1. 애플리케이션 생성(Building an application)

본 문서는 볼랜드사에서 제공하는 JBuilder 9 버전의 문서중
"Introducing JBuilder" 의 Chapter 3. Building an application를
번역한 것입니다.

이 문서는 번역시 오류, 오타 등이 있을 수 있으며, 이 문서를 사용함으로서
발생되는 문제에 대해서 번역자는 책임이 없으며, JAVA를 배우고 개발 툴로
JBuilder를 선택한 초보자를 위한 것이며, 상업적인 목적이 아닌 한 자유롭게
활용및 배포할 수 있습니다.

본 문서에 대한 문의 사항은 아래로 연락바랍니다.

- Http://wowstone.lin4u.com
- wowstone@empal.com

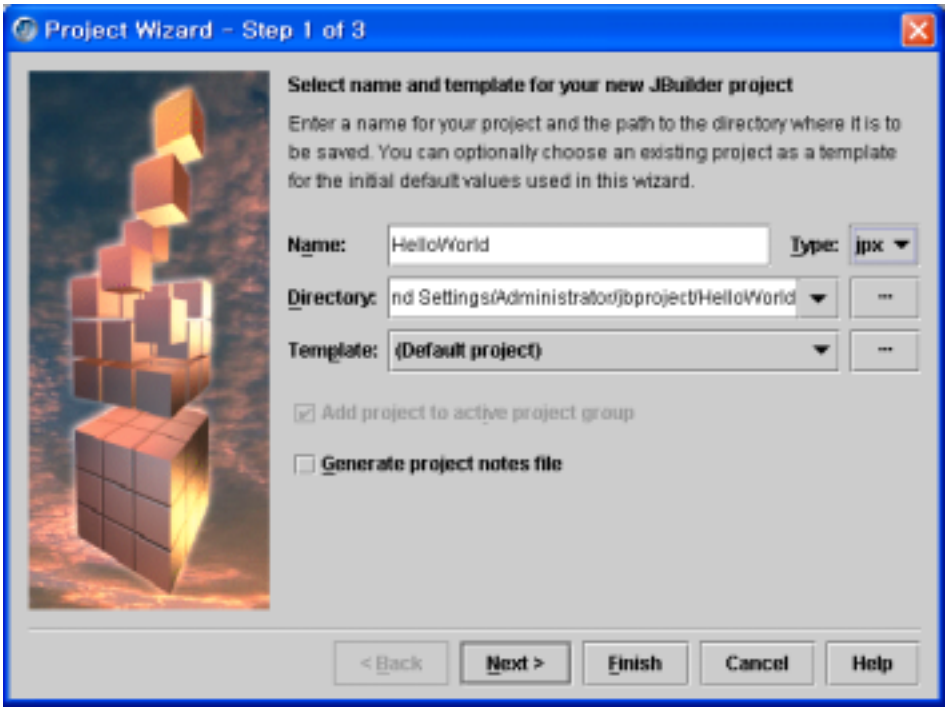
1. 애플리케이션 생성(Building an application)

많은 대부분의 개발자들은 특정 언어를 배울 때 책으로 보기보다는 실제 프로그램 소스를 작성하고 컴파일하고 실행하는 방법을 선호한다. 여기에서는 애플리케이션을 생성하기 위하여 JBuilder가 제공하는 통합개발 환경(IDE, Integrated Development Environment)을 어떻게 이용하는지 방법을 단계별로 자세하게 설명한다. 이 장에서 간단하게 만들어 볼 예제는 아래와 같은 애플리케이션을 완성하는 것이다.

2. 프로젝트 작성(Creating the project)

JBuilder을 이용하여 프로젝트를 생성하기 위해서는 먼저 작업할 프로젝트를 먼저 생성해야 한다. JBuilder는 확장자가 ".jpx"을 가진 프로젝트 파일을 사용하며, 이 파일은 애플리케이션 파일, 설정값, 프로젝트 속성(Property)등으로 구성되어 있다. 보통은 프로젝트를 생성하기 위해서는 프로젝트 마법사(Project wizard)를 사용한다.

1. 메뉴에서 [File|New Project]을 선택하여 프로젝트 마법사를 연다.



프로젝트 마법사를 열면 아래 그림과 같이 프로젝트 마법사 화면이 표시된다. 여기에 필요한 필드의 항목을 아래와 같이 수정해 준다.

항목	설정값	의미
Name	HelloWorld	프로젝트 이름
Type	jpx	프로젝트 파일 형태
Templete	Default project	

Name 항목에 프로젝트 이름을 입력하면, 똑같은 이름이 디렉토리 항목에도 들어간다. Jbuilder는 기본적으로 프로젝트 이름을 프로젝트 디렉토리 이름과 클래스를 위한 패키지(Package)이름으로 같이 사용한다. 패키지 이름은 패키지이름 작성을 위한 자바 관례에 따라 소문자로 변경된다. 프로젝트가 실제 생성되는 곳은 홈 디렉토리에 위치하고 있으며, /<home>/jbproject/ 디렉토리 이다. 윈도우의 경우에는 아래와 같이 HOMEPATH에 존재한다.

Example>
C:\>set
ALLUSERSPROFILE=C:\Documents and Settings\All Users
APPDATA=C:\Documents and Settings\Administrator\Application Data
ClusterLog=C:\WINDOWS\Cluster\cluster.log
CommonProgramFiles=C:\Program Files\Common Files
COMPUTERNAME=KUGSTONE
ComSpec=C:\WINDOWS\system32\cmd.exe

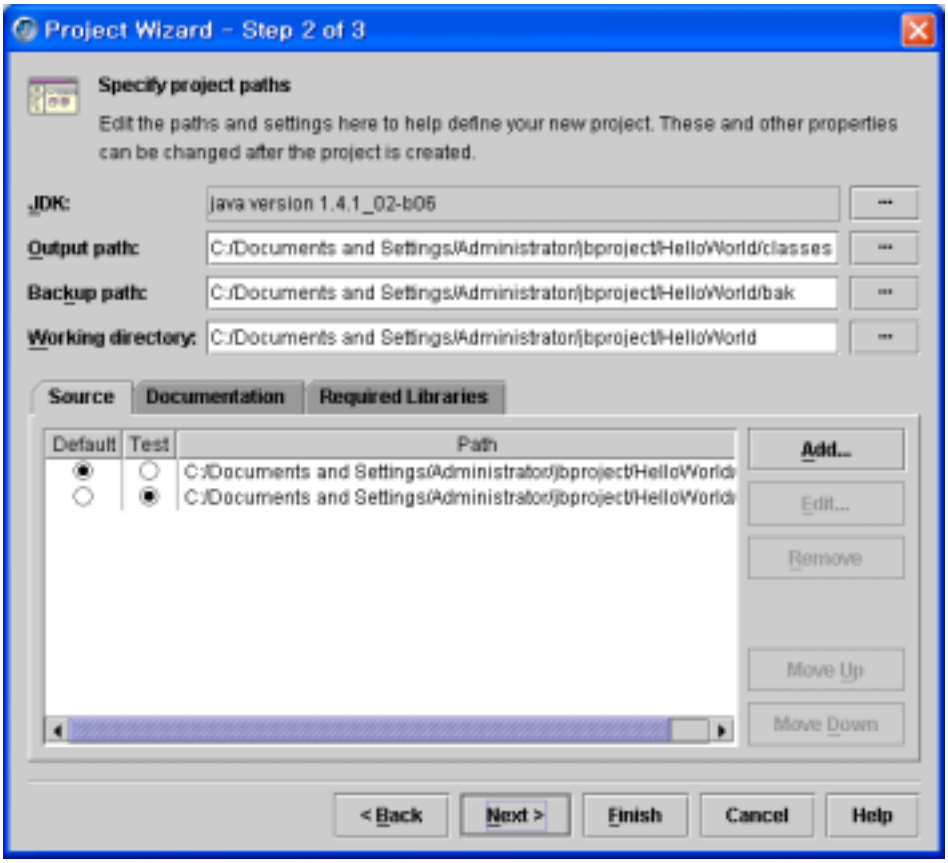
HOMEDRIVE=C:
HOMEPATH=\Documents and Settings\Administrator

LOGONSERVER=\\KUGSTONE
NUMBER_OF_PROCESSORS=1
OS=Windows_NT
Path=C:\Delphi7\Bin;C:\Delphi7\Projects\Bpl\;C:\WINDOWS\system32;C:\WINDOWS;C:\W
INDOWS\System32\Wbem;C:\DJGPP\BIN;C:\HNC
PATHEXT=.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH
PROCESSOR_ARCHITECTURE=x86
PROCESSOR_IDENTIFIER=x86 Family 15 Model 1 Stepping 2, GenuineIntel
PROCESSOR_LEVEL=15
PROCESSOR_REVISION=0102
ProgramFiles=C:\Program Files
PROMPT=PG
SESSIONNAME=Console
SystemDrive=C:
SystemRoot=C:\WINDOWS
TEMP=C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp
TMP=C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp
USERDOMAIN=KUGSTONE
USERNAME=Administrator

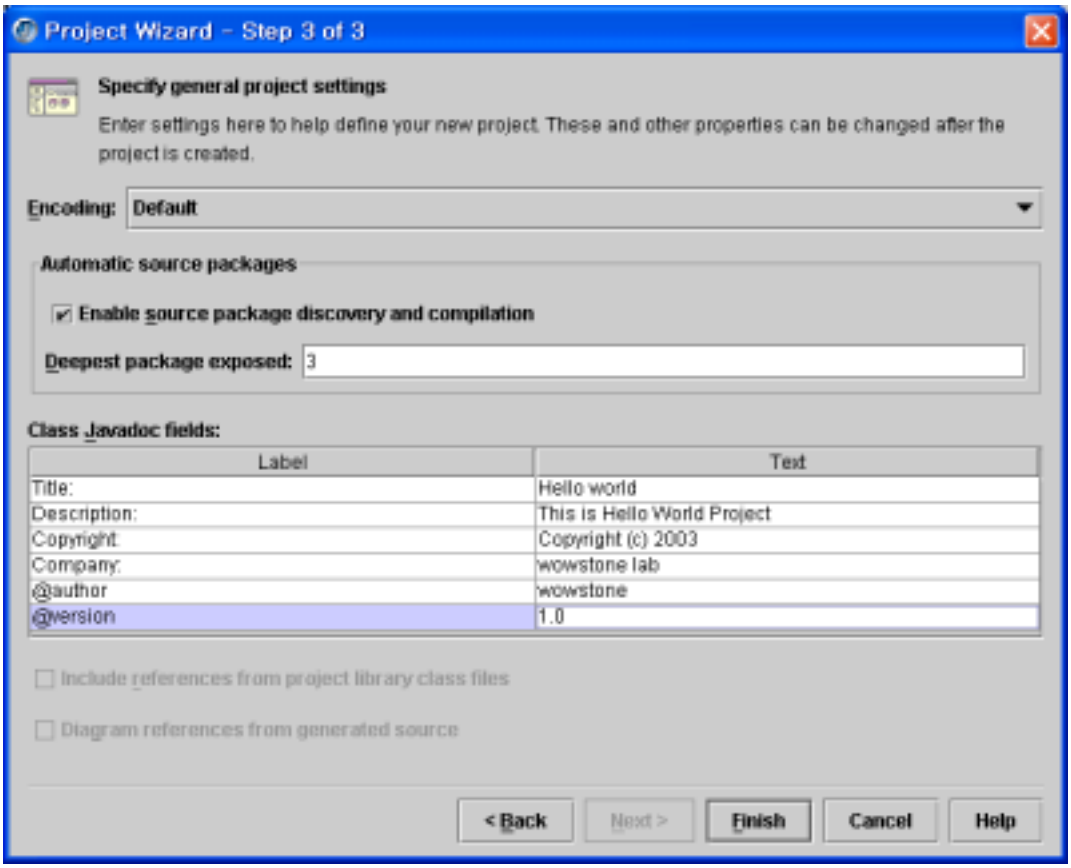
USERPROFILE=C:\Documents and Settings\Administrator
windir=C:\WINDOWS
C:\>

하단에 Generate Project Notes File 옵션은 프로젝트 설명서(Note)를 위한 HTML 파일을 생성하여 프로젝트 파일에 추가하는 것으로 여기를 체크표시를 한 후 다른 옵션들은 기본값으로 설정하고 Next 버튼을 클릭한다.

2. 아래 화면은 작업에 필요한 각종 경로(Path)을 설정하는 곳으로 모두 기본값으로 설정한 후 Next 버튼을 클릭한다. 여기에서는 class files이 어디에서 컴파일이 되는지, 그리고 project files과 source files이 저장되는 위치를 참고한다.

