10. Debugging with Eclipse

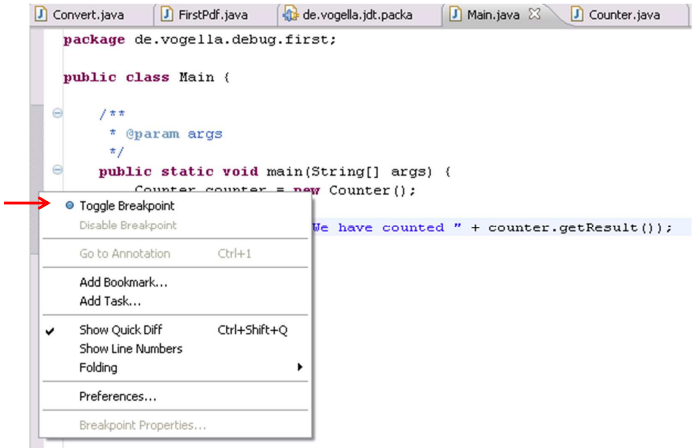
Contents

개요

- **What is debugging?**
  - 프로그램을 대화식으로(interactively) 실행하면서, 소스 코드와 변수의 내용을 살펴보면서 실행 흐름을 추적하고 오류를 찾아 수정함
- **Debugging support in Eclipse**
  - 프로그램을 *Debug mode*로 실행시킴
  - 소스 코드에 breakpoints 를 설정하여 그 위치에서 코드의 실행을 중지시킴
  - *Debug Perspective*를 통해 프로그램의 실행 과정을 제어하고 변수들의 상태를 조사함

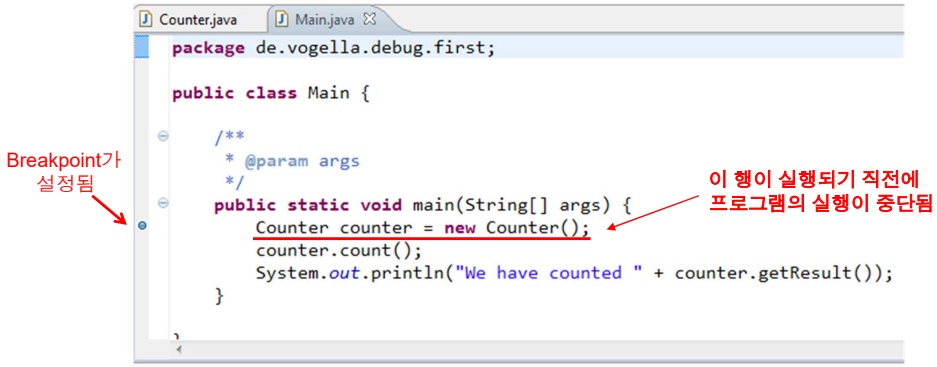
Debugging in Eclipse

- **Setting Breakpoints**
  - 특정 행에 line breakpoints를 생성하기 위해 소스 코드 편집창의 좌측 가장자리를 마우스 우클릭(right-click)하고 *Toggle Breakpoint*를 선택함
    - 또는 그 지점을 double-click함



Debugging in Eclipse

– 예:



