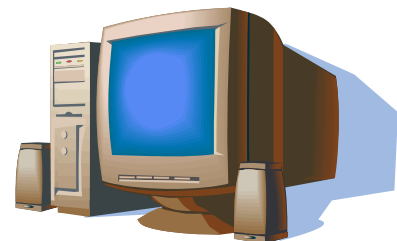


情報①

1学期 第2回

Project.2「変数、文章入出力」



今日の授業のながれ

➤ Project1

「はじめてのJAVAプログラミング」

➤ Project2

「コンピュータに情報を記憶させよう(変数)」

- 配布資料

- 1学期 第2回演習チェックシート part.1-2

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！」

教材 マイレポート 成績▼ **出席** その他▼ コース▼

学生モード 解除

出席

教材名	状態	回数制限	パスワード
» 2022/01/25 出席確認	出席	1回	-

合 計 1 回
必要出席数 1 回

出席:1
遅刻:0
欠席:0

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.1]

- 前回からの続き
- 2.6.3: 色々な図形を描こう①
- 2.6.3: 色々な図形を描こう②
- 2.6.3: 色々な図形を描こう③

最低限
ココはクリア！

[Project.2]

- InputTriangle.java
- InputHouseEx.java

進んでいる
人はチャレ
ンジ！

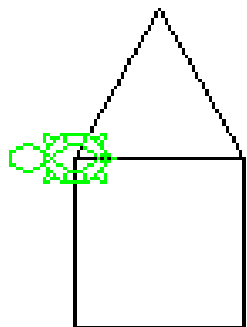
【前回までの復習】

Project1「はじめてのタートル JAVAプログラミング」

学習事項

- プログラムについて
 - 2.2.2.1 ブロック
 - 2.2.2.2 クラス
 - 2.2.2.3 メソッド
- 命令
 - 2.2.2.4 命令
- プログラムの工夫
 - 2.2.2.5 命令の区切り
 - 2.2.2.6 コメント

Java プログラム



```
⊖ /*
   * 家を書く プログラム
   */
   クラスブロック

public class House extends Turtle {

   // 起動処理
   メソッドブロック1
   public static void main(String[] args) {
       Turtle.startTurtle(new House());
   }

   // タートルを動かす処理
   ⊖ void start() {
       メソッドブロック2

       // 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);
       lt(90);
       fd(50);

   }
}
```

プログラムの解説1

- **ブロック**・・・中括弧{と}で囲まれた部分
 - Javaプログラムは複数のブロックから構成される
 - ブロックの中にブロックを入れることができる
- **クラス**・・・Javaプログラムの一つの単位
 - (本格的なプログラムはクラスが複数になるが)授業では一つだけ
 - **public class [クラス名] extends Turtle{** の括弧から、対応する(ファイルの最後にある)括弧 **}**までを**クラスブロック**
 - Javaのプログラムは全て**クラスブロック**の中に書く

プログラムの解説2

- **メソッド**・・・命令をまとめて一つに束ねたもの
 - (1学期は) **void メソッド名 ()**{から始まるブロックのことを**メソッドブロック**という
 - プログラムの命令は, 全て**メソッドブロック**の中に書く
 - **メソッドブロック**はクラスブロックの中に書く必要がある
 - しばらくすると複数のメソッドを扱うが, 1学期はプログラムはstartメソッド(void start())の括弧の中に書くと覚えておけばよい

プログラムの解説3

● 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);↓  
> fd(50);↓  
> }↓  
,
```

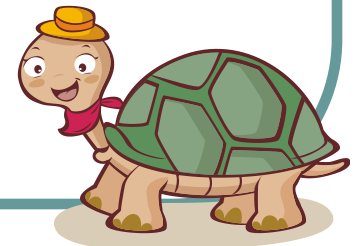


プログラムの解説4

- タートルへの基本命令

- rt(〇〇) タートルを〇〇度右に回らせる
- lt (〇〇) タートルを〇〇度左に回らせる
- fd(〇〇) タートルを〇〇歩前に進ませる
- bk(〇〇) タートルを〇〇歩後に進ませる

- up() タートルがペンを上げる(軌跡を書かなくなる)
- down() タートルがペンを下げる(軌跡を書く)



プログラムの解説5

● 命令の区切り

- Javaのプログラムでは命令と命令の区切りに必ず「;」(セミicolon)を入れる
- これを入れないと、コンピュータがどこで命令が区切られているのか、理解できない
- ちなみにセミicolonが入っていれば、複数の命令を以下のように同じ行に書けるが非推奨！

悪い例) lt(30); fd(50);

Square.java
を見てみよう！

プログラムの解説6

● 文字

- プログラム内は**全て半角文字**で記述し、大文字と小文字に注意
(※コメント部分は日本語OK)
- **ファイル名**も**全て半角文字**で、ファイル名先頭と単語の**区切りに大文字**を使う

例

Square.java

BigHouse.java

プログラムの解説7

● コメント

- コンピュータにプログラム(命令)として考慮されず, 読み飛ばされる部分
- 人間が読めないとメンテナンスに困るためコメントをつける
- 例えばファイルの先頭には, プログラムのタイトル, 名前や日付を書く

範囲指定コメント

- **/*から*/**までコメントになります

行コメント

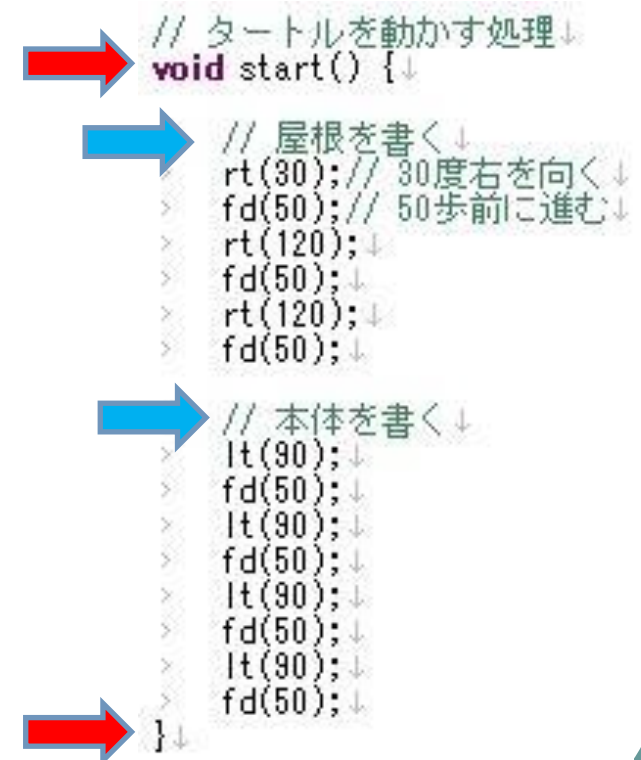
- **//**から行の終わりまでコメントになります

プログラムの解説8

● インデント

- プログラムに空白を挿入して意味のまとまりごとに字下げを行うこと
- 空白はスペースキーではなくtabキーを使った方がきれいに入る

始まりの文字の位置で意味を変えている