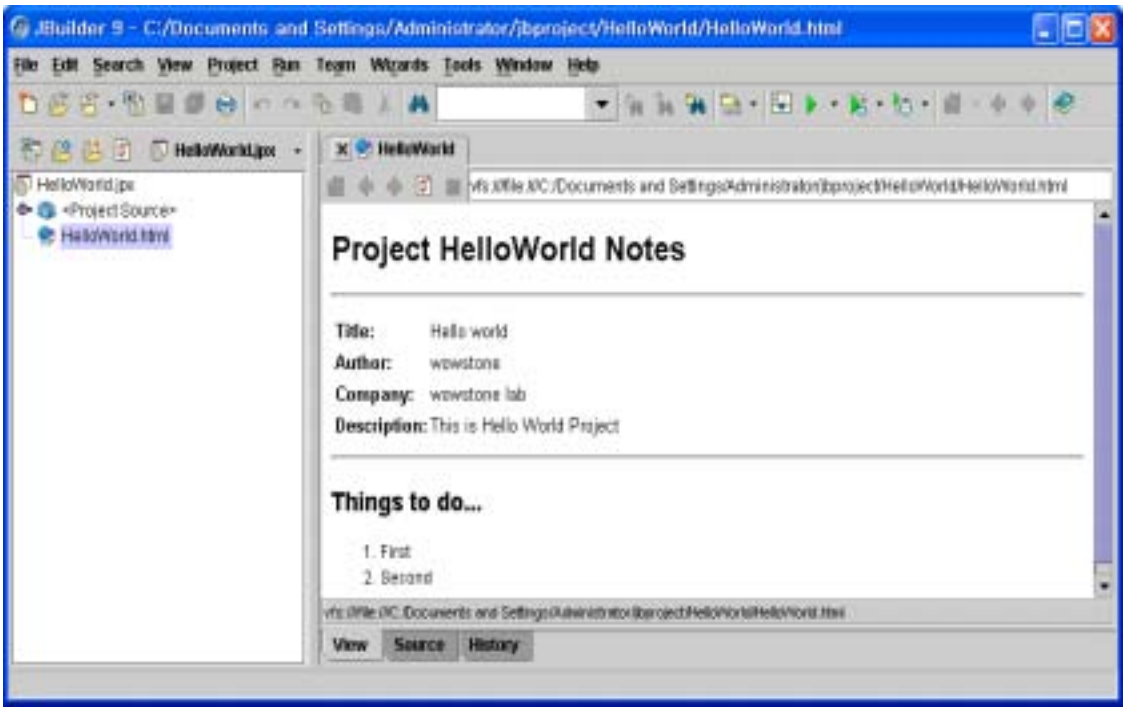


3. 아래 화면은 일반적인 프로젝트 설정값을 할당하는 것으로 Encoding과 Automatic source packages 항목은 기본값으로 놔둔다. Class Javadoc fields에는 아래와 같이 필요한 사항을 입력한다. 여기에 입력된 내용은 프로젝트 HTML 파일에 표시되며, 소스 코드에 헤더 코멘트(Header comment)로 선택적으로 나타나게 된다. 여기에서 Finish 버튼을 클릭한다.



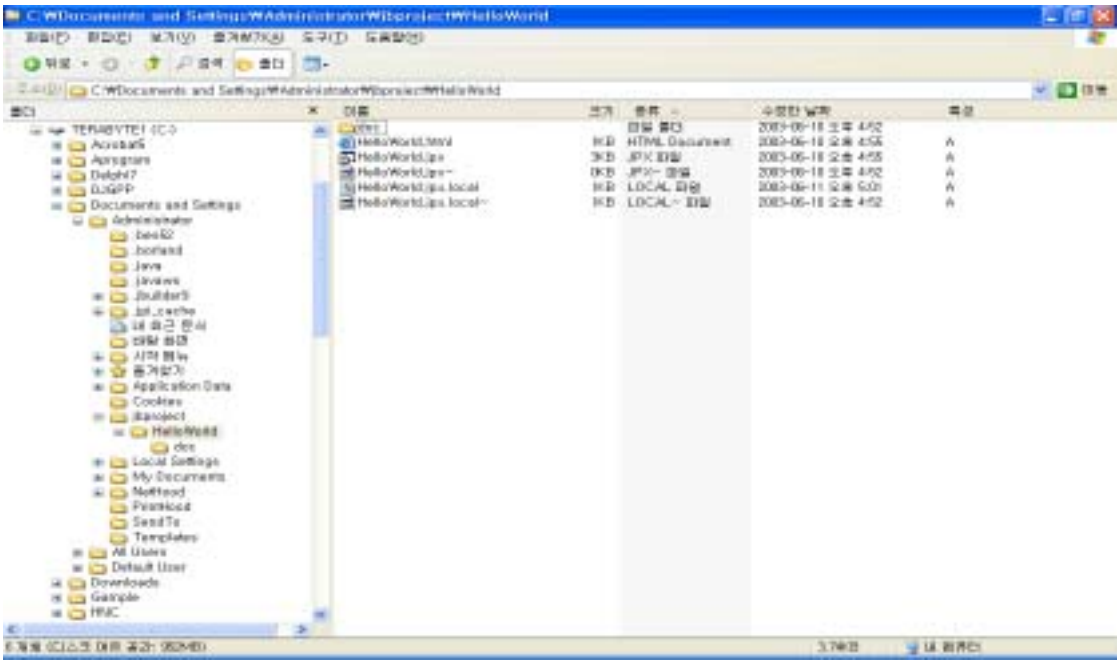
팁>
항목 입력시 해당 항목에 마우스를 더블클릭 해도 되지만 F2을 사용해도 편집할 수 있다.

4. 프로젝트 마법사가 완료되면 아래 화면과 같이 JBuilder 좌측상단에 위치하고 있는 프로젝트 패인에 HelloWorld.jpx와 HelloWorld.html의 두 개의 파일이 프로젝트 마법사에 의해 자동으로 생성된다.



5. 만약 HelloWorld.html를 더블클릭하면 프로젝트 노트파일이 AppBrowser의 중앙에 위치하고 있는 Content pane에 보여지게 된다. 위의 화면에서 보는 것과 같이 노트는 project name, author, company, description 과 같은 정보를 포함하고 있다.

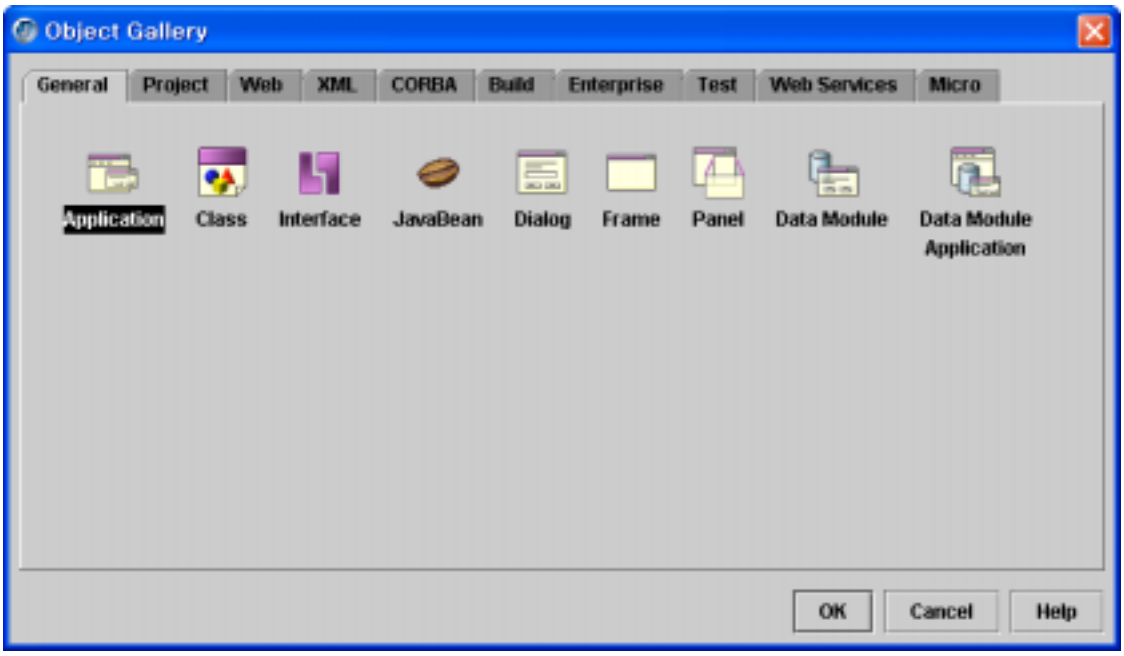
6. 아래화면은 지금까지 생성된 HelloWorld 프로젝트가 생성된 디렉토리내의 파일들을 보여주고 있다.



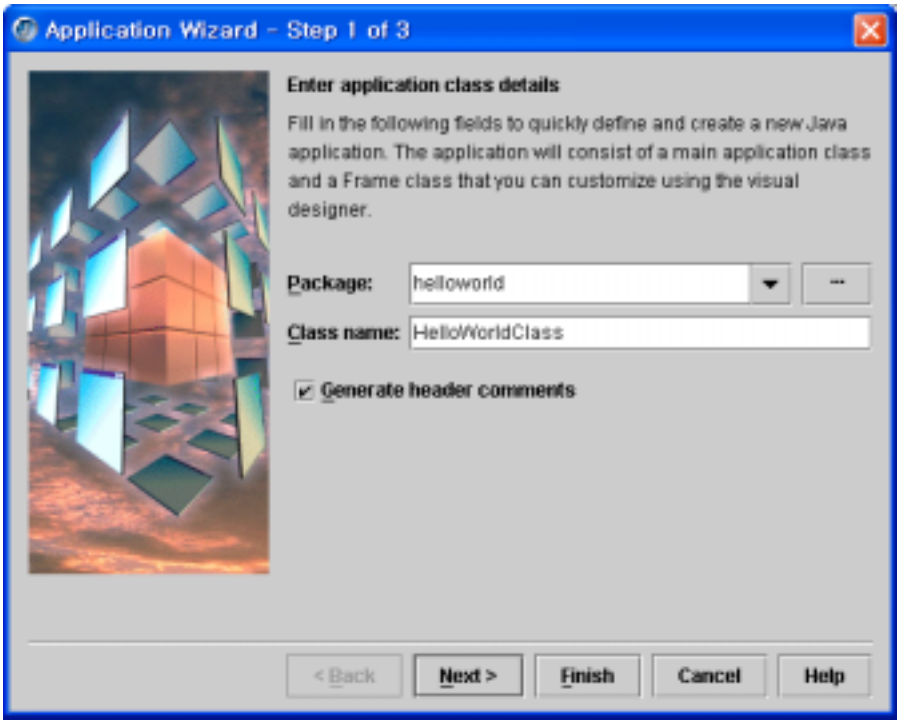
3. 소스파일 생성(Generating your source files)

애플리케이션 마법사는 여러분들이 방금 작성한 프로젝트에 java 확장자를 가진 소스파일을 생성한다. 애플리케션 마법사를 이용하여 사용자 애플리케이션을 위한 소스파일을 생성하기 위해서는 아래와 같은 단계를 이용한다.

1. 메뉴에서 [File|New]나 메인 툴바에서 [New] 버튼을 클릭하여 오젝젝트 겔러리(Object gallery)을 연다. 아래 오브젝트 겔러리 화면에서 [General] 탭을 선택하고 [Application] 아이콘을 더블클릭하여 애플리케이션 마법사를 실행시킨다.

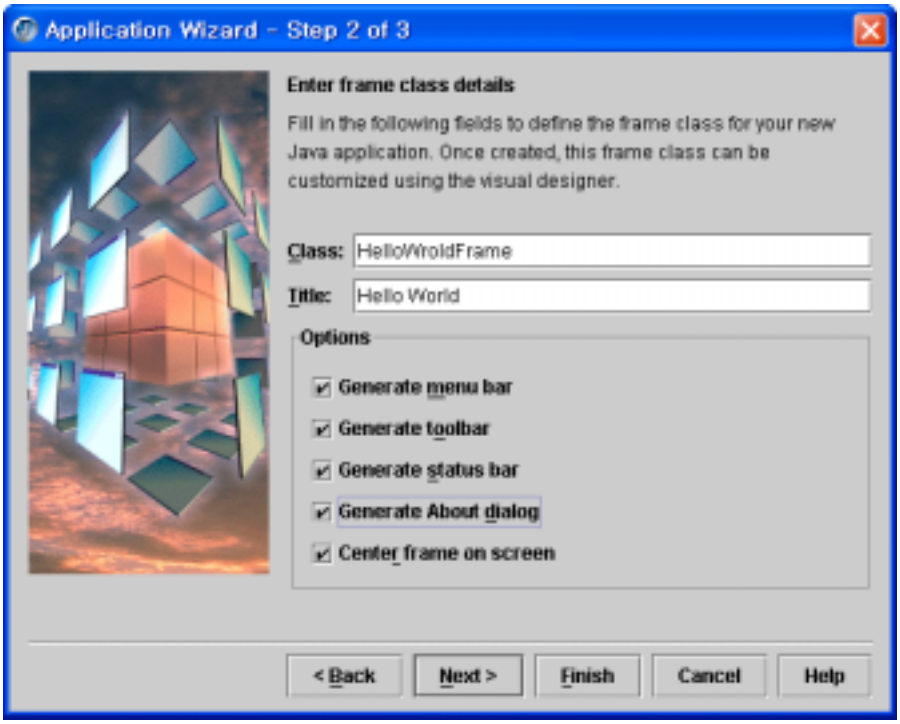


2. 아래 화면과 같이 패키지 이름이 helloworld 라고 표시되는데, 이것은 기본적으로 마법사가 패키지 이름을 프로젝트 파일인 HelloWorld로부터 패키지 이름을 받아오기 때문이다. [Class Name] 필드에 HelloWorldClass 라고 입력한다. 이때 자바 클래스 이름은 대·소문자를 구분하니 조심해야 한다. 그리고 나서 하단에 있는 [Generate Header Comments]를 체크한다. 이 옵션이 체크되면 프로젝트 마법사가 작성한 프로젝트 노트 정보가 애플리케이션 마법사가 작성한 각 소스파일에 선두에 나타나게 된다. 계속하기 위해서 [Next] 버튼을 클릭한다.