Debugging in Eclipse

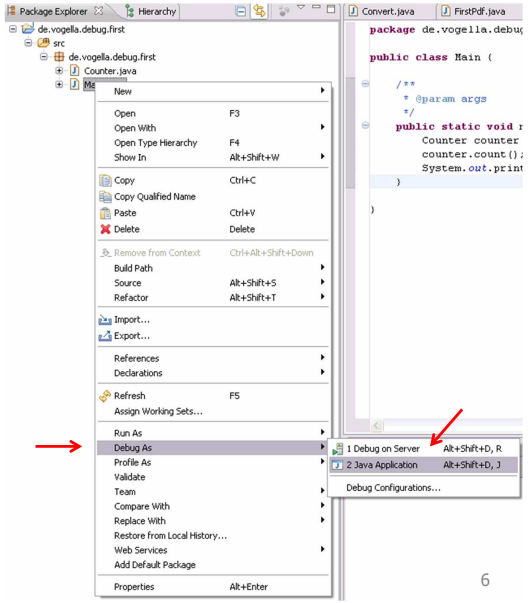
◆ **Starting the Debugger**

– 프로그램을 디버깅하기 위해, main 메소드를 포함하는 Java 클래스 파일을 선택하고 마우스 우클릭 한 후, *Run As* 대신 *Debug As → Java Application* (웹 애플리케이션인 경우는 *Debug on Server*) 을 선택함

▪ Debug mode로 실행

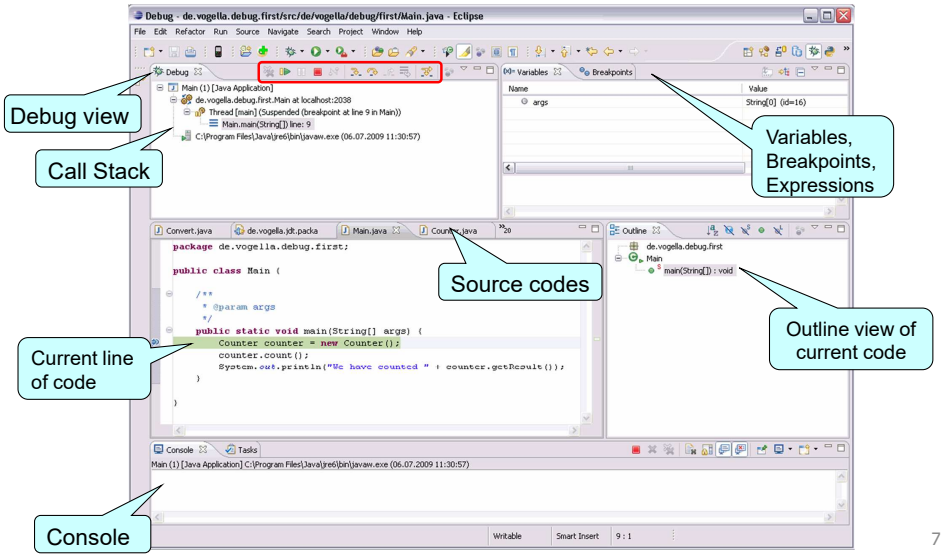
– Breakpoint를 미리 생성하지 않으면 프로그램이 정상적으로 (끝까지) 실행됨

▪ 즉, 프로그램을 디버깅하기 위해서는 하나 이상의 breakpoints를 생성해야 함



Debugging in Eclipse

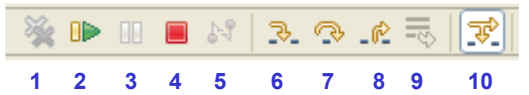
– Debug mode를 최초 실행 시, debug perspective로 전환 선택



