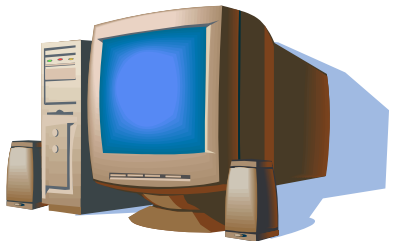
情報①

1学期　第5回

Project.5「アニメーション入門」

(複数オブジェクト)



今日の授業のながれ

- **Project.4**

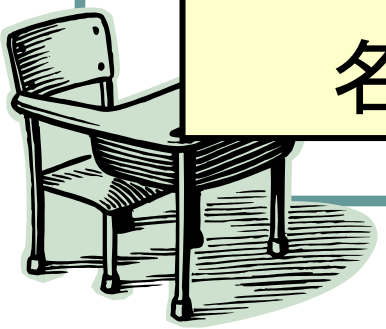
「コンピュータの得意な仕事(繰り返し)」

- **Project.5**

「アニメーション入門(複数オブジェクト)」

  - 配布資料
    - 1学期 第5回演習チェックシート　part.1-5

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。



指示をよく聞いて始める！

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

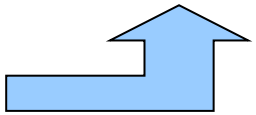
演習の順番

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

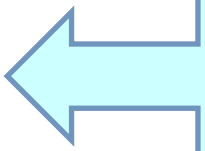
今日の目標

- [Project.4]
- 前回までの残り
- HousePattern.java
- Shell.java



この辺はクリアしたい

- [Project.5]
- ShootingStarEX.java
- RotateStar.java



進んでいる人はココまで挑戦！

前回までの復習

Project.3
「条件分岐」

条件式とは

- ○か×かを確実に判断できる式
  - 『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**
  - **「true」**と記述することでプログラムを止めるまでチクタクと動かし続けることもできる。

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[illegible]

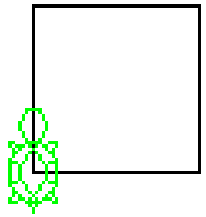
演算子	表記例	意味
&&	A&&B	AとB両方ともtrueの時true
||	A||B	AもしくはBがtrueの時true

Project.4
「繰り返し（ループ）」

繰り返し

- 与えられた条件を満たす限りプログラミングの操作が繰り返し行われる『繰り返し（ループ）』という

- 多くのプログラミング言語において使われており、プログラミングを簡潔にしたり複雑な機能を実装できる等さまざまな利点がある



While文の基本形1

- Circle.javaを見てみましょう。

```
// 円を書く
while (true) {
  fd(1);
  rt(1);
}
```

中括弧で囲まれた部分が一行ずつ
いつまでも繰り返される

While文の基本形2

- WhileHouse.javaを見てみましょう。

```
int i;// ループ用変数

// 屋根を書く
rt(30);
i = 1;
while (i <= 3) {
   fd(50);
   rt(120);
   i = i + 1;
}
```

