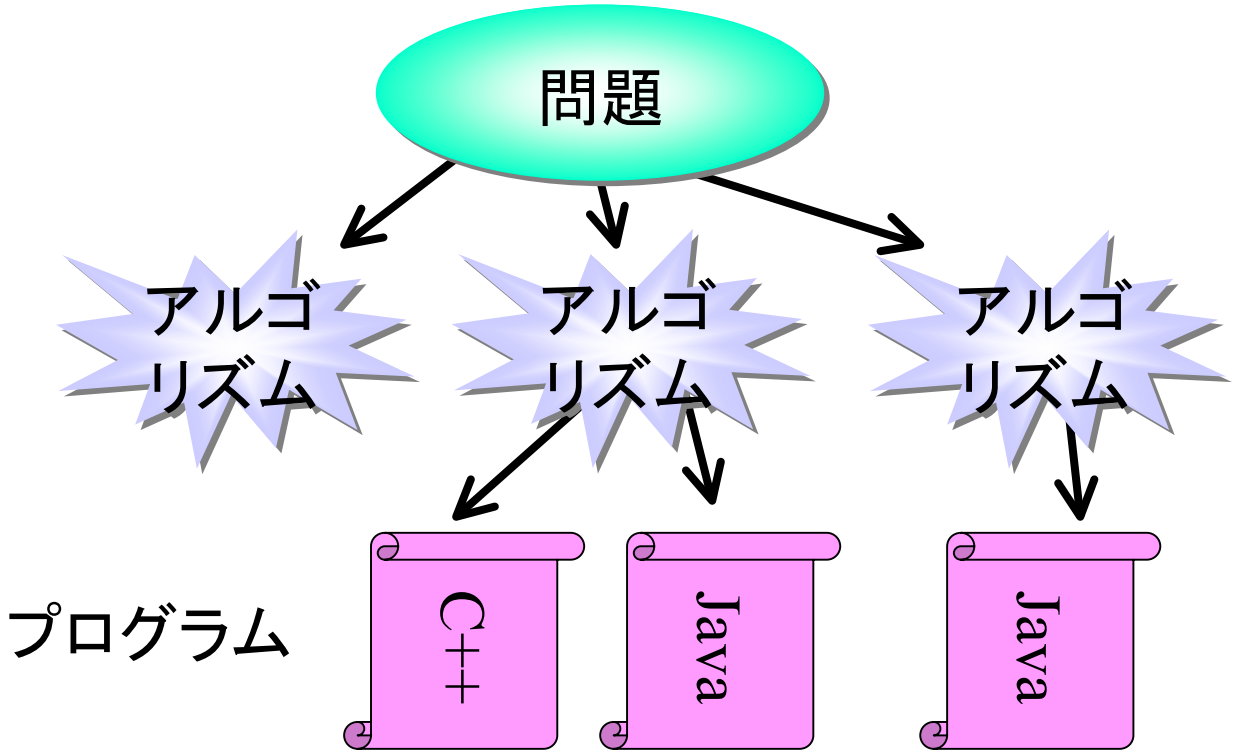
計算機プログラミングI

第4回　2002年10月31日(木)

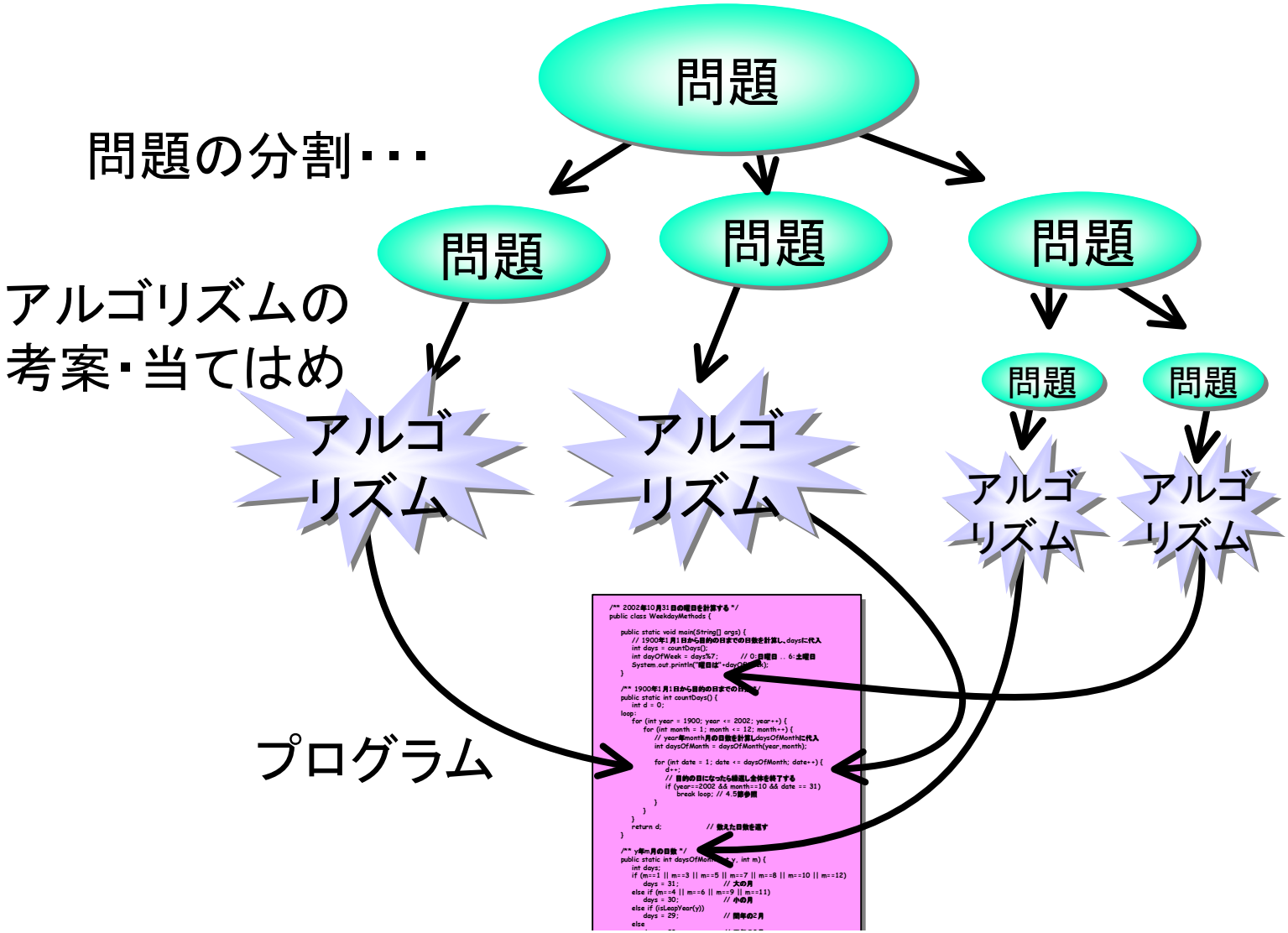
- 問題解決とアルゴリズム
  - 最大公約数
  - (素因数分解)
  - 曜日の計算
- クラスメソッドと手続きの抽象化

