21 実習 Eclipseの下でのJavaプロ

Javaプログラミングは、最近ではEclipse と呼ばれる統合開発環境(IDE)を使って進めることが多くなっている。

- オープンソース... 無償利用可，本体ソースも公開

- プラグイン・アーキテクチャを採用
  $\Longrightarrow$ 拡張性が高い。
  $\Longrightarrow$ プラグインを組み込むことで C，C++，C#，COBOL，PHP，Ruby，Perl，HTML，JSP，XML などの言語での開発にも使える。

- Eclipse本体はJava実行環境（JRE ）上で動作する。
  但し、OSに依存したGUIライブラリも利用するので、ソフトウェアパッケージはOS ごとに用意されている。

- Eclipse上でJavaプログラムをコンパイルする際には、JDK内のjavacではなく、Javaの仕様に準拠したEclipse内部のコンパイラを使うことになる。

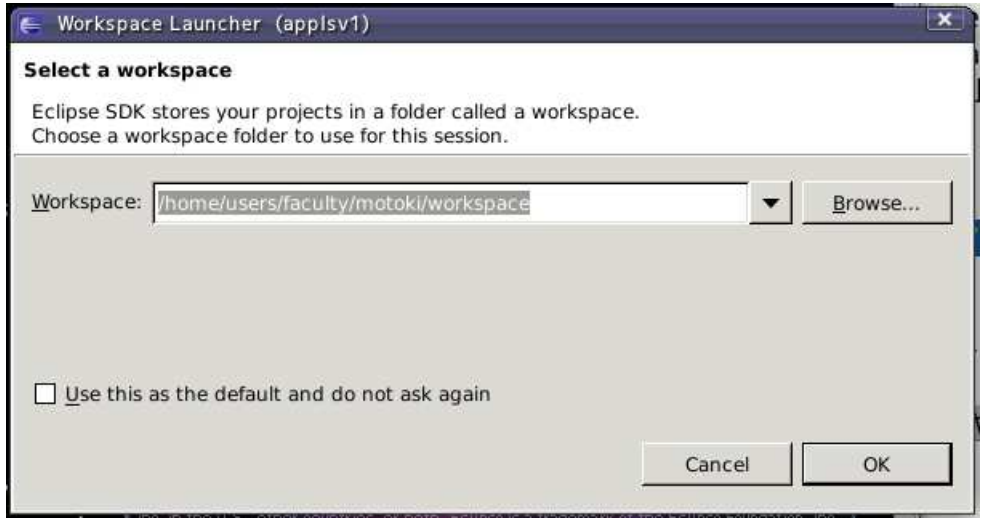
- 実習室に入っているのは Eclipse4.4.2(Luna, 2015)。

21–1 Eclipseの起動，ワークスペースの指定，

Eclipseの起動・終了: それぞれ次の様にする。

- 起動 ··· コマンドライン上で eclipse&
- 終了 ··· メニューバーで File→Exit と選択。

「ワークスペース・ランチャー」ダイアログ: Eclipse起動直後には、「Workspace Launcher」という表題の次の様な小ウィンドウが出る



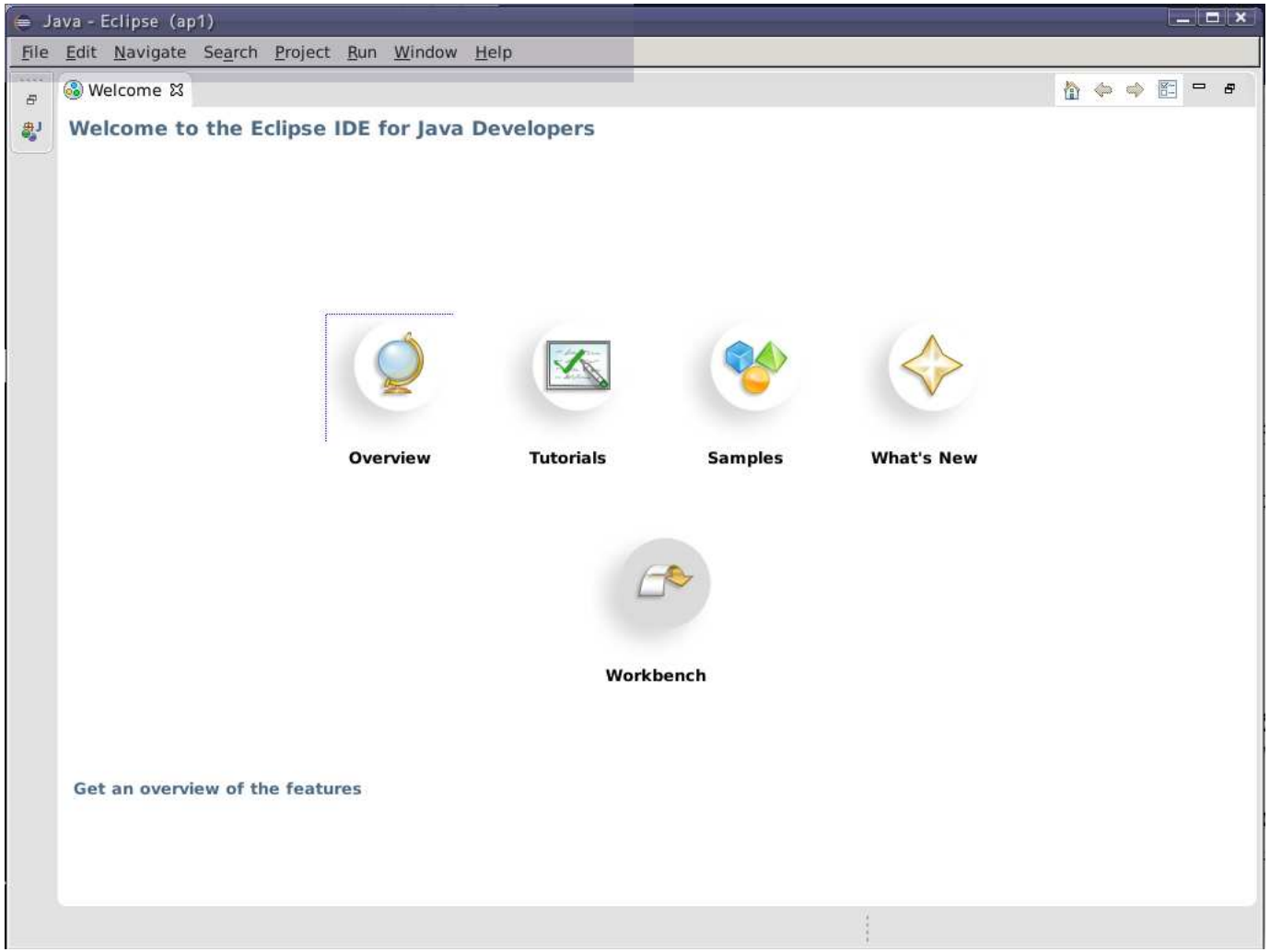
これに対しては、Workspace (i.e. 作業用ディレクトリ) を指定する入力フィールドに例えば

各自のホームディレクトリ /java-lab

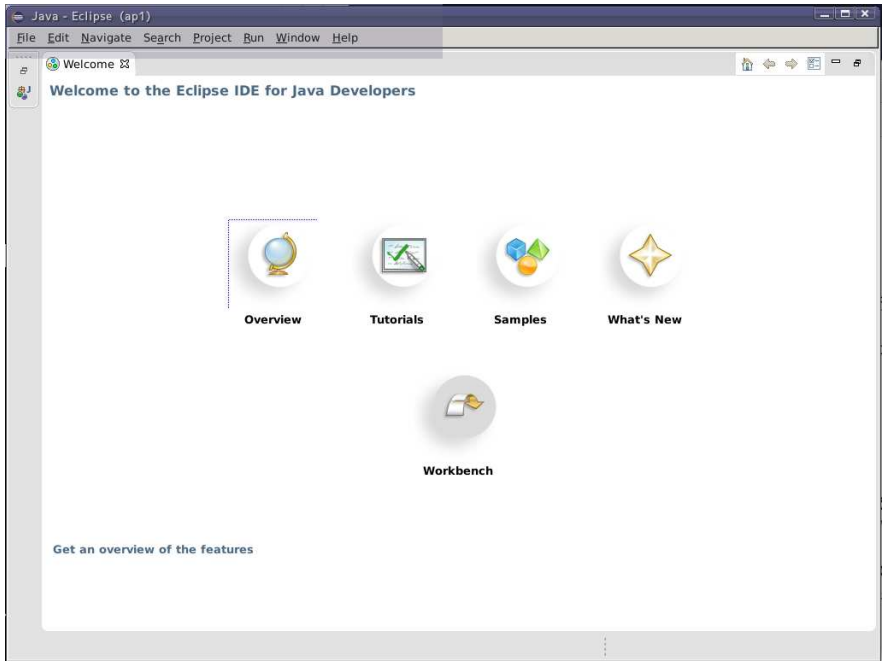
と入れて、右下の OK ボタンを押せば良い。ここで、

- 下の「Use this as the default and do not ask again」という記述の左にチェックマークを入れて OK ボタンを押すと、次回以降にEclipse を起動した際にこの質問小ウィンドウが出なくなる。

「Welcome」タブ: 最初にEclipseを起動した時は、Workspace指定の直後に次の様なウィンドウが現れる。
(メニューバーで Help→Welcome と選択しても表示できる。)



これに対して、...