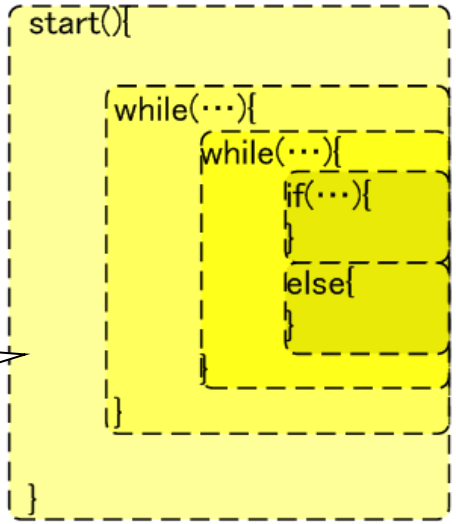
条件を満たす限り
whileに戻って
繰り返し

Project.4.5
「繰り返し（ループ）」

ブロックの入れ子構造

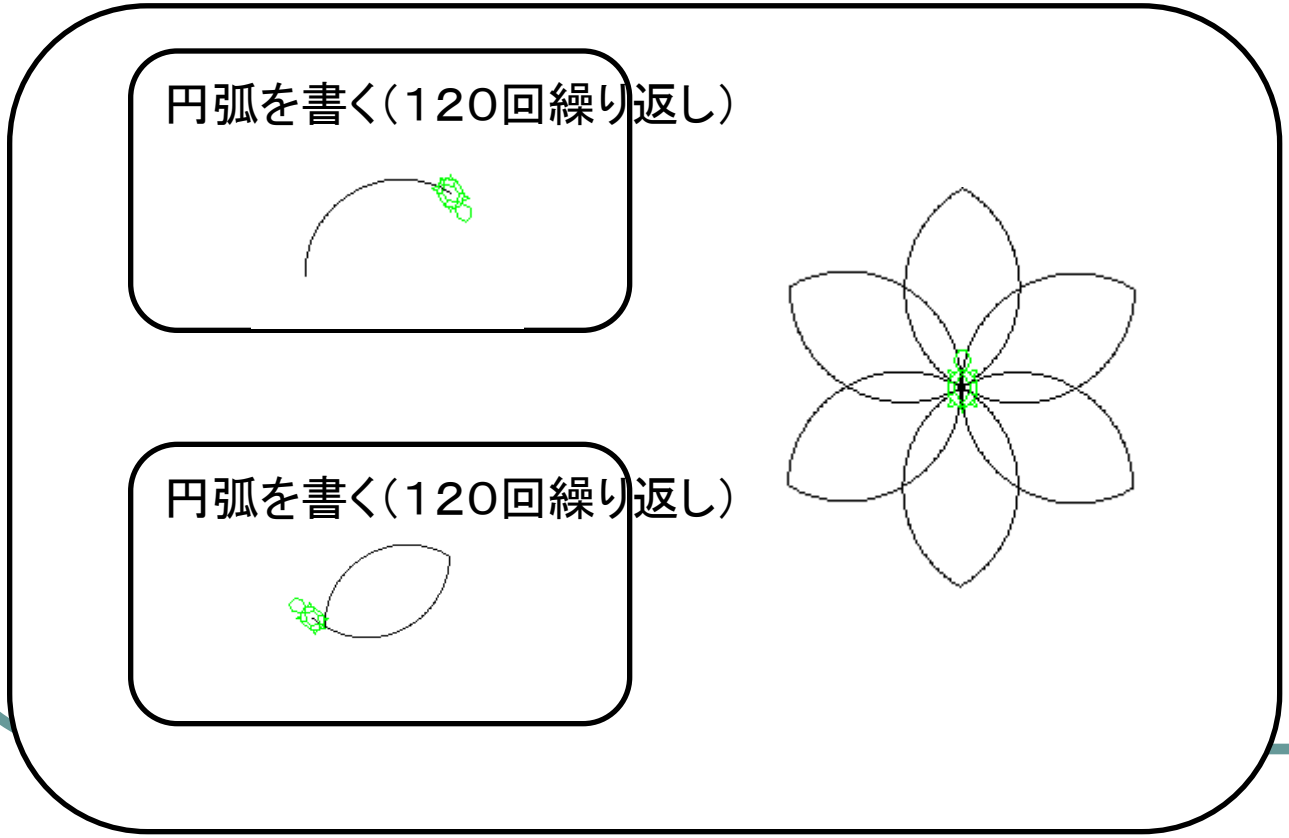
- 実は繰り返しブロックの内側にさらに繰り返しブロックを作ることができる
- このようにブロックの内側に新しいブロックを複数作ることを**「入れ子（ネスト）構造」**
- 繰り返し(while文)に限らず条件分岐(if文)等のブロックは入れ子にすることができる

繰り返しの中に繰り返しを作ったり、さらにその中に条件分岐を作ったりできる



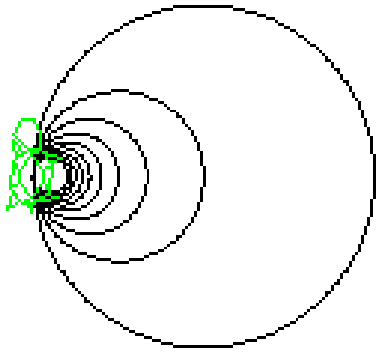
繰り返しを繰り返す

- 例3：Petal.javaを読んでからFlower.javaを見てみよう



入れ子（ネスト）とループ用変数

- 入れ子と先ほど習ったループ用変数を組み合わせるとより複雑な図形が描けます。
  - 例題2：MultiCircle.javaでは、なぜ円が小さくなるのか考えなさい？