問題解決とアルゴリズム

- アルゴリズム: 問題を解くための手順



問題解決の方法


問題
問題の分割・・・
問題
問題
問題
アルゴリズムの
考案・当てはめ
問題
問題
アルゴリズム
アルゴリズム
アルゴリズム
アルゴリズム
プログラム
/** 2002年10月31日の曜日を計算する */
public class WeekdayMethods {
public static void main(String[] args) {
// 1900年1月1日から目的の日までの日数を計算し、daysに代入
int days = countDays();
int dayOfWeek = days%7; // 0:日曜日 , 6:土曜日
System.out.println("曜日は"+dayOfWeek);
}
/** 1900年1月1日から目的の日までの日数 */
public static int countDays() {
int d = 0;
loop:
for (int year = 1900; year <= 2002; year++) {
for (int month = 1; month <= 12; month++) {
// year年month月の日数を計算しdaysOfMonthに代入
int daysOfMonth = daysOfMonth(year,month);
for (int date = 1; date <= daysOfMonth; date++) {
d++;
// 目的の日になったら繰返し全体を終了する
if (year==2002 && month==10 && date == 31)
break loop; // 4.5節参照
}
}
}
return d; // 数えた日数を返す
}
/** y年m月の日数 */
public static int daysOfMonth(int y, int m) {
int days;
if (m==1 || m==3 || m==5 || m==7 || m==8 || m==10 || m==12)
days = 31; // 大の月
else if (m==4 || m==6 || m==9 || m==11)
days = 30; // 小の月
else if (isLeapYear(y))
days = 29; // 閏年の2月
else

最大公約数

- 問題: 2つの整数a, bの最大公約数を求める
  (ただし a < b)
- 問題の分割:
  - 公約数かどうかを調べる
  - 最大の公約数を求める
- アルゴリズム1:
  - i を 1, 2, ..., a の順に変化
  - i が a, b の公約数であるかを調べる
  - 最後に公約数であると分かったときの
    i の値が最大公約数

アルゴリズム1

a=6, b=21の場合

- i=1: i は a,bの公約数である
- i=2: i は a,bの公約数でない
- i=3: i は a,bの公約数である
- i=4: i は a,bの公約数でない
- i=5: i は a,bの公約数でない
- i=6: i は a,bの公約数でない

→ 3が最大公約数

アルゴリズム1のプログラム

```
public class GCD {
  public static void main(String[] args){
      int a = 12, b = 21;
      int lastCommonDivisor = 0; // 一番最近に見つかった公約数
      for (int i = 1; i <= a; i++) {
          if (iはa,bの公約数?) {
              lastCommonDivisor = i; // iは公約数なので覚える
          }
      }
      System.out.println(lastCommonDivisor);
  }
}
```

アルゴリズム1のプログラム

```
public class GCD {
  public static void main(String[] args){
      int a = 12, b = 21;
      int lastCommonDivisor = 0; //
      for (int i = 1; i <= a; i++) {
          if (iはa,bの公約数?) {
              lastCommonDivisor = i; // iは公約数なので覚える
          }
      }
      System.out.println(lastCommonDivisor);
  }
}
```

iはaの約数 && iはbの約数

アルゴリズム1のプログラム

```
public class GCD {
  public static void main(String[] args){
      int a = 12, b = 21;
      int lastCommonDivisor = 0; // iはaの約数 && iはbの約数
      for (int i = 1; i <= a; i++) {
          if (iはa,bの公約数?) {
              lastCommonDivisor = i; // iは公約数なので覚える
          }
      }
      System.out.println(lastCommonDivisor);
  }
}
```

a%i == 0

アルゴリズム2

逆順に探す

- i=6: i は a,bの公約数でない
- i=5: i は a,bの公約数でない
- i=4: i は a,bの公約数でない
- i=3: i は a,bの公約数である

→ 3が最大公約数

アルゴリズム2のプログラム

```
public class GCD2 {
  public static void main(String[] args){
      int a = 12, b = 21;
      int i = a;
      while (! iはa,bの公約数)
          i--;
      System.out.println("最大公約数は" + i);
  }
}
```

ユークリッドの互除法

- 定理: bをaで割った余りをrとする
  aとbの最大公約数とrとaの最大公約数は等しい
- アルゴリズム:
  - bをaで割った余りをrとする
  - rが0でない:　　r,aの最大公約数を求める
  - rで0である:　　aが最大公約数

ユークリッドの互除法

アルゴリズムによって
繰返しの回数が
大きく違う

- 例:a=143, b=1469 の場合

今回のa	今回のb	b÷aの余り
143	1469	39
39	143	26
26	39	13
13	26	0

ユークリッドの互除法

アルゴリズムによって
繰返しの回数が
大きく違う

- 例:a=143, b=1469 の場合

今回のa	今回のb	b÷aの余り
143	1469	39
39	143	26
26	39	13
13	26	0

ユークリッドの互除法

アルゴリズムによって
繰返しの回数が
大きく違う

- 例:a=143, b=1469 の場合

今回のa	今回のb	b÷aの余り
143	1469	39
39	143	26
26	39	13
13	26	0

ユークリッドの互除法

アルゴリズムによって
繰返しの回数が
大きく違う

- 例:a=143, b=1469 の場合

今回のa	今回のb	b÷aの余り
143	1469	39
39	143	26
26	39	13
13	26	0

余りが0になったときのaが最大公約数

ユークリッドの互除法:プログラム

- (略)
- 今回のa,(b/aの余り)を次回のb,aに
  —— 注意が必要

今回のa	今回のb	b÷aの余り
143	1469	39
39	143	26
26	39	13

素因数分解

- 問題の分割
  - 素因数を1つ見つける
  - 素因数を全て見つける

(資料参照)

曜日の計算

- 問題:2002年10月31日は何曜日
- アルゴリズム:
  - 基準となる日から
    目的の日までの日数を数える
  - 7で割った余りを求める

- 小問題:基準日から目的日までの
  日数を数える
- アルゴリズム:
  - 年を基準～目的まで変化
    月を1～12まで変化
    日を1～(月の日数)まで変化
    日数を1増やし
    目的日になったら終了

- 小問題: y年m月の日数
- アルゴリズム:
  - mが大の月: 31 日
  - mが小の月(2月以外): 30日
  - mが2月:
    - yが閏年: 29日
    - yが平年: 28日