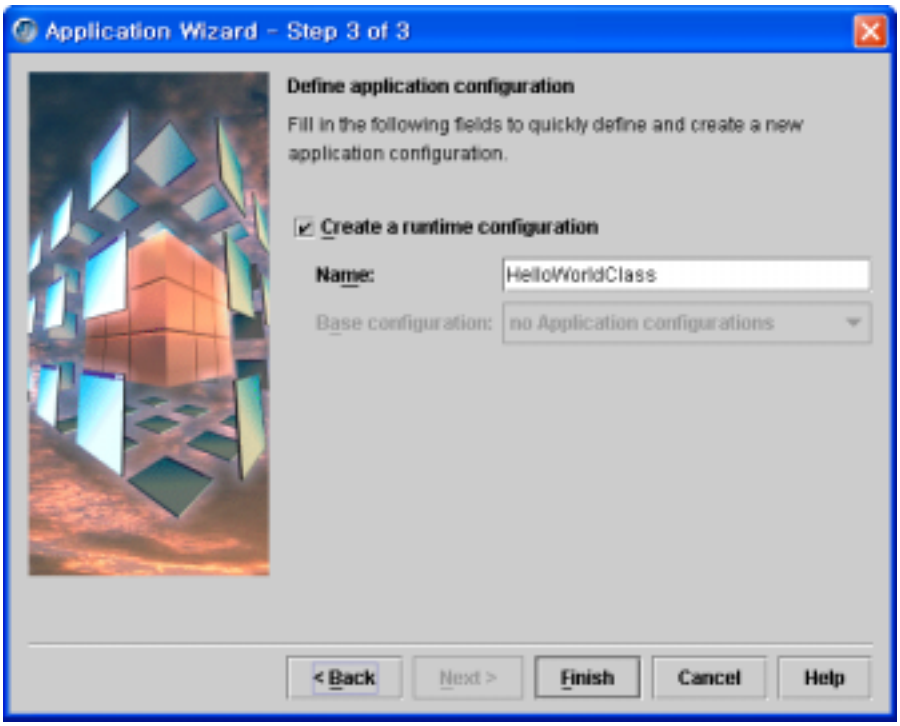


3. 아래 화면에서 [Class] 항목에 프레임 클래스(Frame class)을 나타내는 HelloWorldFrame

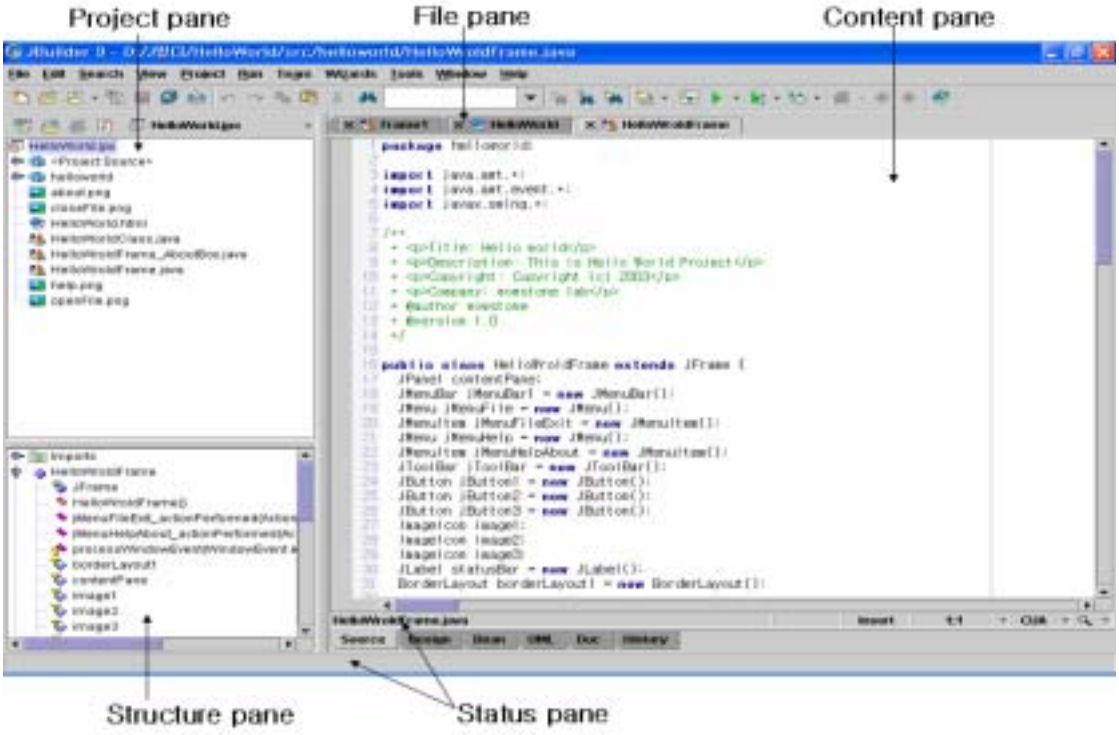
을 입력하고, [Title] 항목에 Hello World 라고 입력한다. Title 항목은 애플리케이션의 창에 Title Bar에 표시되는 텍스트를 말한다. 그리고 하단에 나타나는 Generate Menu Bar, Generate Toolbar, Generate Status Bar, Generate About Dialog, Center Frame On Screen 을 선택한다. 이렇게 선택하면 마법사는 이 항목에 대한 기본적인 기초코드(basic code)를 생성한다.



4. 아래 화면은 애플리케이션 설정을 정의 하는 것으로 이 애플리케이션을 실행하기 위해서는 불 필요하므로 [Finish] 버튼을 클릭한다.



5. 이렇게 소스파일 생성이 정상적으로 실행되면 아래 화면과 같이 AppBrowser 화면이 되시되며, .java 소스파일과 툴바 이미지가 프로젝트에 추가되고, project pane에 node로 표시된다. 앞에서 생성한 HelloWorldFrame.java 의 소스코드는 Content pane에서 볼 수 있다. message pane은 message가 표시 될 때에만 표시된다. project pane에 나타나 helloworld 이름의 자동 소스 패키지 노드 이름은 메뉴에서 [Project | Project Properties] 선택하여 [general] 탭에 있는 [Enable Source Package Discovery And Compilation]옵션이 활성화되어 있을 때 표시된다.



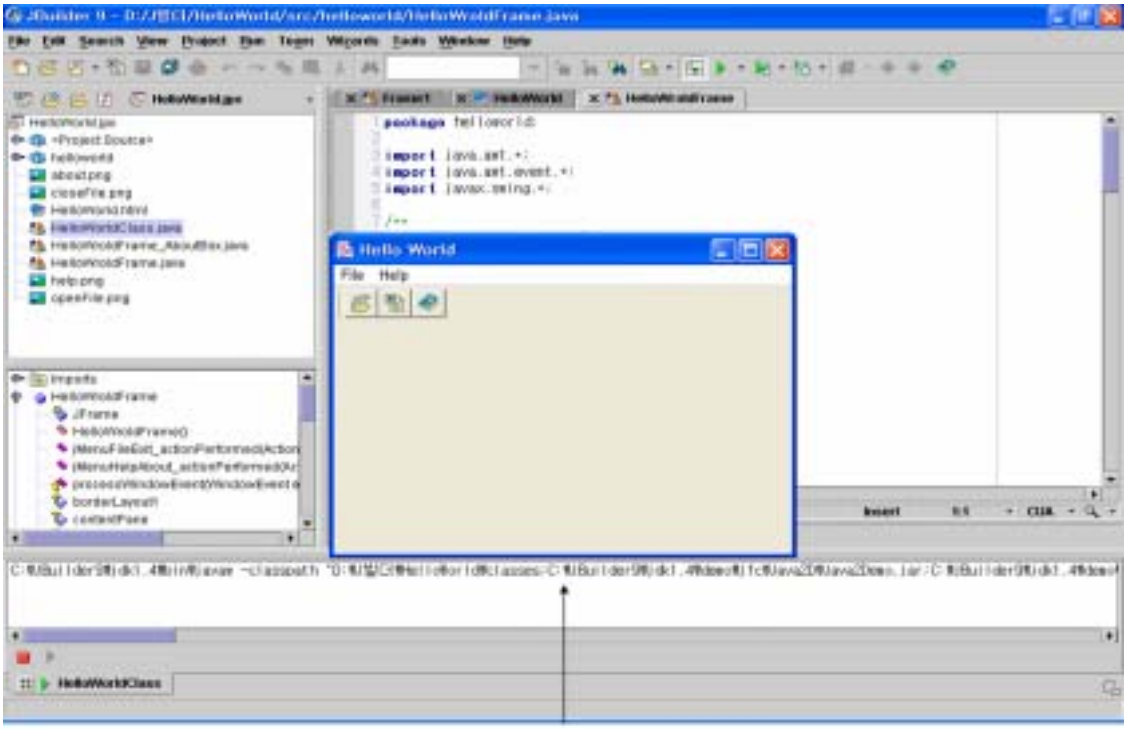
6. 메뉴에서 [File|Save All]을 선택하여 소스파일과 프로젝트 파일을 저장한다. 소스파일은 "/<HOME>/프로젝트명/src/프레임이름"에 자바 컴파일러에 의해 소스파일로부터 생성된 클래스파일들은 "<HOME>/프로젝트명/class/프레임이름"으로 저장된다.

4. 애플리케이션 컴파일 과 실행(Compiling and running your application)

이제 지금까지 작성한 소스파일을 컴파일하고 실행해 볼 차례이다. Compiling은 자바 컴파일러가 수행중인 과정이다. Compiler 는 소스코드를 자바 bytecode로 변환하여 .class 파일을 생성한다.

1. 애플리케이션을 컴파일하고 실행하기 위해 메뉴에서 [Run|Run Project] 나 toobar 에 있는 Run Project 버튼 ()클릭한다. 또한 project pane에 있는 HelloWorldClass.java을 선택하고 마우스 우측버튼을 클릭한후 [Run using "HelloWorldClass"] 메뉴를 선택하여도 된다. 그러면 먼저 컴파일 진행 정도를 표시해 주는 화면이 아래와 같이 표시된다. 여기에는 컴파일동안 발생한 Error와 Warning 개수를 표시해 준다.

컴파일 과정이 종료되면 message pane 이 열리고 난후, 여러분이 생성한 애프리케이션이 아래 화면과 같이 실행된다.



Message pane

만약 Hello World 애플리케이션을 종료하기 위해서는 [File|Exit] 메뉴를 클릭하고, message 을 종료하기 위해서는 message pane 안에 있는 [HelloWorldClass] 탭 좌측에 있는 [Close] 버튼을 클릭한다.

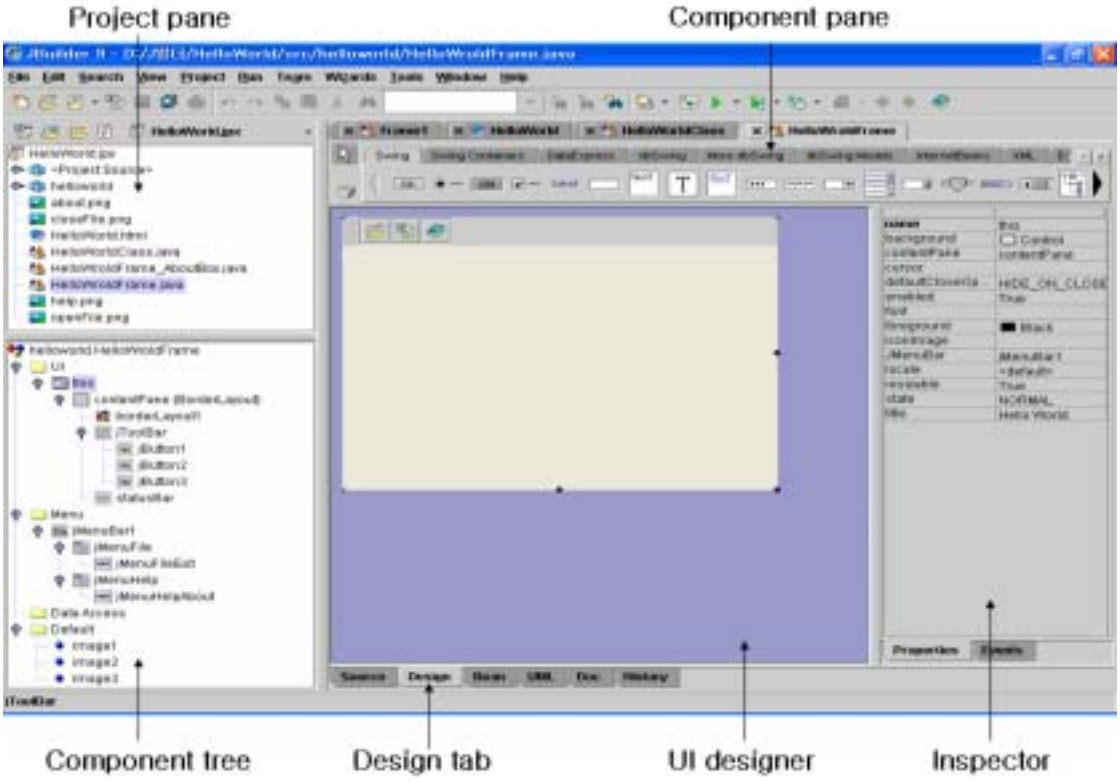
5. 애플리케이션 UI 설정(Customizing your application's user interface)

UI는 사용자 인터페이스를 말하는 것으로 보통 우리가 말하는 GUI(Graphic User Interface) 라고 생각하면 된다. 여기에서는 JBuilder가 제공하는 UI와 이를 다루는 방법에 대하여 설명한다.

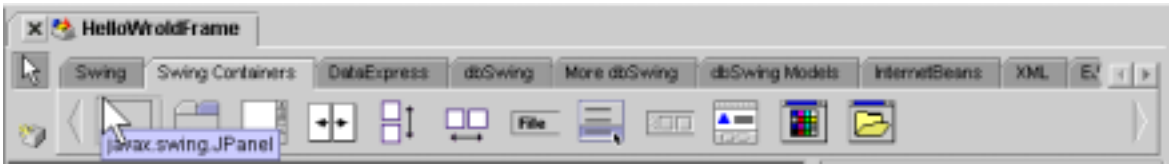
1. UI을 설정하기 위해 project pane에 있는 HelloWorldFrame.java를 더블 클릭한다.

