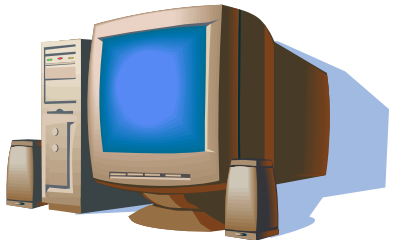
情報①

1学期　第3回

Project.3「条件分岐、乱数」



今日の授業のながれ

- **Project2**

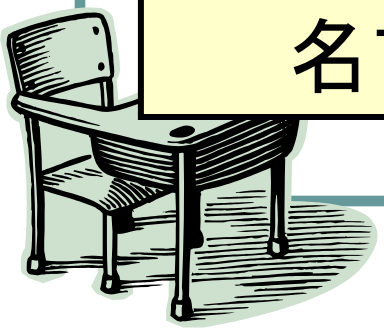
「コンピュータに情報を記憶させよう（変数）」

- **Project.3**

「プログラムに知性を(条件分岐、乱数)」

  - 配布資料
    - 1学期 第3回演習チェックシート　part.1-3

クリアファイルを返却します。
名前を呼ばれたら取りに来てください。



前回の続きから始める！

出欠について

- 授業ではクリアファイルと併せ、[WebClass]でも「出席」を取ります
  - 授業開始[10分後]まで⇒「出席」扱い
  - 以降[授業時間内]⇒「遅刻」扱い
  - 以降[授業終了後]⇒「欠席」扱い

- WebClass上部「出席」⇒「2025/××/××出席確認」⇒「開始」⇒「出席します！」

演習の取り組み方

演習の取り組み方

- 演習は「授業用サイト（オンラインテキスト）」を自分で読み進めて解いていく
  - 1時間目と2時間目に分けているが、自分のペースで進めること

（コンピュータ教室の場合）

- 課題の提出は「演習チェックシート」で行う

演習の取り組み方

- 例題をきちんと作ること
  - テキストは絶対に読み飛ばさない！
  - サンプルプログラムは実行してプログラムの意味を考える（納得して進む。納得できなければ，質問すること）
- 練習問題の数をこなすのが目的でない
  - 早く進むこと≠良いこと（とは限らない）
  - だらだらやること≠良いこと（とはいえない）
  - 一問だけでも，きちんと納得することが大切

演習の順番

① 先にスライドでProjectの流れを掴む
② サンプルプログラムはすべて実行してプログラムの意味を理解し、例題も自分なりに一度考える
③ オンラインテキストを読む
④ 演習課題に取り組む

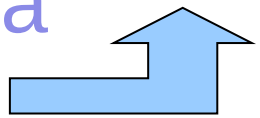
今日の目標

今日の目標

[Project.2]

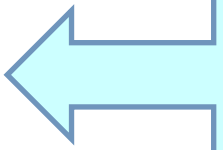
- 前回からの続き
- InputHouseEx2.java
- InputHouseEx3.java

最低限
ココはクリア！



[Project.3]

- OmikujiEX.java
- OmikujiEX2.java
- OmikujiEX3.java

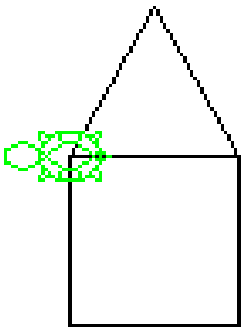


進んでいる
人はチャレ
ンジ！

前回までの復習

Project1
「はじめてのJAVA
タートルプログラミング①」

Java
プログラム



クラスブロック

メソッドブロック1

メソッドブロック2

```
/*
 * 家を書くプログラム
 */
public class House extends Turtle {

    // 起動処理
    public static void main(String[] args) {
        Turtle.startTurtle(new House());
    }

    // タートルを動かす処理
    void start() {

        // 屋根を書く
        rt(30);// 30度右を向く
        fd(50);// 50歩前に進む
        rt(120);
        fd(50);
        rt(120);
        fd(50);

        // 本体を書く
        lt(90);
        fd(50);
        lt(90);
        fd(50);
        lt(90);
        fd(50);
        lt(90);
        fd(50);
    }

}
```

良いコメントとは

- **良いコメント**
  - プログラムの構造が表現できている
  - ぱっと見分かりにくいこと(プログラマーの意図)が説明されている
- **悪いコメント**
  - プログラムを日本語にしただけ

例：fd(100); //亀を前に100歩進める

よい例とは？

良いコメントの例：構造の表現

```
// タートルを動かす処理
void start() {

    //前処理：描きだし位置の調整
    rt(90);//底辺が平らになるように

    //本処理：五角形を描く
    fd(50);
    lt(72);
    fd(50);
    lt(72);
    fd(50);
    lt(72);
    fd(50);
    lt(72);
    fd(50);

}
```

空行と行頭コメントを使って処理が大きく二つに分かれることを表現できている

プログラムの構造が表現できている

良いコメントの例:プログラマの意図

```
// タートルを動かす処理
void start() {

    //前処理：描きだし位置の調整
    rt(90);//底辺が平らになるように

    //本処理：五角形を描く
    fd(50);
    lt(72);
    fd(50);
    lt(72);
    fd(50);
    lt(72);
    fd(50);
    lt(72);
    fd(50);
}
```