これに対して、

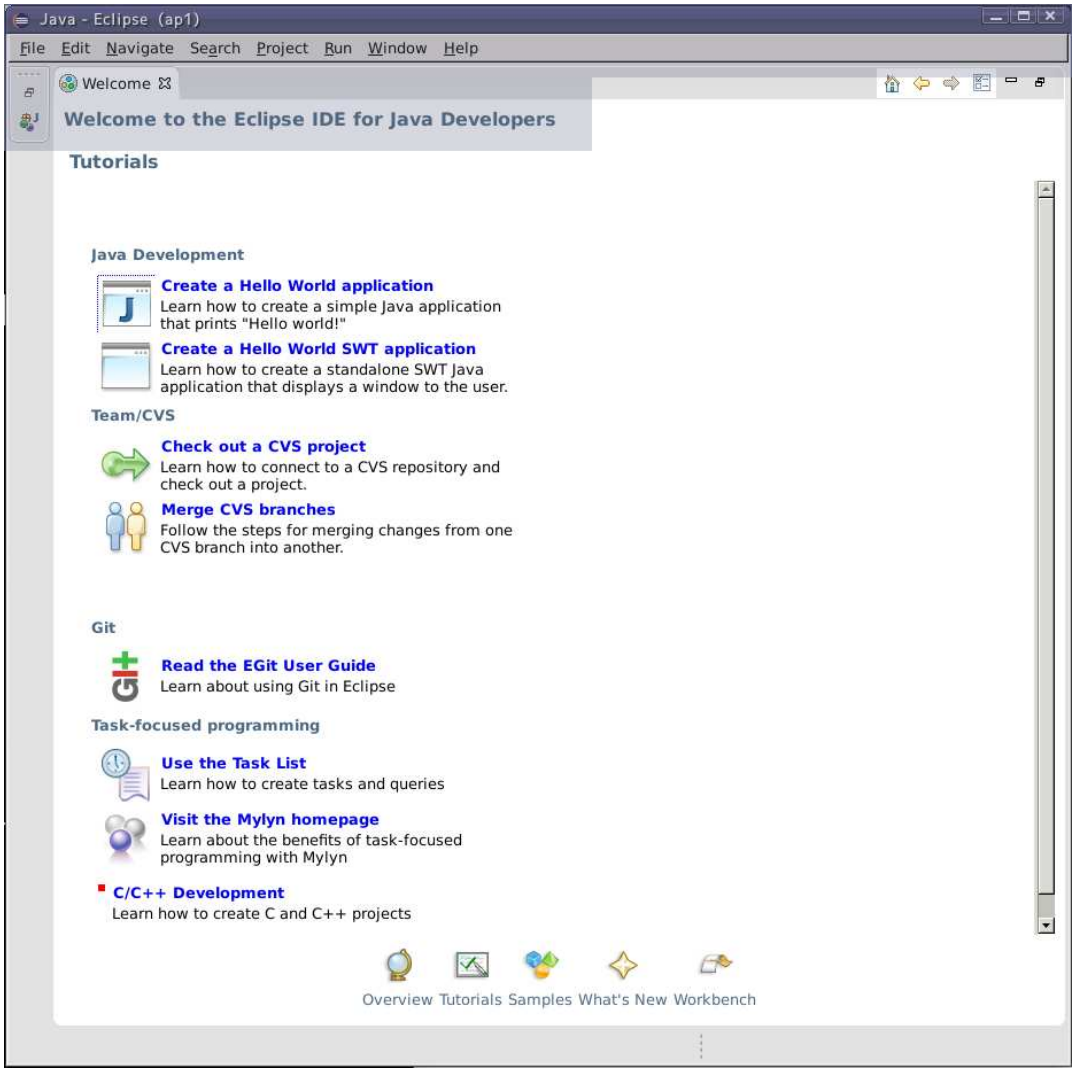
- 右端の 「Workbench」アイコンをクリックすると 、
  ⟶ プログラミング等のための基本画面に切り替わる。

- 「Welcome」タブの右側にある「×」アイコンをクリックしても、
  ⟶ 基本画面に切り替わる。

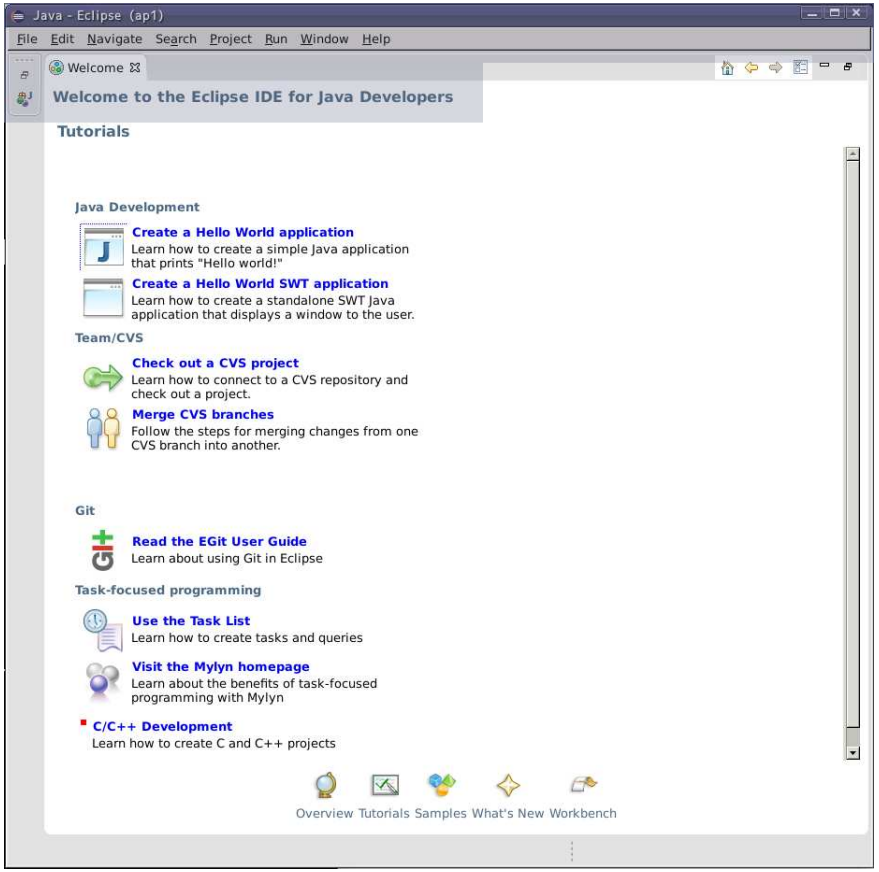
- 中央下付近(右側)の 「Turorials」アイコンをクリックすると 、
  ⟶ 画面内の指示に従ってプログラミング等の作業を行い、
  Eclipseの基本的な使用方法を自習するためのメニュー画面に

チュートリアル(Hello Worldアプリケーションの作成):

Welcomeタブシートで 「Turorials」アイコンをクリックすると 、実際にはWelcomeタブの表示内容が次の様なものに切り替わる。



これに対して、...



これに対して、

左上の「Create a Hello World application」を選ぶと、

⟶画面構成が変わり、Eclipse利用者が

画面右側に表示されたWelcomeタブ内の指示に従って

Javaプログラミングを行う際の基本的操作を経験

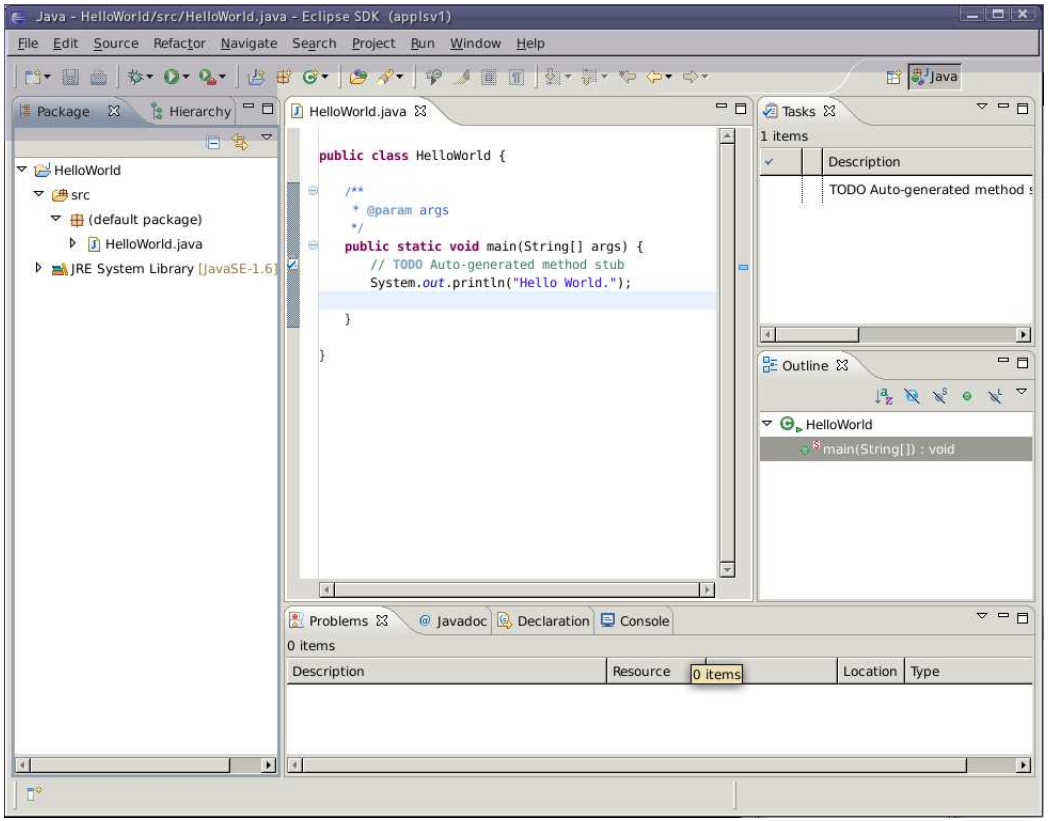
できる様になる。

□演習21．1（チュートリアルの利用）

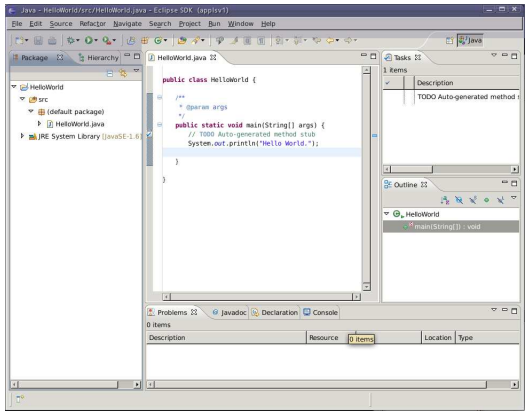
「Turorials」という表題の付いたWelcomeタブシートで、
左上の「Create a Hello World application」を選び、
簡単なJavaアプリケーションの作成・実行を実際にEclipse上で経験してみて下さい。

