Javaプログラミング実習

32. ログ出力

株式会社ジードライブ

今回学ぶこと

- ログ出力ライブラリ「log4j」の使い方
  - ログとは
  - ライブラリの使い方

ログとは

- 何らかの事柄に関する情報や履歴などをテキストとして記録したもの
  - 情報や履歴を記録することをロギングという
  - ロガーと呼ばれるプログラムを使いロギングを行う
- 処理内容や実行中に発生した問題などをログとして記録しておくことで、課題解決の際の手掛かりとすることができる(問題発生前後の状態を比較できる)
  - 開発中：プログラムの動きを把握するために利用
  - 運用開始後：故障や不具合の原因究明に利用

ログ用ライブラリの使用

- System.out.println()を使い様々な情報をログとして出力することも可能だが、情報の重要度に応じて出力を制御することは難しい
  - また出力する情報に「いつ」「どこで」という文脈を付与するには手間がかかる
- Log4jなどのログ用ライブラリを使うことで、この課題を解決することができる
  - Log4j, java.util.logging, commons-logging, SLF4J

Log4j

- Java向けのログ用ライブラリ
  - Apache Software Foundation のチームによって開発されている
- 以下の6段階のレベル(深刻度合い)に応じてログを出力できる

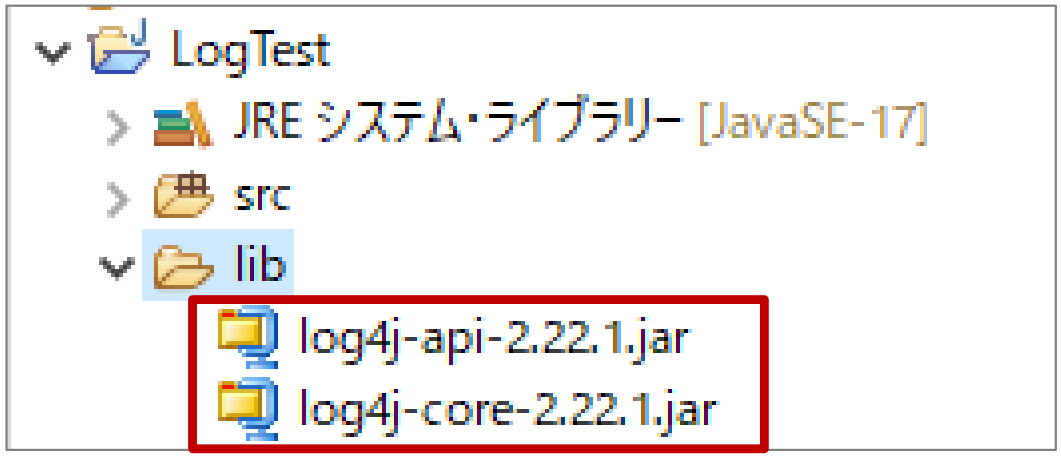
Log4jの導入

- zipファイルをダウンロードして展開する
  - https://logging.apache.org/log4j/2.x/download.html

Distribution	Mirrors
Apache Log4j 2 binary (zip)	apache-log4j-2.22.1-bin.zip
Apache Log4j 2 source (zip)	apache-log4j-2.22.1-src.zip