Debugging in Eclipse

◆ **Stepping through codes**

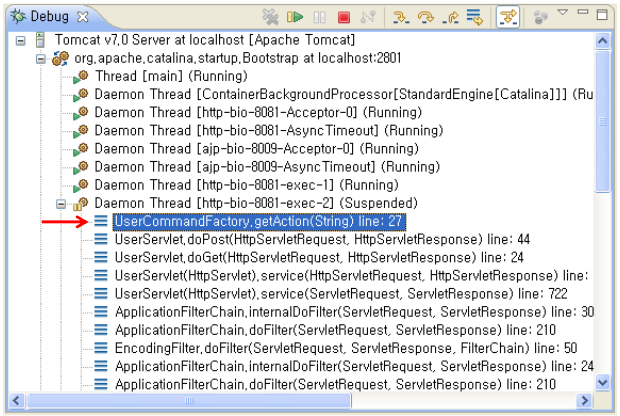
– In Debug View



1. **Remove All Terminated Launches** – 중단된 debug session을 모두 종료함
2. **Resume** – breakpoint를 만나거나 thread가 끝날 때까지 실행을 계속함
3. **Suspend** – 실행중인 thread 중단시킴
4. **Terminate** – 선택된 thread의 실행을 종료함
5. **Disconnect** – remote debugging session으로부터 연결을 끊음
6. **Step Into** – method 안으로 들어가서 첫 번째 코드 행을 실행함
7. **Step Over** – 현재 method의 다음 코드 행을 실행함
8. **Step Return** – 현재 method의 끝까지 실행을 계속함 (return할 때까지)
9. **Drop to Frame** – 이전의 stack frame으로 되돌아 감
10. **Use Step Filters** – filter out되지 않은 다음 코드 행까지 실행을 계속함

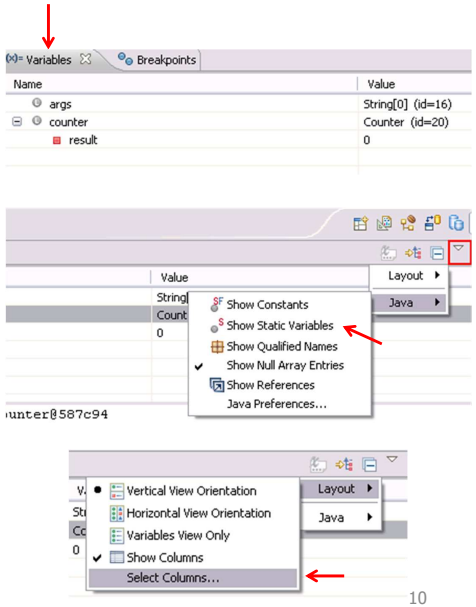
Debugging in Eclipse

- **Call Stack**
  - call stack은 현재까지 실행된 프로그램의 일부, 즉 메소드들과 그것들의 호출 관계를 보여줌
  - Debug View에서 현재의 call stack을 살펴볼 수 있음



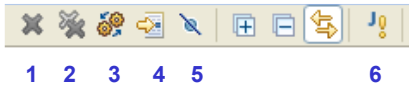
Debugging in Eclipse

- **Evaluating variables**
  - *Variables View*는 클래스의 필드나 현재의 stack으로부터 접근 가능한 지역 변수들을 보여줌
  - 정적 변수(static variables)를 출력하기 위해서는 View menu에서 *Java → Show Static Variables* 선택
  - drop-down menu를 통해 화면에 출력될 컬럼들을 선택할 수 있음
    - 예: 각 변수의 실제 type을 나타내기 위해 *Layout → Select Columns... → Actual Type* 선택



Advanced Debugging

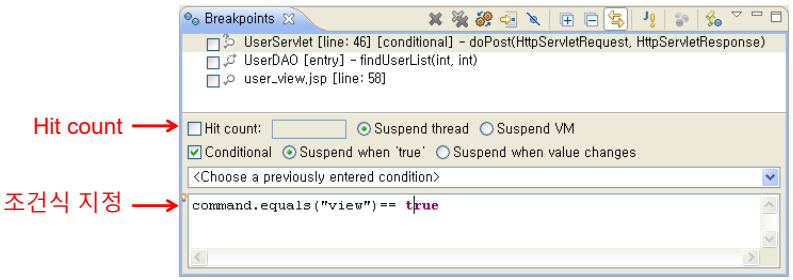
- **Breakpoints 제어**
  - In Breakpoints View



1. Remove Selected Breakpoints
2. Remove All Breakpoints
3. Show Breakpoints Supported by Selected Target
4. Go to File for Breakpoint
5. Skip All Breakpoints
   - ✓ 모든 breakpoint들을 임시적으로 비활성화 함
   - ✓ 이 button을 다시 선택할 경우 breakpoint들이 다시 활성화 됨
6. Add Java Exception Breakpoint
   - ✓ 특정 예외가 발생할 때 프로그램이 중단되도록 breakpoint를 설정