2. design view로 변경하기 위해 content pane의 하단에 있는 [Design] 탭을 클릭한다. 그러면 아래 화면과 같이, 위에 component palette와 오른쪽에 Inspector를 가진 content pane안에 UI designer가 표시된다. 사용자 UI에 component를 추가하기 위해 component palette를 사용하고, 소스코드 안에 property와 event를 추가하여 위해 Inspector를 사용한다. content pane 왼쪽에 위치하고 있는 structure pane은 이제 component와 UI, Menu, Default와 같은

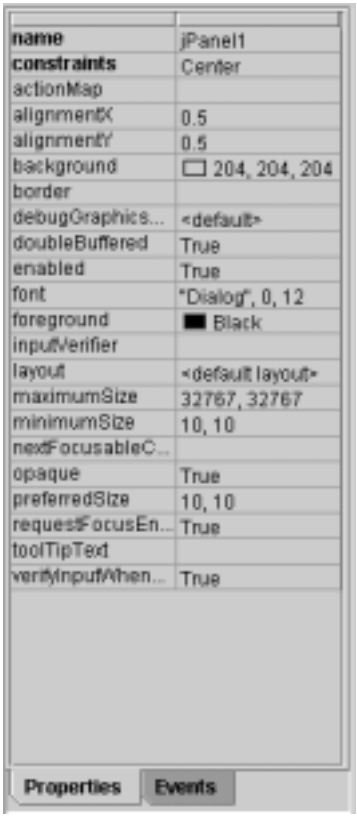
폴더를 포함하고 있는 component tree를 포함한다.



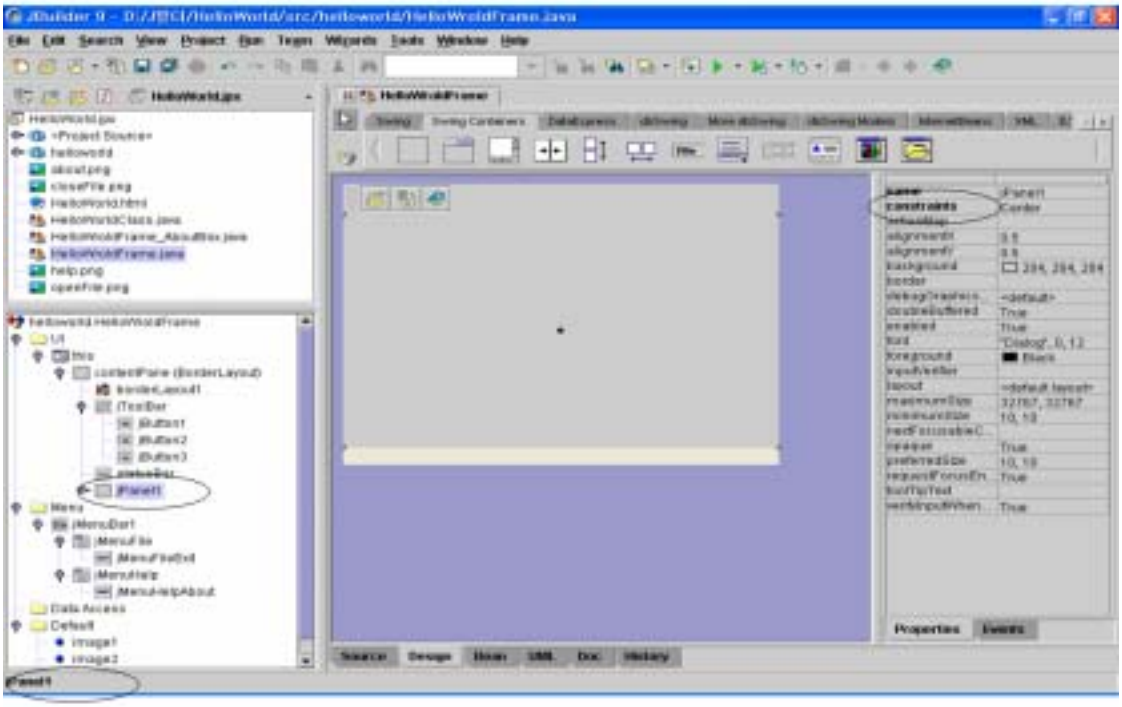
3. UI designer 위에 있는 component palette 중에 Swing Container를 클릭하고, Design 하고자 하는 panel에 추가하기 위해 jPanel component를 선택한다. 만약 component palette에 있는 component 기능을 모를때에는 아래 화면에서와 같이 커서를 해당 component 위에 올려놓으면 tool tip 안에 해당 이름을 알 수 있다.



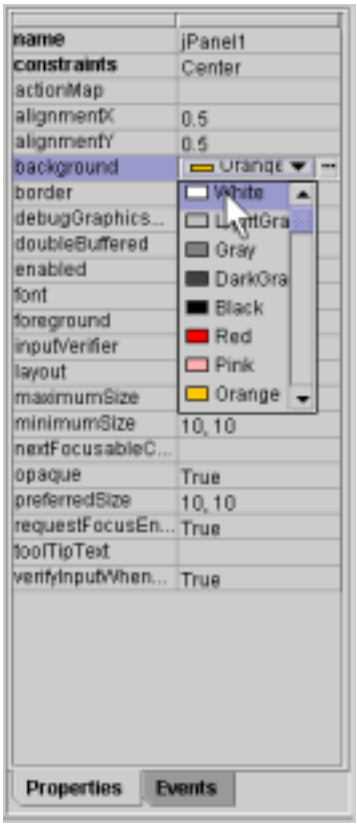
4. 현재 design 하고 있는 프레임 중앙에 component을 위치시키기 위해 UI designer 안에 있는 frame의 중앙을 클릭한다. 한가지 중요한 것은 Inspector 속성 중 [constraints] property는 반드시 [Center]로 설정되어 있어야 한다. 만약 그렇지 않으면, [constraints] property의 오른쪽 column를 클릭하고 drop-down list로부터 [Center]를 선택한다.



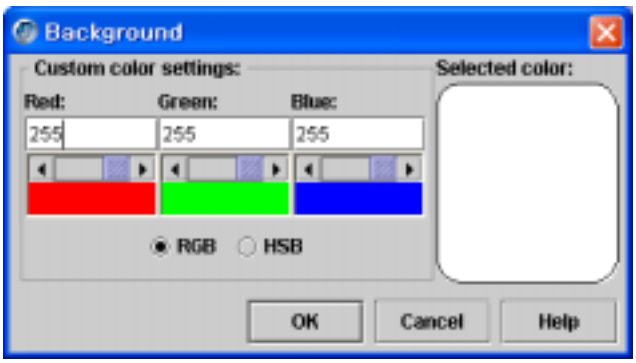
jPanel1은 이제 여러분들의 애플리케이션 design과 component tree에 선택되어 표시된다. component tree와 UI designer에 선택되어진 component는 structure pane아래 status bar에 표시된다.



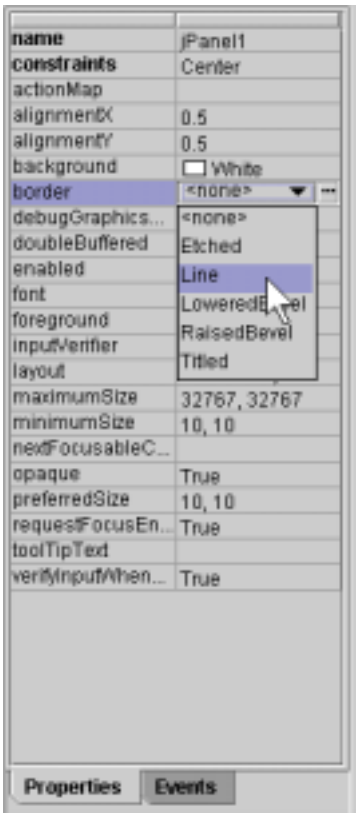
4. 만약 jPanel의 background color을 변경하고자 하면 먼저 Inspector안에 있는 [background] property의 오른쪽 컬럼을 클릭한후, color drop-down list을 열어서 아래 화살표(Down arrow)를 클릭하고, 리스트 꼭대기에 있는 While을 선택한다.



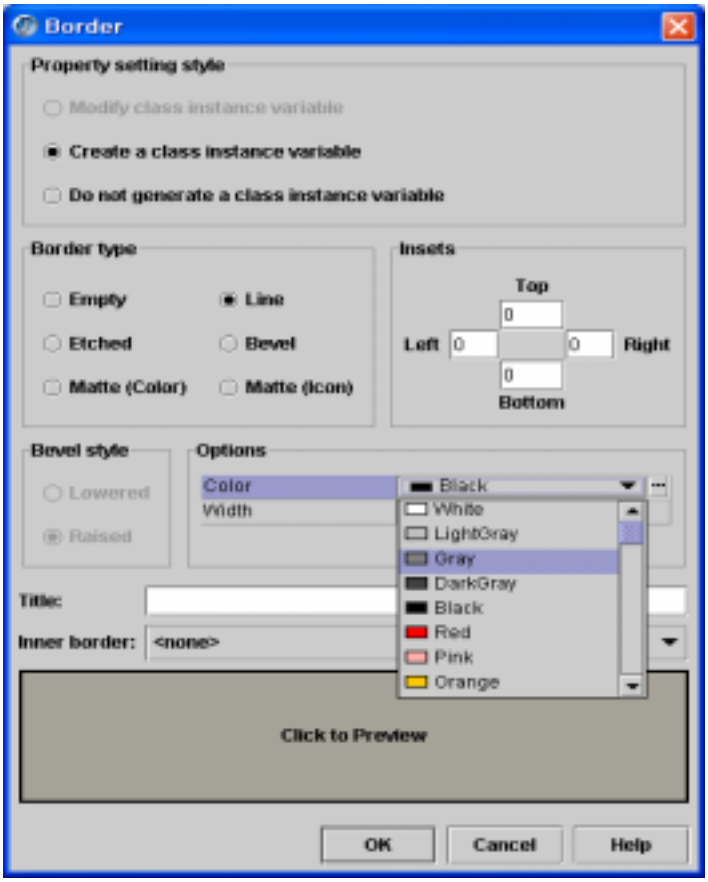
만약 오른쪽 끝에 있는 ... 를 클릭하면 Background dialog box가 아래와 같이 표시되는데, 여기에서 red, green, blue값에 255을 입력하여 흰색을 선택할 수 도 있다.



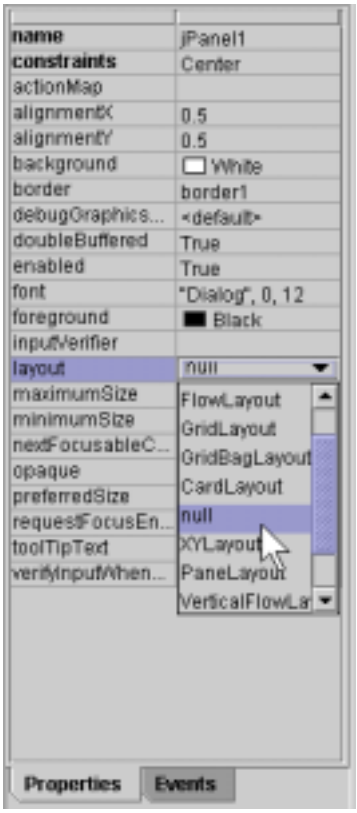
5. 테두리를 추가하고, 테두리 색깔을 Gray로 변경하기 위해서는, 먼저 Inpector의 [border] property에 있는 오른쪽 컬럼을 클릭한후, border drop-down list을 열어서 아래 화살표(Down arrow)를 클릭하고, line을 선택한다.



Border dialog box을 사용하기 위해 ellipsis(...)버튼을 클릭하고, [Option|Color]밑에 있는 black을 클릭하여 color drop-down list에서 Gray로 설정해 주고 dialog box를 닫는다.



6. Inspector의 [layout] property 오른쪽 컬럼을 클릭하여, drop-down list에서 null을 선택해 준다.



layout에 No layout, null을 사용하는 것은 디자인 시 prototyping을 만들어 낼 때 좋은 선택이라고 볼 수 있다. 왜냐하면 null은 layout manager를 사용하지 않기 때문에, 사용자가 원하는 곳에 정확하게 component를 위치시킬 수 있다. 하지만 null이 상대위치 대신 component의 절대위치를 사용하기 때문에 사용자가 애플리케이션 윈도우의 크기를 변경하면 올바르게 변경되지 않는다. 그래서 component를 배치하기 위해 null container를 띄날 필요가 없을 때 이 방법이 권장된다. 나중에 component를 배치하기 전에 유동적으로 layout을 변경하는 것을 배우게 될 것이다.