21–2 Eclipseの画面構成 ---パースペクティブ

Eclipseにおける画面構成の仕方: EclipseでJavaプログラミングを行う際は、Eclipseの標準的な画面構成は次の様になっている。



一般に、Eclipseにおける作業画面(ワークベンチ と呼ぶ)の構成は次の様になっている。



- タイトルバーのすぐ下にメニューバー
- メニューバーのすぐ下に、ツールバー
- 画面中央に、エディタ

- エディタ領域の左側, 下側, 右側に、参考情報を表示するためのサブウィンドウ(ビュー と呼ぶ)。

特にJavaプログラミングの際は、次の様なビューをよく使う。

◇ Packageビュー(左側) ··· プロジェクト内のパッケージやソースファイルの配置を一覧するためのもの。

◇ Hierarchyビュー(左側) ··· 選択したクラスやインタフェースの継承関係を表示するためのもの。

◇ Tasksビュー(右側) ··· コメント部内の "TODO", "FIXMR", "XXX" という文字列(タスクタグという)は、その場所付近に何らかの未対応のプログラミング作業が残っていることを示している。これらの場所等の情報は、この中に一覧表示される。表示された行をダブルクリックすると、エディタ上で対応するタスクタグの場所にカーソルが移動する。

◇ Outlineビュー（右側）··· エディタで開いているクラス内の要素（e.g. フィールド，メソッド）が表示される。表示された要素をクリックするとエディタ上ではその要素の宣言部にカーソルが移動する。

◇ Problemsビュー（下側）··· コンパイルエラーや警告の情報は、このビューの中に一覧表示される。表示された中の行をクリックするとエディタ上では該当する行にカーソルが移動する。

◇ Consoleビュー（下側）··· プログラム実行による標準出力、標準エラー出力への出力は、このビューの中に表示される。 標準入力への入力もこの中にキーボードから打ち込む。

◇ Javadocビュー（下側）··· エディタ上で選択した要素に関するドキュメンテーションコメントが、このビューの中に表示される。

◇ Declarationビュー（下側）··· エディタ上で選択した要素の宣言部分のコードが、このビューの中に表示される。

- 画面最下段に、情報表示用のバー（ステータスバー という）

パースペクティブ： Eclipseにおいては、目的の作業用にメニューバー，ツールバー，エディタ，有用なビューを組み合わせて配置した画面構成をパースペクティブ（perspective）と呼んでいる。Eclipseで標準的に用意されているパースペクティブとしては、次の様なものがある。

- Javaパースペクティブ ··· Javaのプログラム開発時に利用。
- デバッグパースペクティブ ··· デバッグ実行時に利用。
- Java参照パースペクティブ ··· Javaのクラスやメソッドなどのソースコードを検索表示したい時に利用。
- Javaの階層型パースペクティブ ··· 階層ビューの視点でJavaファイルを閲覧・編集したい時に利用。
- リソースパースペクティブ ··· OSのファイルシステムのレベルでファイルを閲覧・編集したい時に利用。
- XMLパースペクティブ ··· XMLファイルを閲覧・編集したい時。
- JavaScriptパースペクティブ ··· Ajaxアプリケーション開発時。
- Webパースペクティブ ···
- ·····

パースペクティブの選択・切替え:

ツールバーの右側にパースペクティブの切替えボタン類があり、現在使用中のパースペクティブについてはボタンが押された状態になっている。多数のパースペクティブを開いていて全てのボタンを表示し切れない場合は、右端に ‘‘>>’’ アイコンが配置されこれで選べる様になっている。

- 既に開いた中の別のパースペクティブに切り替えたい場合 は、
  $\longrightarrow$ 対応する切替えボタンを探して押すだけである。

- Eclipseが提供するパースペクティブを新たに開きたい場合 は、
  $\longrightarrow$ 現在のパースペクティブボタンの左にある  ボタンを押すか、
  メニューバーで Window→Open Perspective→... と選択

パースペクティブのレイアウト変更:

- 新しいビューを開きたい場合 は、
  $\longrightarrow$ メニューバーで `Window→Show View→...` と選択

- ビューの表示位置を移動したい場合 は、
  $\longrightarrow$ ビューのタブの部分をドラッグ&ドロップ

21–3 Eclipseの下でのプログラム作成・実行

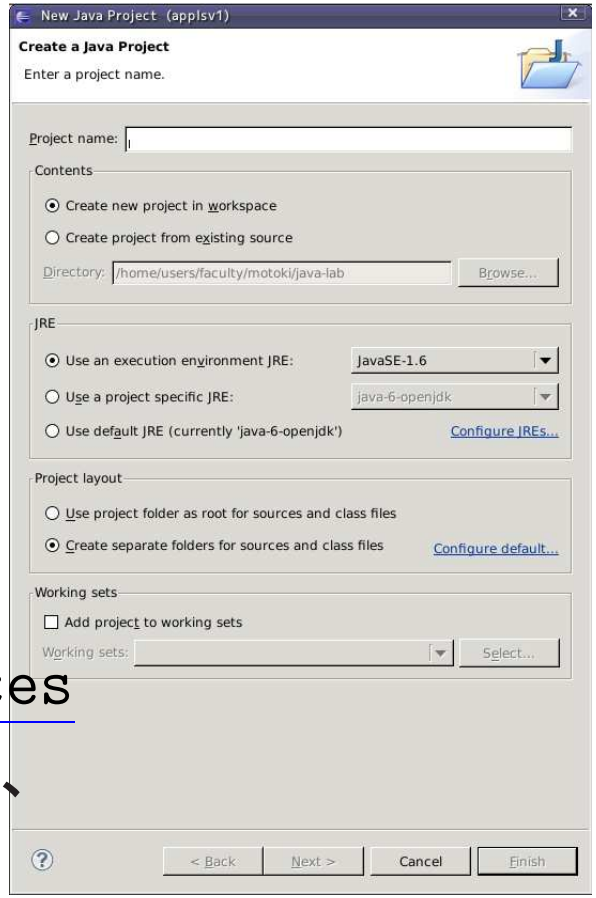
Javaプログラム作成・実行の基本手順:

(0) Eclipseを起動 ··· eclipse&

(1) Javaパースペクティブを開く ··· Eclipse画面がJavaパースペクティブになっていない場合は、例えばメニューバーで
Window→Open Perspective→ other→Java(default) と選択

(2) Javaプロジェクトの作成(未作成の場合) ··· 新規プロジェクトでJavaプログラミングをする場合は、Eclipse起動時に指定したワークスペース内にこのプロジェクト用にディレクトリを設定する。
これをEclipse 上で行うためには 、
⟶ 例えばツールバー内の New Java Projectボタン を押す。
⟶次の様なダイアログ(質問ウィンドウ)が現れる。

このダイアログに対しては、
上部の「Project Name」フィールド
にプロジェクト名を入れ、(必要に応じて)
残りの項目を確認・設定した上で、
ウィンドウ右下の「Finish」ボタン
を押せば良い。



ちなみに、「Project layout」欄で

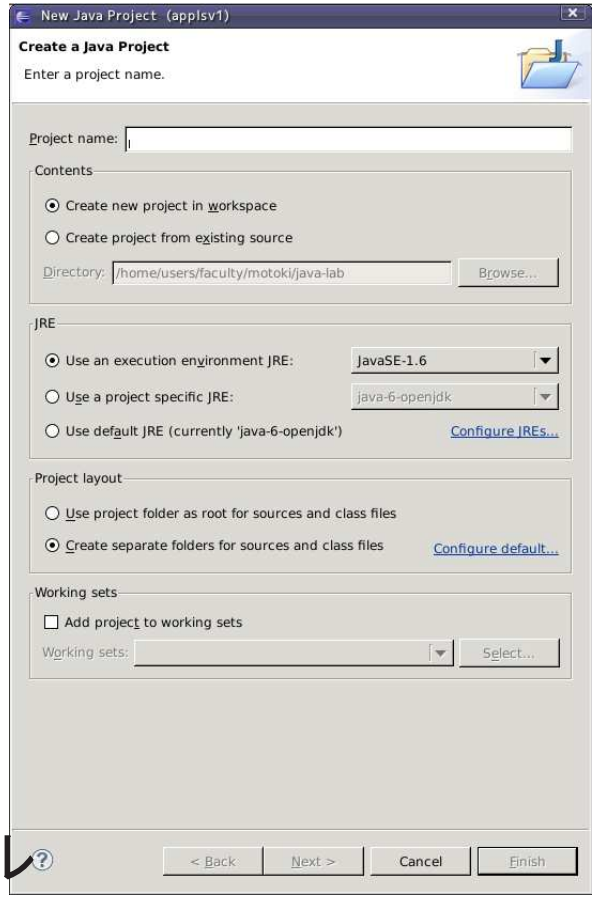
◇Use project folder as root for sources and class files という選択肢を選ぶと、所属パッケージを指定しないプログラムの場合 、作成するソースファイルもコンパイル結果であるクラスファイルも

指定したワークスペース / 指定したプロジェクト名 /

というディレクトリ内に置かれることになる。

◇Create separate folders for sources and class files という選択肢(デフォルト) を選ぶと、...

このダイアログに対しては、
上部の「Project Name」フィールド
にプロジェクト名を入れ、(必要に応じて)
残りの項目を確認・設定した上で、
ウィンドウ右下の「Finish」ボタン...



