for(int i=0; i<n; i++)
                for(int j=0; j<m; j++)
                    for(int k=0; k<n+m; k++)
                        for(int l=0; l<n*m; l++)
                            result *= 2;
            return result;
        }
        public static void main(String[] args) {
            long startTime = System.nanoTime();
            System.out.println("start nano-time = " + startTime);

            doHardWork(100, 100);
            long endTime = System.nanoTime();
            System.out.println("end nano-time = " + endTime);

            long elapsedTime = endTime - startTime;
            System.out.println("total elapsed time = " + elapsedTime);
            System.exit(0);
        }
    }
```

start nano-time = 583086181421940 end nano-time = 583098045092043 total elapsed time = 11863670103

출력 결과를 확인해 보면 대략 11.86초가 경과되었음을 알 수 있습니다.

비교

레퍼런스와 기타 참고문서를 토대로 두 메서드를 비교해 정리하면,

- currentTimeMillis()
    - 날짜와 관련한 시각 계산을 위해 사용(ms 단위)
    - 시스템 시간을 참조(외부데이터)
    - 일반적인 업데이트 간격은 인터럽트의 타이머(10microsec)에 따름
    - GetSystemTimeAsFileTime 메서드로 구현되어 있으며, 운영체제가 관리하는 시간값을 가져옴
- nanoTime()
    - 기준 시점에서 경과 시간을 측정하는데 사용(ns 단위)
    - 시스템 시간과 무관
    - QueryPerformanceCounter/QueryPerformanceFrequency API로 구현되어 있음.

결론

코드의 수행시간 계산은 nanoTime 메서드를, 시각(시/분/초) 계산은 currentTimeMillis 메서드를 사용하자!!

```
package kr.manamana.exam;

import java.text.SimpleDateFormat;
import java.time.LocalDate;
import java.util.Date;

public class Ex05_MiliVsNano {
    public static void main(String[] args) {
        SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd hh:mm:ss.S");
        long currentTimeMillis = System.currentTimeMillis();
        System.out.println(currentTimeMillis);

        long nanoTime = System.nanoTime();
        System.out.println(nanoTime);

        Date date1 = new Date(currentTimeMillis);
        System.out.println(sdf.format(date1));

        Date date2 = new Date(nanoTime);
        System.out.println(sdf.format(date2));

        long day1 = currentTimeMillis/1000/60/60/24; // ms/초/분/시
        System.out.println(day1);
        long day2 = nanoTime/10000/60/60/24; // ns/초/분/시
        System.out.println(day2);

        // 날짜 계산
        LocalDate localDate1 = LocalDate.now();
        System.out.println(localDate1);
        LocalDate localDate2 = localDate1.minusDays(day1);
        System.out.println(localDate2);
    }
}
```

결과

```
1633575725687
10735589735100
2021-10-07 12:02:05.687
2310-03-14 09:15:35.100
18907
12425
2021-10-07
1970-01-01
```