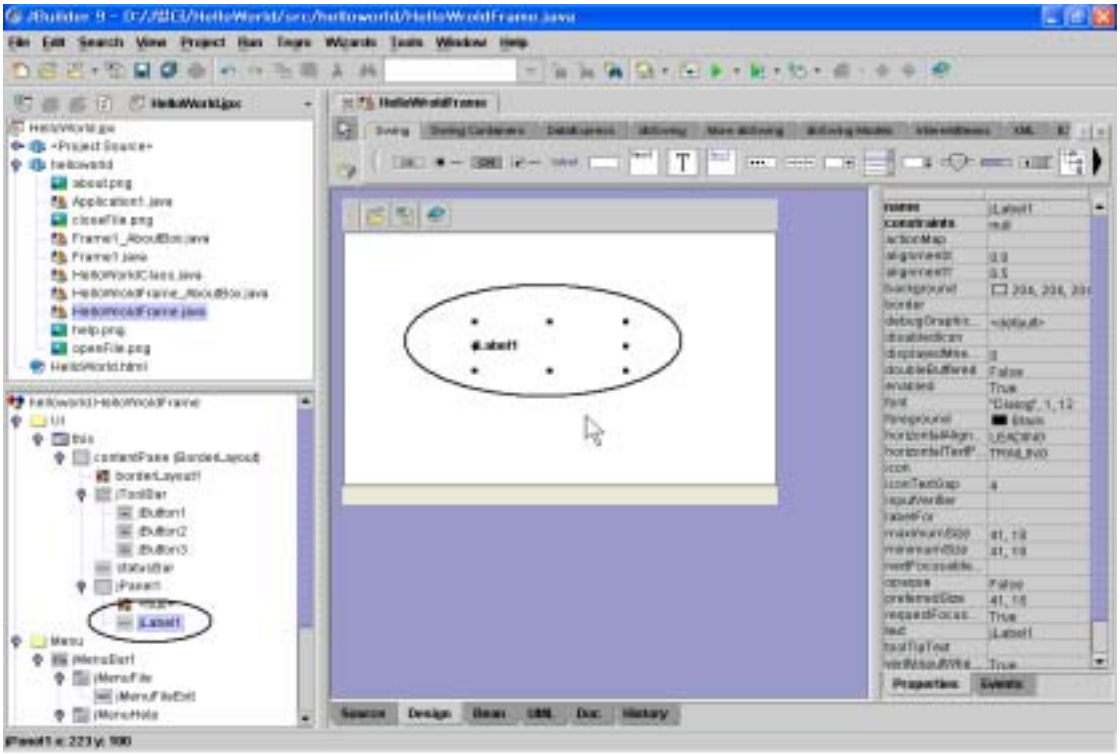
7. 메뉴에서 [File|Save All]이나 toolbar에서 [Save All] 아이콘을 클릭하여 현재 project을 저장한다.

6. 애플리케이션에 컴포넌트 추가(Adding a component to your application)

여기에서는 jPanel component 위에 jLabel component을 추가하기 위해 component palette를 사용하는 것에 대하여 설명한다.

1. component palette에 있는 Swing 탭을 선택한후 jLabel componet을 클릭한다.

2. jLabel이 클릭된 상태에서 마우스를 UI Designer위에 있는 jPanel1 위를 클릭한다. jPanel1를 선택하기 위해서는 좌측에 있는 component tree에 있는 jPanel1을 클릭한다. 위와 같이 정상적으로 component 가 위치하기 되면 왼쪽 component tree에 jPanel1 아래에 jLabel1이 위치하게 된다. 만약 component가 잘못 위치했으면 componet tree나 designer pane에서 해당 component를 클릭한후 [Delete]키를 이용하여 삭제한다.



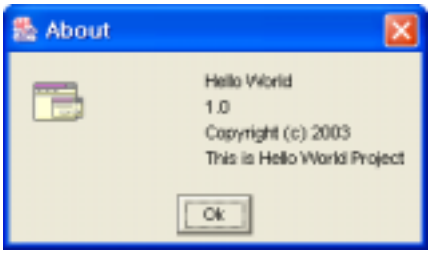
3. designer pane안에 있는 jLabel1 component를 클릭하여 중앙으로 이동시키고 property pane을 이용하여 각종 속성을 아래와 같이 설정한후 메뉴에서 [File|Save All]을 선택하여 저장한다.

항목	설정값	의미
text	Hello World!	표시될 텍스트 명
font	Serif, Bold, Italic, 28	글꼴, 형태, 크기
foreground	Blue	전경색 지정

위와같이 지정하면 jLabel1의 형태는 아래와 같다.

7. 소스코드 수정하기(Editing your source code)

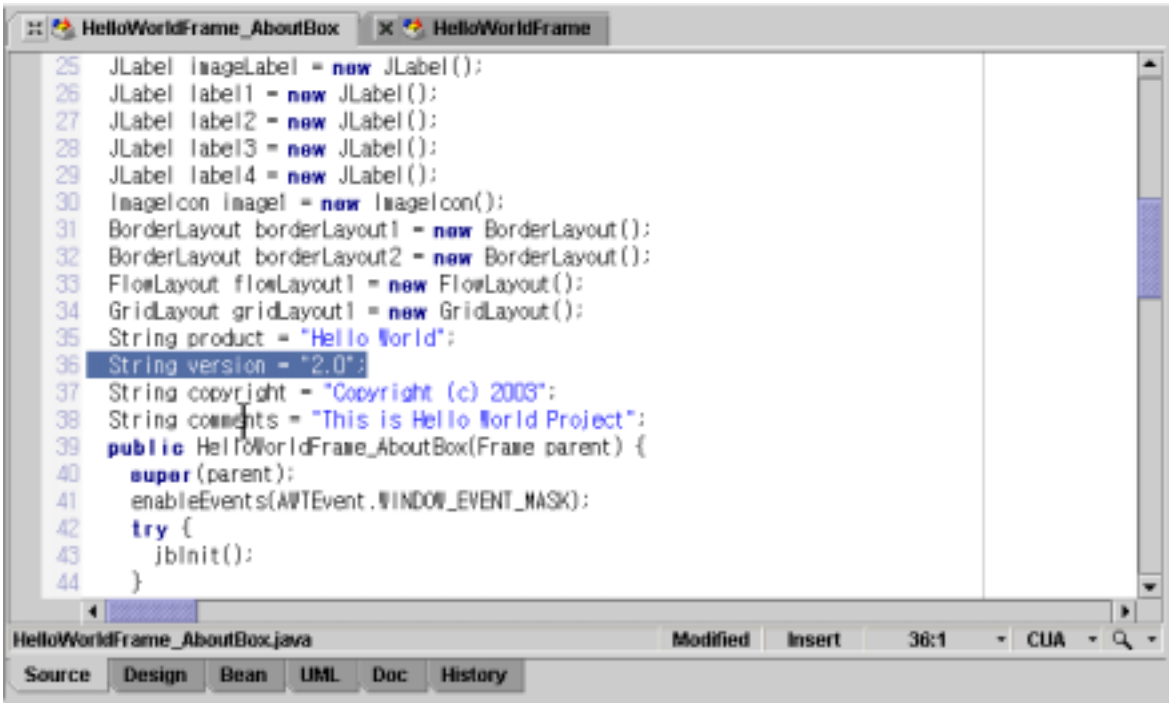
여기에서는 간단하게 아래와 같이 표시되는 [About] box 내의 정보를 변경하기 위해 소스 코드를 직접 수정하는 방법에 대하여 알아본다.



위의 화면은 애플리케이션 마법사에 의해 생성된 기본 애플리케이션 버전이 1.0 이라는 것을 보여주고 있다.

1. 소스코드를 불러들이기 위해 project pane안에 있는 [HelloWorldFrame_AboutBox.java]를 더블클릭한다. 그러면 오른쪽에 content pane이 editor안에 소스코드를 편집할 수 있도록 소스 뷰로 변경된다.

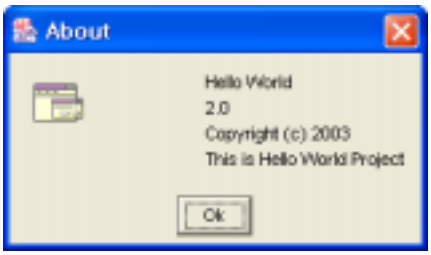
2. 소스코드중에 [String version = "1.0"] 부분을 [String version = "2.0"]으로 수정한후 저장한다.



8. 애플리케이션 컴파일하고 실행하기(Compiling and running your application)

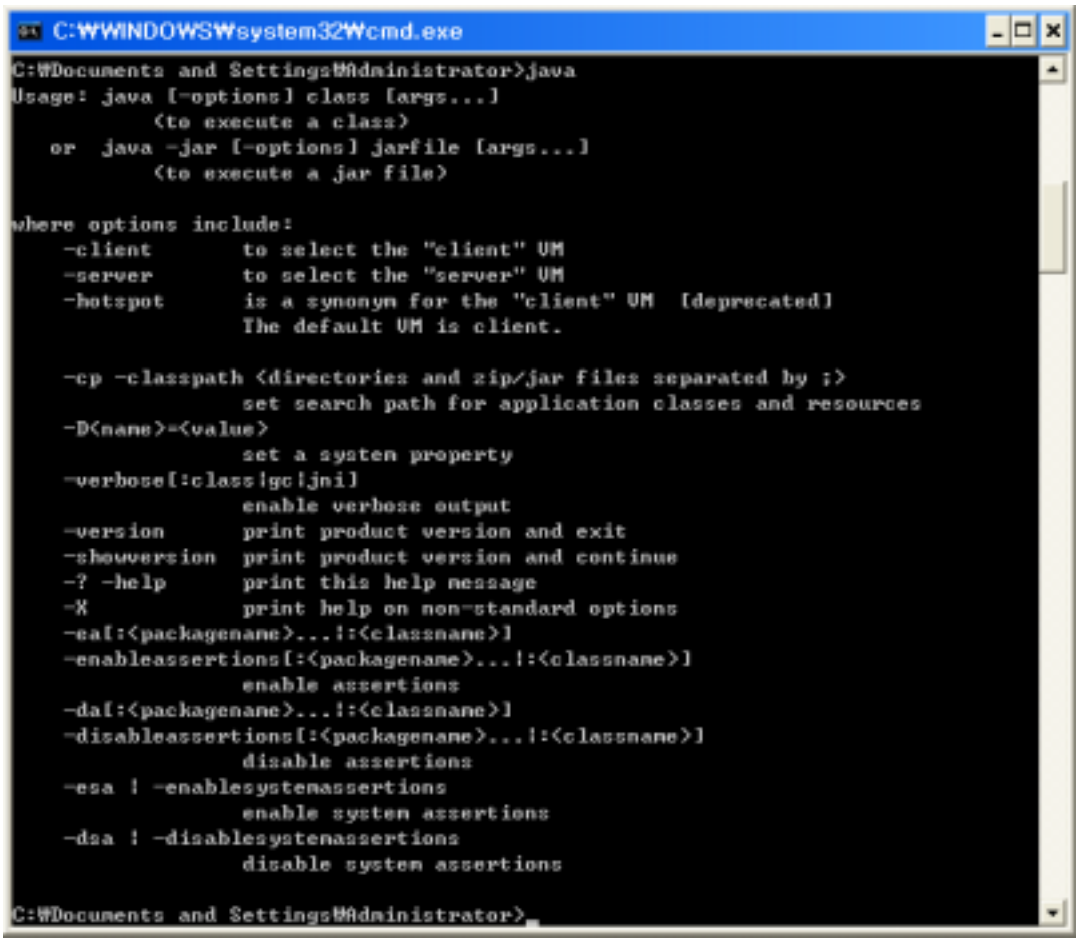
3. 메뉴에서 [Project|Make Project], [Run|Run Project]을 선택하여 컴파일 및 실행을 하면 아래와 같이 애플리케이션이 실행된다.

위 애플리케이션에서 [Help|About]을 선택하면 방금전에 수정한 버전 정보고 dialog box에 아래와 같이 표시된다.



9. 명령행에서 애플리케이션 실행하기(Running your application from the command line)

자바 애플리케이션은 JBuilder 환경이 아닌 명령행에서도 실행될수 있다. java 명령을 포함하고 있는 디렉토리인 <jdk>/bin/ 는 경로(path)에 있어야 한다. 만약 java 명령이 여러분이 지정한 경로에 있으면, 명령행에서 java 명령어를 입력하면 아래화면처럼 command 명령을 설명하는 정보들이 표시된다.



만약 그렇지 않으면 애플리케이션을 실행할 때 <jdk>/bin 디렉토리 안에서 애플리케이션을 실행하여야 한다. 먼저 명령행 윈도우를 열고 아래 명령어를 명령 프롬프트(command prompt)상에 한줄로 입력한다. 이때 윈도우 사용하자는 "/" 대신에 backslash (\)를 사용한다.

```
java -classpath
/<home>/jbproject/HelloWorld/classes helloworld.HelloWorldClass
```

여기에서 <home>은 명령행에서 [set] 명령어를 실행했을 때 표시되는 [HOMEPATH]을 가리킨다. 아래는 필자의 <home>을 보여주고 있다.

```
HOMEPATH=\Documents and Settings\Administrator
```

여기에서 사용된 [java] 명령은 다음과 같은 형식을 가진다.

```
java -classpath classpath package-name.main-class-name
```

여기에서 사용된 각 항목의 의미는 다음과 같다.