行の意図はその行の後にコメントを入れる
一見分かりにくい，90度右へ曲げている意図（目的，なぜそうしているか）を書く

プログラマーの意図が説明されている

Project.2
「コンピュータに情報を記憶させてみよう（変数）」

変数とは？

変数:『値が定まっておらずその時々で変化する値』

Javaでは

「数値や文字列を
しまっておくための箱
（もしくは表）」

変数の型

- 『変数』の中に入るものは整数、実数、文字などいろいろな種類(型)がある
- 『変数』がどの種類(型)を使っているかわかるよう変数の宣言の文頭で書く必要がある

	整数 (負の数含)	実数 (小数含む)	文字列
型	byte , short , int , long	float , double	char, String

変数の型

- 『変数』の中
  などいろ
- 『変数
  るよ

【注意！】
整数はint
実数はdouble
文字列はString
の3つを覚えればOK

型	, short , int , long	double	char, String

変数の作り方

変数を作る　→①変数の名前を決める（**変数の宣言**）
　　②変数の値を決める　（**値の代入**）

③変数の初期化

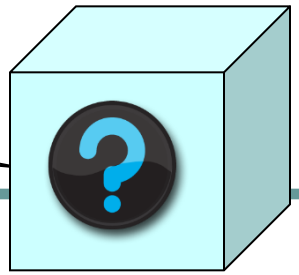
例1）

```
①int X ;
②X = 100;
```

例2）

```
③int X = 100 ;
```

箱の名前は『X』
中身は整数『100』



変数：X

変数の作り方

変数を作る　→①変数の名前を決める（変数の宣言）

③変数

例1）

①in

②X =

代入）

【注意！】

『変数の宣言』は

必ず変数を使う前に

記述しておく!!

箱の名前は

中身は整数

変数：X

【前回の補足】

変数の有効性

計算と変数　2つの方法

- ①計算結果を変数に代入しないで直接使う

```
int x;
x = 10;
fd(x / 2);
```

- ②計算結果を一度変数に代入してから使う

```
int x;
int y;
x = 10;

y = x / 2;
fd(y);
```

何が違うの？

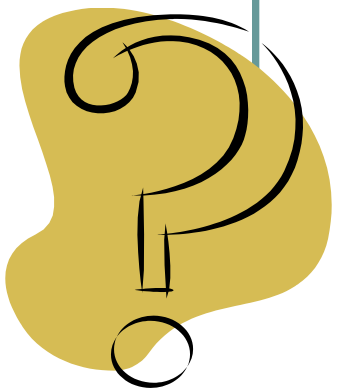
①計算結果を変数に
代入しないで直接使う

クイズ:計算と変数　2つの方法

- Q1.割り算の計算回数は何回？
- Q2.fdの長さを3分の1するには修正箇所は何カ所？

```
int x;
x = 10;
fd(x / 2);
rt(90);
fd(x / 2);
rt(90);
fd(x / 2);
rt(90);
fd(x / 2);
```

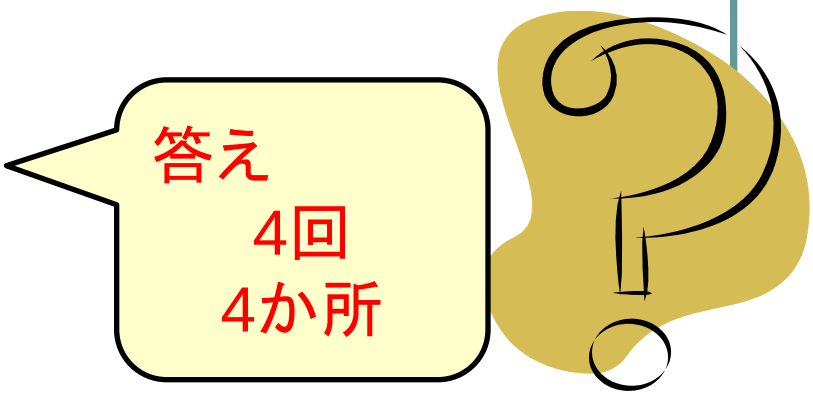
次ページの答えを見る前に自分で考えてみよう!



クイズ:計算と変数　2つの方法

- Q1.割り算の計算回数は何回？
- Q2.fdの長さを3分の1するには修正箇所は何カ所？

```
int x;
x = 10;
fd(x / 2);
rt(90);
fd(x / 2);
rt(90);
fd(x / 2);
rt(90);
fd(x / 2);
```



②計算結果を一度変数に代入してから使う

クイズ:計算と変数　2つの方法

- Q1.割り算の計算回数は何回？
- Q2.fdの長さを3分の1するには修正箇所は何カ所？

```
int x;
int smallX;
x = 10;
smallX = x /2;
fd(smallX);
rt(90);
fd(smallX);
rt(90);
fd(smallX);
rt(90);
fd(smallX);
```

次ページの答えを見る前に自分で考えてみよう!

