객체지향설계및프로그래밍 실습과제#2-2 진행 순서

Eclipse를 켜고 능숙한 솜씨로 새 Java project, Program class, main()을 만든 다음, 아래에 나열된 순서에 따라 차근차근 코드를 작성해 봅시다.

주의: 이 문서는 여러분이 변수 이름 등을 하나도 안 틀리고 똑같이 작성한다는 것을 가정하고 작성되어 있습니다. 자신만의 멋진 이름을 만들고 싶겠지만, 이 문서의 내용을 함께 진행하는 동안에는 가급적 아래 내용 그대로 틀리지 않고 적도록 노력해 주세요.

## 주어진 길이를 갖는 새 배열을 만들고 랜덤 값으로 초기화해 반환하는 메서드 만들기

* 먼저, main() 위에 아래의 코드를 추가하여 새 메서드를 만들어 봅시다:

|  |
| --- |
| static int[] MakeData(int length) { } |

* + 보고 치지 말고 복붙한 다음 Ctrl + Shift + F를 쓰는 것이 몇 만 배는 더 편리합니다.
* (메서드 중괄호 안에서)랜덤 값을 쓰기 위해 Random을 입력하고 Ctrl + Space를 눌러 봅시다.
  + java.util에 있는 Random class를 선택하고 엔터 키를 쳐 줍니다.  
    이전에 Scanner 할 때 그랬듯 맨 위에 import 한 줄이 생기면 성공입니다.
* 이제 Random 형식 변수 rand를 선언하고 새로운 Random 인스턴스로 초기화해 봅시다.
  + 다 완성한 코드는 아래와 같습니다. Ctrl + Space를 중간중간 눌러 주면 간단히 적을 수 있습니다.

|  |
| --- |
| Random rand = new Random(); |

* 이제 rand에 점을 찍으면 여러 가지 랜덤 값들을 꺼내 쓸 수 있습니다.
* 이번에는 새로운 int 배열을 만듭니다. 다음 줄에 아래 코드를 추가해 봅시다:

|  |
| --- |
| int[] result = new int[length]; |

* + 위의 코드는 길이가 length인 새로운 int 배열을 만드는 코드입니다. C랑 약간 다르긴 하지만 눈으로 보고 이해하는 데는 크게 어렵지 않음을 알 수 있습니다.
* 이제 for문을 자동 완성 기능으로 하나 만들어 봅시다.
  + 여러분이 잘 따라왔다면 for문의 조건식에 적절한 식( i < length 또는 i < result.length )이 들어 있는 것을 볼 수 있을 것입니다. 이처럼 for문 자동 완성 기능은 배열 조작에 필요한 코드를 함께 작성해 줍니다.
* 방금 만든 for문 안에 아래 코드를 추가해 봅시다:

|  |
| --- |
| result[i] = rand.nextInt(100); |

* + nextInt()는 주어진 상한값 미만의 랜덤 값을 반환하는 메서드입니다.
* 이제 이 for문이 다 돌면 배열의 모든 칸이 랜덤하게 채워집니다. 이제 완성된 배열을 반환하는 일만 남았군요! 적절한 return문을 추가해서 구현을 완료해 봅시다.
* 여기까지 다 수행하면 MakeData()의 전체 코드는 아래와 같이 완성됩니다:

|  |
| --- |
| static int[] MakeData(int length)  {  Random rand = new Random();    int[] result = new int[length];    for ( int i = 0; i < length; ++i )  {  result[i] = rand.nextInt(100);  }    return result;  } |

* Random class나 nextInt()에 대해 더 알고 싶으면 각 이름에 마우스를 갖다 대 보세요!