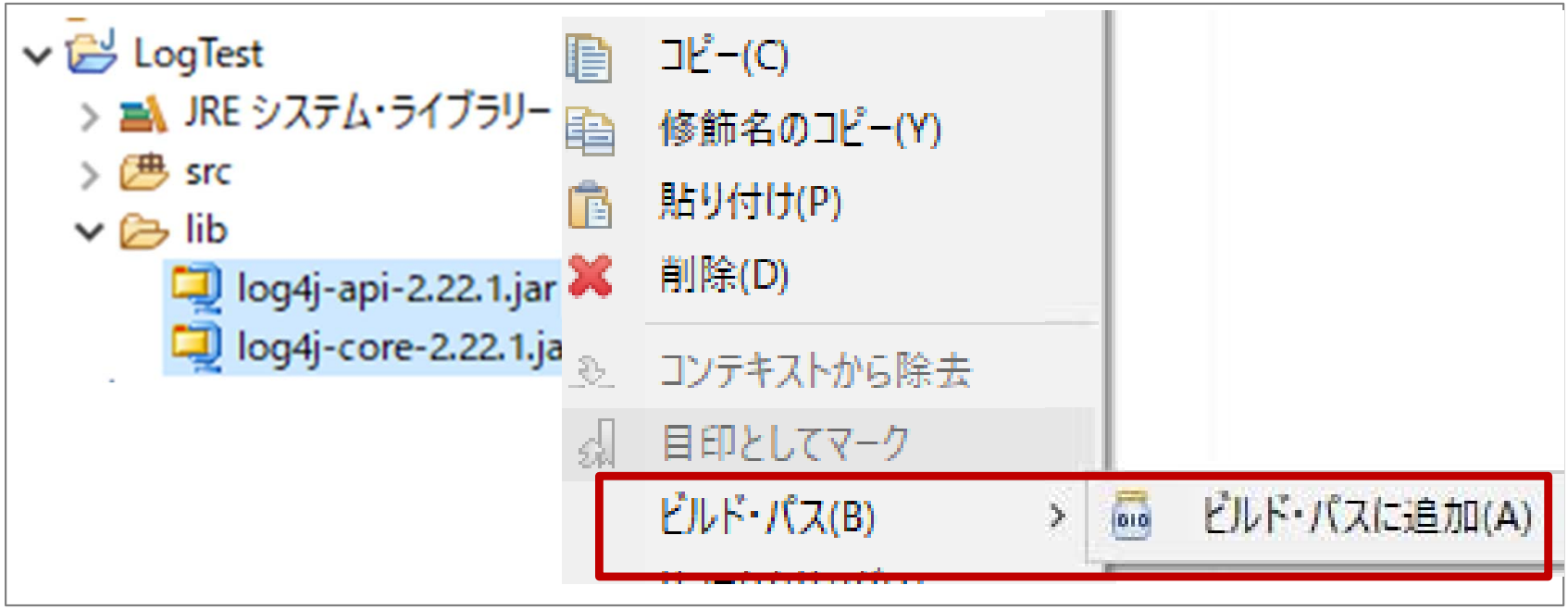
Log4jの導入

- Eclipseのプロジェクト内に libフォルダ を作成し、log4j-api-2.XX.X.jar と log4j-core-2.XX.X.jar をコピーする
  - 作成するフォルダ名は任意(libでなくても可)