クイズ:計算と変数　2つの方法

- Q1.割り算の計算回数は何回？
- Q2.fdの長さを3分の1するには修正箇所は何カ所？

```
int x;
int smallX;
x = 10;
smallX = x /
fd(smallX);
rt(90);
fd(smallX);
rt(90);
fd(smallX);
rt(90);
fd(smallX);
```

『変数』を使うと後々便利!!

答え
1回
1か所

プログラムの開発手順

プログラムの開発手順

①どんなプログラムが必要か考える

②プログラム（ソースコード）を書く（.java）

コンパイルエラー

③機械語に翻訳（コンパイル）する（.class）

実行エラー

④プログラムを実行して動作するか確認

プログラムの開発手順

①どんなプログラムが必要か考える

②プログ

③

④プログラムを実行して動作するか確認

ひとつ命令を書いたら
Ctl+sをする癖をつける
（すぐにコンパイルエラー
を発見して修正ができる）

ラー

class）

実行

プログラミングの開発手順

- 悪い例
  - あまり考えずに全て書く(ぐちゃぐちゃになる)
  - 実行してみる
  - 修正する(どこを修正するか探すのが大変)
- 良い例
  - 全体の構造を考え，どの部分から作るか検討
  - ある一部分を書く
  - 実行する
  - 修正する
  - 残りの部分へ・・・

(正しい部品を組み合わせれば，正しく全体が完成する)

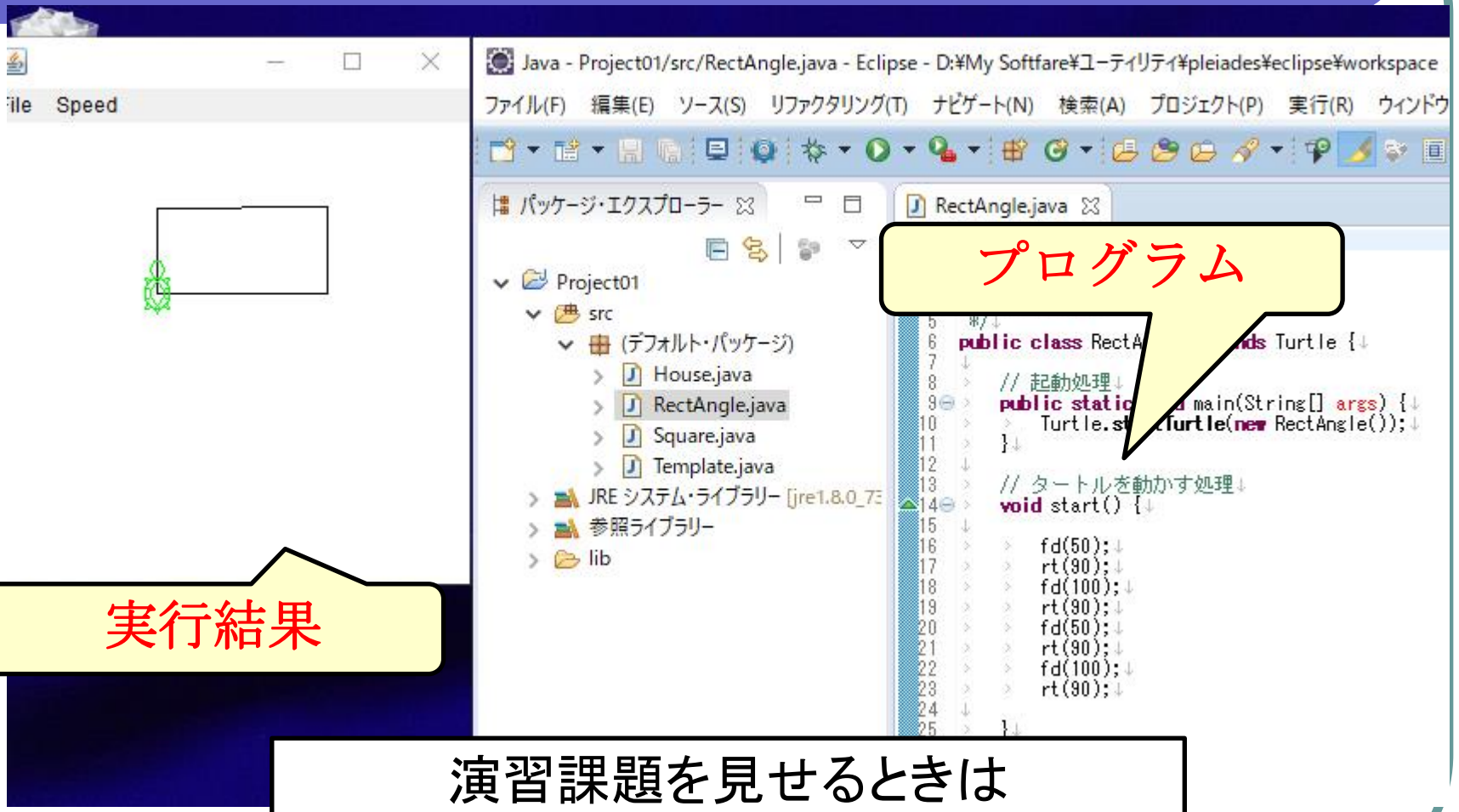
前回の演習

演習について

- **演習課題の解答について**
  - 「実行結果」と「プログラム」の両方を見せる
- **課題用プログラムの作成方法**
  - Template.javaを右クリック→コピー
  - その場で右クリック→貼り付け
  - 名前の競合→(該当の演習課題の)名前を付ける