```
// 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);  
    > }  
}
```

The diagram illustrates the use of indentation in a Java program. A red arrow points to the start of the `void start() {` line. A blue arrow points to the first line of the roof-drawing block (`rt(30);`). Another blue arrow points to the first line of the body-drawing block (`lt(90);`). A final red arrow points to the closing brace of the `start` method (`}`).

House.java
を見てみよう！

【前回の補足】

良いプログラムを書くには

- ファイル名の**最初の一文字と区切りの文字を大文字にする**
 - OPentagon.java → × pentagon.java
 - ORectAngle.java → × rectangle.java
- **良いコメントを書く**
- **空行を入れる**
- **タブキー(Tabキー)で字下げをする**

良いコメントとは

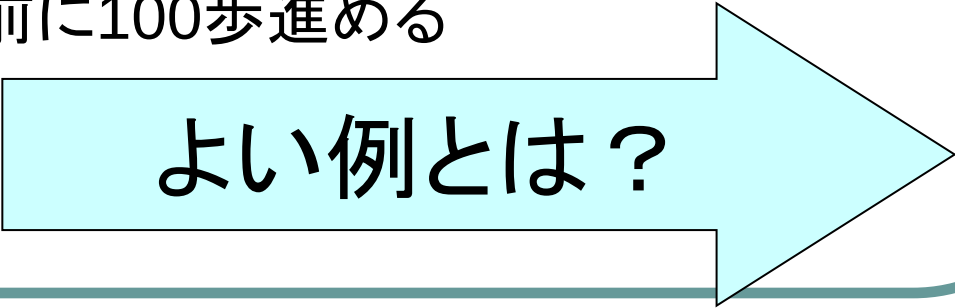
- 良いコメント

- 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かたけ

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: プログラムの意図

```
// 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

良いプログラムを書くには

- 意味のまとまりごとに空行を効果的に利用するだけでなく、タブキー(Tabキー)で字下げ(インデント)で対応する処理を見やすくする

空行で意味のまとまりを変えている
処理の深さに応じて、
字下げをしてカッコの位置を揃えたりしている

後で見直す時の工夫

```
// 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);
}
```

Java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)



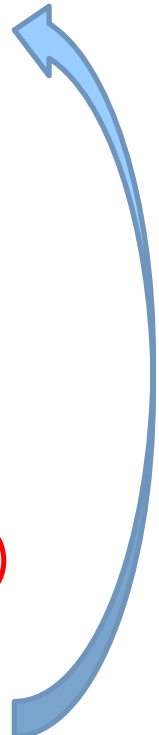
③機械語に翻訳（コンパイル）する(.class)

コンパイルエラー



④プログラムを実行して動作するか確認

実行エラー



Javaプログラムの開発手順

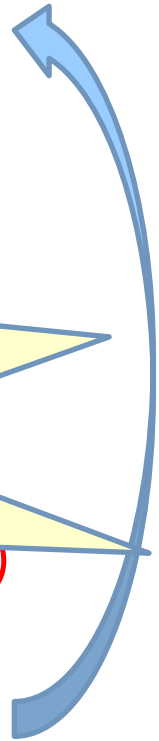
①どんなプログラムが必要かを考える

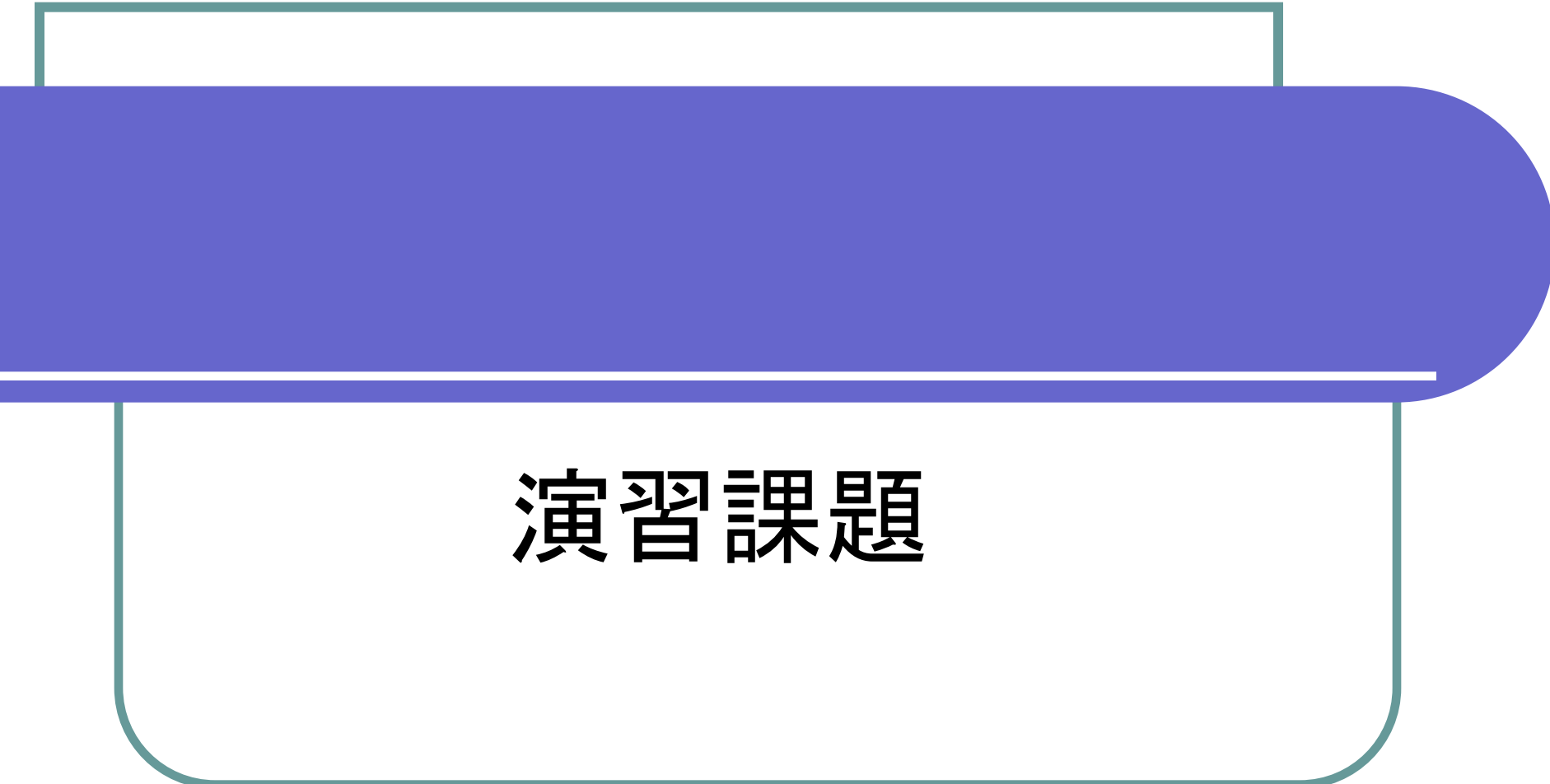
②プログラムを書く

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)

