

プログラミング入門

第4回

箕原辰夫
2005年秋学期

Javaプログラミングの雛形

- 外側にクラスの定義がある
- 内側にメソッドの定義がある
- publicは、外部に公開するという意味

型の変換（暗黙）

- 暗黙の型変換（整数から実数へ）
 - 実数が現れれば、そこから実数として評価される
 - 実数型の変数に代入すれば、実数に変換される

型の変換（明示）

- 明示的な型変換
 - キャスト（型）演算子を使う
 - 変換可能であれば、その型に変換される
- **(int)** 整数への変換
 - 実数の場合は、切り捨てになる
 - 実数から整数に変換するときは必ず必要

文字列との結合

- 文字列同士の足し算
 - 足し合わされた別の文字列が生成される
- 文字列と他の型の値との足し算
 - 他の型は、自動的に文字列に変換される
 - * 数値→その数値を表わす数字などに
- 演算順序に注意！
 - 足し算は、左から行なわれる
 - "文字列"+整数+整数 の場合に注意
 - * "文字列" + (整数+整数) にする

プログラムの状態遷移

- プログラムの状態
 - 変数が一定の状態にあることを指す
- プログラムの状態遷移
 - 変数の値が移り変わっていくことを指す

ブロックと変数の有効領域

- ブロック
 - 波括弧 {} で囲まれた領域を指す
- 変数の有効範囲
 - 変数が宣言されてから、そのブロックでの実行が終了するまで

変数の値を使った繰返し

- ループ変数の値を変えていく
- 最終的に、継続条件を満たさなくする
- ループ変数の値を使ってメソッド呼出しのパラメータなどに使える
- 変数をいくつ使うべきかは、表わしたいデータの種類による

ループ変数のトレース

- ループ変数の値の変化を追う
 - 最初の値（初期値）
 - 途中の増分（差分）値
 - 継続条件が終わるときの値（最終値）
- 値の推移を追っていただければ繰返しはどう動くかわかる

繰返しを作るコツ

- ループ変数がいつかわ継続条件を満たさないように条件を作る
- ループ変数がどの変数か注意する
- 条件はきつめに設定する
- 1 回も繰返さない場合もある

AWTの描画メソッド（１）

- java.awtパッケージのGraphicsクラスにある
- Graphicsクラスは描画領域を表現している
- drawLine
 - 始点から終点に向けて、線を描く
 - 始点と終点を取り替えても可能
- drawRect
 - 3番目と4番目は、幅と高さ
 - パラメータはdrawLineと同じではない

AWTの描画メソッド (2)

- drawOval

- 外接する四角形はdrawRectと同じ
- 幅と高さが同じだと正円
- 幅と高さが異なると楕円

- drawArc

- 四角形に内接する円を描くのはdrawOvalと同じ
- 開始角度と角度差を360°で指定

AWTの描画メソッド (3)

- drawRoundRect
 - drawRectと同じだが、丸みの幅と高さも指定する
- draw3DRect
 - drawRectと同じだが、凸凹を論理値で指定する
 - 灰色などの中間色でないと効果がわからない

AWTの描画メソッド（４）

- 塗りつぶしメソッド
 - それぞれに対応して用意されている
 - fillRect, fillOval, fillArc, fillRoundRect, fill3Drect
- 彩色（setColor）は、描く前に行なう
- その他
 - copyArea, clearRect

グラフィックスと繰返し

- drawLineによる繰返し
 - drawLine(x1, y1, x2, y2);
- 縦線（垂直線）は、x 座標が等しい
- 横線（水平線）は、y 座標が等しい

水平線を使った繰返しと描画

- 両端のy座標は、繰返しに併せて変えて行く
 - ループ変数を使う
- x座標を変化させて、動きを示す
 - 両端が同じように変化→平行四辺形、四角形など
 - 違う形で変化→台形、三角形など

その他のグラフィックス・メソッドを 使った繰り返し

- drawRectで、ピラミッドを上から見たような入れ子四角形
- 幅と高さが、始点の移動よりも2倍小さくなっている

多重の繰返し

- ループ変数を 2 つ使う
- 外側の繰返しと内側の繰返しに分かれる
- 内側の繰返しが行なわれるのは
 - ➔ 外側の繰返しの回数 × 内側の繰返しの回数
- 掛け算になることに注意する
- 内側の繰返しの継続条件を外側の繰返しのループ変数を使って制御できる

条件分岐の意味（縮退した繰返し）

- 繰返しの 1 回版だと思えばよい



Normal



Repetative



Conditional

3つのif文

- 書式 1 : 条件を満足しなければスキップする
 - `if ( 条件 ) { 満足するとき }`
- 書式 2 : 条件を満足する場合としない場合
 - `if ( 条件 ) { 満足するとき } else { しないとき }`
- 書式 3 : 上から条件を満たすものを当てはめていく
 - `if ( 条件 ) { 満足するとき } else if ( 次の条件 ) ...`

よくある文法エラー

- 反対の条件を書いてしまう
- 条件の後に ; (セミコロン) を入れる
- else if の else を忘れる
- 複合条件を, (カンマ) で記述する

繰返しと条件分岐の組合せ

- 繰返しの中のブロック内に、if文を入れる
- ある回だけ、特別な制御をさせたいときに使う
- if文の条件に整数剰余を使うと周期的に何かをさせられる

整数除算・剰余と繰返しの組合せ

- 整数除算

→ 段階的になにかをさせたいときに使う

$x / n * n \rightarrow n$ を1つの段階にできる

- 整数剰余

→ 周期的に何かをさせたいときに使う

$x \% n \rightarrow n$ が周期になる

if文のネスト

- 外側のif文
 - 大きく分けたいときにつかう
- 内側のif文
 - その条件のなかで更に細かく分けたいときにつかう

論理式

- 条件式を複数使うことができる
 - 論理積 `&&` 両方の条件を満たす
 - 論理和 `||` どちらかの条件を満たす
 - 否定 `!( )` 反対の条件にする

論理式の記述の仕方

- かならず論理積・論理和などで結ぶ必要がある
 - $100 < x < 200$ はだめ
 - $100 < x \ \&\& \ x < 200$
- ドモルガンの法則などで変換することが可能
 - $!(\text{条件A} \ \&\& \ \text{条件B}) \Leftrightarrow !\text{条件A} \ || \ !\text{条件B}$
 - $!(\text{条件A} \ || \ \text{条件B}) \Leftrightarrow !\text{条件A} \ \&\& \ !\text{条件B}$

今日の課題

- 宛先：minohara@sfc.keio.ac.jp
- 件名：java-2
- 課題：教科書の課題5-6
- 内容：プログラムをコピー＆ペースト、質問、授業の感想