授業用サイトにコピー方法が載せました

解答例



演習課題を見せるときは

「実行結果」と「プログラム」

両方を提示すること！

演習について

- **コピー＆ペーストを利用しよう（楽をする）**
  - 今回の「Project」のプログラム例から使える場所をコピー＆ペースト
  - 今まで書いた自分のプログラムで重複する部分をコピー＆ペースト　など
- **実行結果画面サイズの変更**
  - 実行結果画面の隅を引っ張ったり、最大化する
  - window.size(640, 360);などと変更する

【基本問題】

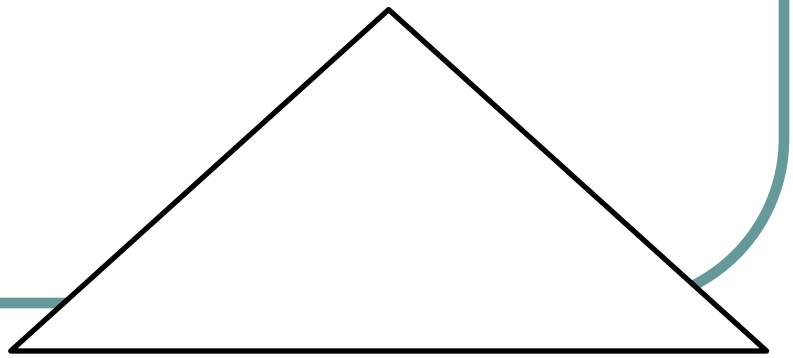
練習問題

- 変数「length」にユーザが値を入力して

自由なサイズの直角二等辺三角形を描くプログラム（InputTriangle.java）を作りなさい

- 実行すると，「1辺の長さを入力してください」と表示
- ユーザーが数字を入力するとその大きさの図形を描く
- 適切なコメントを入力して「読みやすいプログラム」にする
- 図形の向き（角度）も揃える

例）Example4.java
を参考にしよう！



おまけ～ペンの上げ下げ～

- タートルにしばらく図形を描かせない方法

up(); タートルがペンを上げる(軌跡を書かない)
down(); タートルがペンを下げる(軌跡を書く)

- 例　開始位置を移動してから図形を描く
  - **up();**
  - **fd(100);//ここで移動や回転などする**
  - **down();**

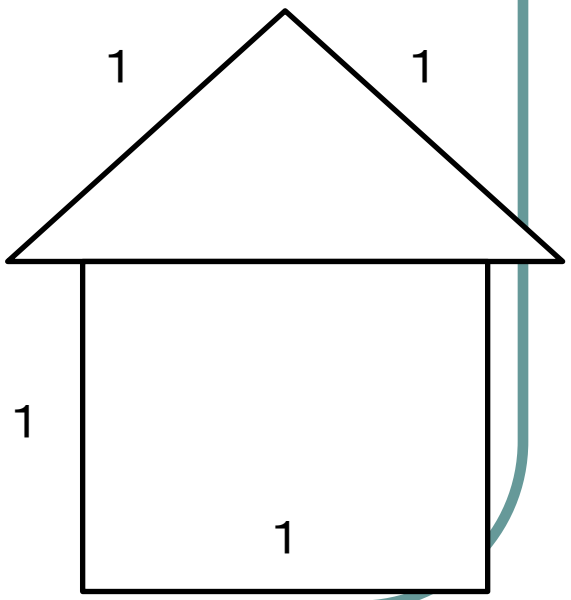
この下に図形を描く

練習問題

- 変数「length」にユーザが値を入力して

自由なサイズの家を書くプログラム (InputHouseEx.java) を作りなさい

- 屋根は直角二等辺三角形
- 壁と窓枠は正方形
- 壁1辺の長さ = 3角形の等辺の長さ



演習の取り組み方

- 早く終わった人は・・・

  【発展問題】にチャレンジ！

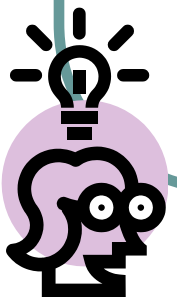
- どうしても分からない人は・・・

  次のページ【ヒント】を参照

ヒント！！！

ヒント

- 直角二等辺三角形について
  - 内角の和は180度で、鋭角の角度は・・・？
  - 辺の長さの比は1:1:√？
  - 整数はintだけど、小数の時はdouble
  - 「ルートの計算」を見直してみよう
- 家について
  - 小かっこを使った細かい計算ミスに注意
  - 「かっこについて」を見直してみよう