③プログラムをコンパイルする

④プログラムを実行して動作するか確認





演習課題

演習について

- 演習課題の解答について

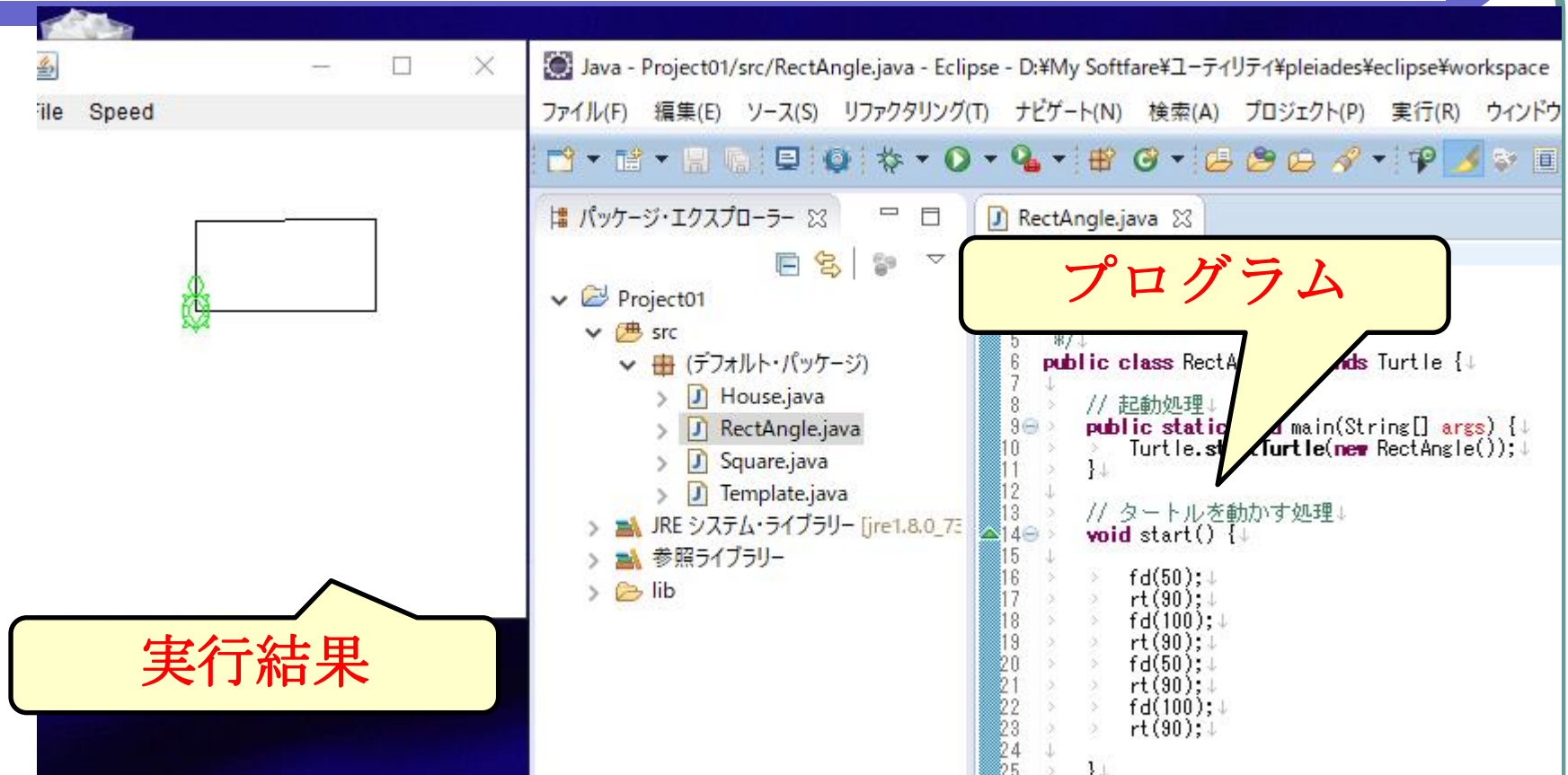
- 「実行結果」と「プログラム」の両方を見せる

- 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！

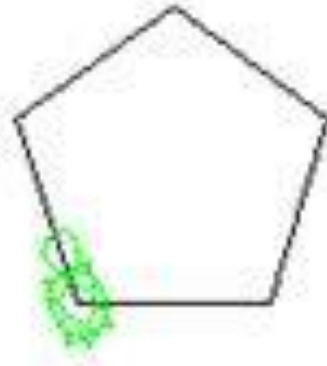
演習について

- 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

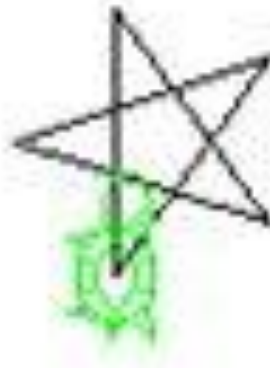
- 五角形を書くプログラム (Pentagon.java) を作りなさい



- Template.javaをコピーして作成する
- 図形の向き(角度)も揃える
- プログラムには適切なコメントを挿入してみよう

練習問題

- 星を描くプログラム (Star.java) を作りなさい



- Template.javaをコピーして作成する
- 図形の向き(角度)も揃える
- プログラムには適切なコメントを挿入してみよう

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！

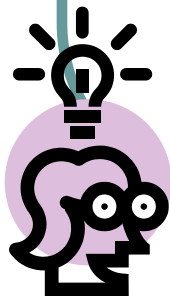
- どうしても分からない人は...

次のページ【ヒント】を参照

ヒント！！！！

ヒント

- 五角形を書くプログラムを作りなさい
 - はじめに平らにするため角度を調整する
 - 内角の和は540度なので一つの角度は・・・？
- 星を書くプログラムを作りなさい
 - 星の末端の角度を求めてみよう。
 - 真ん中にある図形は正五角形だから・・・？

