小テスト

ウォーミングアップ

クラス

オブジェクト指向の入り口へ

分割の限界

メソッドの分割にも
限界がある。

分割の限界

複数のソースファイルに分割して記述！

例6-1

[プロジェクト名] ex6-1
Calc.java

```
public class Calc{
 public static void main(String[] args){
   int a=10, b=12;
   int total = add(a,b);
   int product = prod(a,b);
  }
}
```

add, prodは
別ソースファイルに!!

例6-1

[プロジェクト名] ex6-1

CalcLogic.java

```
public class CalcLogic{
 public static int add(int a, int b){
  return a+b;
 }

 public static int prod(int a, int b){
  return a*b;
 }
}
```

計算処理をまとめたソースファイル

例6-1

[プロジェクト名] ex6-1
Calc.java

```
public class Calc{
 public static void main(String[] args){
   int a=10, b=12;
   int total = add(a,b);
   int product = prod(a,b);
  }
}
```

コンパイルエラー

例6-1

[プロジェクト名] ex6-1

Calc.java

```
public class Calc{
 public static void main(String[] args){
   int a=10, b=12;
   int total = CalcLogic.add(a,b);
   int product = CalcLogic.prod(a,b);
  }
}
```

CalcLogic.java内にあると明示する。

イメージ図

Calcクラス

mainメソッド

```
total=CalcLogic.add(a,b)

product=CalcLogic.prod(a,b)
```

CalcLogicクラス

addメソッド

prodメソッド

Calc.java

CalcLogic.java

「ソース分割」のいいこと

テスト

分業のしやすさ

「ソース分割」のいいこと

テスト

(1)ソースを分割し、
　それぞれのソースを分業で開発する。
(2)各プログラマは、java によってソースを
　コンパイルし、classファイルを作る。
(3) それを１箇所に集めて、
　「クラスファイルの集合体」を
　JAR(Java ARchive)ファイルに圧縮し
　て完成。
(4) 実行時は「mainメソッドが含まれてい
　るクラスファイル」を実行。

分割の限界

もっとでかいくくり

パッケージ

分割の限界

calcapp.mainパッケージ

Calcクラス

mainメソッド

total=CalcLogic.add(a,b)

product=CalcLogic.prod(a,b)

Calc.java

calcapp.logicsパッケージ

CalcLogicクラス

addメソッド

prodメソッド

CalcLogic.java

クラスをパッケージに所属させる

```
package 所属させたいパッケージ名;
```

これをソースの頭に！

例6-2

[プロジェクト名] ex6-2

Calc.java

```
package calcapp.main;

public class Calc{
 public static void main(String[] args){
   int a=10, b=12;
   int total = CalcLogics.add(a,b);
   int product = CalcLogics.prod(a,b);
  }
}
```

例6-1

[プロジェクト名] ex6-1

CalcLogic.java

```
package calcapp.logics;

public class CalcLogic{
 public static int add(int a, int b){
  return a+b;
 }

 public static int prod(int a, int b){
  return a*b;
 }
}
```

例6-2

[プロジェクト名] ex6-2
Calc.java

```
package calcapp.main;

public class Calc{
 public static void main(String[] args){
   int a=10, b=12;
   int total = CalcLogics.add(a,b);
   int product = CalcLogics.prod(a,b);
  }
}
```

コンパイルエラー

例6-2

[プロジェクト名] ex6-2
Calc.java

```
package calcapp.main;

public class Calc{
 public static void main(String[] args){
    int a=10, b=12;
    int total = calcapp.logics.CalcLogics.add(a,b);
    int product = calcapp.logics.CalcLogics.prod(a,b);
  }
}
```

パッケージを明示

メソッド呼び出し

```
パッケージ名.クラス名.メソッド名();
```

FQCN

Full qualified class name

完全限定クラス名

import文

FQCNの省略はimport文!!

import パッケージ名.クラス名;

calcapp.logics.CalcLogic.add(a,b);

import calcapp.logics.CalcLogic;
CalcLogic.add(a,b);

import文

＊で全クラスFQCN省略

import パッケージ名.＊;

calcapp.logics.CalcLogic.add(a,b);

```
import calcapp.logics.*;
CalcLogic.add(a,b);
CalcLogic.prod(a,b);
```

javaの謎

命令が異常に長い

javaの謎

FQCN

java.util.Scanner.nextInt()

パッケージ　クラス　メソッド

javaの謎

FQCN

java.util.Scanner.nextLine()

パッケージ　クラス　メソッド

API

javaに最初から くっついている超重要パッケージ

(Application Programming Interface)

java.lang	javaに欠かせない重要なクラス群
java.util	プログラミングを便利にするさまざまなクラス群
java.math	数学に関するクラス群
java.net	ネットワーク通信などを行うためのクラス群
java.io	ファイルの読み書きなど、データを逐次処理するためのクラス群

API

特にjava.langパッケージは重要すぎるので…

import java.lang.*;

暗黙のimport

正体判明

java.lang.System.out.println();

省略されていた!!

演習7

どんどん質問を!!