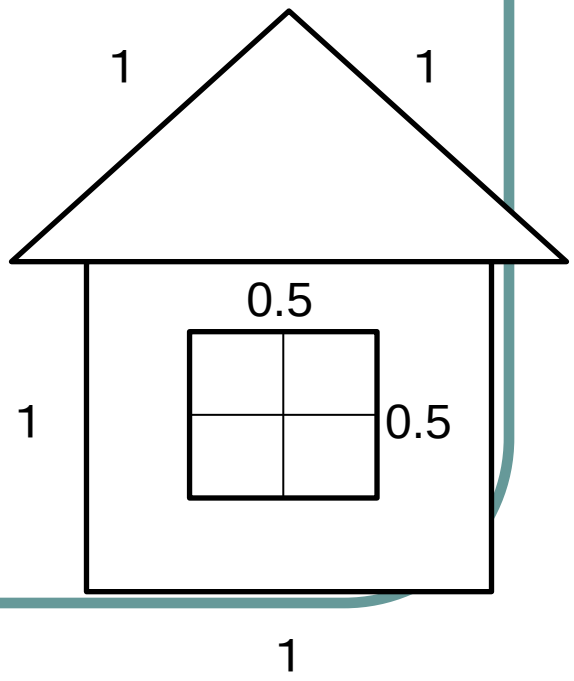


【発展問題】

練習問題

- 家を書くプログラムを改良して窓付きの家(InputHouseEx1.java) を作りなさい
  - 壁1辺の長さ = 窓枠1辺の長さ ×2
  - 窓枠の中はさらに4つの正方形
  - 描き終わると以下のメッセージを表示

「1辺●●の大きさの家を書きました」



おまけ～色の変更方法～

- 色を変更するタイミングで以下の命令を挿入

color(java.awt.Color.カラー名);

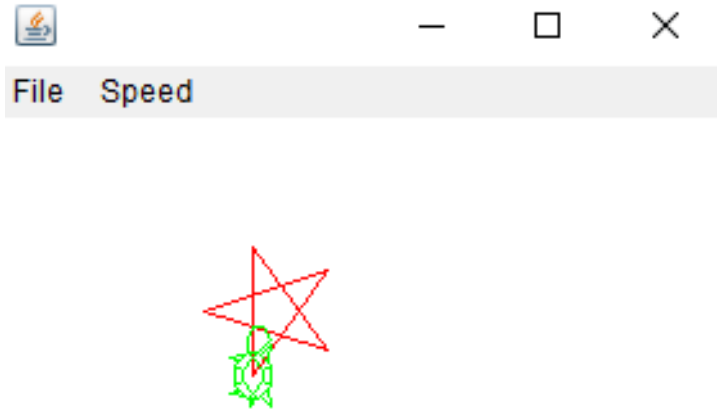
- 例　赤・青・灰色　の場合
  - **color(java.awt.Color.red);**
  - **color(java.awt.Color.blue);**
  - **color(java.awt.Color.gray);**

例題：5
（Example5.java）

例題：5

- **例題5:**

  Example5.javaを赤色の星を描くようにプログラムを書き換えなさい



実行結果

次ページの答えを見る前に自分で考えてみよう!

解答例

```
// タートルを動かす処理
void start() {

    print("星の大きさを半角数字で書いてEnterキーを押してください。");

    // ユーザーからの値の入力を変数「length」に代入する
    int length;
    length = input();

    // 例題5:以下にペンの色を変更して赤い星を描くようにプログラムを書き換えなさい！
    color(java.awt.Color.red);

    // 自由なサイズの星を描く
    fd(length);
    rt(144);
    fd(length);
    rt(144);
    fd(length);
    rt(144);
    fd(length);
    rt(144);
    fd(length);
    rt(144);

}
```

プログラム

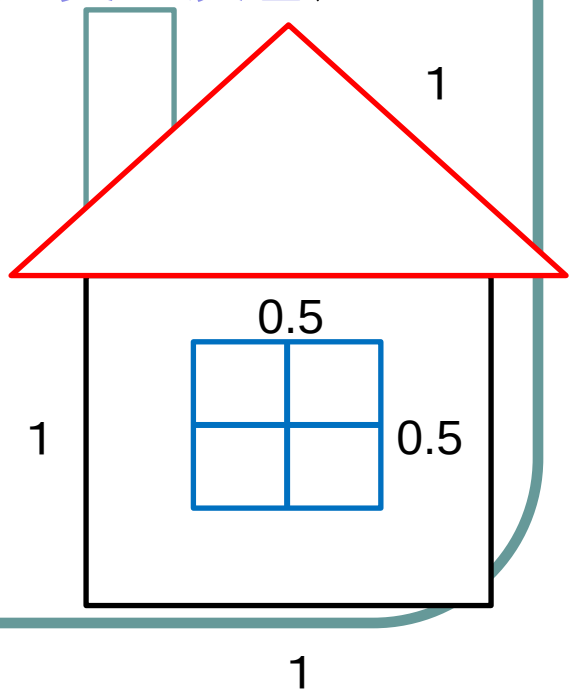
問題 | @ Javadoc | 宣言 | コンソール

Example5 [Java アプリケーション] C:¥Program Files (x86)¥Java¥jre1.8.0_101¥bin¥javaw.exe (2020/05/23 16:24:52)

```
Turtle Version: 1.0.5 (2007/12/21)
星の大きさを半角数字で書いてEnterキーを押してください。
50
```