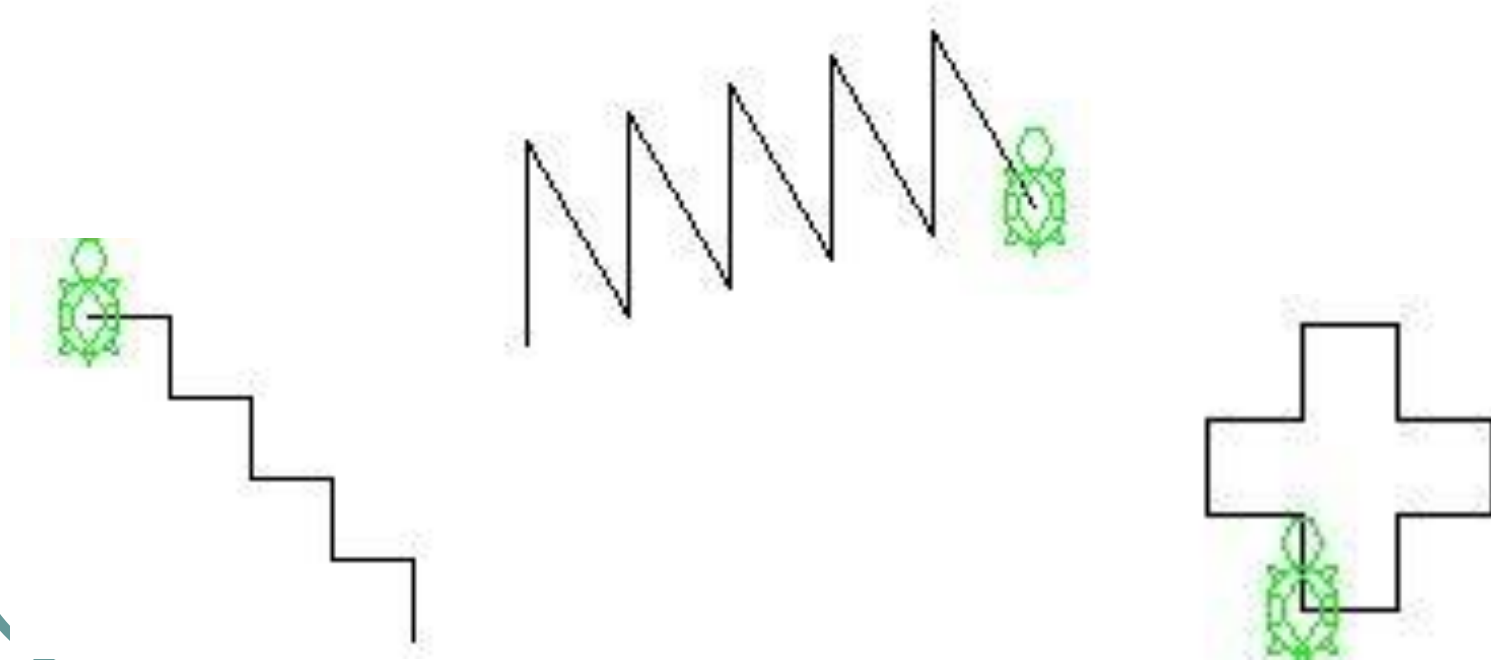


【発展問題】

練習問題

- 練習問題 2.1

- 色々な図形を描いてみましょう①～③





Project2

コンピュータに情報を記憶させてみよう(変数)

學習事項

学習事項

- 変数
 - 変数とは
 - 変数の型
 - 変数の作り方
- 変数を使った計算
 - 四則演算
 - ルートの計算
- 文章入出力
 - 文章出力
 - 文章入力

I . 変数

変数

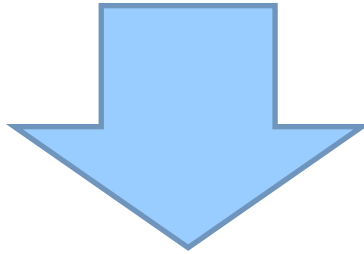
- タートルを動かすには`rt(90)`のように()内に数字を直接書き込んだ
- 一方、Javaなどのプログラミングの世界にはコンピュータに数字や文字を記憶して呼び出しをする『変数』と呼ばれる機能がある
- ここでは、『変数』について学習していく



変数とは？

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

変数の型

- 『変数』という箱に入るものは**整数、実数、文字**など様々な**型(種類)**がある
- Javaでは『変数』を新しく作る場合はどの**型(種類)**を使うか必ず書く必要がある

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

変数の型

- 『変数』に入るものは**整数**、**小数**、**文字列**など様々な型(種類)があります

【注意！】

整数は**int**

小数(実数)は**double**

文字列は**String**

の3つを覚えればOK

int, long

double

char,
String

変数の作り方

変数の作り方

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

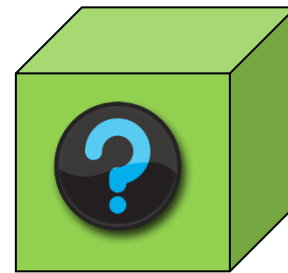
例1)

```
①int Hatena;  
②Hatena = 100;
```

箱の名前は『Hatena』
中身は整数『100』



①・②をまとめて行うことを
③変数の初期化と呼ぶ



変数: Hatena

例2)

```
③int Hatena = 100 ;
```

例) VariableHouse.java
を見てみよう！

例について

- ① VariableHouse.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
 - 「変数」を使ったプログラミング
 - Project.1のHouse.javaとの違い





【注意】

変数の注意点

- 『①変数の宣言』は変数を使うよりも手前に記述しておく必要がある
- 変数の中身はプログラムの途中でどんどん変化させることができる(詳細は後日)
- 変数は1つだけでなく、複数作ってもよい

例) MultiVariableHouse.java
を見てみよう！

例について

- ① MultiVariableHouse.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
 - 「変数(複数)」を使ったプログラミング
 - VariableHouse.javaとの違い



変数(複数ある場合のイメージ)

- 「変数」はプログラムで使うデータを保存しておくための「表」にも例えられる
- この表には複数の「変数名」と「値」を保存・読み出すことができる

変数名	値
x	←

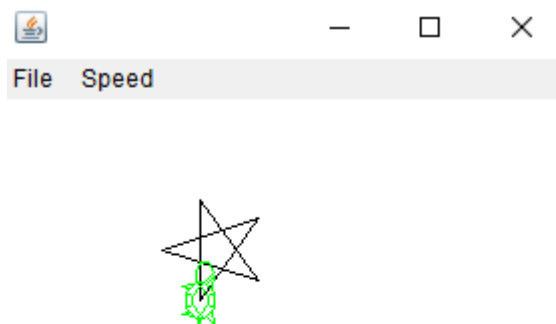
{ 30
50
100

などの値が代入される

例題: 1
(Example1.java)

例題: 1

- Example1.javaを
整数が入る変数「length」を利用して
星を描くプログラムに書き換えなさい



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

プログラム

```
// ①変数「length」の定義と値(50)の代入をする(もしくは①変数の初期化)  
int length;  
length = 50;
```

```
// 以下の『星を描くプログラム』のタートルが進んでいる値を変数「length」を使うように書き換える  
fd(length);  
rt(144);  
fd(length);  
rt(144);  
fd(length);  
rt(144);  
fd(length);  
rt(144);  
fd(length);  
rt(144);
```

```
}
```

Ⅱ．計算式について

計算について

- コンピュータは「電子計算機」なので、Javaでも式を用いて「計算」することができる
- 「計算式」の内以下の四則演算(+1)の記号を算術演算子と呼ぶ

演算子	演算
+ (shift + れ)	足し算をします
-	引き算をします
* (shift + け)	掛け算をします
/	割り算をします
% (shift + 5)	割り算の余りを求めます