- 小問題: y年が閏年か調べる
- アルゴリズム:
  - yが400の倍数、または
  - yが100の倍数でなく4の倍数

曜日を計算するプログラム

```
/** 2002年10月31日の曜日を計算する */
public class Weekday {
    public static void main(String[] args) {
        int days = 0;
        // 1900年1月1日から目的の日までの日数を計算し、daysに代入
    loop:
        for (int year = 1900; year <= 2002; year++) {
            for (int month = 1; month <= 12; month++) {
                int daysOfMonth;
                // year年month月の日数を計算しdaysOfMonthに代入
                if (month==1 || month==3 || month==5 || month==7
                    || month==8 || month==10 || month==12)
                    daysOfMonth = 31;
                else if (month==4 || month==6 || month==9 || month==11)
                    daysOfMonth = 30;
                /* 2月の場合、閏年かどうかを調べる (練習問題3.6-2) */
                else if (year%400 == 0 || (year%100 != 0 && year%4 == 0))
                    daysOfMonth = 29; // 閏年
                else
                    daysOfMonth = 28; // 平年
                for (int date = 1; date <= daysOfMonth; date++) {
                    days++;
                    // 目的の日になったら繰返し全体を終了する
                    if ((year==2002) && (month==10) && (date == 31))
                        break loop; // 4.5節参照
                }
            }
        }
        int dayOfWeek = days%7; // 0:日曜日 .. 6:土曜日
        System.out.println("曜日は"+dayOfWeek);
    }
}
```

- 基準となる日から目的の日までの日数を数える
- y年m月の日数
- y年が閏年か調べる
- 7で割った余りを求める

練習

クラスメソッド

- 問題を分割——プログラムは分割されていない
  → クラスメソッドを使うと分割できる

曜日の計算

- 問題:2002年10月31日は何曜日
- アルゴリズム:
  - 基準となる日から目的の日までの日数を数える
  - 7で割った余りを求める

- 小問題:基準日から目的日までの日数を数える
- アルゴリズム:
  - 年を基準～目的まで変化
    - 月を1～12まで変化
      - 日を1～(月の日数)まで変化
        - 日数を1増やし
        - 目的日になったら終了

- 小問題: y年m月の日数
- アルゴリズム:
  - mが大の月: 31日
  - mが小の月(2月以外): 30日
  - mが2月:
    - yが閏年: 29日
    - yが平年: 28日

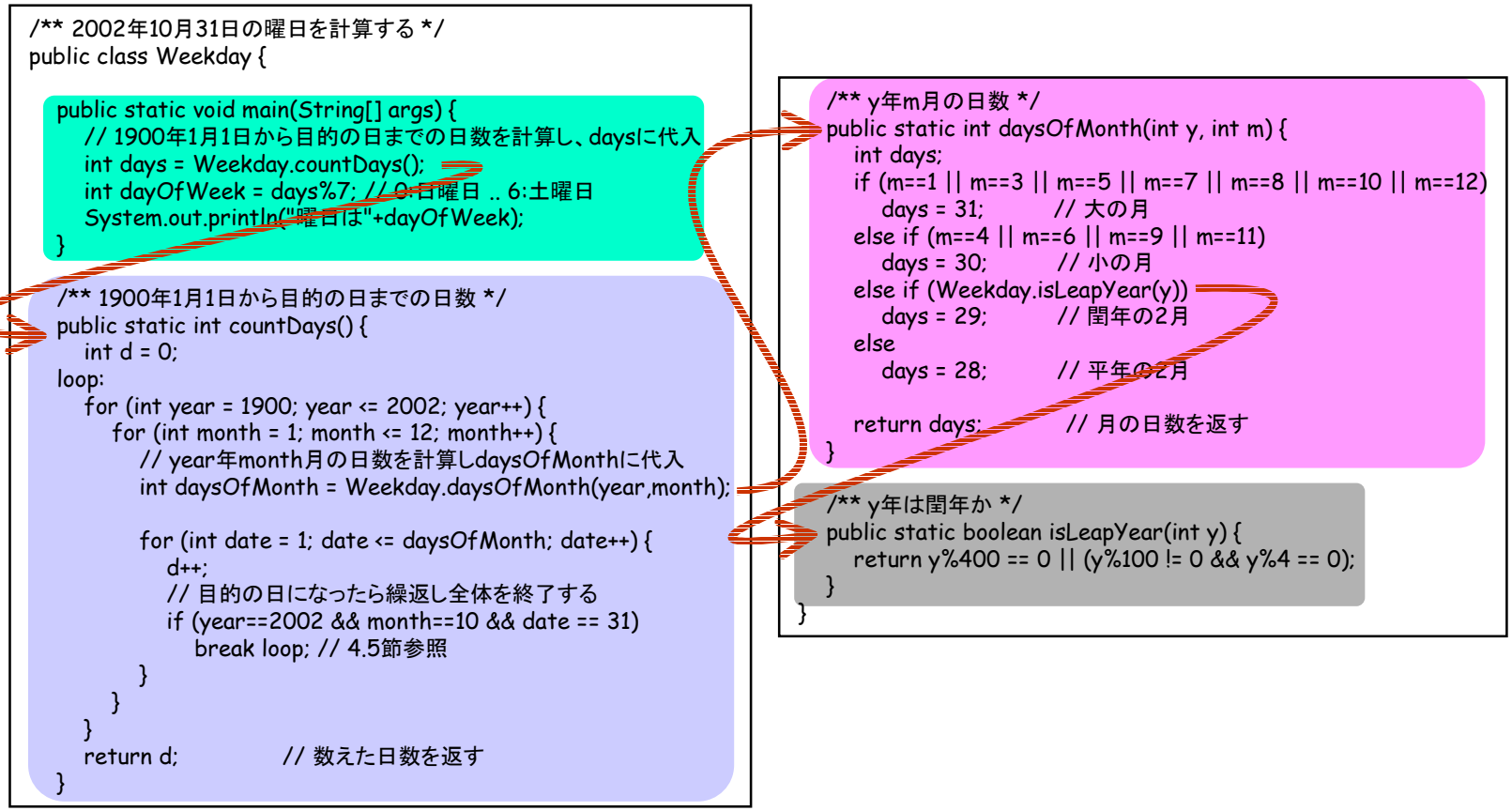
- 小問題: y年が閏年か調べる
- アルゴリズム:
  - yが400の倍数、または
  - yが100の倍数でなく4の倍数