ちなみに、「Project layout」欄で
◇Use project folder ... を選ぶと、

.....

◇Create separate folders for sources and class files という選択肢(デフォルト)を選ぶと、所属パッケージを指定しないプログラムの場合、
作成するソースファイルは

指定したワークスペース / 指定したプロジェクト名 /src/ ,

コンパイル結果であるクラスファイルは

指定したワークスペース / 指定したプロジェクト名 /bin/

というディレクトリ内に置かれることになる。

(3) パッケージの作成（所属パッケージを指定したいが未作成の場合）・・・

所属パッケージを指定するプログラムについては 、

パッケージ毎にプログラムを分類保存するために、

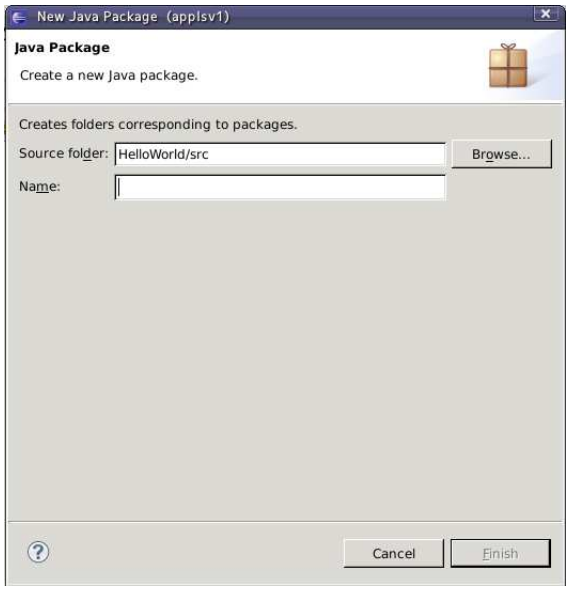
指定したワークスペース / 指定したプロジェクト名 /

というディレクトリ内に所属パッケージ名に対応したサブディレクトリを設定する。

これをEclipse上で行うためには 、

⟶例えばツールバー内の New Java Packageボタン を押す。

⟶次の様なダイアログ（質問ウィンドウ）が現れる。



このダイアログに対しては、...

このダイアログに対しては、
「Name」フィールドにパッケージ名を入れ、
「Source folder」フィールドの内容を
確認・設定した上で、ウィンドウ
右下の「Finish」ボタンを押せば良い。

ちなみに、 所属パッケージを指定する
プログラムについては 、
プロジェクト生成時に「Project layout」欄で

◇Use project folder as root for sources and class files という選択肢を選んでいた場合は、作成するソースファイルもコンパイル結果であるクラスファイルも

指定したワークスペース / 指定したプロジェクト名
/ 所属パッケージ名中の"."を"/"に置き換えた文字列 /

というディレクトリ内に置かれることになる。

◇Create separate folders for sources and class files という選択肢（デフォルト）を選んでいた場合は、...

このダイアログに対しては、
「Name」フィールドにパッケージ名を入れ、

. . . . . .

右下の「Finish」ボタンを押せば良い。

ちなみに、 所属パッケージを指定する
プログラムについては 、
プロジェクト生成時に「Project layout」欄で
◇Use project folder ... を選んでいた場合は、...
◇Create separate folders for sources and class files という選択肢を選んでいた場合は、作成するソースファイルは

指定したワークスペース / 指定したプロジェクト名 /src
/ 所属パッケージ名中の"."を"/"に置き換えた文字列 / ,

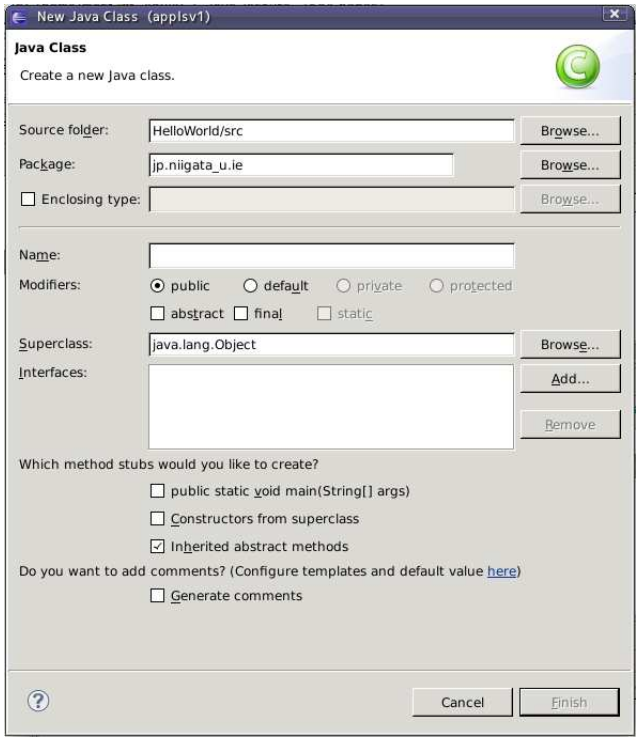
コンパイル結果であるクラスファイルは

指定したワークスペース / 指定したプロジェクト名 /bin
/ パッケージ名中の"."を"/"に置き換えた文字列 /

というディレクトリ内に置かれることになる。

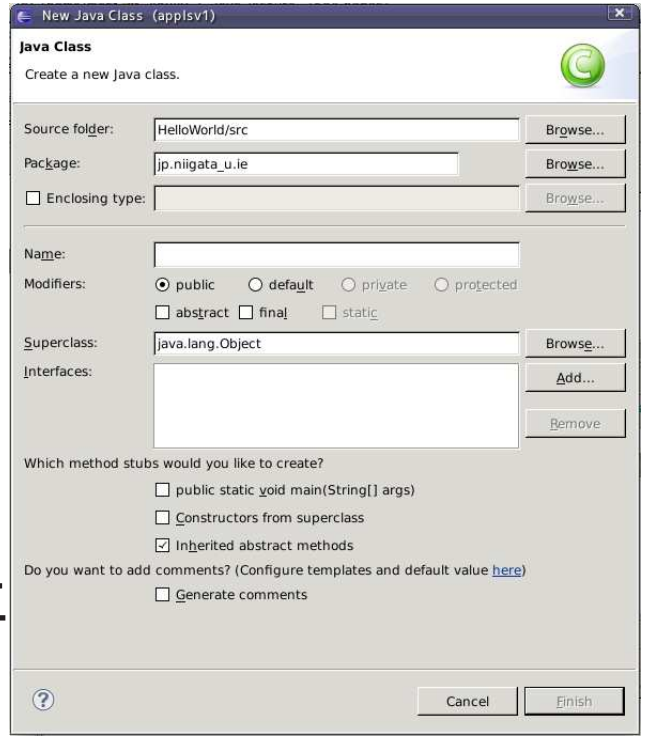
(4) Javaクラスの作成（未作成の場合）··· Eclipse上でクラス定義を新たに始めたい場合は、例えば
左側の「Package」ビュー内で
該当するパッケージ名のアイコンを選んだ後に
ツールバー内の New Java Classボタン を押す。

⟶次の様なダイアログ（質問ウィンドウ）が現れる。



このダイアログに対しては、...

このダイアログに対しては、
「Name」フィールドにクラス名を入れ、
残りの項目を確認・設定した上で、
右下の「Finish」ボタンを押せば良い。



ちなみに、

◇「Which method stubs ... 」欄で、
「public static void main(...)」に
チェックマークを入れると、mainメソッド定義の枠組みが最初からソースファイルの中に組み込まれる。

◇「Do you want to add comments?」欄で、
「Generate comments」にチェックマークを入れると、
クラス宣言の直前にドキュメンテーションコメントを入れるための行が最初からソースファイルの中に組み込まれる。

◇「Enclosing type」欄の 左にチェックマーク を入れ、この
フィールドに既存クラスを指定 すると、この既存クラスの中に
新たな（入れ子）クラスを定義する枠組みが挿入される。

(5) Java コードの編集 ··· 左側の「Package」ビュー内で編集したいクラスを選びソースを画面中央のエディタ領域に表示して編集、という作業を繰返す。 エディタ内では 、

◇ 行末で Enter を打つと、次の行の適切な字下げが行われた場所にカーソルが移動する。

◇ 普通に打ち込んでも良いが、コンテンツアシスト（コードアシスト, 補完機能）やクイックフィックス（エラー修正候補表示とその選択に基づく修正）といった機能を活用することで、キーボードからのコード入力作業を大幅に減らすことができる。

例えば、s と入力して Alt-/キーを押すと、
⟶ 入力候補が表示される。
続いて yso と入力し "System.out.println();" という入力候補が表示された時点で Enter キーを押せば、
⟶ 実際にキーボードから入力した "syso" という文字列は "System.out.println();" に変化