Excelと(ほぼ)同じ
演算子を利用

変数を使った計算

変数を使った計算

- 数値を使った計算

- 例: タートルを「5×5」歩進める

`fd(5*5);`

今まで数値を書いていたところは計算式でも良い

- 計算結果を変数に代入

- 例: 変数「answer」に『計算結果』を代入

`int answer = 5*5;`

// answerの中は計算結果25が代入

変数を使った計算

- 数値だけでなく変数を使った計算

- 例:変数「answer」に『変数と数値の計算結果』を代入

- `int x = 50 ;` //変数 x の値は 50
- `int answer = x + 200 ;` // answerに250が代入

- 例:変数「answer2」に『変数同士の計算結果』を代入

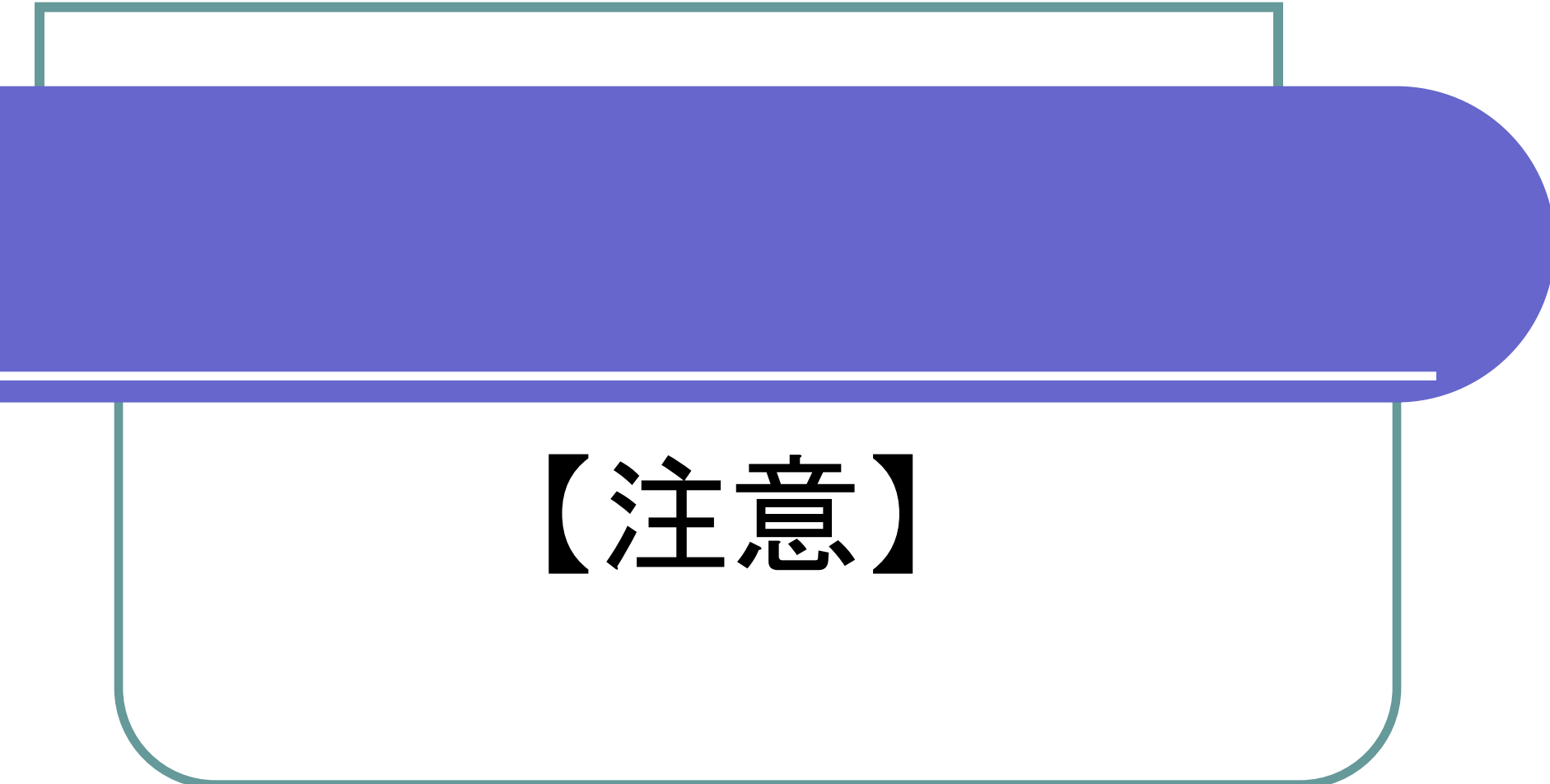
- `int y = 50 ;` //変数 y の値は 50
- `int z = 100 ;` //変数 z の値は 100
- `int answer2 = y + z ;` // answerに150が代入

例) DoubleHouse.java
を見てみよう！

例について

- ① DoubleHouse.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
 - 「**計算式**」を使ったプログラミング
 - ※実行時画面サイズを広くしてみる





【注意】

かっこについて

- Javaでの計算の順序は数学と同じ順序で行われるが、中カッコ{}や大カッコ[]を使うことができないので、代わりに小カッコ()を複数用いて計算します
 - 計算例: $\{1 + (2 - 3) \times 4\} \div 5$
 $(1 + (2 - 3) * 4) / 5$

ルートの計算

変数を使った計算

● ルートの計算方法

- 例:変数「A」に『 $\sqrt{2}$ 』,変数「B」に『 $A\sqrt{5}$ 』を代入

- `Int A = Math.sqrt(2)` ; //変数A に $\sqrt{2}$ を代入

ルート2

- `Int B = A * Math.sqrt(5)` ; //変数B に $A\sqrt{5}$ を代入

A かける ルート5

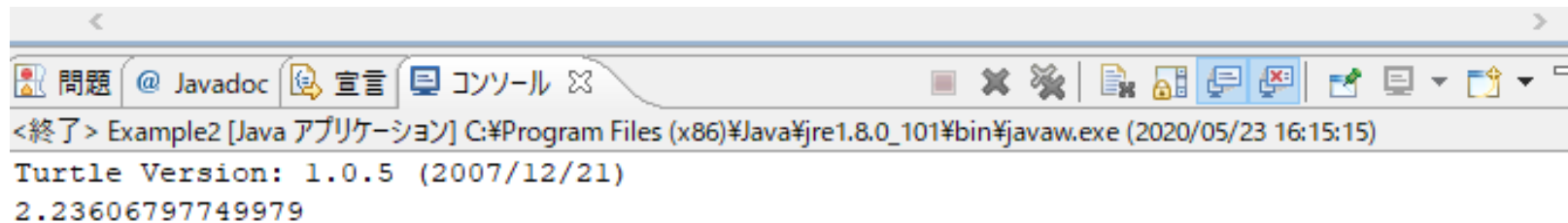


例題: 2

(Example2.java)