曜日を計算するプログラム

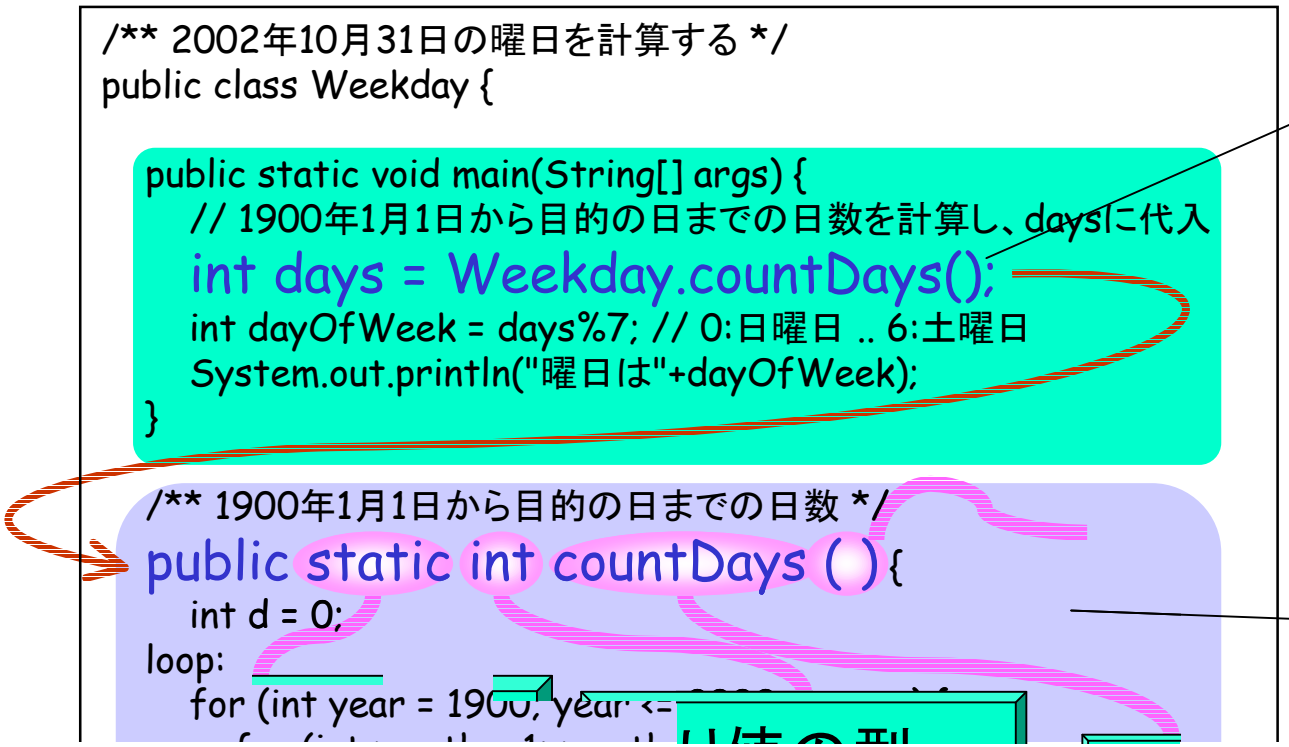
```
/** 2002年10月31日の曜日を計算する */
public class Weekday {
  public static void main(String[] args) {
    int days = 0;
    // 1900年1月1日から目的の日までの日数を計算し、daysに代入
loop:
    for (int year = 1900; year <= 2002; year++) {
      for (int month = 1; month <= 12; month++) {
        int daysOfMonth;
        // year年month月の日数を計算しdaysOfMonthに代入
        if (month==1 || month==3 || month==5 || month==7
          || month==8 || month==10 || month==12)
          daysOfMonth = 31;
        else if (month==4 || month==6 || month==9 || month==11)
          daysOfMonth = 30;
        /* 2月の場合、閏年かどうかを調べる (練習問題3.6-2) */
        else if (year%400 == 0 || (year%100 != 0 && year%4 == 0))
          daysOfMonth = 29; // 閏年
        else
          daysOfMonth = 28; // 平年
        for (int date = 1; date <= daysOfMonth; date++) {
          days++;
          // 目的の日になったら繰返し全体を終了する
          if ((year==2002) && (month==10) && (date == 31))
            break loop; // 4.5節参照
        }
      }
    }
    int dayOfWeek = days%7; // 0:日曜日 .. 6:土曜日
    System.out.println("曜日は"+dayOfWeek);
  }
}
```