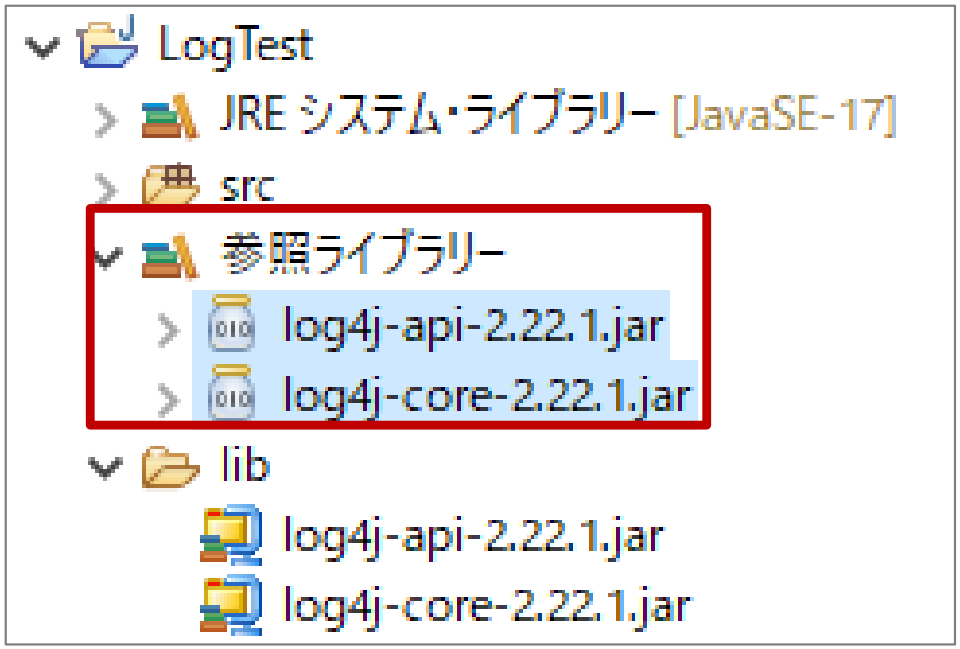
Log4jの導入

- コピーしたjarファイルを選択、右クリックして、ビルド・パス ⇒ ビルド・パスに追加



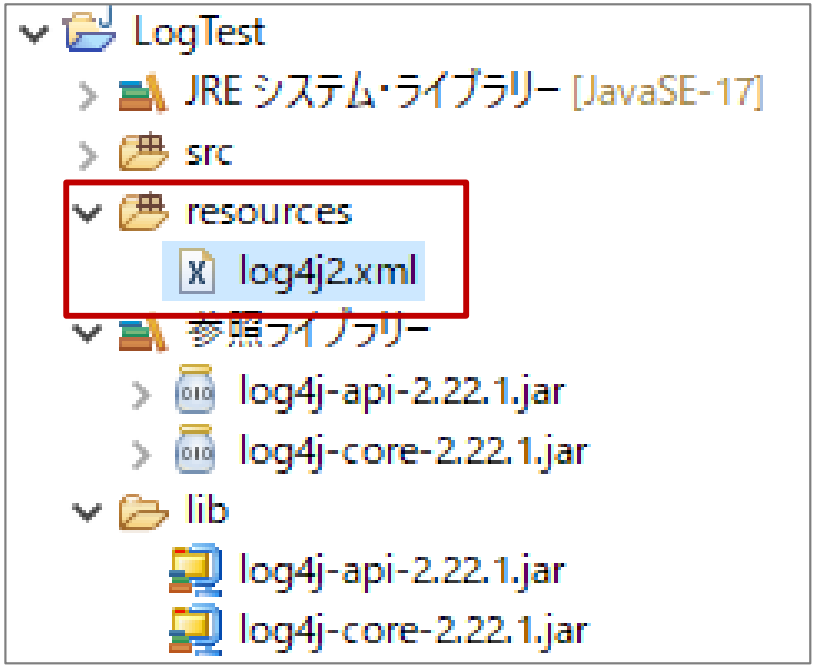
Log4jの導入

- 参照ライブラリーに、jarファイルが登録されていることを確認する



設定ファイルの準備

- ソースフォルダを作成し、**log4j2.xml** という名前のXMLファイルを格納する
  - 通常のフォルダではなく「ソースフォルダ」を作る
  - ソースフォルダ名は任意(下図では「resources」としている)



log4j2.xml

- 全体をConfigurationタグで囲む
  - status属性は通常offを指定する
  - Properties, Appenders, Loggers要素を内包する

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration status="off">
    <Properties>
        <!-- Appenders内で利用するプロパティの設定 -->
    </Properties>
    <Appenders>
        <!-- 出力先や形式に関する設定 -->
    </Appenders>
    <Loggers>
        <!-- 出力レベルの設定 -->
    </Loggers>
</Configuration>
```

Properties要素

- Propertyタグを使いプロパティを設定できる
- 設定した変数は ${プロパティ名} で利用できる