例題: 2

- Example2.javaを
変数「answer」に「ルート5の計算結果」を代入
するようにプログラムを書き換えなさい



The screenshot shows a Java IDE window with the title bar "Example2 [Java アプリケーション] C:\Program Files (x86)\Java\jre1.8.0_101\bin\javaw.exe (2020/05/23 16:15:15)". The console output displays the text "Turtle Version: 1.0.5 (2007/12/21)" followed by the number "2.23606797749979".

```
<終了> Example2 [Java アプリケーション] C:\Program Files (x86)\Java\jre1.8.0_101\bin\javaw.exe (2020/05/23 16:15:15)  
Turtle Version: 1.0.5 (2007/12/21)  
2.23606797749979
```

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

プログラム

```
// タートルを動かす処理
```

```
void start() {
```

```
    // 例題2: 以下の変数「answer」に「ルート5の値」を代入されるようにプログラムを書き換えなさい！
```

```
    // 変数「answer」にルート5の値を代入
```

```
    double answer;
```

```
    answer = Math.sqrt(5);
```

```
    // 値の出力
```

```
    print(answer);
```

```
    System.exit(0);
```

```
}
```

```
}
```

Ⅲ. 文章入出力

文章入出力

- TurtleJavaではコンソール上にメッセージ(文字列または変数の値)を**文章出力**(表示)させることができる
- またユーザー(人)に**文章入力**(インプット)させることで、まるでユーザと対話しているようなインタラクティブなプログラムを作れる

文章出力

文章出力の方法①

- ①「文字列」のみを表示させたい場合：

ダブルクォーテーションで「文字列」を囲む

```
print("ここに出したいメッセージを書く");
```

- 例) コンソール上に「私の名前はEclipse」と表示

- ```
print("私の名前はEclipse");
```

出力結果：  
私の名前はEclipse



## 文章出力の方法②

- ②「変数の値」のみを出力させたい場合：  
ダブルクォーテーションなしで「変数名」を書く

**print(変数名);**

- 例) コンソール上に「変数xの値」を表示

- `int x = 50 ;`

- **print(x);**

出力結果：  
50

(※ `print("x");`では『 x 』出力されてしまうので注意！)

## 文章出力の方法③

- ③「変数の値」と「文字列」を出力させる場合：

「変数」と「文字列」を一緒に出力する場合は+（プラス記号）を使ってつなげる

```
print("メッセージ1" + 変数名 + "メッセージ2") ;
```

- 例)「xの値は50です」と表示させたい場合

- `int x = 50 ;`

- `print("xの値は" + x + "です");`

出力結果：  
xの値は50です

(※`print("xの値はxです");`では『xの値はxです』と出力されてしまう!!)



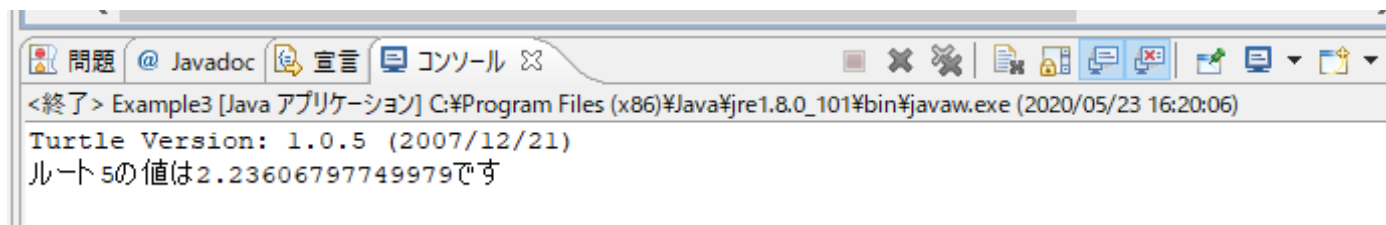
例題: 3  
(Example3.java)

# 例題: 3

- Example3.javaの

コンソール上の「ルート5の値は●●です」

●●の部分が正しいルートの値(2.236...)まで表示されるようにプログラムを書き換えなさい



```
<終了> Example3 [Java アプリケーション] C:\Program Files (x86)\Java\jre1.8.0_101\bin\javaw.exe (2020/05/23 16:20:06)
Turtle Version: 1.0.5 (2007/12/21)
ルート5の値は2.23606797749979です
```

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

```
* 例題3 (文字列と変数を組み合わせた文章出力)
*/
public class Example3 extends Turtle {

 // 起動処理
 public static void main(String[] args) {
 Turtle.startTurtle(new Example3());
 }

 // タートルを動かす処理
 void start() {

 // 変数「answer」にルート5の値を代入
 double answer;
 answer = Math.sqrt(5);

 // 例題3: 以下の文章出力を上記の変数「answer」を使って「ルート5の値は●●です」が計算される。
 print("ルート5の値は" + answer + "です");

 System.exit(0);
 }
}
```

プログラム

# 文章入力

# ユーザの入力した値を記憶する

- プログラムでコンソール画面から値を入力させるには、input命令を使う

**input();**

- 実際は「**input();**」単体で使うことはできないので、次のように**入力用の変数**を用意して入力を行う

# ユーザの入力した値を記憶する

- 例：1辺の長さを表す変数「length」を作って、その値をユーザに自由に入力させる

- `int length;` //入力用変数「length」の設定
- `length = input();`

「length」の中に  
1辺の長さをユーザが入力

もしくは

(2つをまとめて)