- 基準となる日から目的の日までの日数を数える
- y年m月の日数
- y年が閏年か調べる
- 7で割った余りを求める

クラスメソッドによる
プログラムの分割



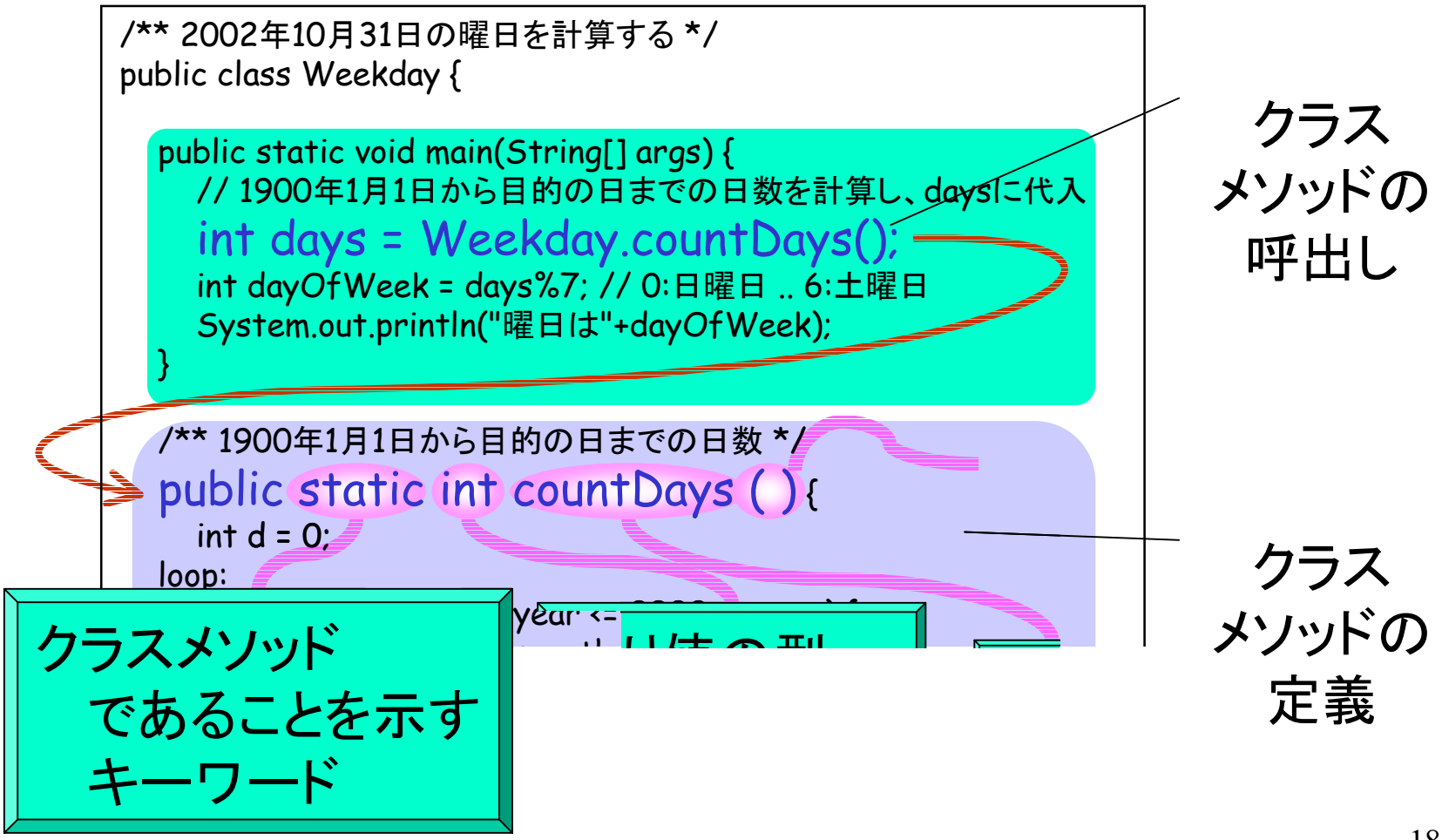
クラスメソッドの定義



クラスメソッドの呼出し

クラスメソッドの定義

クラスメソッドの定義


/** 2002年10月31日の曜日を計算する */
public class Weekday {
public static void main(String[] args) {
// 1900年1月1日から目的の日までの日数を計算し、daysに代入
int days = Weekday.countDays();
int dayOfWeek = days%7; // 0:日曜日 .. 6:土曜日
System.out.println("曜日は"+dayOfWeek);
}
/** 1900年1月1日から目的の日までの日数 */
public static int countDays ( ) {
int d = 0;
loop:
クラスメソッド
であることを示す
キーワード
クラス
メソッドの
呼出し
クラス
メソッドの
定義

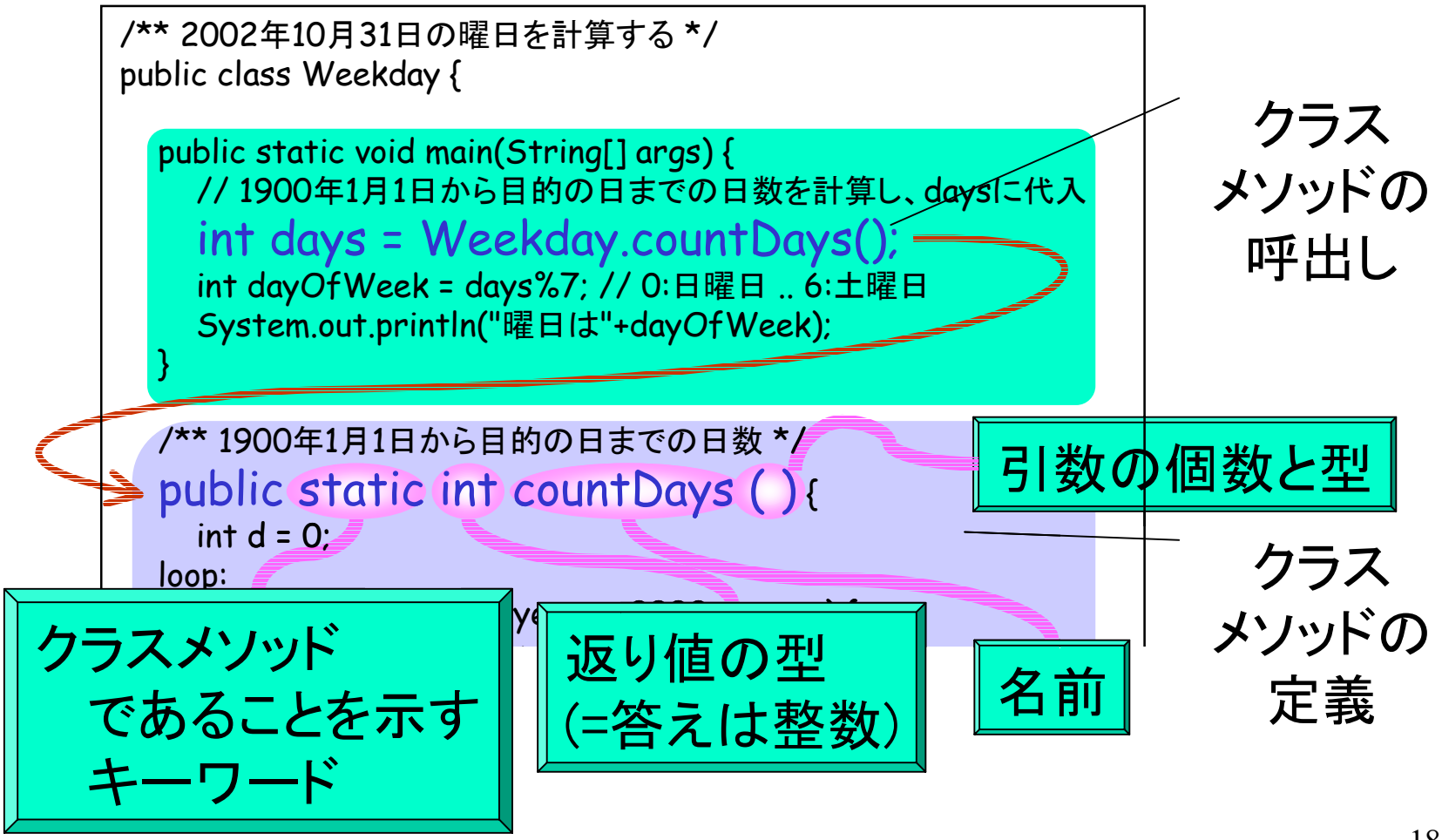
クラスメソッドの定義


/** 2002年10月31日の曜日を計算する */
public class Weekday {
public static void main(String[] args) {
// 1900年1月1日から目的の日までの日数を計算し、daysに代入
int days = Weekday.countDays();
int dayOfWeek = days%7; // 0:日曜日 .. 6:土曜日
System.out.println("曜日は"+dayOfWeek);
}
/** 1900年1月1日から目的の日までの日数 */
public static int countDays ( ) {
int d = 0;
loop:
クラスメソッド
であることを示す
キーワード
返り値の型
(=答えは整数)
クラス
メソッドの
呼出し
クラス
メソッドの
定義

クラスメソッドの定義


/** 2002年10月31日の曜日を計算する */
public class Weekday {
public static void main(String[] args) {
// 1900年1月1日から目的の日までの日数を計算し、daysに代入
int days = Weekday.countDays();
int dayOfWeek = days%7; // 0:日曜日 .. 6:土曜日
System.out.println("曜日は"+dayOfWeek);
}
/** 1900年1月1日から目的の日までの日数 */
public static int countDays ( ) {
int d = 0;
loop:
クラスメソッド
であることを示す
キーワード
返り値の型
(=答えは整数)
名前
クラス
メソッドの
呼出し
クラス
メソッドの
定義

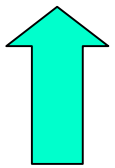
クラスメソッドの定義


/** 2002年10月31日の曜日を計算する */
public class Weekday {
public static void main(String[] args) {
// 1900年1月1日から目的の日までの日数を計算し、daysに代入
int days = Weekday.countDays();
int dayOfWeek = days%7; // 0:日曜日 .. 6:土曜日
System.out.println("曜日は"+dayOfWeek);
}
/** 1900年1月1日から目的の日までの日数 */
public static int countDays ( ) {
int d = 0;
loop:
クラスメソッドであることを示すキーワード
返り値の型(=答えは整数)
名前
引数の個数と型
クラスメソッドの呼出し
クラスメソッドの定義