◇クラス内の各々の要素のコードを折りたたみ表示（i.e.1行にまとめて簡略表示）できる。 左端付近の縦長の領域内 に置かれた

$\ominus$ ··· 要素のコードが折りたたまれていないことを表し、
$\oplus$ ··· 要素のコードが折りたたまれていることを表す。

これらのアイコンをクリック ⟶切り替え。
$\oplus$アイコンの上にカーソルを移動
⟶ 本来のコード内容がポップアップで表示される。

◇エディタ領域 左端にある縦長の棒を右クリック し、
出てきたメニューで 「Show Line Numbers」を選択
⟶ 行番号を表示 するかどうかの切り替え

(6) ソースファイル保存 ···

例えばツールバー内の「Save」ボタンを押す。

⟶ 自動的にコンパイルも行われる。

エラーが検出されると 、

⟶ その行の左に赤い⊗印。ここで、

◇ この ⊗印の上にマウスカーソルを移動 させると、

⟶ エラー内容（や対処法）が表示される。

◇ ⊗印の横に 電球も表示されている場合 は、

電球アイコンをクリック して

表示される 修正候補の中から適切なものを選ぶ ことによって、意図した修正をソースコード上に反映させることもできる。

(7) 実行 ··· 例えば、

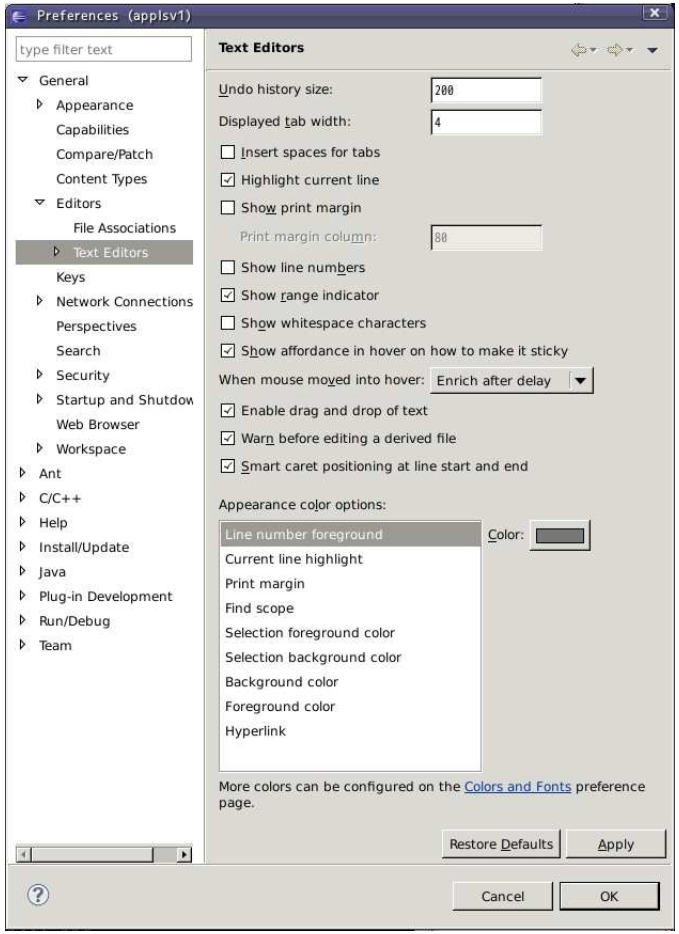
画面左の 「Package」ビュー内のソースファイルを右クリック し
現れるメニューで 「Run as」→「Java Application」 と選択。

⟶ 正常に実行が進んだ場合 、
実行結果は下部の「Console」ビュー内に表示される。

標準入力への入力が必要な場合
「Console」ビューの領域の中に入力文字列を打ち込む。

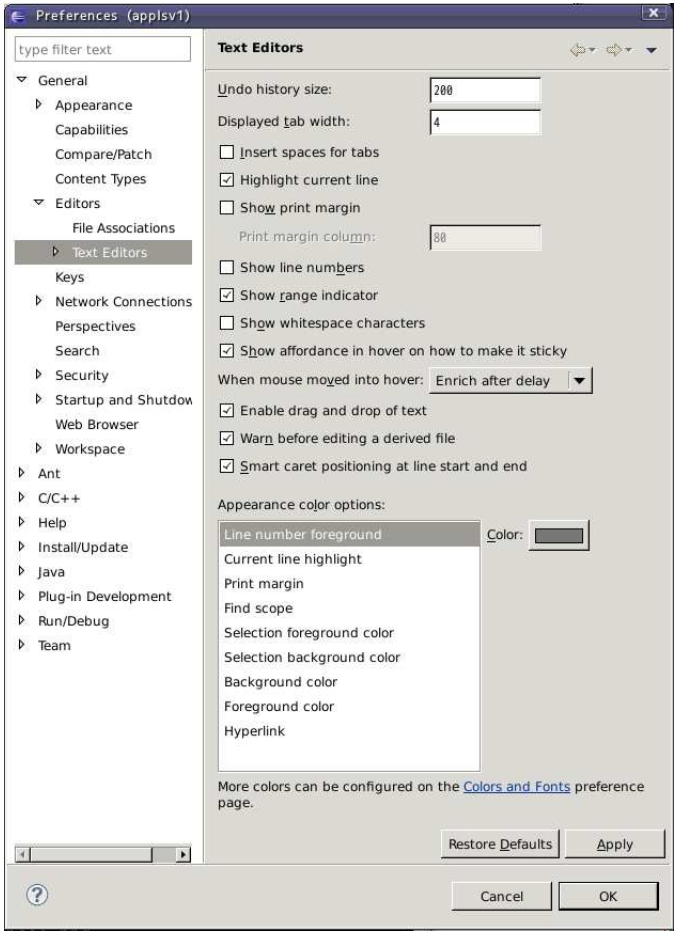
エディタの設定：

メニューバーで Window→Preferences と選択し、
出てきた「Preferences」ウィンドウ左の階層構造内で
General→Editors→Text Editors と選択
⟶ 「Preferences」ウィンドウは次の様に変わる。



ここで、このウィンドウ中の .....

ここで、このウィンドウ中の



◇ 「Show line numbers」にチェックマークを入れて「OK」ボタンを押すと、

⟶ エディタ上で行番号 が表示される様に。

> 補足：
> エディタ領域左端にある縦棒を右クリックし、出てきたメニューで
> 「Show Line Numbers」を選択してもよい。

◇ 「Show print margin」にチェックマークを入れて「OK」ボタンを押すと、

⟶ エディタ上に印刷マージン

(i.e. 印刷の際のおおよその限界ライン，デフォルトでは80桁まで)

が表示される様になる。

プログラム作成支援の機能： Eclipseにおいては、プログラム作成を支援する機能として次の様なものが備わっている。

- コンテンツ・アシスト（コード・アシスト，補完機能）・・・

  Alt-/ （Windowsでは Ctrl-Space）を押すと、

  ⟶ 現在入力中の文字列を補完する入力候補が一覧表示される。

  この中から目的のメソッドやクラス，テンプレート（雛形）を 選択 し Enter キー

  ⟶ 入力した文字列を選択した記述列に置き換え

- クイック・フィックス（即時修正）・・・

  警告のある部分にカーソルを移動し、Ctrl-1 キーを押すと、

  ⟶ 修正方法の候補が一覧表示される。

  この中から妥当そうな項目を 選ぶ ことで、

  ⟶ 修正をソースコード上に反映

- テンプレートの生成 ··· よく使う定形コードの生成は、
  エディタの ソース領域内を右クリック して現れるメニューで
  （あるいはメニューバーで）
  Source→目的に応じた項目 と選択することによって行行える。

- リファクタリング ··· ソフトウェアの外部的振舞いを保ったままで内部の構造を改善していく作業を、一般にリファクタリングという。

  具体的にEclipseでリファクタリングを進めるには 、

  ◇ Packageビュー内のソースファイルを右クリック して現れるメニューで Refactor→目的に応じた項目 と選択したり、

  ◇ ソースコード内の関連箇所にカーソルを移動し右クリック して現れるメニューで Refactor→目的に応じた項目 と選択したり、あるいは

  ◇ ソースコード内の関連箇所にカーソルを移動 し Ctrl-1 キーを押したりする。

コード閲覧支援の機能： Eclipseにおいては、コードを読むのを支援する機能として次の様なものが備わっている。

- **Packageビュー** ··· プロジェクト全体 の状況(e.g.パッケージ階層)を把握するのに役立つ。
- **Outlineビュー** ··· 編集中の クラスの概要 を把握するのに役立つ。
- **宣言部の参照** ··· Outlineビュー内で 要素をクリック すると、
  ⟶ Declarationビュー内にその宣言部が表示される。
- **宣言部への移動** ··· ソースコード内の関連箇所にカーソル移動した後に、F3キーを押すか 右クリックして現れるメニュー で「Open Decl
  という項目を選択する。
- **メソッドの呼出し階層を開く** ··· ソースコード内のメソッド呼出し部にカーソル移動した後に、 右クリックして現れるメニュー で「Open Call Hierarchy」という項目を選択すると、Call Hierarchyビュー(無ければ生成される)内に階層が表示される。

- 型階層の表示 ··· ソースコード内の変数等にカーソル移動した後に、右クリックして現れるメニュー で「Open Type Hierarchy」という項目を選択すると、Hierarchy ビュー内に関連した型の階層が表示される。また、同じメニューで「Quick Type Hierarchy」という項目を選択すると、関連した型の階層が一時的にポップアップ表示される。

- ·····

(その他、
ソースコード内の関連箇所にカーソル移動した後に、
右クリックして現れるメニュー で色々選択できる。)

Eclipseの主要ショートカット一覧：

メニューバーで Help→KeyAssist と選ぶと、

⟶ ショートカットキーの一覧が表示される。

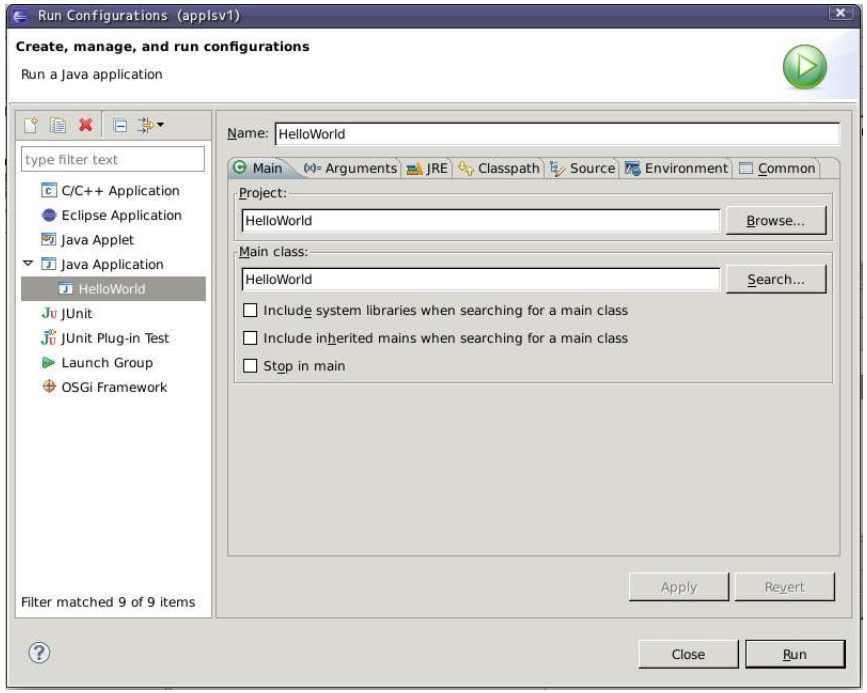
このうちの主要なものを次に示す。

ショートカット名称	キー	概要
Delete Line	Ctrl-D	行削除
Undo	Ctrl-Z	前操作の取消
Next Page	Alt-F7	次ページ
Previous Page	Shift-Alt-F7	前ページ
Last Edit Location	Ctrl-Q	前回の編集場所
Save	Ctrl-S	ファイル保存
Run Java Application	Shift-Alt-X J	アプリケーションプログラムを実行
Debug Java Application	Shift-Alt-D J	デバッグモードでアプリケーションプログラムを実行