```
<Properties>
  <Property name="fmt" value="%d{yyyy-MM-dd HH:mm:ss} %m%n" />
</Properties>

<Appenders>
  <Console name="Console" target="SYSTEM_OUT">
    <PatternLayout pattern="${fmt}" />
  </Console>
</Appenders>
```

String fmt = "%d{yyyy-MM-dd HH:mm:ss} %m%n";
のようなイメージ

ログの出力形式の指定

Appenders要素

- ログ出力先の設定を記述する
  - Console要素で、コンソール出力の設定を行う
  - File要素でファイル出力の設定を行う

```
<Appenders>
  <Console name="console" >
    <PatternLayout  pattern="${fmt}" />
  </Console>
  <File name="logfile"  fileName="${file}">
    <PatternLayout  pattern="${fmt}" />
  </File>
</Appenders>
```

Console要素

- コンソール出力の設定を記述する
  - name属性でLoggers要素内で参照される際の名称を設定する
  - PatternLayout要素でログの表示形式を設定する

```
<Console name="console">
    <PatternLayout pattern="${fmt}" />
</Console>
```

表示形式を指定

表示形式の例：

```
%d{yyyy-MM-dd HH:mm:ss} %-5p %m%n
```

%d：日付の表示。{ } 内は日付の形式
%p：DEBUG, WARNなどのログレベルの表示。%-5p とすることで、左詰めの表記になる
%m：ログメッセージ
%n：改行

File要素

- ファイル出力時のパスやログファイルの管理についての設定を記述する
  - name属性でLoggers要素内で参照される際の名称を設定する
  - fileName属性でログ用ファイルのパスを設定する
  - PatternLayout要素でログの表示形式を設定する

```
<File name="logfile" fileName="C:/User/Taro/mylog.log">
    <PatternLayout  pattern="${fmt}" />
</File>
```

Loggers要素

- AppenderRef要素でConsole要素やFile要素との紐づけを行う
  - level属性で、どのレベル以上のログを出力するかを設定する
  - 出力しない場合は、offを指定する

```
<Loggers>
    <Root level="trace">
        <AppenderRef ref="logfile" />
        <AppenderRef ref="console" level="warn"/>
    </Root>
</Loggers>
```

基本設定：TRACEレベル以上のログを出力

個別設定：コンソールには、WARNレベル以上のログを出力

ログの出力

- Loggerオブジェクトのもつ、各ログレベルに応じた出力用メソッドを利用する

```
import org.apache.logging.log4j.Logger;
import org.apache.logging.log4j.LogManager;

public class LogTest {
    private static Logger logger = LogManager.getLogger();

    public static void main(String[] args) {
        logger.trace("trace メッセージです");
        logger.debug("debug メッセージです");
        logger.info("info メッセージです");
        logger.warn("warn メッセージです");
        logger.error("error メッセージです");
        logger.fatal("fatal メッセージです");
    }
}
```

ログの出力

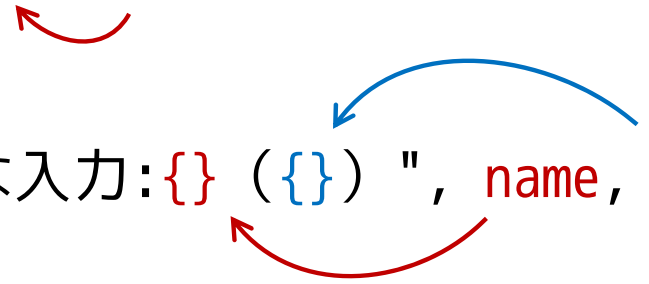
- ログメッセージの{}には変数を入れることができる

```
System.out.println("名前を入力してください");
String name = scanner.nextLine();

System.out.println("年齢を入力してください");
int age = scanner.nextInt();

logger.debug("入力値:{}", age);

if (age < 0) {
    logger.warn("不正な入力:{} ({}) ", name, age);
}
```



練習

- 練習32-1