メソッドの返り値

```
public static void main(String[] args) {
    // 1900年1月1日から目的の日までの日数を計算し、daysに代入
    int days = Weekday.countDays();
    int dayOfWeek = days%7; // 0:日曜日 .. 6:土曜日
    System.out.println("曜日は"+dayOfWeek);
}
```

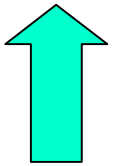
```
/** 1900年1月1日から目的の日までの日数 */
public static int countDays() {
    int d = 0;
loop:
    for (int year = 1900; year <= 2002; year++) {
            ...
        }
    }
    return d;                    // 数えた日数を返す
}
```



メソッドの返り値

```
public static void main(String[] args) {
    // 1900年1月1日から目的の日までの日数を計算し、daysに代入
    int days = Weekday.countDays();
    int dayOfWeek = days%7; // 0:日曜日 .. 6:土曜日
    System.out.println("曜日は"+dayOfWeek);
}
```

```
/** 1900年1月1日から目的の日までの日数 */
public static int countDays() {
    int d = 0;
loop:
    for (int year = 1900; year <= 2002; year++) {
            ...
        }
    }
    return d;                    // 数えた日数を返す
}
```

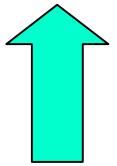


dの値を計算

メソッドの返り値

```
public static void main(String[] args) {
    // 1900年1月1日から目的の日までの日数を計算し、daysに代入
    int days = Weekday.countDays();
    int dayOfWeek = days%7; // 0:日曜日 .. 6:土曜日
    System.out.println("曜日は"+dayOfWeek);
}
```

メソッド呼出し式の値になる



```
/** 1900年1月1日から目的の日までの日数 */
public static int countDays() {
    int d = 0;
loop:
    for (int year = 1900; year <= 2002; year++) {
            ...
        }
    }
    return d;                    // 数えた日数を返す
}
```

dの値を計算

クラスメソッドの引数

```
/** 1900年1月1日から目的の日までの日数 */
public static int countDays() {
    int d = 0;
loop:
    for (int year = 1900; year <= 2002; year++) {
        for (int month = 1; month <= 12; month++) {
            // year年month月の日数を計算しdaysOfMonthに代入
            int daysOfMonth = Weekday.daysOfMonth(year,month);

            for (int date = 1; date <= daysOfMonth; date++) {
                d++;
                // 目的の
                if (year=
                    break
            }
        }
    }
    return d;
}
```

```
/** y年m月の日数 */
public static int daysOfMonth(int y, int m) {
    int days;
    if (m==1 || m==3 || m==5 || m==7 || m==8 || m==10 || m==12)
        days = 31;          // 大の月
    else if (m==4 || m==6 || m==9 || m==11)
        days = 30;          // 小の月
    else if (Weekday.isLeapYear(y))
        days = 29;          // 閏年の2月
    else
        days = 28;          // 平年の2月
```

クラスメソッドの引数

```
/** 1900年1月1日から目的の日までの日数 */
public static int countDays() {
    int d = 0;
loop:
    for (int year = 1900; year <= 2002; year++) {
        for (int month = 1; month <= 12; month++) {
            // year年month月の日数を計算しdaysOfMonthに代入
            int daysOfMonth = Weekday.daysOfMonth(year,month);

            for (int date = 1; date <= daysOfMonth; date++) {
                d++;
                // 目的の
                if (year=
                    break
            }
        }
    }
    return d;
}
```

```
/** y年m月の日数 */
public static int daysOfMonth(int y, int m) {
    int days;
    if (m==1 || m==3 || m                    10 || m==12)
        days = 31;          //
    else if (m==4 || m==6
        days = 30;          //
    else if (Weekday.isL
        days = 29;          //
    else
        days = 28;          // 平年の2月
```

- 引数が2つ
- 整数型
- yとmにしまう

クラスメソッドの引数

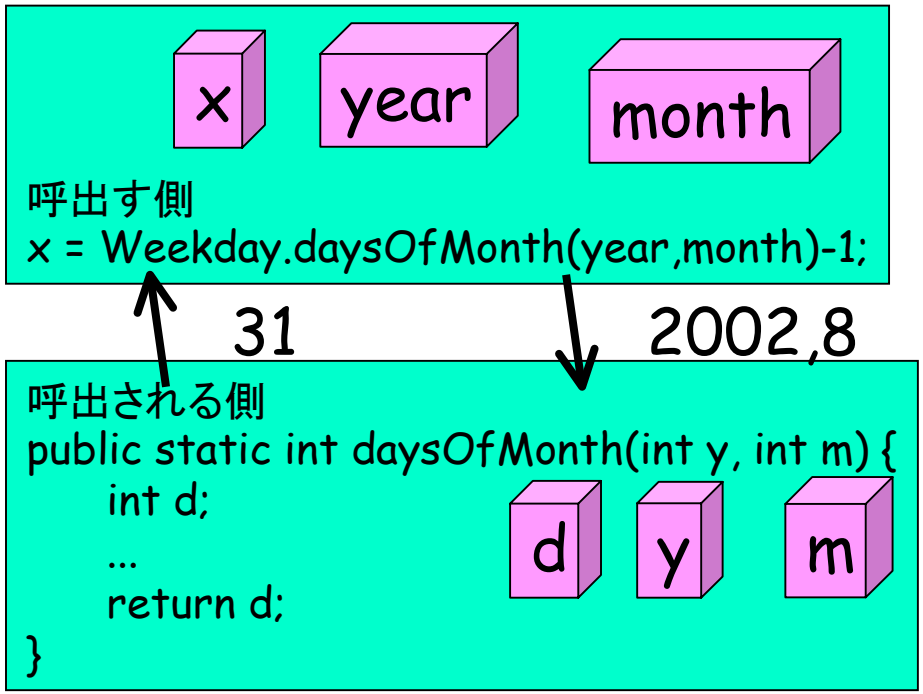
```
/** 1900年1月1日から目的の日までの日数
public static int countDays() {
    int d = 0;
loop:
    for (int year = 1900; year <= 2002; ye
        for (int month = 1; month <= 12; month++) {
            // year年month月の日数を計算しdaysOfMonthに代入
            int daysOfMonth = Weekday.daysOfMonth(year,month);

            for (int date = 1; date <= daysOfMonth; date++) {
                d++;
                // 目的の
                if (year=
                    break
            }
        }
    }
    return d;
}
```

