Eclipseの使い方

2015/10/1版　藤村 晃洋

目次

バグとデバッグ

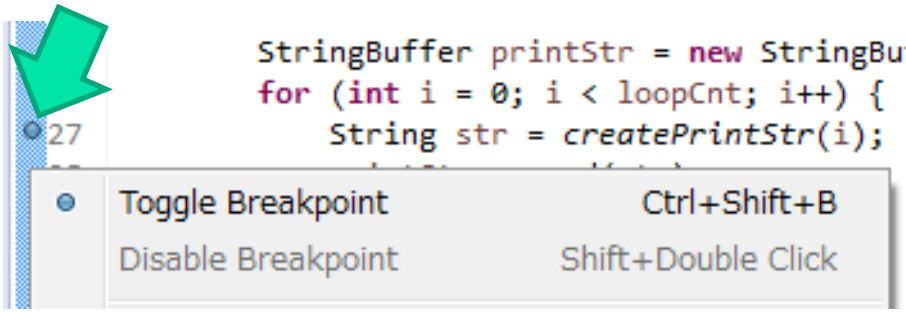
- バグ
  - プログラムが仕様通りに動かないこと
  - 例：小数点以下が四捨五入で計算される仕様のはずが切り捨てされている
- デバッグ
  - バグの原因を調査し、修正すること
- デバッガ
  - デバッグ作業を支援するツールのこと
  - Eclipseでは高機能なデバッガが付属している

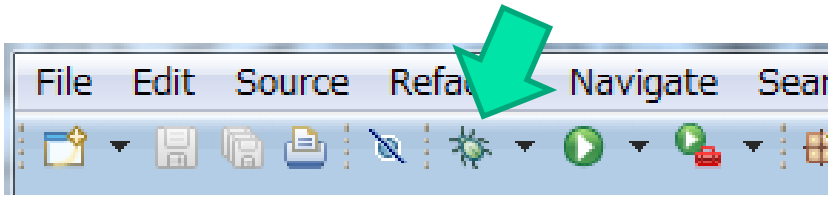
デバッガを使ったデバッグ

- バグの原因
  - 条件分岐などで想定されるパスを通らない
  - 変数に意図した値が入らない
- バグの原因調査に使えるデバッガの機能
  - ブレークポイント
    - プログラムの実行を指定して位置で停止させる
    - 停止した時点での変数の値を確認できる
  - ステップ実行
    - プログラムを一行ずつ実行できる
- 怪しい箇所にブレークポイントを付け、ステップ実行で意図した動作をしているかを確認していく

ブレークポイント 1/2

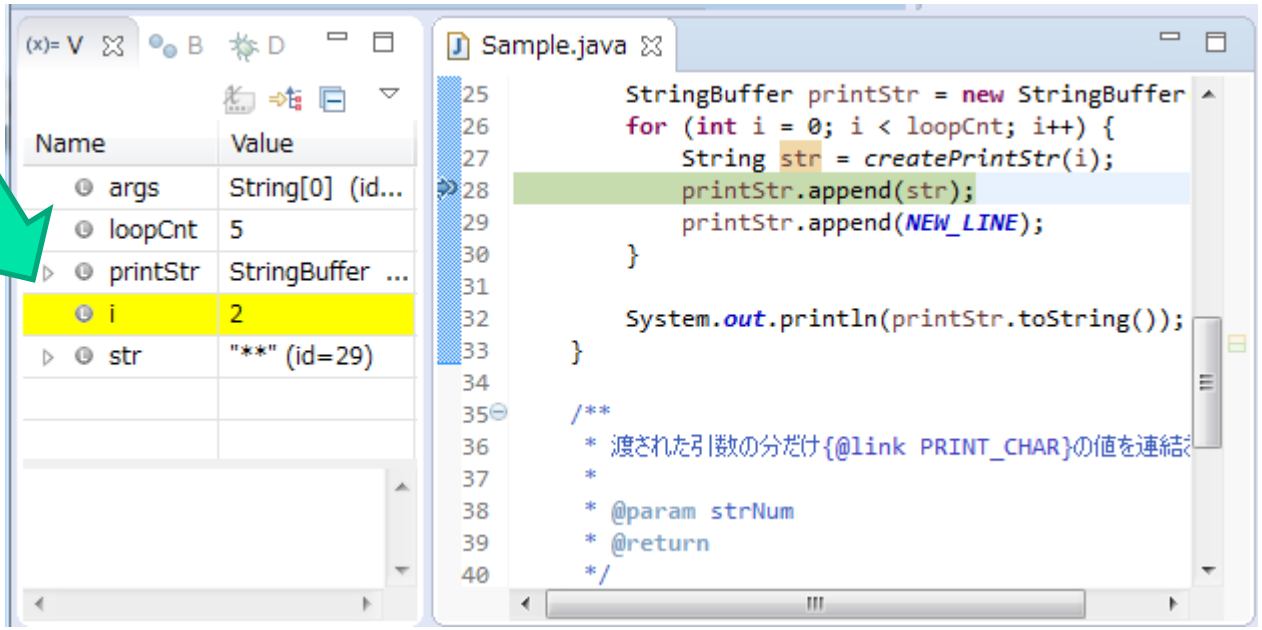
- プログラムの実行を途中で停止できる
  - ソースコードタブの左端をダブルクリック or
    右クリックメニュー→ブレークポイントの切り替えでブレークポイントを作成
  - デバッグ実行(虫のアイコンから実行 or F11)
    するとブレークポイントを設定した箇所で
    プログラムの動作が停止する