- `int length = input();`

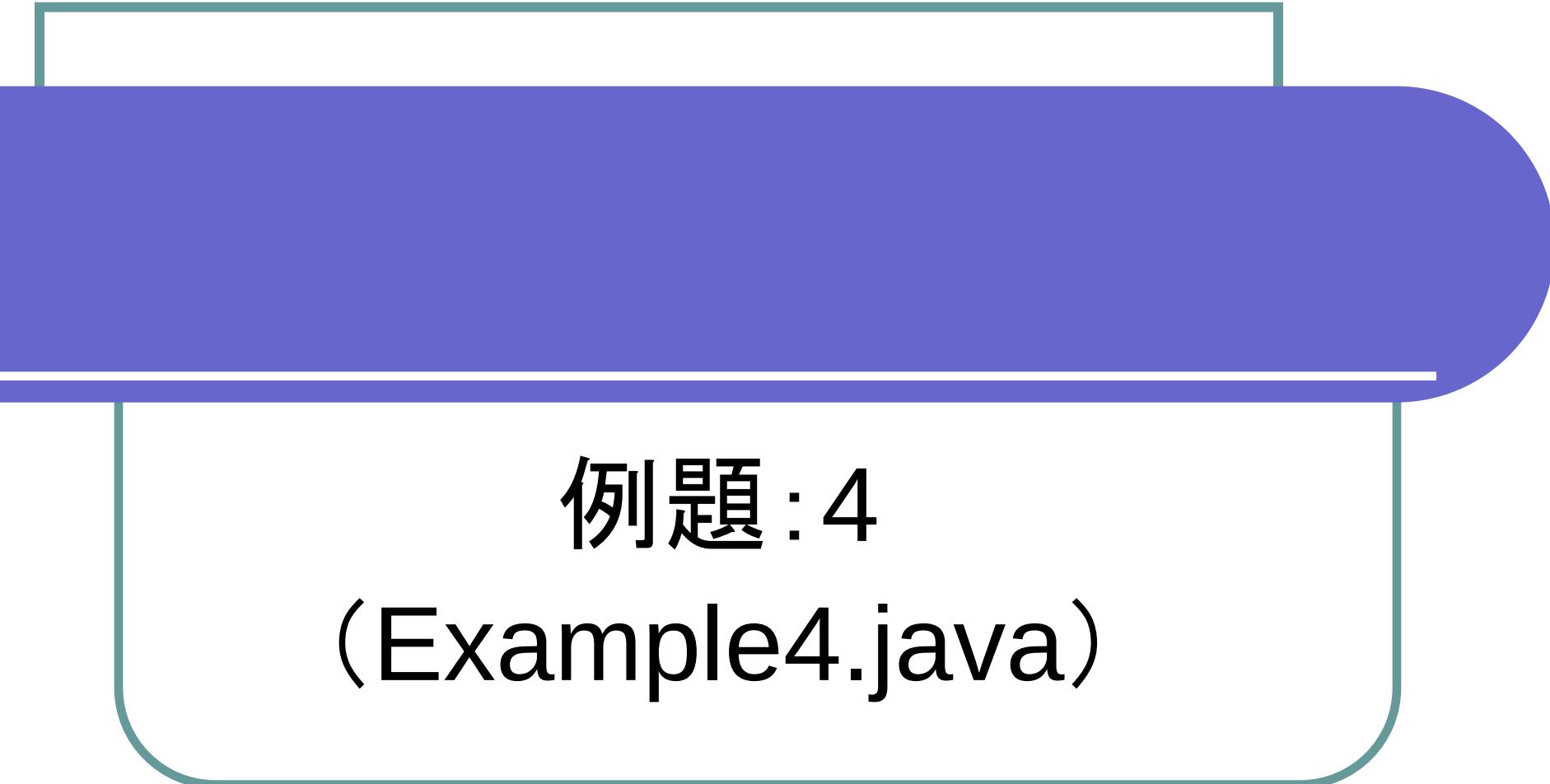
例) InputHouse.java  
を見てみよう！



# 例について

- ① InputHouse.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
  - 「文字入出力」を使ったプログラミング
  - ※コンソールに数字を入力してエンターキー





## 例題: 4

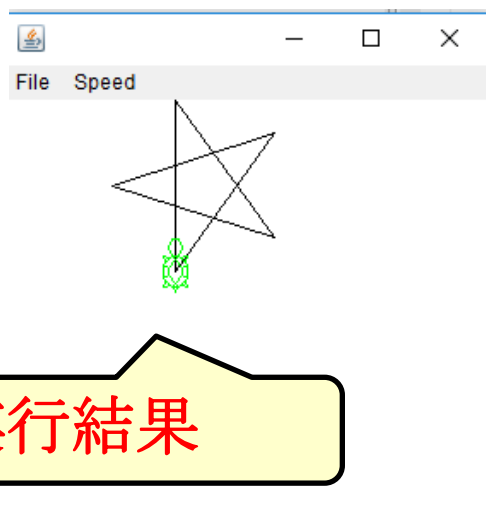
(Example4.java)

# 例題: 4

- Example4.javaを

変数「length」にユーザが値を入力して

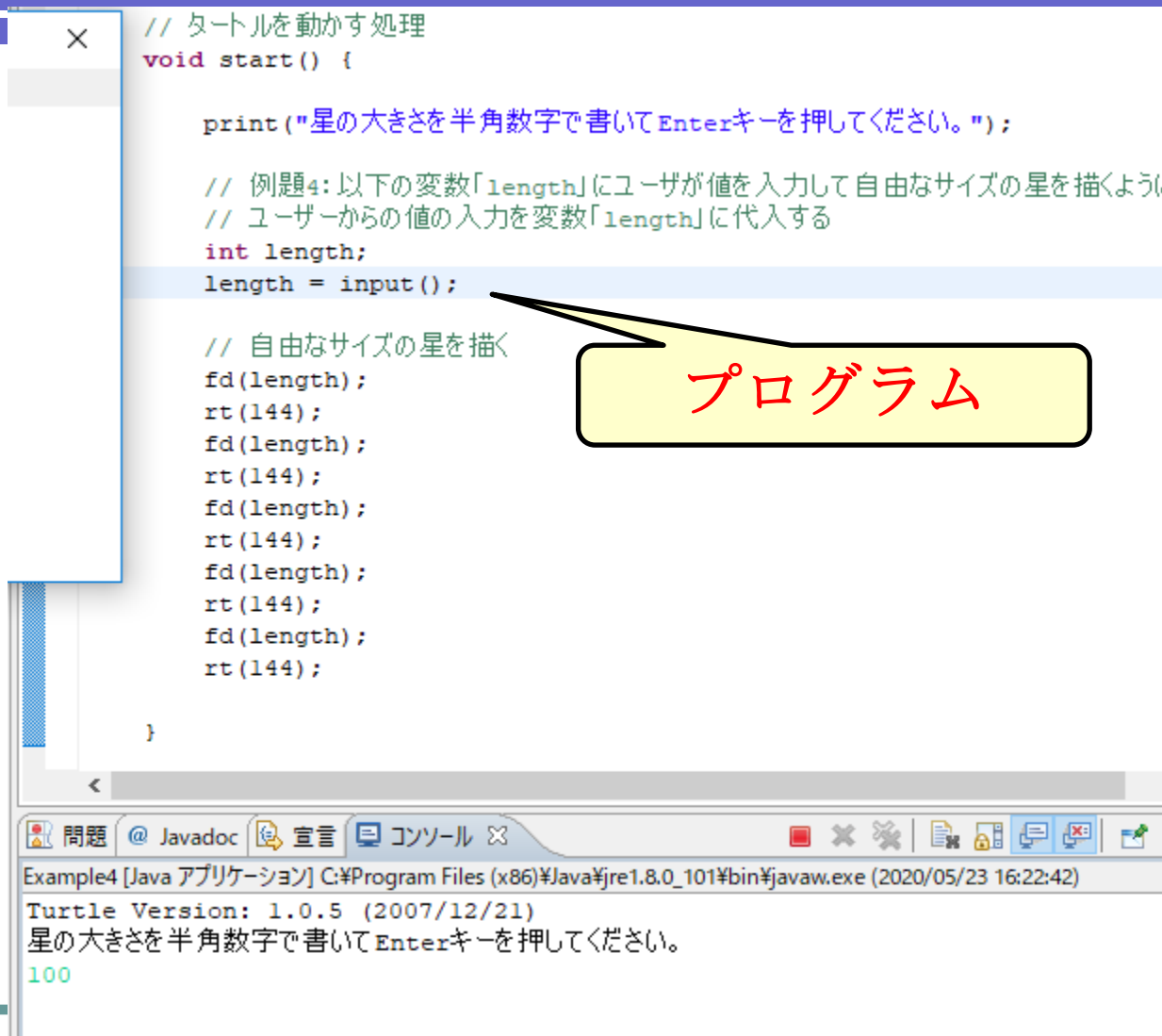
自由なサイズの星を描くようにプログラムを書き換えなさい



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。");

