



JAVAの活用

オブジェクト指向プログラミング特論

只木進一:工学系研究科

目的

- Javaの様々な機能を使う
 - 便利な機能
 - C++と違うところ
- スマートなプログラム開発
 - ライブラリの活用



PACKAGE

- プログラム全体をモジュールに分ける
- 関連の強いクラスをpackageにまとめる
- ソースファイルのフォルダに対応
 - 階層化できる
- ソースファイル先頭でpackage宣言
 - プロジェクトフォルダ中の相対位置に対応
- 修飾子 (public, private)が無いと、package内に公開



IMPORT

- 使用するライブラリクラスの名前解決を支援
 - クラス名が重複しているときには注意が必要
- 特定のクラス: `import` クラス名
 - 例: `import java.util.List;`
 - プログラム中の `List` クラスは、`java.util.List` になる
- パッケージ: `import` パッケージ名
 - 例: `import java.awt.*;`



修飾子

- アクセス制限: `public`, `private`, `protected`
 - C++と同様な制限
- 定数: `final`
 - コンストラクタ内で一度だけ値を設定することも可能
- クラスに属する変数、メソッド: `static`
 - クラスインスタンスを作らなくても存在



例外処理

- 実行時に発生するエラーへの対処を記述する
 - ファイルが開かない
 - 正しく値を変換できない
- メソッドが例外を返す: **throws**
- メソッドの呼出側がそれ进行处理する

```
try{  
    処理  
} catch(例外 ex){  
    例外処理  
}
```



例外処理: 例

```
public class FileIOMain {  
  
    private File inFile;  
    private File outFile;  
    private final String nl = System.getProperty("line.separator");  
  
    public FileIOMain() throws FileNotFoundException {  
        inFile = new File("input.txt");  
        if (!inFile.isFile() || !inFile.canRead()) {  
            String message = "入力ファイル"  
                + inFile.getAbsolutePath() + "がありません。";  
            throw new java.io.FileNotFoundException(message);  
        }  
        outFile = new File("output.txt");  
    }  
}
```

入力ファイルinput.txtが存在しないと
例外FileNotFoundExceptionをthrow



```
public static void main(String[] args) throws IOException {  
    FileIOMain fileIOMain = null;  
    try {  
        fileIOMain = new FileIOMain();  
    } catch (FileNotFoundException ex) {  
        Logger.getLogger(  
            FileIOMain.class.getName()).log(  
                Level.SEVERE, null, ex);  
    }  
    if (fileIOMain != null) {  
        int n = fileIOMain.getData();  
        System.out.println(n);  
    }  
}
```

FileIOMainのコンストラクタから
例外が出ると、エラーログを生成



型パラメタの利用 (GENERIC)

- クラスが保持するデータの型を明示する
 - コンパイル時に整合性を確認する
 - コレクションライブラリ(リストなど)で重要
- 例: IntegerクラスのインスタンスからなるArrayList

```
java.util.ArrayList<Integer> list  
    = new java.util.ArrayList<>();  
.....  
Integer a = list.get(0);
```



BOXING

- 原始型: int, double, boolean, char など
- 対応するクラス: Integer, Double, Boolean, Character
- 自動でキャストできる

```
int n = Integer.valueOf(10);  
Integer nInt = n;
```



拡張されたFOR

- コレクションの全要素に対するfor

```
List<Integer> list;  
...  
for(Integer e:list){  
    処理  
}
```



GRAPHICAL USER INTERFACE

- javaでは、標準にGUIコンポーネントがある
- 基本的コンポーネント: `javax.swing.*`
- 描画用クラス
 - `java.awt.*`;
 - `java.awt.geom.*`;
 - `java.awt.Graphics.*`;
 - `java.awt.Graphics2D.*`;