練習問題

- 家を書くプログラム改良して色と煙突付きの家(InputHouseEx2.java)に改良しなさい
  - 煙突を追加（※高さは三角形、左辺は壁と同じx座標）
  - 色をつけなさい（屋根は赤、窓は青、煙突は灰色）



Project.3
「条件分岐」

学習事項

学習事項

- 条件分岐
  - 条件式
- if文
  - 変数の有効範囲
- 乱数
- etc
  - おまけ
  - 画像の表示方法

Ⅰ．条件分岐
（場合分け）

復習

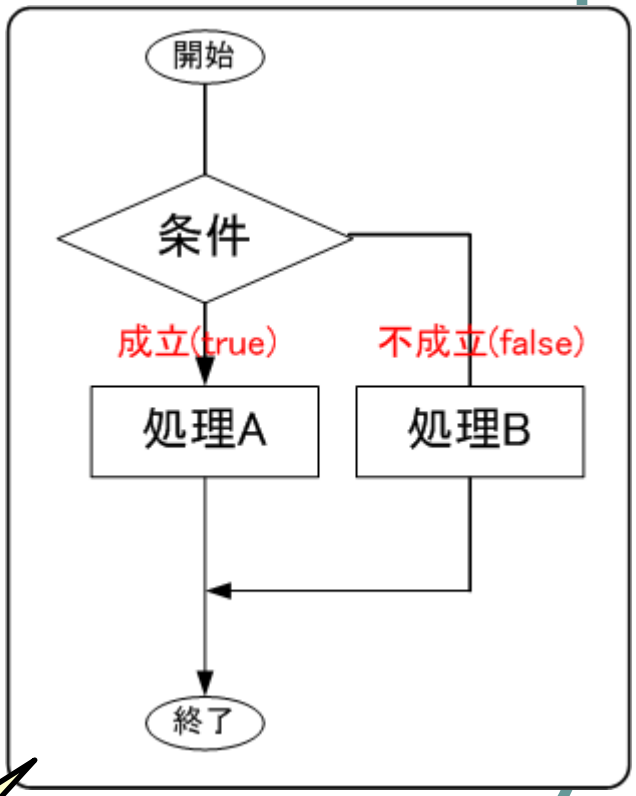
- 命令
  - startメソッドブロックの中にある，rt(〇〇)やfd(〇〇)がタートルに対する指示・命令
  - ルール
    メソッド内に書かれた命令は，必ず上から順番に実行される（**順次実行**）

```
void start() {

    // 屋根を書く
    rt(30);// 30度右を向く
    fd(50);// 50歩前に進む
    rt(120);
    fd(50);
    rt(120);
    fd(50);

    // 本体を書く
    lt(90);
    fd(50);
    lt(90);
    fd(50);
    lt(90);
    fd(50);
    lt(90);
```

この順番以外は？

条件分岐

- **『条件分岐』**を用いることでプログラムの流れに変化を与えることができる。
- このような効果を持つ命令文を**「制御文」**と呼ぶ



フローチャート

条件分岐とは

- 代表的な条件分岐構文は以下の3つ
  - if文
  - switch文
  - for文(繰り返しも含む)

- 今回は最も簡単な**「if文」**についてオンラインテキストとサンプルプログラムを見ながら勉強しましょう！

条件分岐とは

条件分岐とは

- 『ある特定の条件を満たしたときだけ実行させたい（あるいはさせたくない）動作』があるとき、**「条件式」**を用いてプログラムの流れを制御することができる。

- これを『条件分岐（場合分け）』と呼ぶ。

条件式とは

条件式とは

- **『条件分岐（場合分け）』は○か×かを確実に判断できる条件が必要。**
  - 『等号・不等号（== ,<,>）』等の**比較演算子**
  - 集合関係を利用した**論理演算子**

などが条件式に用いられます。

true は ○
false は ×
のこと

表 4.3.1.1 Javaで使える数値の比較演算子

演算子	表記例	評価
==	A==B	AとBが同じ値の時成立(true)
!=	A!=B	AとBが同じ値でない時成立(true)
>	A>B	Aが Bより大きい時成立(true)
<	A<B	Aが Bより小さい時成立(true)
>=	A>=B	Aが B以上の時成立(true)
<=	A<=B	Aが B以下の時成立(true)

表 4.3.2.1 Javaで使える論理演算子

演算子	表記例	意味
&&	A&&B	AとB両方ともtrueの時true
||	A||B	Aもしくは Bが trueの時true

例題 : 1

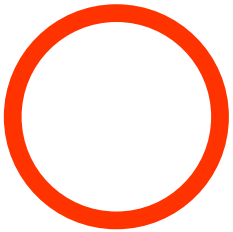
例題：1

- 『xが50以上100未満』を
  条件式（演算子）を用いて表すには？

次ページの答えを見る前
に自分で考えてみよう!