 // 例題4: 以下の変数「length」にユーザが値を入力して自由なサイズの星を描くよう
 // ユーザーからの値の入力を変数「length」に代入する
 int length;
 length = input();

 // 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);
 fd(length);
 rt(144);

}
```

プログラム

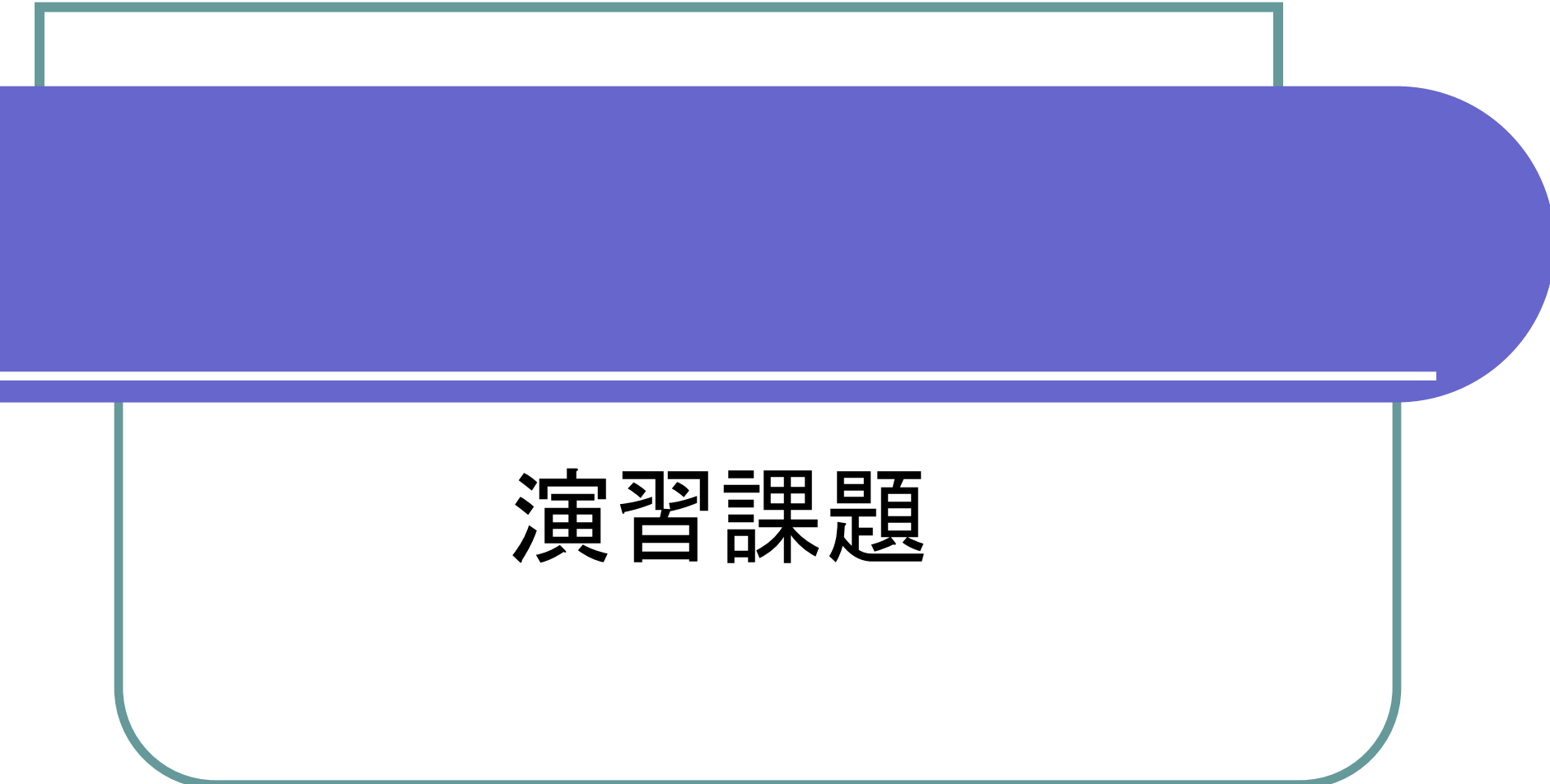
問題 @ Javadoc 宣言 コンソール

Example4 [Java アプリケーション] C:\Program Files (x86)\Java\jre1.8.0\_101\bin\javaw.exe (2020/05/23 16:22:42)

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。

100

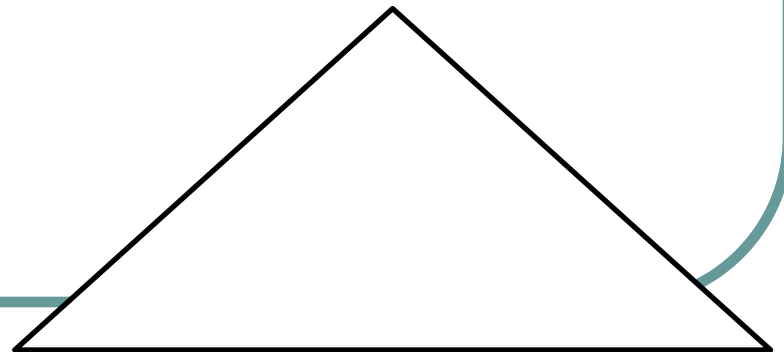


# 演習課題

# 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

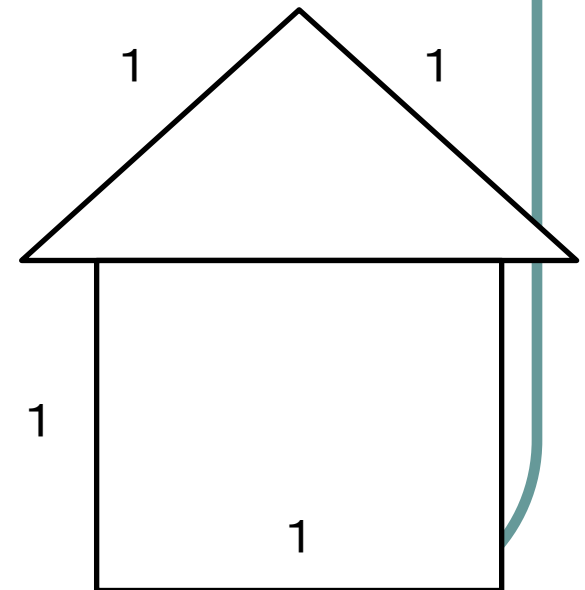
**up();** タートルがペンを上げる(軌跡を書かない)  
**down();** タートルがペンを下げる(軌跡を書く)

- 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





# 次回予告

# 次回予告

- Project.2 「コンピュータに情報を記憶させてみよう(変数)」
- Project.3 「条件分岐(if文,乱数)」

クリアファイルを  
提出して帰ること！

以上

NEXT  
TIME

おつかれさまでした。  
それでは、また次回！！