【前回の補足】

ショートカットキー

＜エディタ編＞

①(Ctrl + A)･･･すべて選択

②(Ctrl + C)･･･コピー

③(Ctrl + V)･･･貼り付け

④(Ctrl + X)･･･切り取り

⑥(Ctrl + F)･･･検索/置換

⑦(Ctrl + S)･･･上書き保存+コンパイル

⑧(Ctrl + Z)･･･直前の動作の取り消し

⑨(Ctrl + Y)･･･直前の動作のやり直し

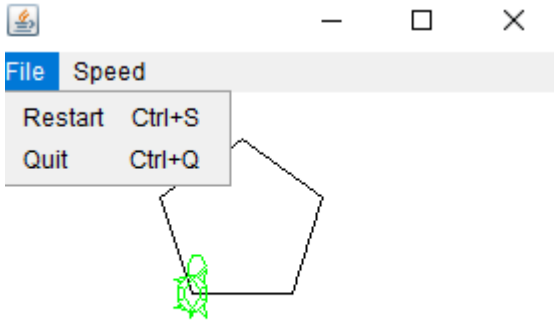
⑩(Ctrl + W)･･･タブを閉じる

⑪(Ctrl + F11)･･･プログラムの実行

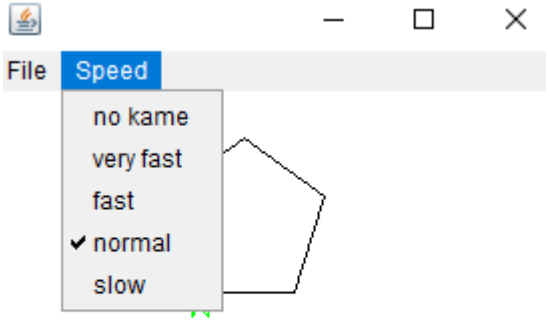
⑫(Ctrl + Shift + F)･･･自動的にフォーマット

ウィンドウ画面

＜ウィンドウ編＞
①(　Ctrl + S　)・・・プログラムのリスタート
②(　Ctrl + Q　)・・・画面を閉じる



＜亀の速度＞
[Speed]をクリックして
① no kame
② very fast
③ fast
④ normal
⑤ slow



演習課題

演習について

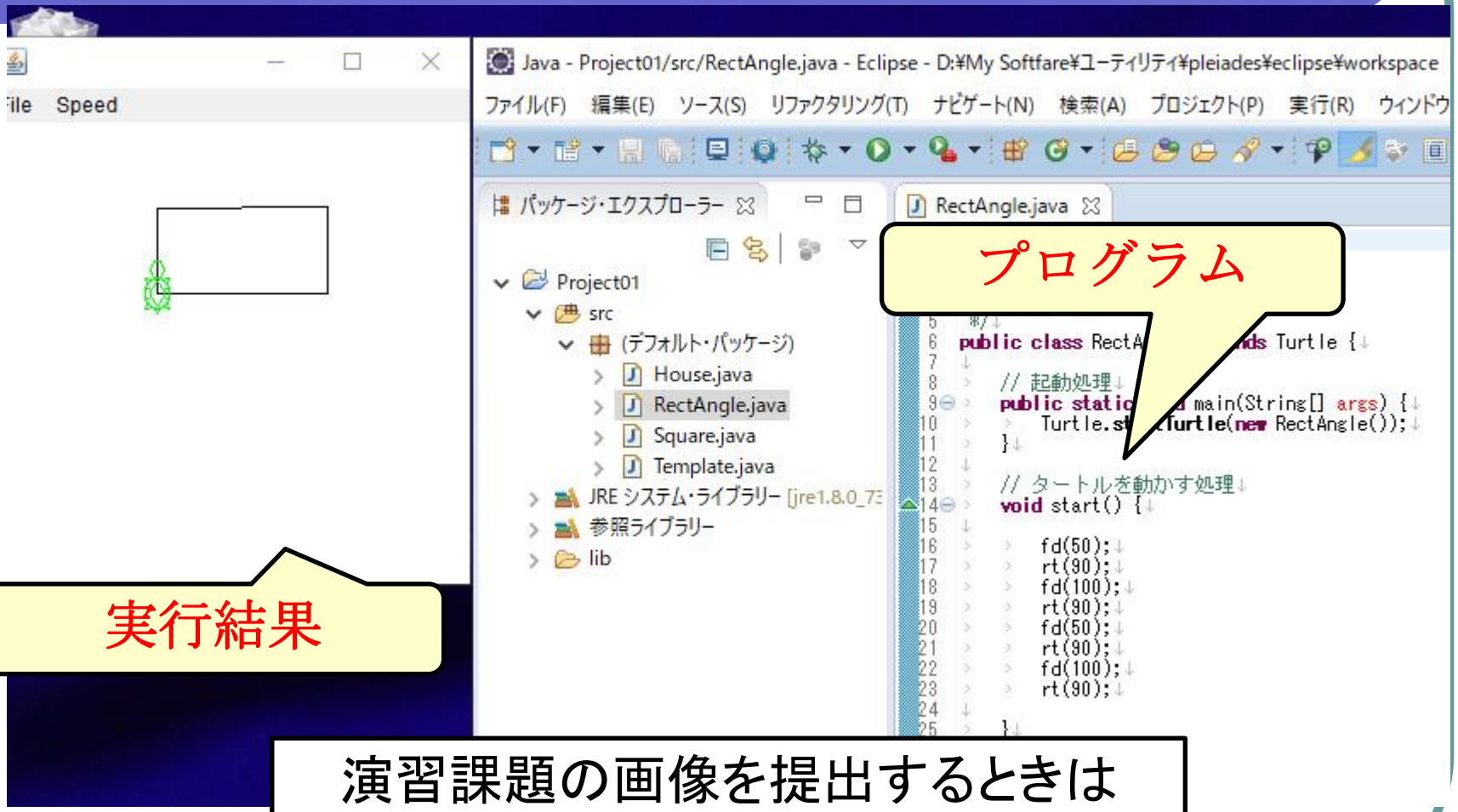
- **演習課題の提出画像について**
  - 「実行結果」と「プログラム」の両方を写す
  - ファイル名には「組番号」を付ける
- **課題用プログラムの作成方法**
  - Template.javaを右クリック→コピー
  - その場で右クリック→貼り付け
  - 名前の競合→(該当の演習課題の)名前を付ける