- 引数を計算
  （year,monthの値をとり出す）
- メソッドに渡す

```
/** y年m月の日数 */
public static int daysOfMonth(int y, int m) {
    int days;
    if (m==1 || m==3 || m                    10 || m==12)
        days = 31;          //
    else if (m==4 || m==6
        days = 30;          //
    else if (Weekday.isL
        days = 29;          //
    else
        days = 28;          // 平年の2月
```

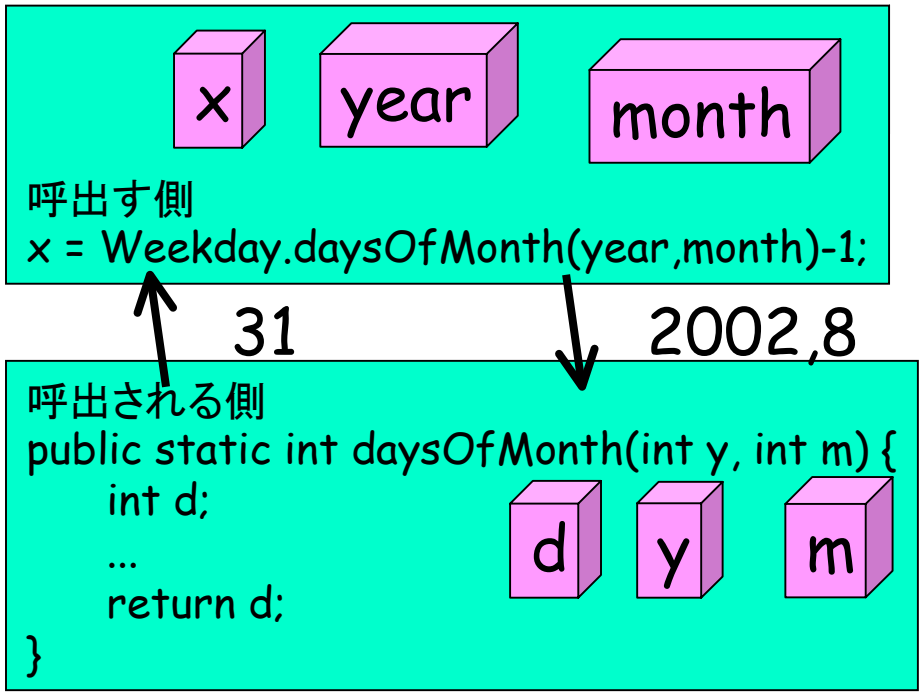
- 引数が2つ
- 整数型
- yとmにしまう

クラスメソッドのまとめ



変数は別々!

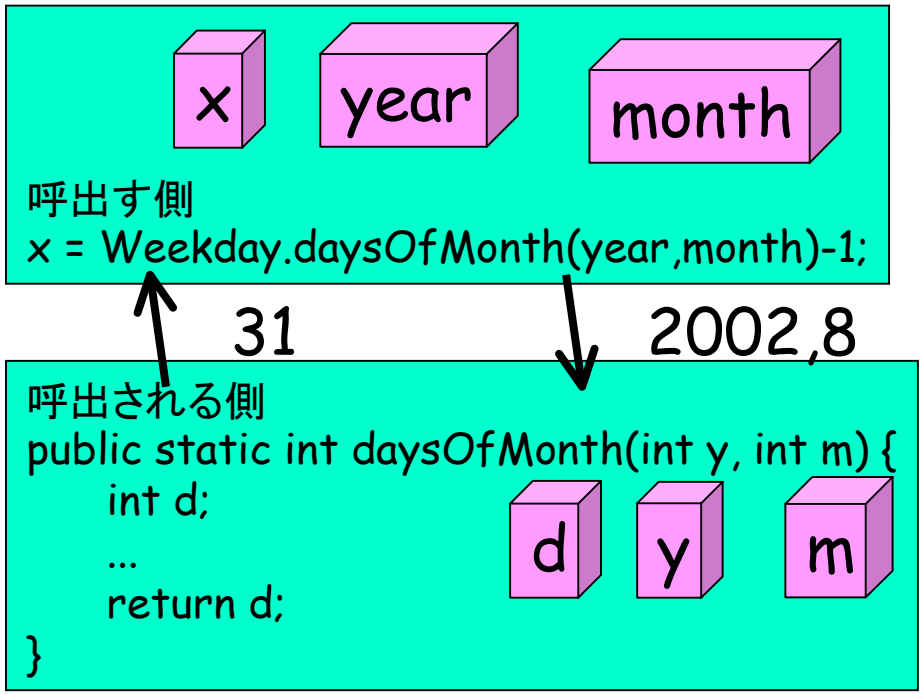
クラスメソッドのまとめ



- 呼出す側で引数の値が計算される

変数は別々!

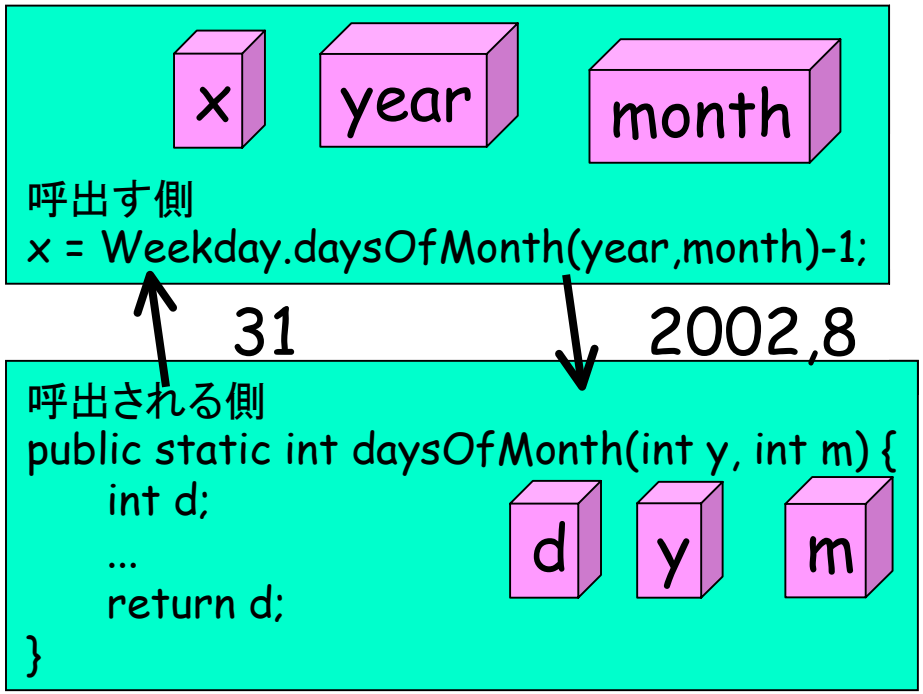
クラスメソッドのまとめ



- 呼出す側で引数の値が計算される
- 呼出される側の変数に引数がしまわれる

変数は別々!