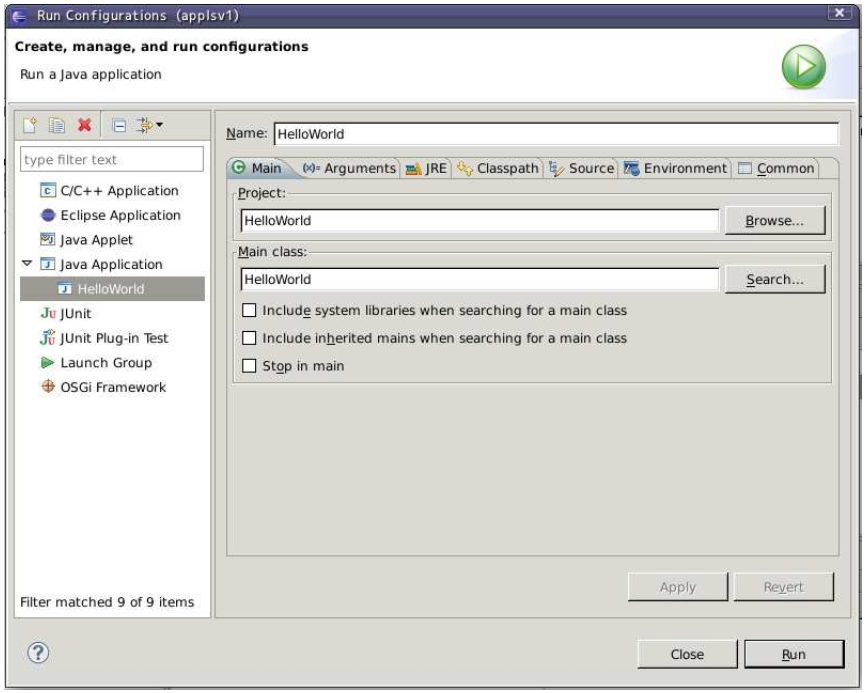
ショートカット名称	キー	概要
Content Assist	Alt-/	補完（候補案を提示）。Windows上では Ctrl-Space
Quick Fix	Ctrl-1	エラー等の修正（候補案を提示）
Quick Assist -- Assign to field	Ctrl-2 F	フィールドへの代入式を記述
Quick Assist -- Assign to local variable	Ctrl-2 L	局所変数への代入式を記述
Quick Assist -- Rename in file	Ctrl-2 R	
Quick access	Ctrl-3	文字列検索を使ってEclipseの機能にアクセス
Open Declaration	F3	指定した要素の宣言に移動
Open Type Hierarchy	F4	指定した要素に関連した型階層を表示

ショートカット名称	キー	概要
Refresh	F5	Packageビューの更新(最新表示)
Open Type	Ctrl-Shift-T	型を指定してそのソースファイルを開く。
Quick Outline	Ctrl-O	Outlineビューの内容が一時的にポップアップ表示される。この表示内で要素を選んでその宣言部に移動することも可能。
Quick Hierarchy	Ctrl-T	型階層の表示
Go to Line	Ctrl-L	行番号を指定してファイル内を移動
Backward History	Alt-←	ナビゲーション操作による移動履歴に基づいて、以前の場所に戻る
Forward History	Alt-→	ナビゲーション操作による移動履歴に基づいて、先の場所に進む

作成プログラムの起動に関する設定： プログラムに引数等を指定したい時は、 Package ビュー内のソースファイルを右クリック して現れるメニューで Run As→Run Configurations... と選択する。これを行うと細部を設定してプログラム実行を行うための次の様なウィンドウが現れる。



これに対して、...



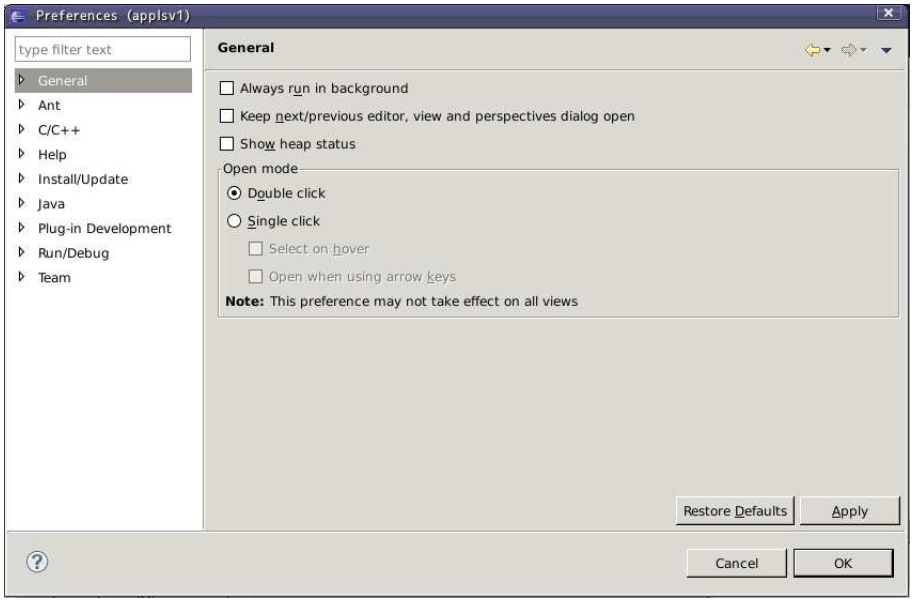
これに対して、

◇ プログラムに引数を指定したい場合は、

⟶ ウィンドウ内の Arguments タブのシートを開いて、
「Program arguments:」という行の下のデータ入力欄に
引数として与える文字列を空白で区切って並べた上で、
ウィンドウ右下の Run ボタンを押せば良い。

コンパイラとコードフォーマッタの設定確認：

メニューバーで Window→Preferences と選択することにより、コンパイラとコードフォーマッタの設定を確認することができる。これを行うと次の様なウィンドウが現れる。



これに対して、...

これに対して、

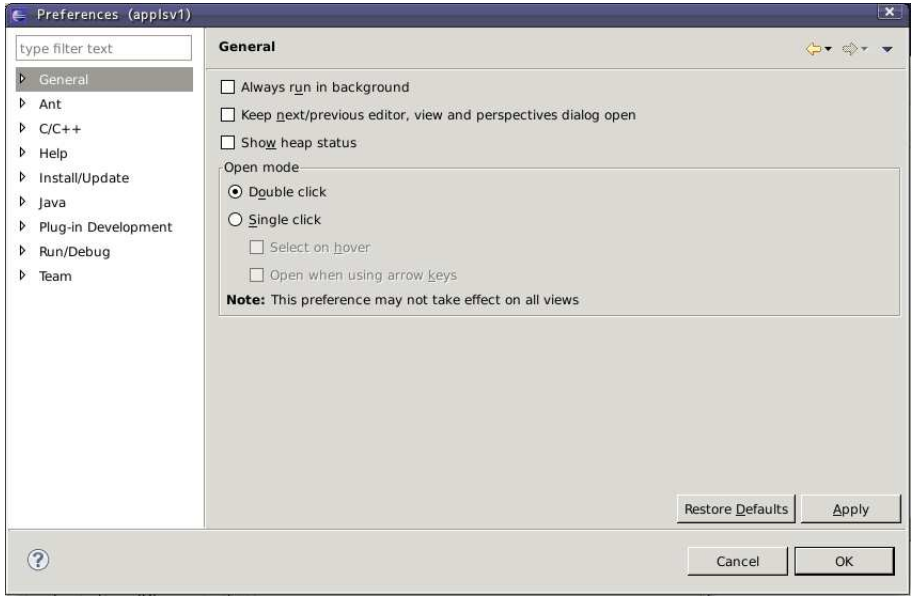
◇ コンパイラの警告レベルの設定を見たい場合 は、

⟶ 画面左の階層構造内で

Java→ Compiler

→Errors/Warnings

と選択する。



◇ コードフォーマッタの設定を見たい場合 は、

⟶ 画面左の階層構造内で Java→Code Style →Formatter

と選択し、現れる表示右上 Edit... ボタンを押す。

このシートについては設定変更しない方が無難である。

◇ import 文の書き方に関する設定を見たい場合 は、

⟶ .....

◇ Save 時の自動的コード整形に関する設定を見たい場合 は、

⟶ .....