授業用サイトにコピー方法が載せました

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

解答例



演習課題の画像を提出するときは

「実行結果」と「プログラム」

両方が写るようにすること！

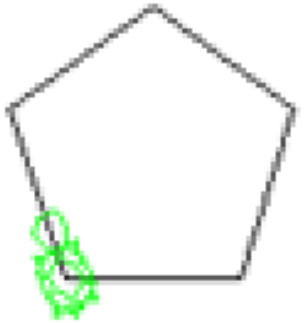
【基本問題】

演習課題

- PentagonEX.java
  - Pentagon.javaを改良し、繰り返し(while文)を使って正五角形を書くプログラムにしなさい

ポイント

- 一周したらプログラムを止めるよう改良

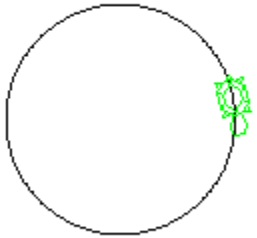


演習課題

- CircleEX.java
  - Circle.java（円を描くプログラム）を1周したら亀が止まるように改良しなさい

ポイント

- 一周したらプログラムを止めるよう改良

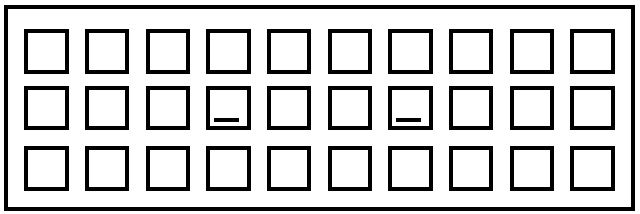


演習課題

- Keyboard.java
  - キーボードを描くプログラムを作りなさい

ポイント

- 横10列　縦3列のキー配列
  - キーの大きさ「lengthは10」キー間隔「marginは5」
- キーボードの外枠と印はプログラム済（省略）



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！

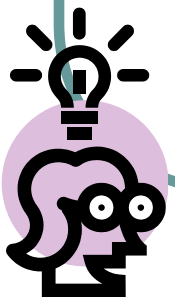
- どうしても分からない人は・・・

  次のページ【ヒント】を参照

ヒント！！！

ヒント

- PentagonEX.javaについて
  - 例題1を参考にする
- CircleEX.javaについて
  - 例題1を参考にする
  - 1周回るということは何回繰り返すだろう・・・？
- Keyboard.javaについて
  - HundredBlock.javaを参考にする



【発展問題】

おまけ～割り算のあまり～

- 剰余演算子(%)を使うと、割り算のあまりを求めることができる
  - 例　9÷2の余りを求める
    - **int amari = 9 % 2 ;//amariは1**

