객체지향설계및프로그래밍 실습과제#3-2 진행 순서

오늘도 역시 Eclipse를 켜고 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.

## Toy class를 (default package) 안에 만들고 생성자 추가해 주기

* Eclipse의 Package Explorer(보통 코드 창 왼쪽에 있는 것)을 보면,  
  Program.java가 (default package) 안에 들어 있는 것을 확인할 수 있습니다.  
  이 package를 우클릭한 다음 평소처럼 새 클래스를 만들고 이름을 Toy로 정해 줍니다.
* Toy class 안에 아래의 코드를 추가해 봅시다:

|  |
| --- |
| int numberOfHeads; int numberOfArms; int numberOfLegs; |

* + 평소와는 다르게, 이 ‘필드’들에는 static을 붙이지 맙시다.
* 방금 추가한 코드 아래에서 Ctrl + Space를 누르고 즉시 엔터 키를 쳐 봅시다.  
  아래와 같은 코드가 뜨면 성공입니다:

|  |
| --- |
| public Toy() {   } |

* + 이렇게 만들어진, return type 없고 이름이 클래스명과 동일한 메서드를 ‘생성자’라 부릅니다.
* 생성자는 이 클래스의 인스턴스(new 하면 나오는 실제 물건)에 대한 초기 상태를 지정해 주기 위해 사용되는 메서드입니다. 우리의 장난감은 만들어질 때 머리 하나, 팔 두 개, 다리 두 개로 구성되어 있는 지극히 정상적인 장난감이어야 합니다. 이러한 초기 조건을 만족할 수 있도록 방금 만든 생성자 안에 적절한 구문을 추가하여 각 필드들이 적절한 값을 갖도록 설정해 줍시다.

## Toy class에 toString() 추가해 주기

* 지난 시간에 배열을 sysout으로 출력했더니 이상한 숫자가 나오는 것을 볼 수 있었습니다. 이는 Java 배열이 ‘문자열로 출력’을 위한 적절한 기능을 제공하고 있지 않기 때문입니다. 이 단원에서는 우리가 방금 만든 Toy class가 적절한 문자열 표현을 가질 수 있도록 toString()을 추가해 봅니다.
* 방금 만든 생성자 코드 아래에서 toString을 입력하고 엔터 키를 쳐 봅시다.  
  아래와 같은 코드가 뜨면 성공입니다:

|  |
| --- |
| @Override  public String toString()  {  // TODO Auto-generated method stub  return super.toString();  } |

* + @Override나 super 키워드와 같은 생소한 친구들은, 일단 지금은 신경쓰지 않아도 좋습니다.
* 이제 자동 완성된 toString()의 내용물을 모두 지우고, 아래의 코드를 추가해 봅시다:

|  |
| --- |
| return String.format("머리 %d개, 팔 %d개, 다리 %d개 달린 장난감",  numberOfHeads, numberOfArms, numberOfLegs); |

* String.format()은 C의 printf()와 유사한 규칙을 통해 ‘서식에 맞는 새로운 문자열’을 만들어 반환해 주는 메서드입니다. 위의 코드는 현재 이 인스턴스(장난감)이 가진 머리/팔/다리 수를 반영한 새로운 문자열을 만들어 반환하고 있습니다.
* 이제 Program.java의 main()으로 돌아와서 아래의 작업을 수행해 봅시다:
  + Toy class의 새로운 인스턴스를 만들고,
  + sysout을 통해 해당 인스턴스를 출력해 봅시다.
* 여러분이 예상한 대로, 콘솔 창에 적절한 장난감 설명이 보이면 성공입니다!  
  여기까지 진행하면 실습과제#3-2가 모두 끝납니다.

## 끝?

네, 정말 끝입니다. 오늘 살펴 본 내용들은 앞으로 한 학기 동안 몇 백 번은 더 써 볼 내용들이고, 이제 막 시작한 시점에서는 ‘이거 이렇게 해서 엔터 치면 만들 수 있구나’까지만 알아 두는 것으로 일단은 충분합니다.

‘하지만 난 고작 이 정도 해 보려고 5만원 내고 이 수업 온 게 아니에요’라고 생각하는 친구들은, 이어지는 실습과제#3-3을 도전해 보거나, 아래의 추가 작업들을 수행해 보세요. 특히, ‘따라 하라 해서 따라 했는데 난 이걸 좀 더 이해하고 가고 싶어요’ 하는 학생들은, 아래의 내용 또는 실습과제#3-3이 큰 도움이 될 것입니다!

* main()에서 새로운 Toy 인스턴스 두 개를 생성한 다음,  
  각각의 이름에 점을 찍어 머리/팔/다리 수를 바꾸어 보고  
  이들을 sysout으로 각각 출력해 보세요.
* 위의 작업이 완료되면,  
  Toy.java에 있는 Toy class의 모든 필드들을 static 필드로 바꾼 다음 다시 실행해 보세요.
  + 차이가 느껴지나요? Java의 static 키워드는 C의 static 키워드와 반쯤 동일한 효과를 냅니다!
* 자신이 원하는 값으로 머리/팔/다리 수를 바꾸기 위한 public Fix(int, int, int) 메서드를 Toy class에 추가해 보세요. main()이 직접 Toy 인스턴스의 필드를 건드리는 대신 이 메서드를 통해 해당 값들을 바꾸도록 변경해 보세요.
* 변경이 끝나면 이제 Toy 인스턴스의 모든 필드를 private으로 바꾸고 다시 실행해 보세요.  
  오류 없이 정상적으로 실행되면 성공입니다.
* 이제 거꾸로, 필드가 private인 상태에서 Toy 인스턴스에 . 을 찍어 해당 필드가 ‘안 보인다’는 것을 확인해 보세요.
  + private 키워드는 이처럼, 클래스 밖에서 해당 요소를 볼 수 없도록 만들어 주는 키워드입니다.
* 결과가 지극히 당연하겠지만, 생성자를 private으로 설정하고 오류가 나는 것을 관찰해 보세요.
  + 생성자가 private이면 ‘클래스 자신’밖에 새 인스턴스를 만들 수 없습니다. 이런 걸 왜 쓰는지 궁금하다면, 실습과제#3-3에 도전해 보면 조금은 감이 올 수 있겠군요.

이 정도 추가로 살펴 보면 오늘 해 볼 만한 것들은 모두 완료한 듯 하군요!