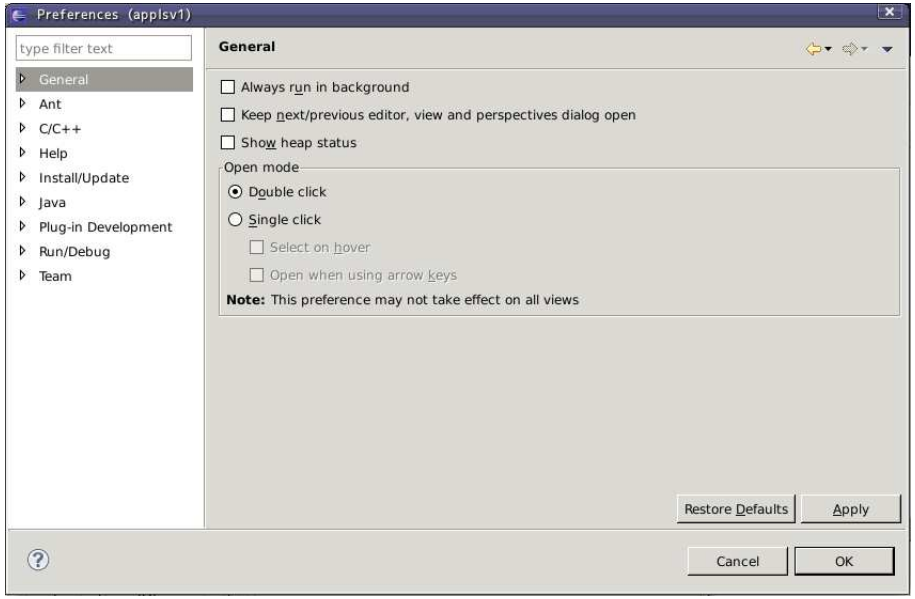
これに対して、

◇ コンパイラの警告レベルの設定を見たい場合 は、

⟶ .....

◇ コードフォーマッタの設定を見たい場合 は、

⟶ .....



◇ import 文の書き方に関する設定を見たい場合 は、

⟶ 画面左の階層構造内で Java→Code Style→Organize Import と選択する。

このシートについても設定変更しない方が無難である。

◇ Save 時の自動的コード整形に関する設定を見たい場合 は、

⟶ 画面左の階層構造内で Java →Editor→Save Actions と選択する。

21–4 デバッガの利用

デバッガ利用の基本手順:

(1) 中断点（ブレークポイント）の設定 ···

設定したい 行の左側

（コード領域外，エディタ領域の左端にある縦長の領域）

を ダブルクリック すると、

⟶ 中断点（breakpoint）を設定するかどうかの切り替えになる。

（ 中断点が設定 された行 . . . . . 左端に青丸の印）

(2) デバッグモードでプログラム実行 ···

Packageビュー内のソースプログラム名を右クリック して現れるメニューで、「Debug As」→「Java Application」と選択。

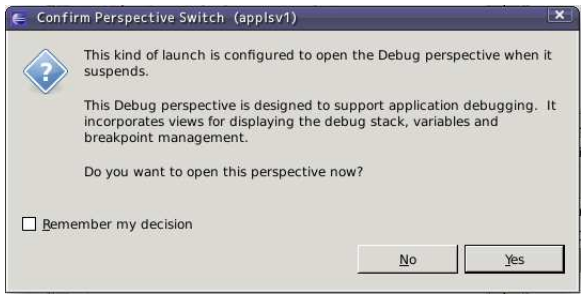
前ステップ(1)で

中断点が全然設定されていない場合 は、

⟶ プログラムが通常実行されるだけである。

中断点が設定されて最初にデバッグ実行しようとした時 は、

⟶ 「Confirm Perspective Switch」という表題の次の様なダイアログが出る。



これに対しては、右下の「Yes」ボタンを押す。

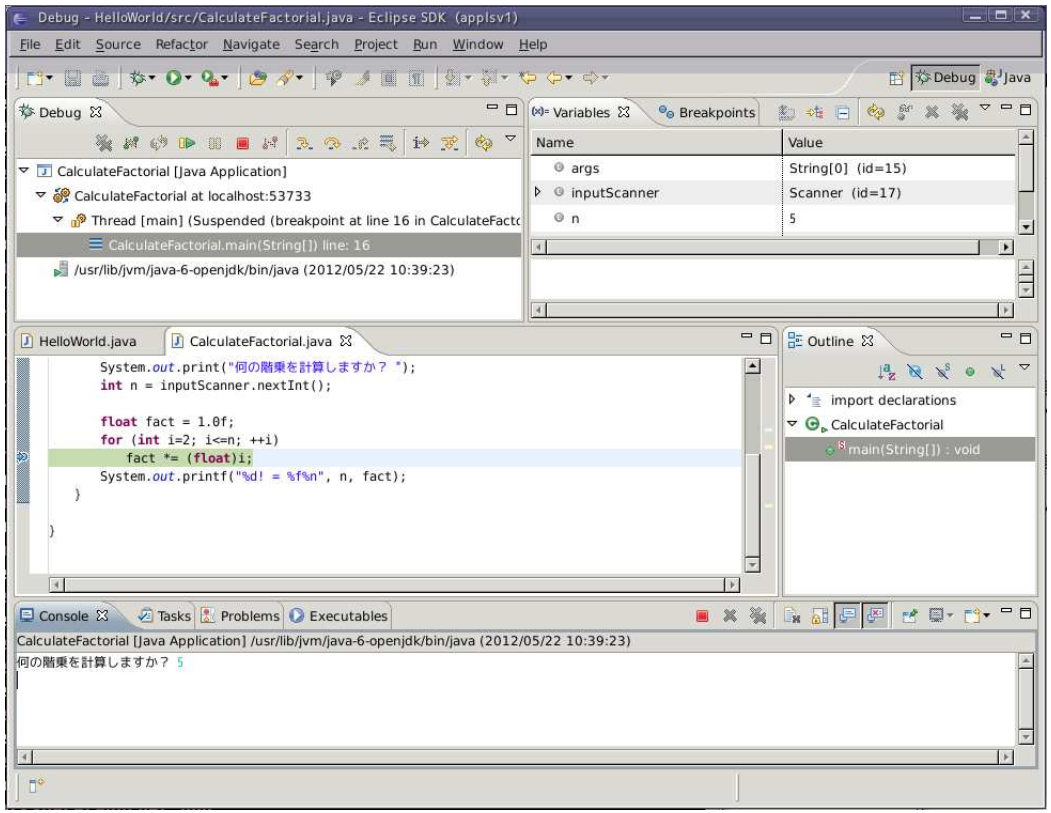
(2) デバッグモードでプログラム実行 ···

Packageビュー内のソースプログラム名を右クリック して現れるメニューで、「Debug As」→「Java Application」と選択。

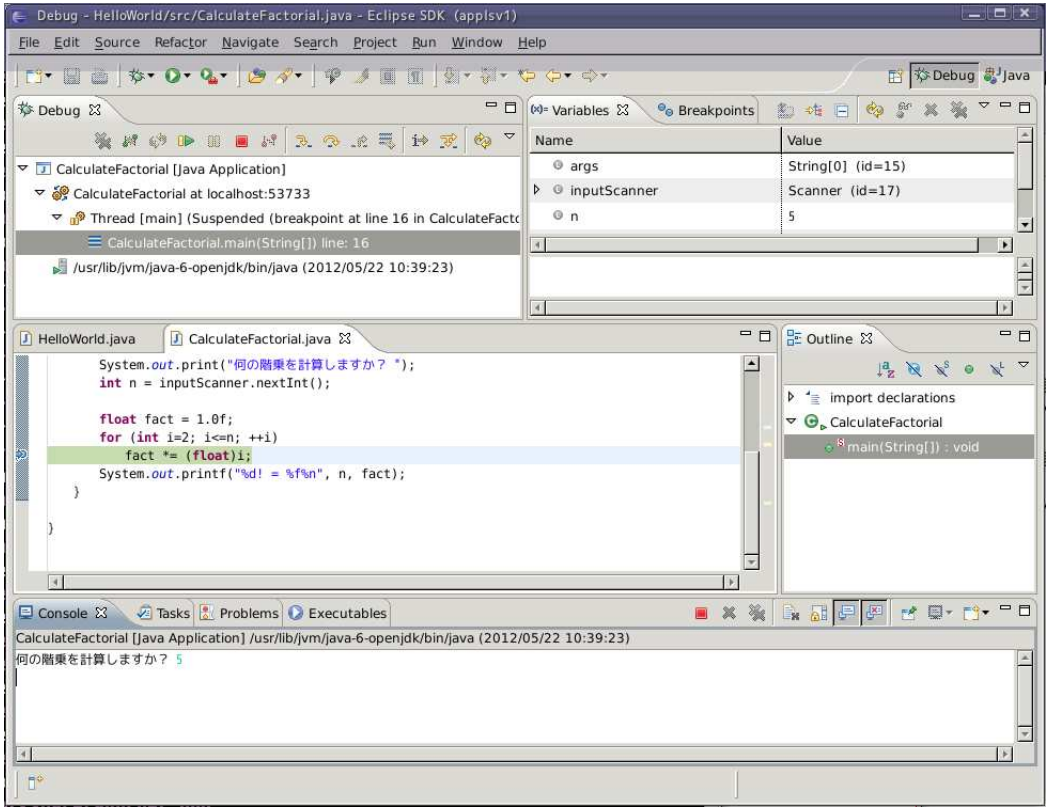
. . . . . .