- これを応用して、奇数と偶数も判定ができる

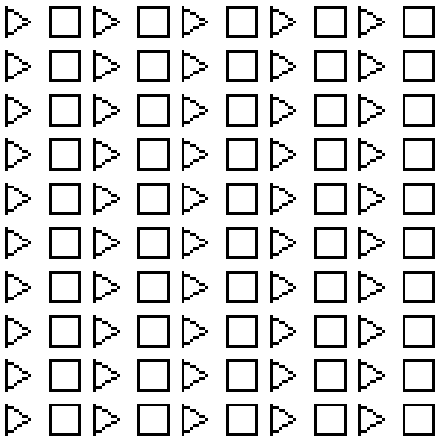
例) SquareAndTriangle.java
を見てみよう！

演習課題

- HousePattern.java
  - 四角形と三角形を交互に100個描くプログラムを作りなさい

ポイント

- 入れ子を使ったプログラミング
- 5セット×10回

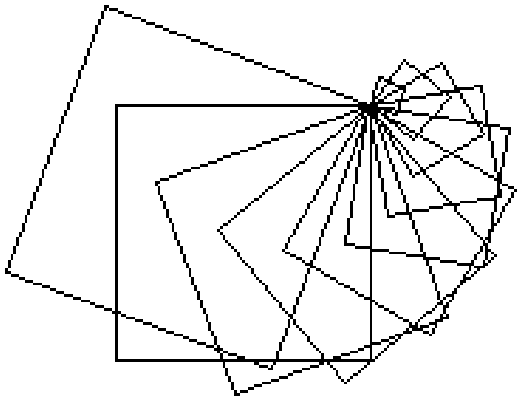


演習課題

- Shell.java
  - 貝殻を描くプログラムを作りなさい

ポイント

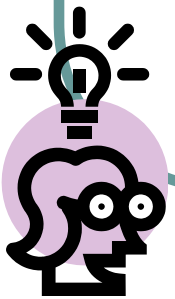
- 入れ子を使ったプログラミング
- 「ループ用変数」を変化



ヒント！！！

ヒント

- HousePattern.javaについて
  - HundredBlock.javaを参考にする
  - 偶数奇数の判定はSquareAndTriangle.java
- Shell.javaについて
  - 例題4を参考にする
  - 同じ大きさの正方形を繰り返しで描くことからはじめて、徐々に小さくするには・・・？



Project.5
「アニメーション入門」

学習事項

学習事項

- 複数の(タートル)オブジェクト
  - オブジェクトとは
  - タートルオブジェクトと図形の描画
- アニメーション入門
  - アニメーションの基本構造
  - 図形オブジェクトとアニメーション
  - 座標系について
  - 画像オブジェクトとアニメーション

Ⅰ．複数の(タートル)オブジェクト

アニメーションについて

- 情報2の授業では学期末ごとにゲーム制作をすることが一つの目標
- 今までは自動生成される1匹のタートル(**オブジェクト**)に色々な図形を描かせてきた
- 一方、ゲーム制作ではキャラクターや部品などの**複数のオブジェクト**（物体）を**アニメーション**で動かす必要がある

アニメーションについて

- 作品に登場する複数のオブジェクト（物体）
  - 主人公（プレイヤー）
  - 敵
  - アイテム
  - 点数表示，ボタン　など
- 今回はまず複数のオブジェクトを扱う方法、そしてアニメーションについて勉強する

①オブジェクトとは？

オブジェクトとは

- 「**オブジェクト**」とはプログラムで扱うことのできるパーツ（部品）のようなもの
  - 「**①オブジェクトの生成**」
    - オブジェクトの種類を指定し、名前を付けて、作成
  - 「**②オブジェクトに命令**」
    - 作成した「**オブジェクト**」を指定し、プログラムを実行
- このようなプログラミングの手法を「**オブジェクト指向プログラミング**」という

オブジェクトの種類

オブジェクトとは

【(タートルJavaの)オブジェクトの種類】

	タートル	図形	画像	テキスト
オブジェクトの種類	**Turtle**	**???**	**ImageTurtle**	**TextTurtle**

- 「タートルオブジェクト」については例題1
- 「図形オブジェクト」については例題2
- 「画像オブジェクト」については例題3
- 「テキストタートル」については次回

オブジェクトの生成と命令

複数の場合の命令の仕方

- 命令の書式

  ①オブジェクトの生成

  オブジェクトの種類　名前　= new オブジェクトの種類();

  ②命令を出す

  名前.命令;

【注意】:ピリオドを忘れないように

【注意】名前について

【注意】オブジェクトの名前

- オブジェクトの名前はある程度自由につけてよいが、後で見てわかりやすく重複のないようにする

- 「int、if、whileなど」Javaであらかじめ指定されている単語（**予約語**）は避けること

②タートルオブジェクトと図形の描画

タートルオブジェクト

- これまで図形を描いていたタートルに名前を付け、複数のタートルオブジェクトとして使うことが可能

- ①オブジェクトの生成
  - オブジェクトの種類にTurtleを記述して生成

Turtle 名前を付ける = new Turtle();

