Chapter 5. 형식 맞추기(Formatting)

목적

- 시간에 따라 코드는 쉽게 바뀌어도 스타일은 쉽게 바뀌지 않는다.
- 유지보수에 지속적인 영향

수직 형식 맞추기

신문기사처럼 작성하라

- 첫 문단은 전체 기사를 요약한다
- 내려가면서 세부적인 사실이 들어난다. (Top down)

개념은 빈 행으로 분리하라

- 생각과 생각 사이는 빈 행으로 구분한다.

```
package fitnesse.wikitext.widgets;

import java.util.regex.*;

public class BoldWidget extends ParentWidget {
  public static final String REGEXP = "'''.+?'''";
  private static final Pattern pattern =
Pattern.compile("'''(.+?)'''",
    Pattern.MULTILINE + Pattern.DOTALL
  );

  public BoldWidget(ParentWidget parent, String text)
throws Exception {
  super(parent);
    Matcher match = pattern.matcher(text);
    match.find();
    addChildWidgets(match.group(1));
```

```
  }

  public String render() throws Exception {
    StringBuffer html = new StringBuffer("<b>");
    html.append(childHtml()).append("</b>");
    return html.toString();
  }
}
```

세로 밀집도

- 밀접한 개념은 세로로 가까이 두어야 한다.
- 변수는 사용하는 위치에 최대한 가까이 선언한다.

```
public int countTestCases() {
  int count= 0;
  for (Test each : tests)
    count += each.countTestCases();
  return count;
}
```

- 인스턴스 변수는 클래스의 처음이나 마지막에 둔다.
- 한 함수가 다른 함수를 호출한다면 두 함수는 세로로 가까이 배치한다.

```
public class WikiPageResponder implements
SecureResponder {
  protected WikiPage page;
  protected PageData pageData;
  protected String pageTitle;
  protected Request request;
  protected PageCrawler crawler;

  public Response makeResponse(FitNesseContext
context, Request request) throws Exception {
```

```
    String pageName = getPageNameOrDefault(request,
"FrontPage");

    loadPage(pageName, context);
    if (page == null)
      return notFoundResponse(context, request);
    else
      return makePageResponse(context);
  }

  private String getPageNameOrDefault(Request
request, String defaultPageName) {
    String pageName = request.getResource();
    if (StringUtil.isBlank(pageName))
      pageName = defaultPageName;
    return pageName;
  }

  protected void loadPage(String resource,
FitNesseContext context) throws Exception {
    WikiPagePath path = PathParser.parse(resource);
    crawler = context.root.getPageCrawler();
    crawler.setDeadEndStrategy(new
VirtualEnabledPageCrawler());
    page = crawler.getPage(context.root, path);
    if (page != null)
      pageData = page.getData();
  }

  private Response notFoundResponse(FitNesseContext
context, Request request) throws Exception {
    return new
NotFoundResponder().makeResponse(context, request);
  }

  private SimpleResponse
makePageResponse(FitNesseContext context) throws
Exception {
    pageTitle =
```

```
PathParser.render(crawler.getFullPath(page));
    String html = makeHtml(context);
    SimpleResponse response = new SimpleResponse();
    response.setMaxAge(0);
    response.setContent(html);
    return response;
  }
...
```

- 개념적으로 유사하다면 가까이 둔다.

```
public class Assert {
  static public void assertTrue(String message,
boolean condition) {
    if (!condition)
      fail(message);
  }

  static public void assertTrue(boolean condition) {
    assertTrue(null, condition);
  }

  static public void assertFalse(String message,
boolean condition) {
    assertTrue(message, !condition);
  }

  static public void assertFalse(boolean condition) {
    assertFalse(null, condition);
  }
...
```

- *세로 순서
  - Top-down, Bottom-up
  - 책에선 top-down을 권고

가로 형식 맞추기

가로 공백 밀집도

- 할당문은 왼쪽과 오른쪽을 구분한다.
- 함수와 인수는 서로 밀접하다

```
private void measureLine(String line) {
  lineCount++;
  int lineSize = line.length();
  totalChars += lineSize;
  lineWidthHistogram.addLine(lineSize, lineCount);
  recordWidestLine(lineSize);
}
```

- 승수 사이는 공백이 없다. 곱셈은 우선순위가 가장 높으므로

```
public static double root1(double a, double b, double
c) {
  double determinant = determinant(a, b, c);
    return (-b + Math.sqrt(determinant)) / (2*a);
  }
  public static double root2(int a, int b, int c) {
    double determinant = determinant(a, b, c);
    return (-b - Math.sqrt(determinant)) / (2*a);
  }
  private static double determinant(double a, double
b, double c) {
    return b*b - 4*a*c;
  }
}
```

- *Lint오류가 생기는 경우도 있음

가로 정렬

```
public class FitNesseExpediter implements
ResponseSender {
  private Socket          socket;
  private InputStream     input;
  private OutputStream    output;
  private Request         request;
  private Response        response;
  private FitNesseContext context;
  protected long          requestParsingTimeLimit;
  private long            requestProgress;
  private long            requestParsingDeadline;
  private boolean         hasError;

public FitNesseExpediter(Socket s,
 FitNesseContext context) throws Exception {
  this.context = context;
  socket = s;
  input = s.getInputStream();
  output = s.getOutputStream();
  requestParsingTimeLimit = 10000;
}
```

- 코드가 엉뚱한 부분을 강조해 진짜 의도가 사라진다.
- *정렬이 필요할 정도로 목록이 길다면 문제는 목록 길이이지 정렬이 아니다.* 쪼개라.

들여쓰기

- 들여쓰는 정도는 계층에서 코드가 자리 잡은 수준에 비례한다.
- 들여쓰기를 무시하고 싶은 유혹을 이겨라

```
public class CommentWidget extends TextWidget
{
  public static final String REGEXP = "^#[^\r\n]*(?:
```

```
(?:\r\n)|\n|\r)?";
  public CommentWidget(ParentWidget parent, String
text){super(parent, text);}
  public String render() throws Exception {return "";
}
}
```

```
public class CommentWidget extends TextWidget {
  public static final String REGEXP = "^#[^\r\n]*(?:
(?:\r\n)|\n|\r)?";
  public CommentWidget(ParentWidget parent, String
text) {
    super(parent, text);
  }
  public String render() throws Exception {
    return "";
  }
}
```

팀 규칙

- 위에서 뭐라고 했든, 팀에 속한다면 자신이 선호해야 할 규직은 바로 팀 규칙이다.