例題：1

- 『xが50以上100未満』を
  条件式（演算子）を用いて表すには？



答え

50<=x && x <100

よくある間違え

50<= x <100

Ⅱ．if文

①2択の条件分岐
（if ~ else 文）

2択の条件分岐

```
if（条件式）{
        処理A
}else{
        処理B
}
```

「条件式の結果が〇なら処理A

×なら 処理Bが行われる」

例）Condition.java
を見てみよう！

例について

① Condition.javaを実行してみる
② プログラムをよく読んでみる
③ プログラムの意味を理解する
- 条件分岐（if～else文）
- 文字入出力

例題 : 2

例題:2

```
if( [条件式] ) {
    処理A
} else{
    処理B
}
```

```
int x = 1; // x の初期化

if (x == 1) {
 print("xは1です");
} else {
 print("xは1ではありません");
}
```

○ならば

× ならば

例題2:この場合の出力結果は？

次ページの答えを見る前に自分で考えてみよう!

例題：2

```
if( [条件式] ) {
    処理A
} else{
    処理B
}
```

```
int x = 1; // x の初期化

if (x == 1) {
 print("xは1です");
} else {
 print("xは1ではありません");
}
```

○ならば

× ならば

例題2：この場合の出力結果は？

Xは1なので・・・

【実行結果】

xは1です

②2択の条件分岐
（if ~ 文（else無し））

2択の条件分岐

```
if（条件式）{
        処理A
}
```

「条件式の結果が〇なら処理A」

例）Rohrer2.java
を見てみよう！

例について

① Rohrer2.javaを実行してみる
② プログラムをよく読んでみる
③ プログラムの意味を理解する
- 論理演算子

例題 : 3

例題:3

```
if( [条件式] ) {
    処理A
}
```

```
int x = 2; // x の初期化

if (x == 1) {
 print("xは1です");
}
```

〇ならば

例題3:この場合の出力結果は？

次ページの答えを見る前に自分で考えてみよう!

例題:3

```
if( [条件式] ) {
    処理A
}
```

```
int x = 2; // x の初期化

if (x == 1) {
 print("xは1です");
}
```

〇ならば

例題3:この場合の出力結果は？

Xは2なので・・・

【実行結果】

（何も出力されない）

③複数の条件分岐
（if ~ else if ~ else 文）

条件分岐とは

- ○○かどうか調べて，
  - □□だった場合は，××をする
  - △△だった場合は，★★をする
  - それ以外の場合は，■■をする

複数の条件に応じた『条件分岐（場合分け）』

どうすればよいか？

2択の条件分岐

```
if( [条件式①] ) {
    処理A
} else if( [条件式②] ) {
    処理B
} else{
    処理C

}
```

「条件式①の結果が
〇なら処理A
×なら・・・
条件式②の結果が
〇なら処理B
×なら 処理C
が行われる」

例) Rohrer.java
を見てみよう！

例について

① Rohrer.javaを実行してみる
② プログラムをよく読んでみる
③ プログラムの意味を理解する
- 比較演算子

例題 : 4

例題：4

```
if( [条件式①] ) {
    処理A
} else if( [条件式②] ) {
    処理B
} else{
    処理C
}
```

```
int x = 3;

if (x == 1) {
 print("xは1です");
} else if (x == 2) {
 print("xは2です");
} else {
 print("xは1か2以外です");
}
```

例題4：この場合の出力結果は？

次ページの答えを見る前に自分で考えてみよう！

例題：4

```
if( [条件式①] ) {
    処理A
} else if( [条件式②] ) {
    処理B
} else{
    処理C
}
```

```
int x = 3;

if (x == 1) {
 print("xは1です");
} else if (x == 2) {
 print("xは2です");
} else {
 print("xは1か2以外です");
}
```

例題4：この場合の出力結果は？

Xは3なので・・・

【実行結果】

xは1か2以外です

【注意事項】

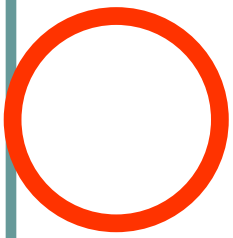
注意事項

- **else if([条件式③])**…と条件式を増やせば、4択以上の**『複数の条件分岐（場合分け）』**を行うことができる

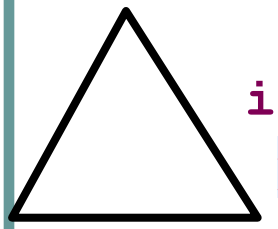
- 条件式の中にさらに条件を組み込む（**入れ子状**にする）こともできる

注意事項

- 条件式の**直後の処理が1行**で表せる場合は**{}の省略**もできてしまうが・・・



```
if (x == 1) {
 print("xは1です");
}
```



```
if (x == 1)
 print("xは1です");
```