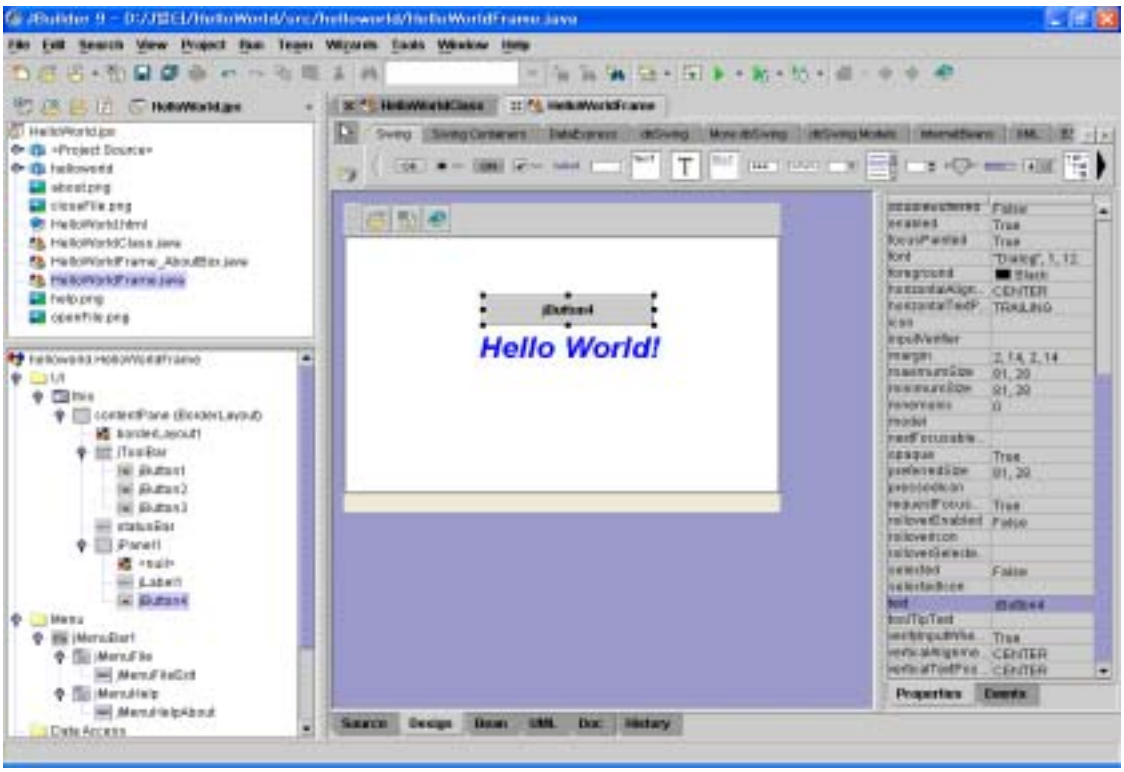
항목	의미
java	자바애플리케이션을 실행한다
-classpath	application classes and resources을 위한 검색경로를 설정하기위한 옵션이다.
classpath	컴파일된 클래스를 포함하고 있는 디렉토리를 가리킨다.
package-name	여기에서는 helloworld를 가리킨다.
main-class-name	여기에서는 HelloWorldClass를 가리킨다.

10. 버튼에 이벤트 추가하기(Adding an event to a button)

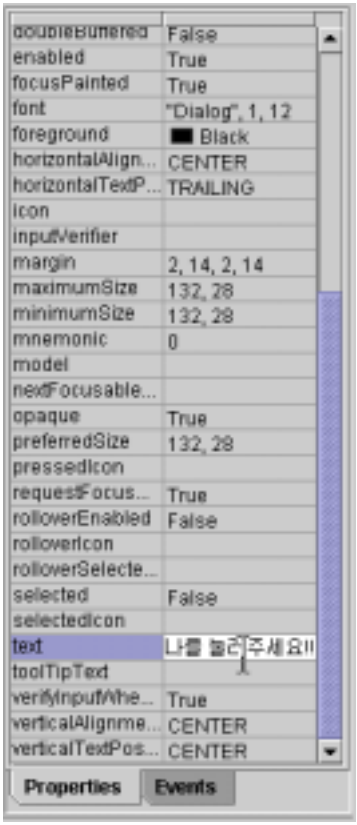
여기에서는 번트에 이벤트를 추가하기 위해 또다른 swing component를 사용할 것이다.

1. project pane에서 HelloWorldFrame.java 파일을 열고 UI designer로 변경하기 위해 Design Tab를 클릭한다.

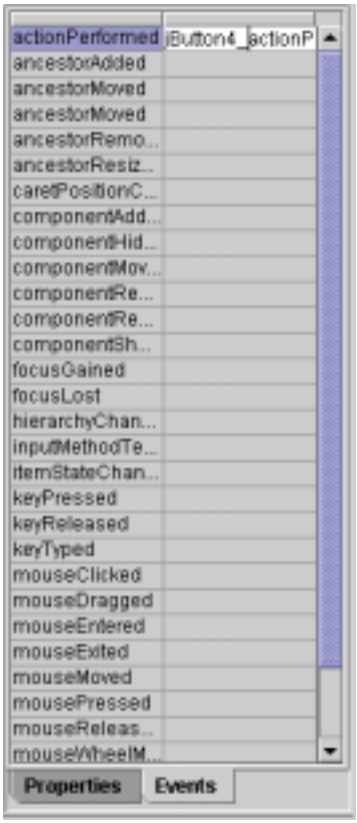
2. component pallete Swing tab에 있는 jButton (OK) component를 클릭하고, component tree의 jPanel1이나, design pane에 있는 panel에 위치시킨후 화면중앙에 위치시키다. 그러면 아래 화면과 같이 jButton4가 componet tree 의 jPanel1 아래에 추가된다.



3. Inspector의 [Text] 속성을 아래 화면에서 보는것과 같이 jButton4에서 "나를 눌러주세요!!"로 변경한다. 만약 버튼에 해당 문자가 일부 보이지 않으면 jButton4를 클릭하여 문자가 보일때까지 크기를 조정한다.



3. jButton4가 눌러 졌을때 해당 작업을 지정하기 위해 Inspector pane의 [Event] 탭을 클릭한후, [ActionPerformed] 이벤트 속성을 더블클릭한다.



그러면 아래와 같은 함수원형만 있는 코드(skeletorn code)가 [ActionPerformed] 이벤트가 추가됨과 동시에 에디터 상태로 변경된다.

```
void jButton4_actionPerformed(ActionEvent e) {
}
```

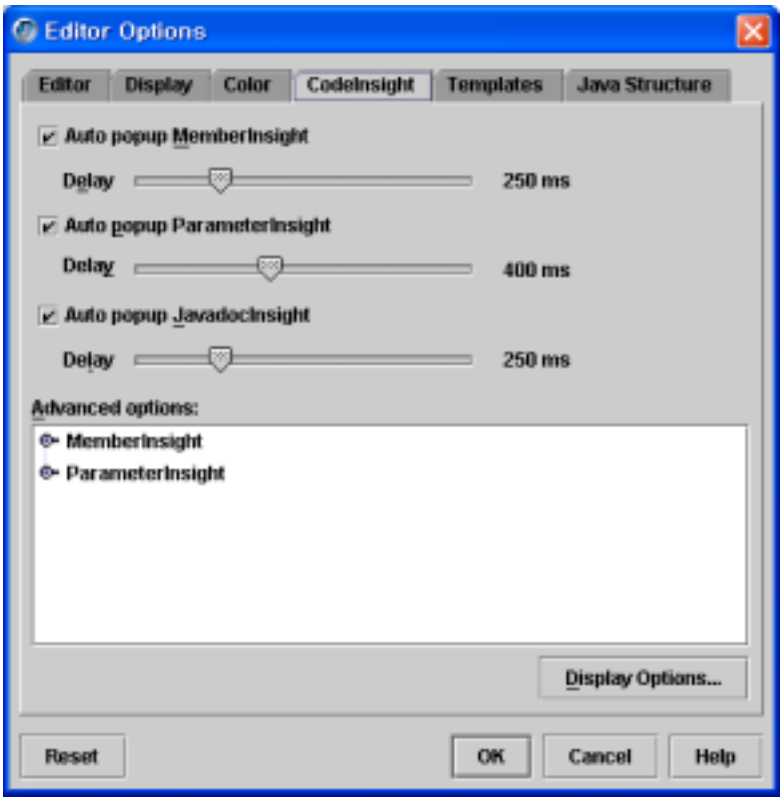
위의 코드에서 아래 부분을 추가하여 수정한다.

```
void jButton4_actionPerformed(ActionEvent e) {
        jLabel1.setForeground(new Color(255,0,0));
}
```

4. 모든 파일을 저장하고 실행시킨다. 방금 만든 버튼을 클릭하면 "Hello World!"의 텍스트 색상이 적색으로 변경되는 것을 볼 수 있을 것이다.

팁>
JBuilder에서는 코드 자동완성 기능(CodeInsight)을 제공해 준다. 위에서 jLabel1. 까지만 입력하면 자동적으로 아래 화면과 같이 팝업메뉴가 표시된다. 또는 [CTRL+spacebar]을 눌러도 된다. 화살표키을 이용하여 해당 코드나 함수를 선택하고 [Enter]키를 누르면 코드가 완성된다.

만약 코드 자동완성기능이 동작하지 않으면 메뉴에서 [Tools|Editor Options]을 선택 아래 대화창의 [CodeInsight]기능을 확인해 본다.



11. UI 완성하기(Completing your UI)

최종적인 애플리케션을 완성하기 위해서는 지금까지 설정한 layout를 portable layout으로 변경해야 한다. 만약 여러분들이 iPanel1의 layout를 null이나 no layout으로 설정해 놓으면 아래 화면에서 보는 것과 같이 애플리케이션이 실행중 일 때 윈도우 크기를 변경하더라도, 여러분의 애플리케이션은 윈도우 크기가 변하지 않을 뿐만 아니라, 해당 componet의 위치도 변경되지 않는다. 왜냐하면 null 은 절대위치를 참조하기 때문에 아무리 윈도우를 변경하더라도 항상 똑같은 위치에 고정되어 있기 때문이다. 그래서 null은 여러분의 애플리케이션에 component를 마지막으로 배치(Deployment)하기 위해서는 좋은 layout manager가 아니라고 볼 수 있다.

null이 아닌 Portable layout 은 동적인(dynamic) 상대위치(relative positioning)를 사용한다. portable layout 상에서의 component 는 애플리케이션의 윈도우 크기가 변하더라고 정

확하게 component 위치를 재 배치한다.

1. project pane에서 HelloWorldFrame.java 파일을 열고 UI designer로 변경하기 위해 Design Tab를 클릭한다.

2. component tree에서 jPanel1을 선택하고, Inspector의 Property tab을 클릭한다.

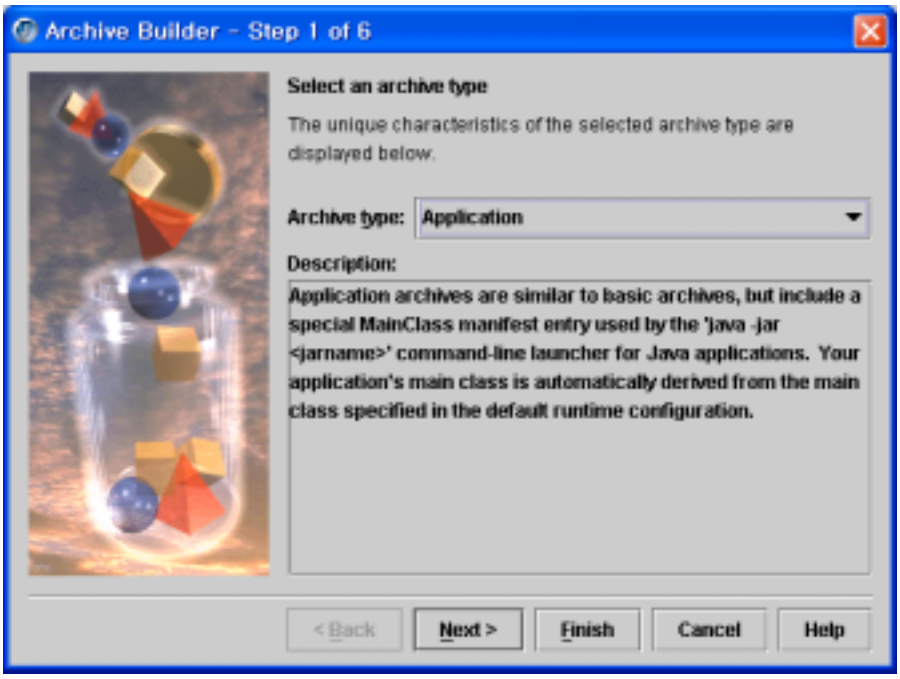
3. Inspector의 jPanel1의 layout 속성을 [GridbagLayout]으로 변경한다. designer안에 있는 각각의 component를 선택하여 각 component가 차지하고 있는 grid 영역을 확인해 보아라

4. 이렇게 변경하고 애플리케이션을 실행한다. 애플리케이션이 실행중에 윈도우 크기가 변경되면 자동으로 각각의 component의 위치가 변경되어 아래 화면과 같이 재 배열 되는 것을 볼 수 있다.

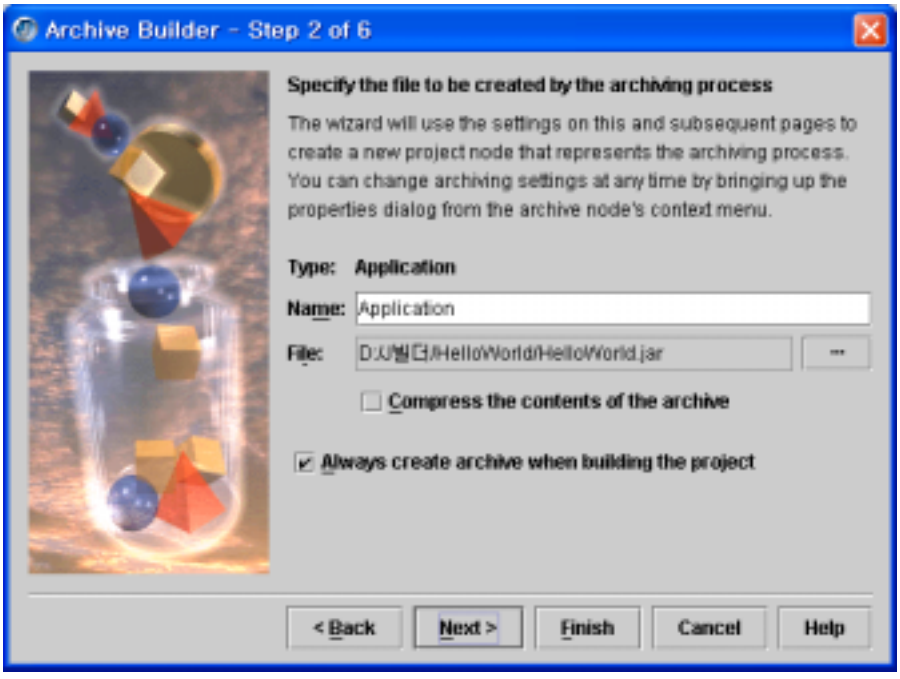
12. 배포를 위한 애플리케이션 준비(Preparing your application for deployment)

Archive Builder 는 사용자 프로그램을 배포하기 위하여 필요한 모든 파일들을 모으며, 그들을 Java archive file 인 JAR 파일로 기록한다. 여기에서는 이와같이 JBuilder 에서 제공하는 JAR 파일을 만드는 방법에 대하여 설명한다.