タートルオブジェクトの生成

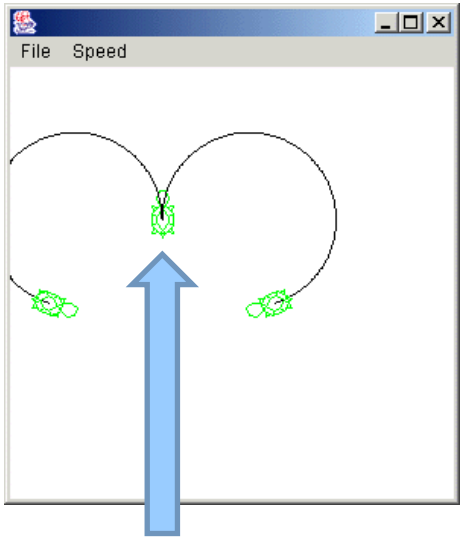
命令の書式

- ②命令を出す

.（ピリオド）を忘れないように

kameTaro.fd(1);

タートルの名前　命令



※デフォルトの亀は名前を付けなくても存在するので、hide();で隠すと良い

複数のタートルオブジェクト

- 複数の亀を扱いたい場合

①オブジェクトの生成

```
Turtle kameTaro = new Turtle();
   //Turtleという種類のkameTaroというオブジェクトを生成
Turtle kameJiro = new Turtle();
   //Turtleという種類のkameJiroというオブジェクトを生成
```

②命令を出す

```
kameTaro.fd(100);       //kameTaroが100前に進む
kameJiro.fd(50);        //kameJiroが50前に進む
```

例) TwoTurtles.java
を見てみよう！

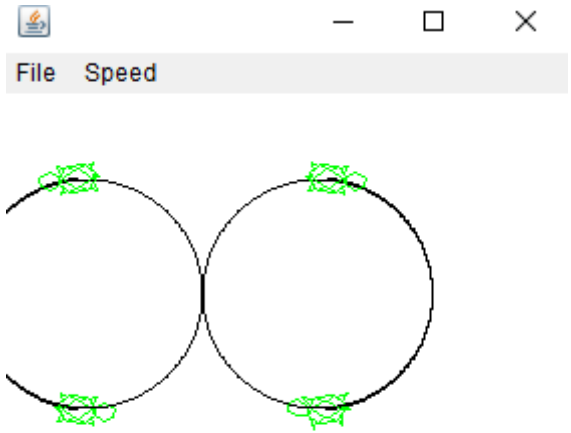
例について

① TwoTurtles.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
- 「複数のオブジェクト」を使ったプログラミング
- タートルオブジェクト(Turtle)

例題1：

例題 : 1

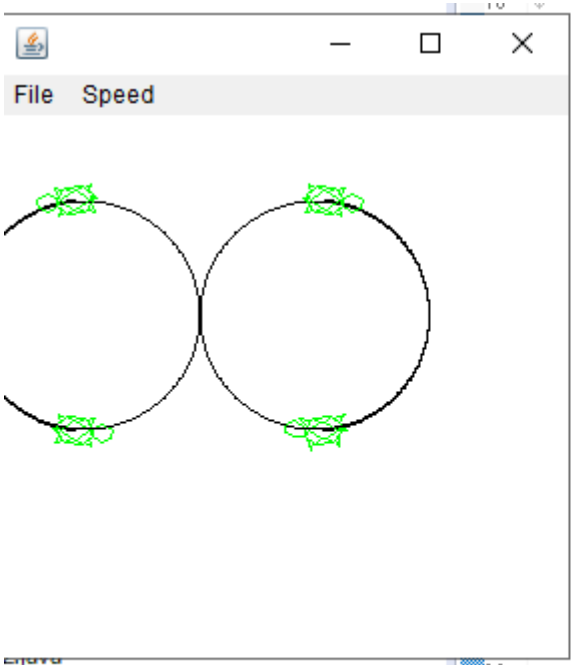
- TwoTurtles.javaの亀の数を4匹に増やし、それぞれ逆方向に動かしてみよう
  - 亀三郎は右逆回り
  - 亀四郎は左逆回り



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



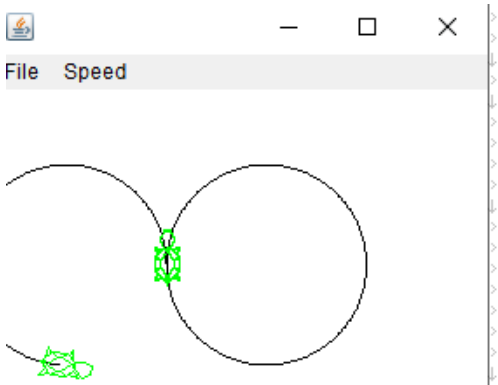

```
// タートルを動かす処理
public void start() {

    hide();// 元々の亀を消す

    Turtle kameTaro = new Turtle(); // 亀太郎を生成
    Turtle kameJiro = new Turtle(); // 亀次郎を生成
    Turtle kameSabro = new Turtle(); // 亀三郎を生成
    Turtle kameShiro = new Turtle(); // 亀四郎を生成

    int i = 0;
    while (i < 360) {

        // 亀太郎は右回り
        kameTaro.rt(1);
        kameTaro.fd(1);

        // 亀次郎は左回り
        kameJiro.lt(1);
        kameJiro.fd(1);

        // 亀三郎は右逆回り
        kameSabro.rt(1);
        kameSabro.bk(1);

        // 亀四郎は左逆回り
        kameShiro.lt(1);
        kameShiro.bk(1);

        i++;
    }
```