## 입력받은 배열의 모든 수에 1을 더한 새로운 배열을 반환하는 메서드 만들기

* 이번에는, 위의 for문을 응용하여 제목에 적힌 동작을 수행하는 AddOne()을 만듭니다.  
  아래 코드를 MakeData() 바로 아래에 추가해 봅시다:

|  |
| --- |
| static int[] AddOne(int[] arr) { } |

* 이전과 동일한 방법으로, AddOne() 안에서 int 배열 result를 생성해 봅시다.  
  new 부분에 들어가는 길이 값은 arr.length를 이용하면 됩니다.
  + Java에서는 배열 이름에 점을 찍어서 몇 가지 추가 기능을 쓸 수 있습니다.
* 능숙하게 for문 자동 완성을 수행하고, 아마도 익숙할 ‘i번째 것을 i번째에’ 할당식을 적어 줍니다.
  + 과제#2-1에서 만들었던 초보자용 for문 말고 일반 array용 for문을 쓰면 편합니다.
  + 물론, 우리는 지금 ‘1씩 더하는’ 메서드를 만들고 있으니 그 코드를 빼먹으면 안 되겠지요.
* 다 되었으면 result를 반환해 줍시다.
* 여기까지 진행하면 AddOne()은 다음과 같이 완성됩니다:

|  |
| --- |
| static int[] AddOne(int[] arr)  {  int[] result = new int[arr.length];    for ( int i = 0; i < arr.length; i++ )  {  result[i] = arr[i] + 1;  }    return result;  } |

## 입력받은 배열의 모든 수를 더한 결과를 담는 1칸짜리 배열을 반환하는 메서드 만들기

* 이번엔 약간 변형된 for문을 사용해서, 제목에 적힌 대로 동작하는 Sum()을 만듭니다.  
  방금 완성한 AddOne() 다음 줄에 아래 코드를 추가해 봅시다:

|  |
| --- |
| static int[] Sum(int[] arr) { } |

* 새 배열은 천천히 만들고, 일단 합을 계산할 int sum = 0; 한 줄을 추가해 쓰도록 합시다.
* 이번에는 for 말고 fore까지만 친 다음에 Ctrl + Space를 눌러 봅시다. foreach가 선택되면 엔터 키를 쳐서 foreach문을 완성해 봅시다.
* foreach문은 for문과 유사하게 생겼지만 약간 사용법이 다릅니다. 우선 아래와 같이 전체 구문을 완성해 보도록 합시다:

|  |
| --- |
| for ( int i : arr ) {  sum += i; } |

* for문의 경우 인덱스 값을 우리가 직접 배열 대괄호에 넣어서 원하는 값을 꺼내 옵니다. foreach문은 이를 간소화하여 ‘arr 안에 있는 각 int값에 대해’, ‘그 값 하나하나를 i라고 이름 붙여 사용’할 수 있게 해 줍니다. 따라서 위의 코드는 아래 for문과 동일한 효과를 가집니다:

|  |
| --- |
| for ( int idx = 0; idx < arr.length; idx++ )  sum += arr[idx]; |

* foreach문은 for문과 동일한 작업을 좀 더 간편하게 적을 수 있는 좋은 친구입니다. 다만 foreach문은 ‘i번째’라는 개념을 쓸 수 없기 때문에 AddOne()처럼 한 칸 한 칸 직접 들어 옮겨야 하는 상황에서는 사용할 수 없습니다.
  + 왜 이렇게 만들어 놓았는지는 이해하기 조금 어려우니, 일단 오늘은 ‘제한적이지만 편하게 쓸 수 있는 멋진 차선책’ 정도로만 생각하고 가도록 합시다.
* 여튼, 이제 sum에는 우리가 구하고자 한 값이 들어 있습니다. 새로운 1칸짜리 배열을 만들고 그 안에 sum의 값을 담은 다음 반환하면 전체 메서드 구현이 끝납니다:
* 여기까지 진행하면 Sum()은 아래와 같이 완성됩니다:

|  |
| --- |
| static int[] Sum(int[] arr)  {  int sum = 0;    for ( int i : arr )  {  sum += i;  }    int[] result = new int[1];    result[0] = sum;    return result;  } |

## 이후 할 일

이제, 위에서 만들어 둔 각 메서드들을 적절히 호출하여, 원본 배열, 각 연산이 끝난 후의 결과 배열을 콘솔 창에 예쁘게 출력해 주도록 main()을 구성해 봅시다. 여러분이 원한다면 PrintData()와 같은 메서드를 추가로 구현해서 배열 출력에 사용해도 좋습니다. 구성이 끝나면 제대로 잘 실행되나 한 번 테스트해 본 다음, AddOne()과 Sum()을 응용하여 자신만의 배열 연산 메서드를 만들어 보도록 합시다.