ブレークポイント 2/2

- ブレークするとその時点での変数の値が見える



- 発展：条件付きブレークポイント
  - 変数の値など条件を満たす場合のみブレークできる

ステップ実行

- ブレーク後、プログラムを１行ずつ実行できる
- ステップイン(F5)
  - １行実行。実行行がメソッドならその中に入る
- ステップオーバー(F6)
  - １行実行。実行行がメソッドならその中を一気に実行する
- ステップリターン(F7)
  - 現在のメソッドの処理を全て実行し、メソッド呼び出し元に戻る
- 再開(F8)
  - 次のブレークポイントまで実行する

ショートカット 1/3

- 型宣言を開く(F3)
  - 変数、メソッドに対して使う→宣言箇所に移動
  - クラスに対して使う→該当ファイルを開く
  - デコンパイラ(後述)があれば、classファイルも見れる
    - JD-Eclipse
- 型階層を開く(F4)
  - クラスの継承ツリーが見れる
- クイック型階層(Ctrl + T)
  - メソッドをオーバーライドしているクラスが見れる

ショートカット 2/3

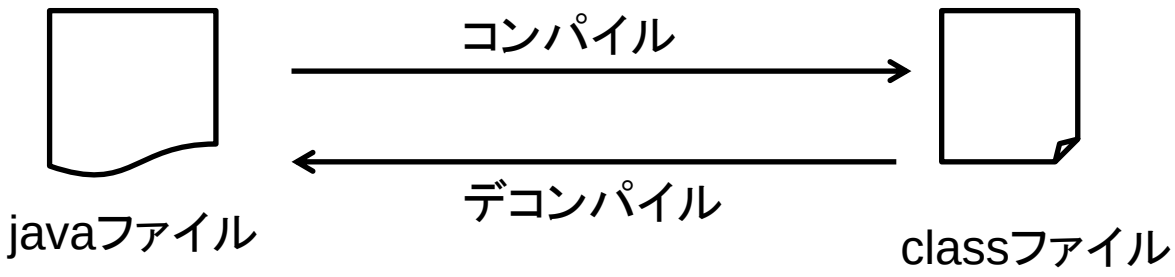
- 呼び出し階層を開く(Ctrl +Alt + H)
  - メソッドを呼び出している箇所を全て表示
  - 知らないメソッドの使い方を知れる
- コード補完(Ctrl + Space)
  - クラス名や変数名を書いている途中に押すと、候補を表示してくれる。インポートもしてくれる
    - [syso]と打ってコード補完すると・・・
- 名前変更(Alt + Shift + R)
  - 変数名やクラス名を変更し、参照しているコードも全て修正してくれる

ショートカット 3/3

- リソースを開く(Ctrl + Shift + R)
  - ワークスペースのファイルを検索
  - ワイルドカードの利用が可能
- 型を開く(Ctrl + Shift + T)
  - jarなどの中のクラスを開ける
  - 自分が作ったクラス以外を探すならこっち
- 進む、戻る(Alt + Right, Alt + Left)
  - ジャンプする前の場所に戻ることが可能
  - 再度ジャンプ先に進むことが可能

デコンパイラ

- Javaプログラムはjavaファイルをコンパイルし、classファイルにして実行している
- 他者が作成したライブラリを利用する場合はclassファイルしかなく、javaファイルがない場合がある
  - この場合デバッガを使ってもソースコードを見れない
- デコンパイラはclassファイルからjavaファイルを復元する
- デコンパイラを入れることで、.classファイルしかないライブラリの中身もデバッグすることが可能



フォーマッタ

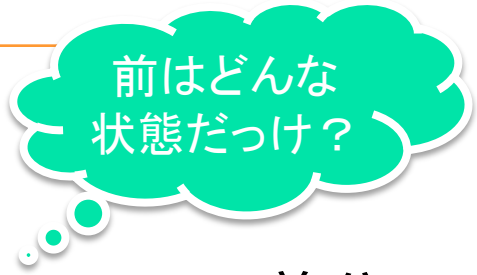
- ソースコードを適切なインデントで整形する
  - ウィンドウ→Java→コードスタイル→フォーマッター
- 保存時に自動で整形する
  - ウィンドウ→Java→エディター→補完アクション

```
StringBuffer      printStr=new  StringBuffer();
for (int i=0; i<  loopCnt; i++) {
    String str = createPrintStr(i);
    printStr
    .append(str);
    printStr
    .
    append
    (
            NEW_LINE
            )
            ;
}
```

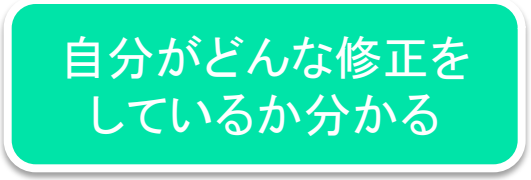
```
StringBuffer printStr = new StringBuffer();
for (int i = 0; i < loopCnt; i++) {
    String str = createPrintStr(i);
    printStr.append(str);
    printStr.append(NEW_LINE);
}
```

- 整形されていない場合、マージ時に大量にコンフリクトが出てしまい手間がかかってしまう

SVN操作 1/2



- コミット履歴の確認
  - 右クリック→チーム→Show History
  - 履歴を二つ選択してCompare with each otherで差分が見れる



- ローカルファイルとの差分
  - 最後に更新した時点との差分
    - 右クリック→比較対象→Base from Working Copy
  - リポジトリの最新との差分
    - 右クリック→比較対象→Latest fromRepository

SVN操作 2/2

- 最後に更新した時点に戻す
  - 右クリック→チーム→Revert
- リポジトリの最新で上書きする
  - 右クリック→置換対象
    →Latest fromRepository
- 管理対象外とする
  - チーム→Add to svn:ignore