– [jre1.8.0_73]

【注意】
タートルの動きの違い

解答例？

- 解答例と次の解答例？を実行して比べてみよう




```
// タートルを動かす処理
public void start() {

    hide();// 元々の亀を消す

    Turtle kameTaro = new Turtle(); // 亀太郎を生成
    Turtle kameJiro = new Turtle(); // 亀次郎を生成
    Turtle kameSabro = new Turtle(); // 亀三郎を生成
    Turtle kameShiro = new Turtle(); // 亀四郎を生成

    int i = 0;
    while (i < 360) {
        // 亀太郎は右回り
        kameTaro.rt(1);
        kameTaro.fd(1);
        i++;
    }

    i = 0;
    while (i < 360) {
        // 亀次郎は左回り
        kameJiro.lt(1);
        kameJiro.fd(1);
        i++;
    }

    i = 0;
    while (i < 360) {
        // 亀三郎は右逆回り
        kameSabro.rt(1);
        kameSabro.bk(1);
        i++;
    }

    i = 0;
    while (i < 360) {
        // 亀四郎は左逆回り
        kameShiro.lt(1);
        kameShiro.bk(1);
        i++;
    }

}
```

プログラム

タートルを同時に動かすために

- 1つ目の解答例では、タートルを細かく切り替えて少しずつ動かすことによって、4匹のタートルが同時に動いているように見せることができます
- 2つ目の解答例？では、1匹のタートルが動作が終わるまで次のタートルが動作できないため不自然な動きになっています
- これはコンピュータは同時に2度の命令をできないためです

Ⅱ．アニメーション入門

アニメーションについて

- 続いて**複数のオブジェクト**（物体）を**アニメーション**で動かす勉強をしていこう。

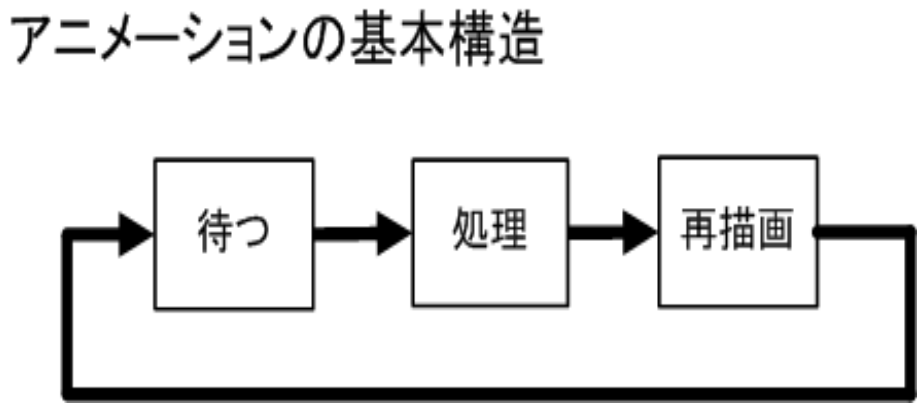
- アニメーションでは
  - 図形
  - 画像
  - 文字(次回)など

  様々なものを動かすことができる。

①アニメーションの基本構造

アニメーションの基本構造

- どんなアニメーションをするプログラムでも基本構造は同じ




```
while (true) {

        // 待つ
        sleep(0.1);

        // 処理を行う



        // 再描画
        update();
}
```