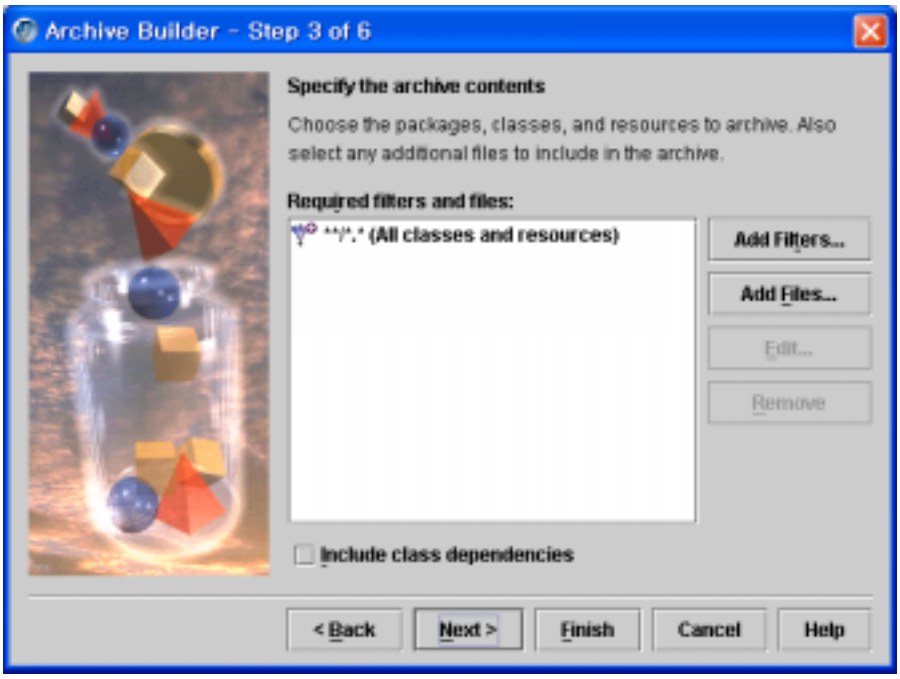
1. 메뉴에서 [Wizards|Archive Builder]을 선택하여 Archive Builder 을 실행한 다음, 아래 화면이 표시되면 Archive Type drop-down list 박스에서 [Application]를 선택한후 [Next] 버튼을 클릭한다.



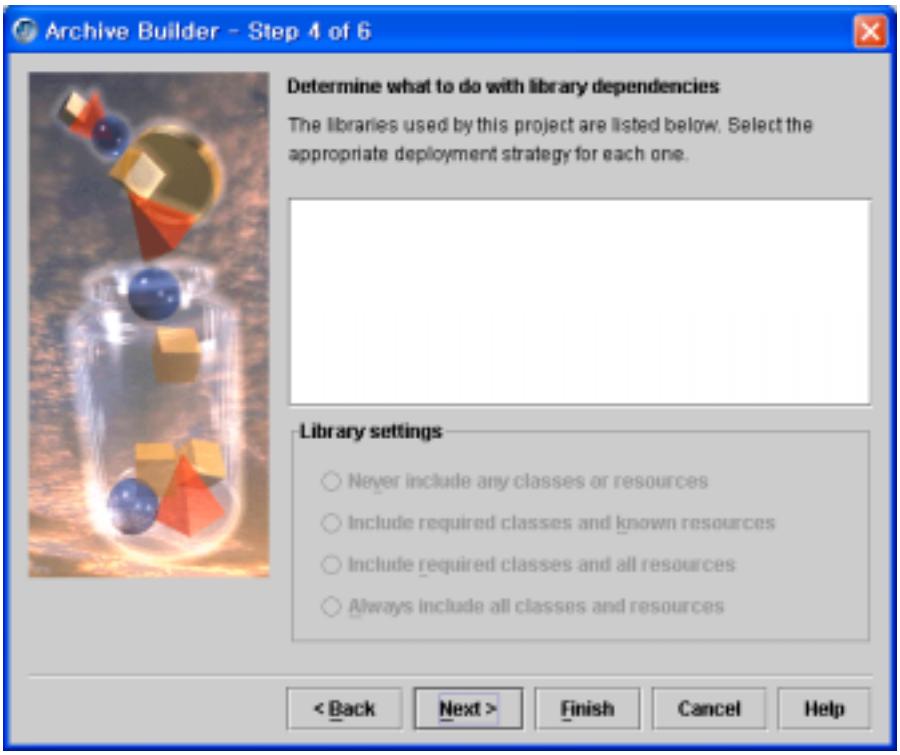
2. 아래 화면이 표시되면 모든 설정값을 기본값으로 하고 [Next] 버튼을 클릭한다. 여기에서 HelloWorld.jar 파일은 HelloWorld 디렉토리에 생성된다.



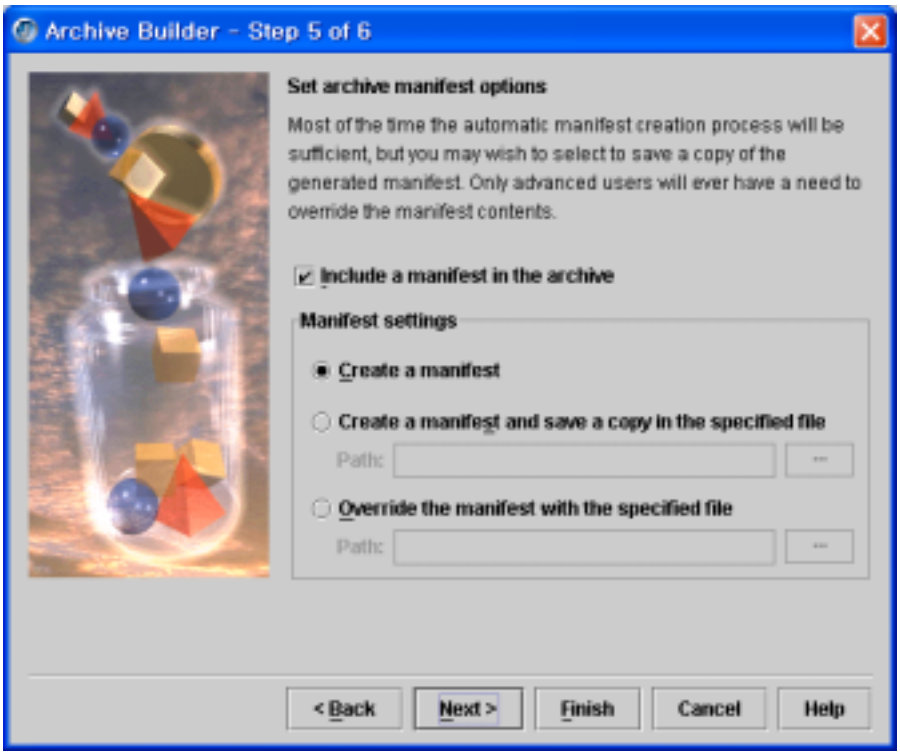
3. 아래 화면이 표시되면 모든 설정값을 기본값으로 하고 [Next] 버튼을 클릭한다.



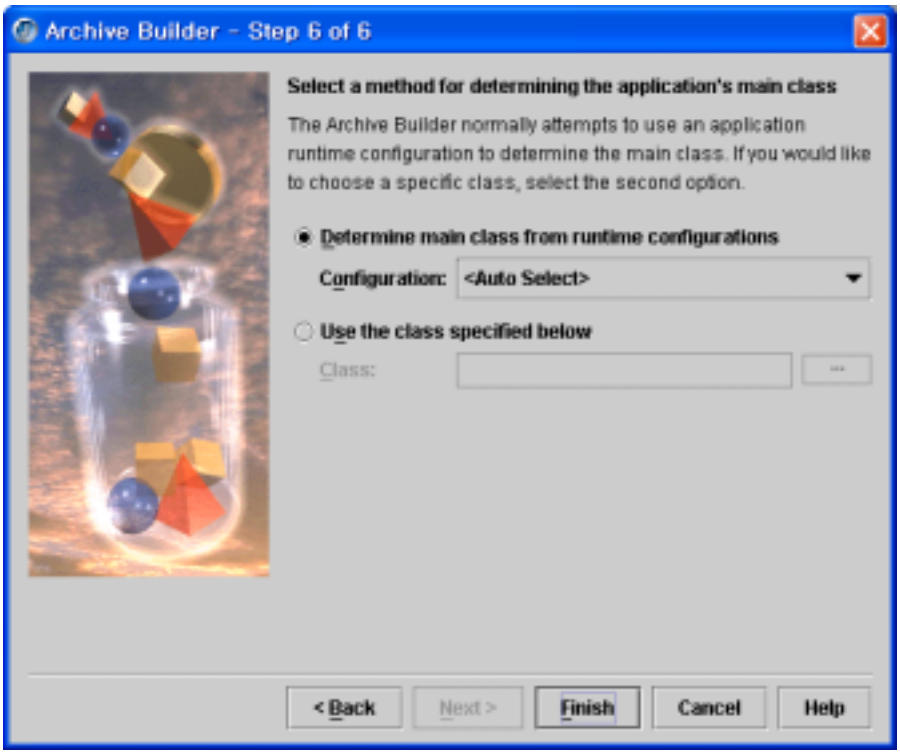
4. 아래 화면이 표시되면 모든 설정값을 기본값으로 하고 [Next] 버튼을 클릭한다.



5. 아래 화면이 표시되면 모든 설정값을 기본값으로 하고 [Next] 버튼을 클릭한다.



6. Archive Builder를 종료하기 위하여 아래 화면에서 [Finish] 버튼을 클릭한다. 그러면 [Application] 불리우는 Archive node 가 project pane에 표시되면 사용자는 마우스 오른쪽 버튼을 이용하여 수정하거나 속성을 변경할 수 있다.

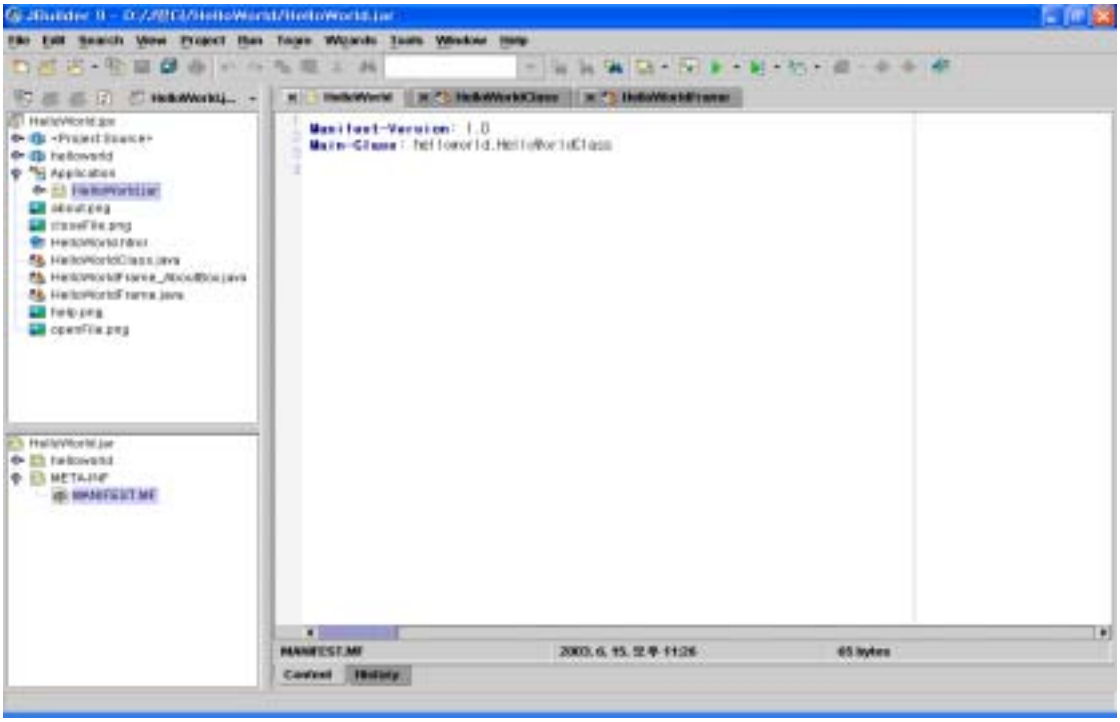


사용자 애플리케이션을 컴파일하고 JAR 파일을 생성하기 위하여 Application node을 선택, 마우스 오른쪽 버튼을 이용하거나, 메뉴에서 [Project | Make Project] 선택한다. Archive Builder 는 메뉴에서 [Project|Project Properties | Paths] 를 선택했을때 나타나는 project

output path안에 있는 모든 파일을 JAR 파일로 모은다

7. project pane에서 [Applicaiton] archive node를 펼치면 archive file인 HelloWorld.jar 파일이 생성된다.