前ステップ(1)で 中断点が設定されていれば 、

⟶ 結果的にデバッグパースペクティブ に切り替わり、
プログラムが最初の中断点手前で止まった状態が表示



デバッグパースペクティブの下では、次の様な操作が可能である。

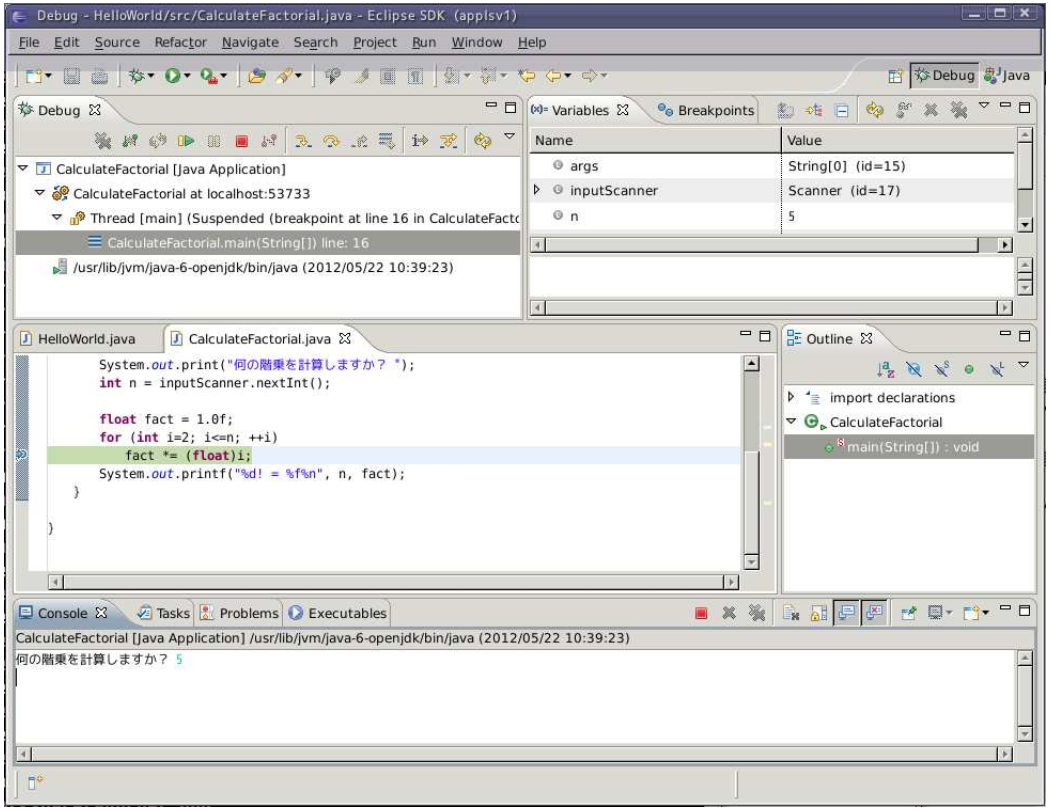


◇ 実行中断した時点での変数値の確認

パースペクティブ右側に配置される Variables ビュー 内を見る。

◇ 指定変数のエントリを値追跡のための Expressions ビュー の中に追加

Variables ビュー内の 変数名を右クリック して現れるメニューから 「Watch」 という項目を選ぶ。Expressions ビューが無かった場合は、これで生成される。



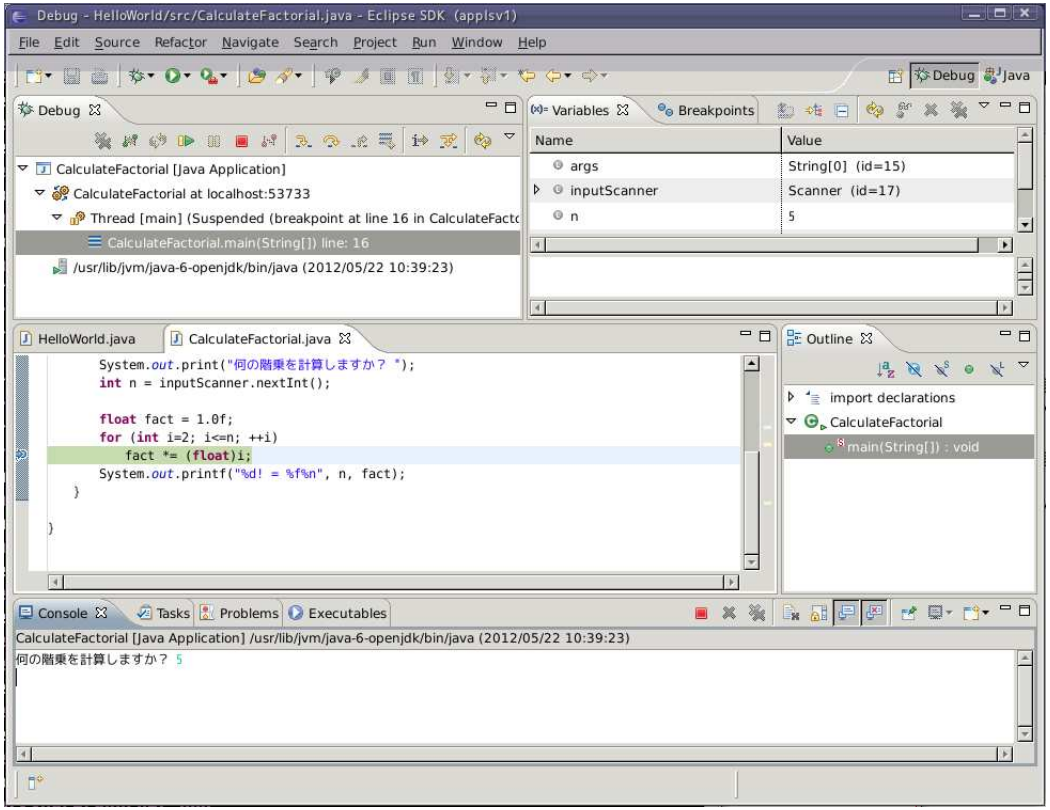
◇ 次の中断点まで実行を進める

Debugビューのツールバー内にある「Resume」ボタン を押す。

◇ ステップ実行

Debugビューのツールバー内にある次の3つのボタンによって、それぞれ実行を1ステップだけ進めることができる。

- Setp Over … 次の行（の手前）まで
- Step Into … 次がメソッド呼出しの場合その中に入ってステップ実行
- Step Return … メソッド呼び出し元に戻る



◇ 指定行まで実行

行にカーソルを合わせて
右クリックして現れる
メニューから 「Run to Line」という項目を選ぶ。

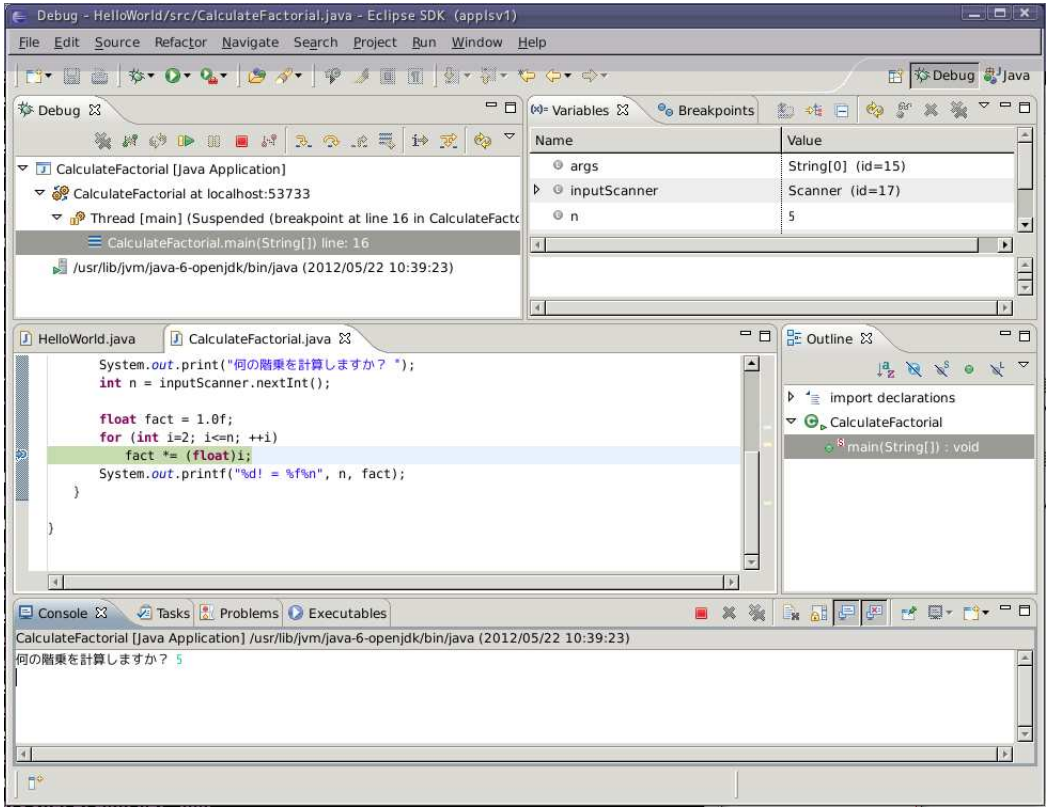
◇ 中断点を追加、除去

行だけでなくメソッドやフィールド，例外を中断点に設定できる。

例えば 例外の場合 は、 メニューバーで

Run→「Add Java Exception Breakpoint...」

と選択することにより行う。



◇ 変数の値を強制的に変更

「Variables」ビュー内で
変数を選択 した後に、
ビュー下部に表示される 値を変更 し、
右クリック→「Assign Value」と選択する。

◇ 実行中止

Debugビューのツールバー内にある 「Terminate」ボタン

◇ 再度実行

Eclipseツールバー内の 「Debug」ボタン を押す。

21–5 JUnit を用いた単体テスト

参考文献:
- 木村聡「Eclipse で学ぶ はじめての Java 第 2 版」ソフトバンククリエイティブ，2010．第 **19** 章「テスト」(**pp.375-402**)
- 日経ソフトウェア編「Java ツール完全理解」日経 BP 社，2011．第 **1** 部 **2** 章 (**pp.20-29**)
- 宮本信二「Eclipse3.6 完全攻略」ソフトバンククリエイティブ，2010．
  6.1 節「**JUnit** による単体テスト」, **6.2** 節「**JUnit** の基本」, **6.3** 節「**JUnit3** の利用」(まとめて **pp.321-334**)
- 森直樹「Java で学ぶ遺伝的アルゴリズム」共立出版，2007．
  4.5 節「テストファースト開発」(**pp.67-73**), **4.6** 節「リファクタリング」(**pp.73-74**)

内容を検討中