慣れない内はたとえ一行でも省略せずに{}を書くことを強くお勧めする！

変数の有効範囲とブロックについて

- 変数には**有効範囲（スコープ）**が存在する

- **変数の有効範囲（スコープ）**は**定義されたブロックの中** (「{}」で囲まれた範囲) のみ有効
  - 範囲外で変数を呼び出すと、コンパイルエラー
  - 範囲外なら同じ変数名を使うことができる
  - その場合、それぞれの変数が違う値を持つ

クラスブロック

```
/**
 * ○×ゲーム省略版
 */
public class MaruBatsuGame extends Turtle {

        // 個々のマスの見た目（3種類）を定義する
        ImageTurtle kara = new ImageTurtle("kara.gif");
        ImageTurtle maru = new ImageTurtle("maru.gif");
        ImageTurtle batsu = new ImageTurtle("batsu.gif");

        // 起動処理
        public static void main(String[] args) {
                Turtle.startTurtle(new MaruBatsuGame());
        }

        // タートルを動かす処理
        public void start() {
                initialize();// 初期化
                doGame();// ゲームをする
        }

        void doGame() {

        // 勝敗判定
        int winnerNumber = checkWinner();

        }
```

メソッドブロック1

メソッドブロック2

メソッドブロック3

①ココで変数の定義

⇩

プログラム全体で有効

②ココで変数の定義

⇩

このブロックのみ有効

Ⅲ. 乱数

乱数について

- 実行するたびに異なる数字を出したい場合には「乱数」を利用するとよい。
- 例）

```
int x;
x= random(100);
```

0～99の乱数を生成して，xに代入

（実行するたびに異なる数字が入るが，同じ数字が入ることもある）

乱数は0からスタートしていることに注意！

例) RandomStar.java Janken.java を見てみよう！

例について

① RandomStarを実行してみる
② プログラムをよく読んでみる
③ プログラムの意味を理解する
- 乱数

例について

① Janken.javaを実行してみる
② プログラムをよく読んでみる
③ プログラムの意味を理解する
- 乱数
- 複数の条件分岐（if～else if～else文）

例題 : 5

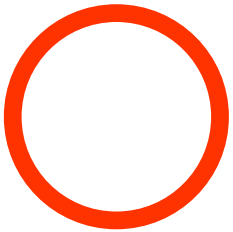
例題:5

- Javaで0からスタートする乱数ではなく『-20～20』の乱数を発生させるには？

次ページの答えを見る前に自分で考えてみよう!

例題：5

- Javaで0からスタートする乱数ではなく『-20～20』の乱数を発生させるには？



答え

random(41)-20;

よくある間違え

random(40)-20;

-20から19まで

【おまけ】

- ① タートルを隠す/表示する

課題によってはタートルを表示しないほうが良い場合もある。
その場合、以下の命令でタートルを隠すことができる。

hide();

逆に表示させたい場合には以下を記入する。

show();

（※　初期設定で,タートルは表示されるようになっているので通常は記入する必要はない。）

- ② ウィンドウのサイズを変更する

  ウィンドウ画面のサイズを調整するには以下の命令で可能である。

  window.size(数値, 数値);

  ちなみに、ウィンドウの表示位置を変更するのは以下の通り。

  window.warp(数値, 数値);

- ③　ウィンドウ画面を閉じる

  プログラムを書いているとウィンドウが溜まっていくことがある。そうならないためにはプログラムが終了する直前にウィンドウを閉じる命令を書いてしまえば良い。

  ウィンドウ画面は以下の命令で閉じることができる。

  System.exit(0);

  （※厳密にはウィンドウだけでなくプログラム自体を終了）

演習課題

課題 おみくじプログラム　解説

- Omikuji.javaを使って，おみくじプログラムを作る

- 大まかな処理の流れは以下のとおり
  - 1.前処理（プログラム済）
  - 2.おみくじを引く（乱数の利用）
  - 3.おみくじの結果を表示する（条件分岐、画像の表示）

課題 おみくじプログラム　解説

- 画像を表示するための命令

  例）

  ```
  new ImageTurtle(“daikichi.jpg”);
  update();
  ```

  - 画像はプログラム(Omikuji.java等)の一つ上の階層のフォルダに置く
  - 演習課題で必要な4種類の画像

  （chuukichi.jpg、daikichi.jpg、kyou.jpg、shoukichi.jpg）
  はすでにその場所に準備済み

演習問題

- **初級（OmikujiEX.java）**
  - 実行するたびに異なるおみくじの結果（画像）が表示されるようにしなさい
- **中級（OmikujiEX2.java）**
  - おみくじの結果（画像）に加えてコンソールに一言気のきいたコメントを表示させなさい（print文を使う）
- **上級（OmikujiEX3.java）**
  - 大吉，中吉，小吉，凶がでる確率を10%，40%，40%，10%になるように調整しなさい

次回予告

次回予告

- Project.3「プログラムに知性を(条件分岐)」
- Project.4「コンピュータの得意な仕事（繰り返し）」

クリアファイルを提出して帰ること！

NEXT TIME

以上

おつかれさまでした。

それでは、また次回！！