8. project pane에서 JAR 파일을 더블클릭한다. 그러면 아래 화면과 같이 content pane에 manifest파일이 표시되며, structure pane에 JAR 파일의 항목이 표시된다.



참고>
Archive로 작성된 애플리케이션 배포판을 생성하기 위해서는 위에서 설명한 JBuilder 를 사용하지 않고 명령행을 이용할 수도 있다. 이 애플리케이션의 명령행을 사용하는 것은 앞에서 설명한 것 처럼 일반적인 자바 프로그램을 컴파일하고 실행시키는 과정은 똑 같다. 다만, classpath 옵션에 일반 자바 파일을 컴파일한 class 파일을 사용하지만, Archive로 작성된 애플리케이션은 방금 작성한 JAR 파일을 포함시켜야 한다는 것만 다르다.

13. 명령해에서 배포 애플리케이션 실행(Running your deployed application from the command line)

14. HelloWorld 스스코드(HelloWorld Source code)

여기에서는 지금까지 작업한 Helloworld 애플리케이션을 작성할때, JBuilder 가 생성한 대표적인 소스코드인 HelloWorldFrame.java, HelloWorldClass.java,

HelloWorldFrame_AboutBox.java 에 대한 소스코드를 보여주고 있다. 사용자는 이러한 소스코드를 여기에서는 이해하실 필요가 없으며, JBuilder가 이렇게 코드를 기본적으로 생성하고 이러한 구조를 가지고 있구나 하는 정도로만 이해하면 된다.

Example>

```
/*------------------- filename : HelloWorldFrame.java -------------------------*/
package helloworld;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.border.*;

/**
 * <p>Title: Hello World</p>
 * <p>Description: This is Hello World Project</p>
 * <p>Copyright: Copyright (c) 2003</p>
 * <p>Company: wowstone lab</p>
 * @author wowstone
 * @version 1.0
 */

public class HelloWorldFrame extends JFrame {
  JPanel contentPane;
  JMenuBar jMenuBar1 = new JMenuBar();
  JMenu jMenuFile = new JMenu();
  JMenuItem jMenuFileExit = new JMenuItem();
  JMenu jMenuHelp = new JMenu();
  JMenuItem jMenuHelpAbout = new JMenuItem();
  JToolBar jToolBar = new JToolBar();
  JButton jButton1 = new JButton();
  JButton jButton2 = new JButton();
  JButton jButton3 = new JButton();
  ImageIcon image1;
  ImageIcon image2;
  ImageIcon image3;
  JLabel statusBar = new JLabel();
  BorderLayout borderLayout1 = new BorderLayout();
  JPanel jPanel1 = new JPanel();
  Border border1;
  JLabel jLabel1 = new JLabel();
  JButton jButton4 = new JButton();
  GridBagLayout gridBagLayout1 = new GridBagLayout();

  //Construct the frame
  public HelloWorldFrame() {
```

```
    enableEvents(AWTEvent.WINDOW_EVENT_MASK);
    try {
      jbInit();
    }
    catch(Exception e) {
      e.printStackTrace();
    }
  }
  //Component initialization
  private void jbInit() throws Exception  {
    image1 = new
ImageIcon(helloworld.HelloWorldFrame.class.getResource("openFile.png"));
    image2 = new
ImageIcon(helloworld.HelloWorldFrame.class.getResource("closeFile.png"));
    image3 = new ImageIcon(helloworld.HelloWorldFrame.class.getResource("help.png"));
    contentPane = (JPanel) this.getContentPane();
    border1 = BorderFactory.createLineBorder(Color.gray,2);
    contentPane.setLayout(borderLayout1);
    this.setSize(new Dimension(400, 300));
    this.setTitle("Hello World");
    statusBar.setText(" ");
    jMenuFile.setText("File");
    jMenuFileExit.setText("Exit");
    jMenuFileExit.addActionListener(new
HelloWorldFrame_jMenuFileExit_ActionAdapter(this));
    jMenuHelp.setText("Help");
    jMenuHelpAbout.setText("About");
    jMenuHelpAbout.addActionListener(new
HelloWorldFrame_jMenuHelpAbout_ActionAdapter(this));
    jButton1.setIcon(image1);
    jButton1.setToolTipText("Open File");
    jButton2.setIcon(image2);
    jButton2.setToolTipText("Close File");
    jButton3.setIcon(image3);
    jButton3.setToolTipText("Help");
    jPanel1.setBackground(Color.white);
    jPanel1.setBorder(border1);
    jPanel1.setLayout(gridBagLayout1);
    jLabel1.setFont(new java.awt.Font("Dialog", 3, 28));
    jLabel1.setForeground(Color.blue);
    jLabel1.setOpaque(false);
    jLabel1.setText("Hello World!");
    jButton4.setActionCommand("나를 눌러주세요!!");
    jButton4.setText("나를 눌러주세요!!");
    jButton4.addActionListener(new HelloWorldFrame_jButton4_actionAdapter(this));
    jToolBar.add(jButton1);
```

```
    jToolBar.add(jButton2);
    jToolBar.add(jButton3);
    jMenuFile.add(jMenuFileExit);
    jMenuHelp.add(jMenuHelpAbout);
    jMenuBar1.add(jMenuFile);
    jMenuBar1.add(jMenuHelp);
    this.setJMenuBar(jMenuBar1);
    contentPane.add(jToolBar, BorderLayout.NORTH);
    contentPane.add(statusBar, BorderLayout.SOUTH);
    contentPane.add(jPanel1, BorderLayout.CENTER);
    jPanel1.add(jButton4, new GridBagConstraints(0, 0, 1, 1, 0.0, 0.0
            ,GridBagConstraints.CENTER, GridBagConstraints.NONE, new Insets(55, 122,
0, 113), 26, 2));
    jPanel1.add(jLabel1, new GridBagConstraints(0, 1, 1, 1, 0.0, 0.0
            ,GridBagConstraints.CENTER, GridBagConstraints.NONE, new Insets(9, 112,
107, 98), 21, 5));
  }
  //File | Exit action performed
  public void jMenuFileExit_actionPerformed(ActionEvent e) {
    System.exit(0);
  }
  //Help | About action performed
  public void jMenuHelpAbout_actionPerformed(ActionEvent e) {
    HelloWorldFrame_AboutBox dlg = new HelloWorldFrame_AboutBox(this);
    Dimension dlgSize = dlg.getPreferredSize();
    Dimension frmSize = getSize();
    Point loc = getLocation();
    dlg.setLocation((frmSize.width - dlgSize.width) / 2 + loc.x, (frmSize.height -
dlgSize.height) / 2 + loc.y);
    dlg.setModal(true);
    dlg.pack();
    dlg.show();
  }
  //Overridden so we can exit when window is closed
  protected void processWindowEvent(WindowEvent e) {
    super.processWindowEvent(e);
    if (e.getID() == WindowEvent.WINDOW_CLOSING) {
      jMenuFileExit_actionPerformed(null);
    }
  }

  void jButton4_actionPerformed(ActionEvent e) {
    jLabel1.setForeground(new Color(255,0,0));
  }
}
```

```
class HelloWorldFrame_jMenuFileExit_ActionAdapter implements ActionListener {
  HelloWorldFrame adaptee;

  HelloWorldFrame_jMenuFileExit_ActionAdapter(HelloWorldFrame adaptee) {
    this.adaptee = adaptee;
  }
  public void actionPerformed(ActionEvent e) {
    adaptee.jMenuFileExit_actionPerformed(e);
  }
}

class HelloWorldFrame_jMenuHelpAbout_ActionAdapter implements ActionListener {
  HelloWorldFrame adaptee;

  HelloWorldFrame_jMenuHelpAbout_ActionAdapter(HelloWorldFrame adaptee) {
    this.adaptee = adaptee;
  }
  public void actionPerformed(ActionEvent e) {
    adaptee.jMenuHelpAbout_actionPerformed(e);
  }
}

class HelloWorldFrame_jButton4_actionAdapter implements java.awt.event.ActionListener {
  HelloWorldFrame adaptee;

  HelloWorldFrame_jButton4_actionAdapter(HelloWorldFrame adaptee) {
    this.adaptee = adaptee;
  }
  public void actionPerformed(ActionEvent e) {
    adaptee.jButton4_actionPerformed(e);
  }
}

/*------------------- filename : HelloWorldClass.java ------------------------*/
package helloworld;
import javax.swing.UIManager;
import java.awt.*;

/**
 * <p>Title: Hello World</p>
 * <p>Description: This is Hello World Project</p>
 * <p>Copyright: Copyright (c) 2003</p>
 * <p>Company: wowstone lab</p>
 * @author wowstone
 * @version 1.0
 */
```

```
public class HelloWorldClass {
  boolean packFrame = false;

  //Construct the application
  public HelloWorldClass() {
    HelloWorldFrame frame = new HelloWorldFrame();
    //Validate frames that have preset sizes
    //Pack frames that have useful preferred size info, e.g. from their layout
    if (packFrame) {
      frame.pack();
    }
    else {
      frame.validate();
    }
    //Center the window
    Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
    Dimension frameSize = frame.getSize();
    if (frameSize.height > screenSize.height) {
      frameSize.height = screenSize.height;
    }
    if (frameSize.width > screenSize.width) {
      frameSize.width = screenSize.width;
    }
    frame.setLocation((screenSize.width - frameSize.width) / 2, (screenSize.height -
frameSize.height) / 2);
    frame.setVisible(true);
  }
  //Main method
  public static void main(String[] args) {
    try {
      UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
    }
    catch(Exception e) {
      e.printStackTrace();
    }
    new HelloWorldClass();
  }
}

/*------------------- filename : HelloWorldFrame_AboutBox.java ----------*/
package helloworld;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.border.*;
```

```
/**
 * <p>Title: Hello World</p>
 * <p>Description: This is Hello World Project</p>
 * <p>Copyright: Copyright (c) 2003</p>
 * <p>Company: wowstone lab</p>
 * @author wowstone
 * @version 1.0
 */

public class HelloWorldFrame_AboutBox extends JDialog implements ActionListener {

  JPanel panel1 = new JPanel();
  JPanel panel2 = new JPanel();
  JPanel insetsPanel1 = new JPanel();
  JPanel insetsPanel2 = new JPanel();
  JPanel insetsPanel3 = new JPanel();
  JButton button1 = new JButton();
  JLabel imageLabel = new JLabel();
  JLabel label1 = new JLabel();
  JLabel label2 = new JLabel();
  JLabel label3 = new JLabel();
  JLabel label4 = new JLabel();
  ImageIcon image1 = new ImageIcon();
  BorderLayout borderLayout1 = new BorderLayout();
  BorderLayout borderLayout2 = new BorderLayout();
  FlowLayout flowLayout1 = new FlowLayout();
  GridLayout gridLayout1 = new GridLayout();
  String product = "Hello World";
  String version = "2.0";
  String copyright = "Copyright (c) 2003";
  String comments = "This is Hello World Project";
  public HelloWorldFrame_AboutBox(Frame parent) {
    super(parent);
    enableEvents(AWTEvent.WINDOW_EVENT_MASK);
    try {
      jbInit();
    }
    catch(Exception e) {
      e.printStackTrace();
    }
  }
  //Component initialization
  private void jbInit() throws Exception  {
    image1 = new ImageIcon(helloworld.HelloWorldFrame.class.getResource("about.png"));
    imageLabel.setIcon(image1);
    this.setTitle("About");
```

```
    panel1.setLayout(borderLayout1);
    panel2.setLayout(borderLayout2);
    insetsPanel1.setLayout(flowLayout1);
    insetsPanel2.setLayout(flowLayout1);
    insetsPanel2.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10));
    gridLayout1.setRows(4);
    gridLayout1.setColumns(1);
    label1.setText(product);
    label2.setText(version);
    label3.setText(copyright);
    label4.setText(comments);
    insetsPanel3.setLayout(gridLayout1);
    insetsPanel3.setBorder(BorderFactory.createEmptyBorder(10, 60, 10, 10));
    button1.setText("Ok");
    button1.addActionListener(this);
    insetsPanel2.add(imageLabel, null);
    panel2.add(insetsPanel2, BorderLayout.WEST);
    this.getContentPane().add(panel1, null);
    insetsPanel3.add(label1, null);
    insetsPanel3.add(label2, null);
    insetsPanel3.add(label3, null);
    insetsPanel3.add(label4, null);
    panel2.add(insetsPanel3, BorderLayout.CENTER);
    insetsPanel1.add(button1, null);
    panel1.add(insetsPanel1, BorderLayout.SOUTH);
    panel1.add(panel2, BorderLayout.NORTH);
    setResizable(true);
  }
  //Overridden so we can exit when window is closed
  protected void processWindowEvent(WindowEvent e) {
    if (e.getID() == WindowEvent.WINDOW_CLOSING) {
      cancel();
    }
    super.processWindowEvent(e);
  }
  //Close the dialog
  void cancel() {
    dispose();
  }
  //Close the dialog on a button event
  public void actionPerformed(ActionEvent e) {
    if (e.getSource() == button1) {
      cancel();
    }
  }
}
```

지금까지 여러분들은 JBuilder를 이용하여 첫 번째로 애플리케이션을 작성해 보았다. 이제 여러분들은 JBuilder 개발 환경과 어느 정도 친숙해 졌을 거이며, 많은 시간을 절약하여 프로그래밍을 더 쉽게 할 수 있었던 것을 발견할 수 있을 것이다.

연습문제>
1. JBuilder에서 말하는 component 란 무엇인가?
1. 컴파일, byte code란 무엇인가?