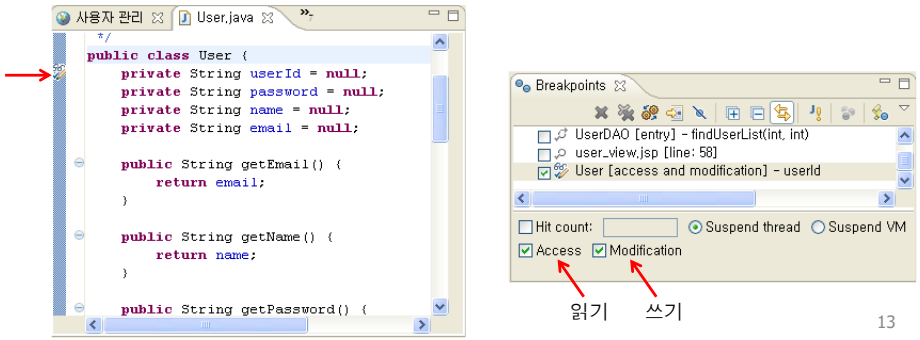
Advanced Debugging

- **Breakpoint 속성**
  - breakpoint를 정의한 후, *Breakpoints View*에서 breakpoint에 관한 속성들을 설정할 수 있음
    - breakpoint를 선택 후 *right-click → Breakpoint Properties*
  - breakpoint의 활성화 시점을 제한하는 조건을 정의할 수 있음
    - 예: breakpoint가 12번 이상 도달한 후에 활성화 되도록 하거나(Hit Count 이용), 또는 조건식(conditional expression)을 정의하여 그 조건이 참이 될 때만 breakpoint에서 실행이 중단되도록 함



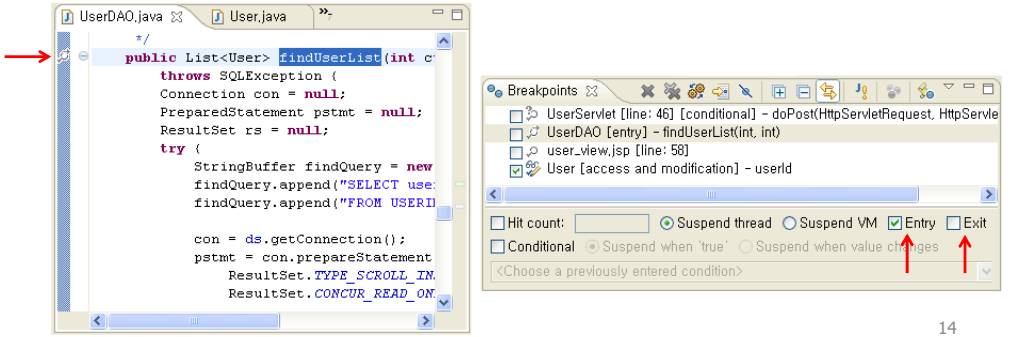
Advanced Debugging

- **Watchpoint**
  - 필드에 설정된 breakpoint
    - 그 필드 값이 읽히거나 변경될 때 실행이 중단됨
  - 생성 방법: 필드 선언(declaration) 옆의 왼쪽 가장자리를 double-click 함
  - watchpoint의 속성으로, 읽기(Access) 또는 쓰기(Modification) 접근에 대해 선택적으로 실행이 중단되도록 설정 가능



Advanced Debugging

- **Method Breakpoint**
  - 메소드에 설정된 breakpoint
    - 그 메소드가 호출되어 실행될 때 중단됨
  - 생성 방법: 메소드의 선언부(header) 옆 왼쪽 가장자리를 double-click 함
  - 메소드 안으로 들어가거나 메소드를 벗어날 때 프로그램이 중단되도록 설정 가능