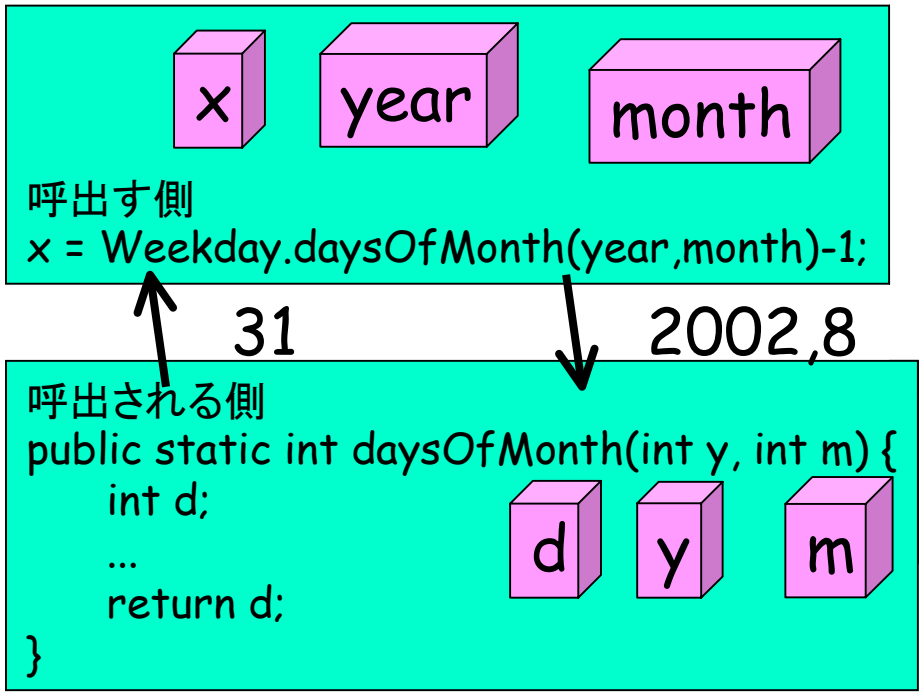
クラスメソッドのまとめ



- 呼出す側で引数の値が計算される
- 呼出される側の変数に引数がしまわれる
- 呼出された側が実行

変数は別々!

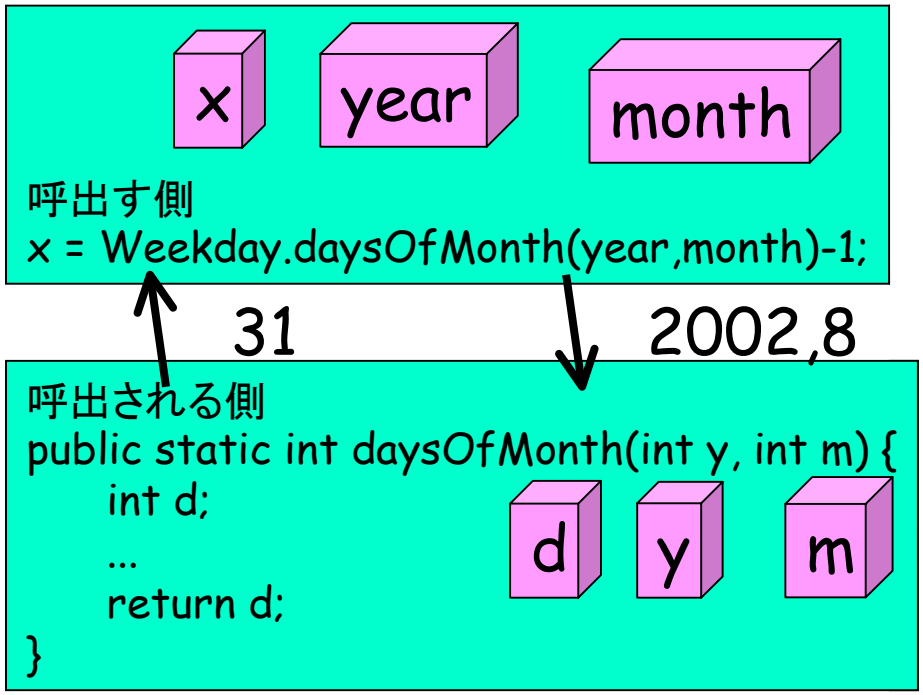
クラスメソッドのまとめ



- 呼出す側で引数の値が計算される
- 呼出される側の変数に引数がしまわれる
- 呼出された側が実行
- return文の式の値が計算される

変数は別々!

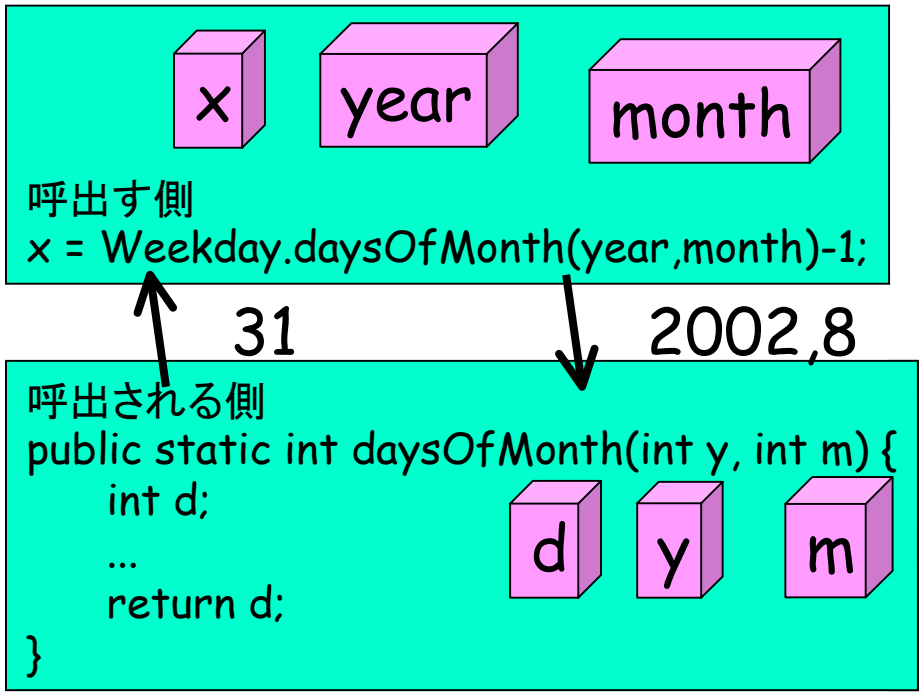
クラスメソッドのまとめ



- 呼出す側で引数の値が計算される
- 呼出される側の変数に引数がしまわれる
- 呼出された側が実行
- return文の式の値が計算される
- 返り値が呼出した側の式の値になる

変数は別々!

クラスメソッドのまとめ



変数は別々！

- 呼出す側で引数の値が計算される
- 呼出される側の変数に引数がしまわれる
- 呼出された側が実行
- return文の式の値が計算される
- 返り値が呼出した側の式の値になる
- 呼出した側の実行が再開

練習