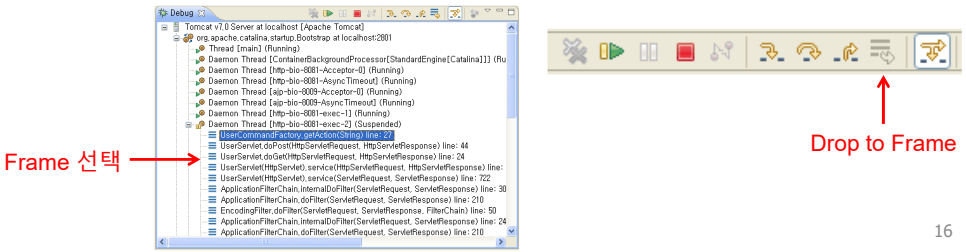
Advanced Debugging

- **Step Filter**
  - 디버깅 시 skip할 package들을 지정 가능
  - 예를 들어, 프로그램에서 test framework(library)을 이용할 때 그것에 속한 클래스들 안으로 진입하지 못하도록 설정할 수 있음
  - *Window → Preferences → Java → Debug → Step Filtering* menu를 통해 지정

- **Hit Count**
  - 모든 breakpoint에 대해 hit count 속성을 설정 가능
  - Hit count에 지정된 횟수만큼 breakpoint에 도달했을 때 프로그램 실행이 중단됨

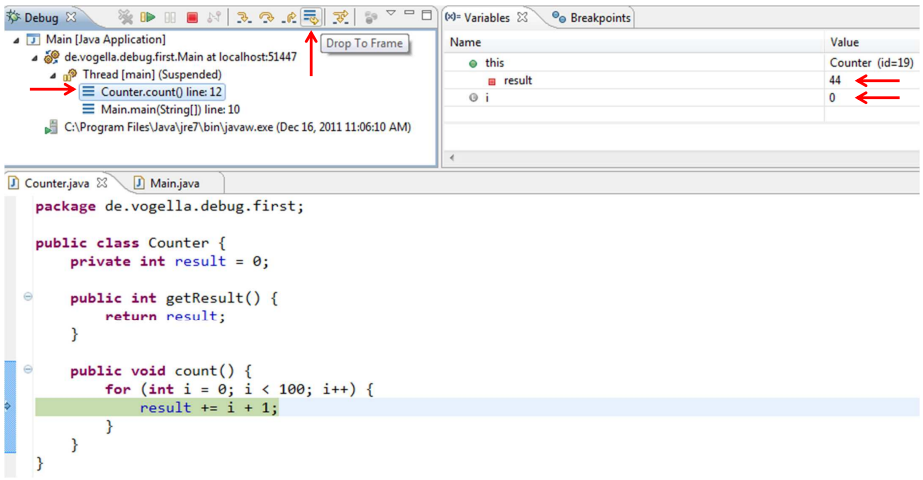
Advanced Debugging

- **Drop to frame**
  - 디버깅 중 call stack 안의 특정 level(frame)을 선택하여 그 위치부터 JVM이 재시작하도록 설정
    - 프로그램의 일부를 재실행할 수 있음
  - stack 안의 특정 level을 선택한 후 *Debug View*의 toolbar에서 *Drop to Frame* button을 click
  - 주의: 기 실행된 코드에 의해 변경된 결과는 그 상태를 유지함
    - 이전 frame으로 돌아갈 때 (전역)변수 또는 외부 데이터(예: files, databases)에 대한 변경은 reset되지 않음



Advanced Debugging

- 예: Counter.count() 실행 frame의 "for" loop를 처음부터 재시작할 수 있음
  - 그러나 "result" 변수 값은 reset되지 않음



References

- Java Debugging with Eclipse – Tutorial
  - http://www.vogella.com/articles/EclipseDebugging/article.html