ココに1コマで行う
具体的な処理を書く

新しい命令ついて

アニメーションのための新しい命令

- **sleep([秒の指定]) ;**

指定された秒数だけ止まる

- **update() ;**

画面を書き換える

（※ 画面を書き換えないと、fd(5)などで移動しても、画面が動かないので注意）

【注意】

【注意】秒の指定について

- sleep( )命令の「秒の指定」では1を入れれば1秒止まり、0.1を入れれば0.1秒止まる

- この数字を小さくすることによって、理論的にアニメーションを早くできるが、せいぜい0.04秒～（秒間25コマ～）

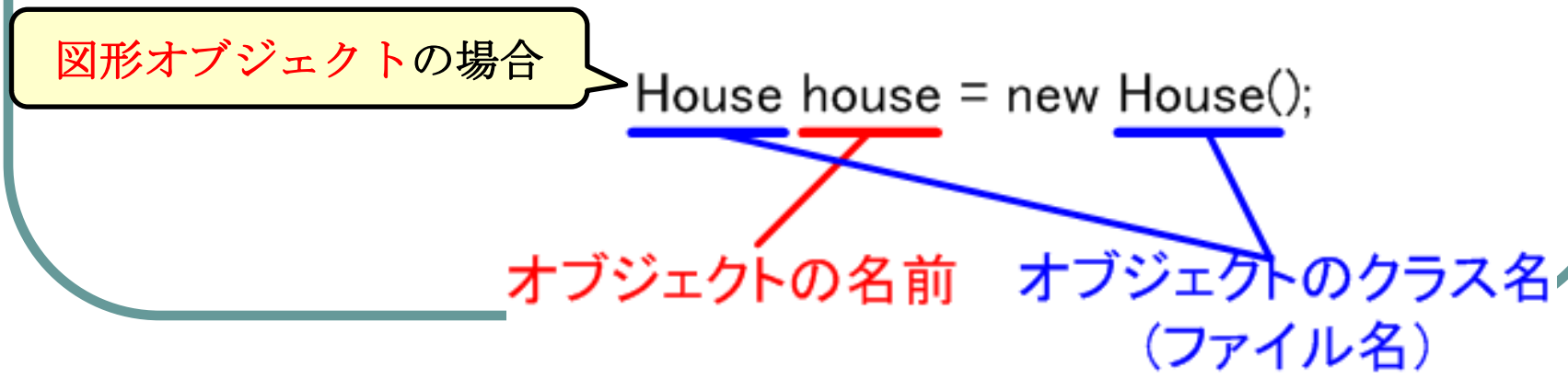
②図形オブジェクトとアニメーション

図形オブジェクト

- タートルで作った図形（家など）に名前を付け、図形オブジェクトとして使うことが可能

- ①オブジェクトの生成（⇒②命令を出すは省略）
  - オブジェクトの種類に図形のファイル名(.javaを除く)記述
    - (例)House.javaを使いたい場合

House名前を付ける = new House();



図形オブジェクト

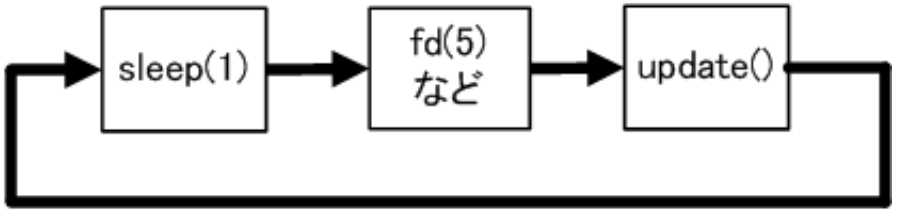
- 使いたい図形のファイル(House.javaなど)は、プログラミング中のファイルと**同じフォルダ**に置く

練習問題を解く際にはこちらで準備しているため
特に気にしないで大丈夫です

※ 図形オブジェクトなどでペンの軌跡を描く場合、タートルオブジェクトと違い、初めにdown( )命令が必要となる

例) MoveRightHouse1.java
MoveRightHouse2.java
を見てみよう！

アニメーションの基本構造(具体的な処理)



while(true)による
無限の繰り返し

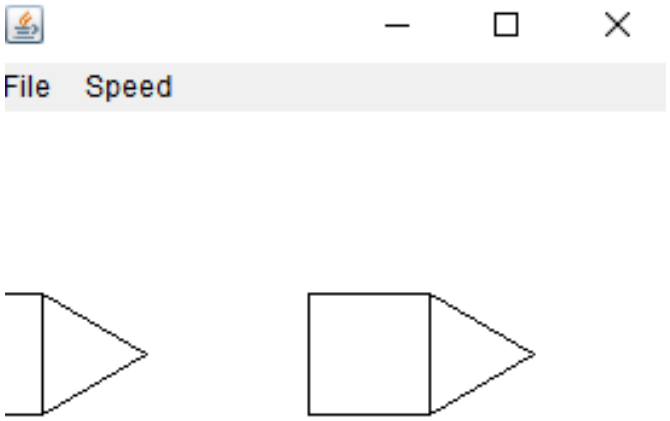
例について

① MoveRightHouse1.javaを実行してみる
② MoveRightHouse2.javaを実行してみる
③ プログラムをよく読んでみる
④ プログラムの意味を理解する
- 「アニメーションの基本構造」
- 図形オブジェクト(House)
- 家の向きが異なる

例題2:

例題：2

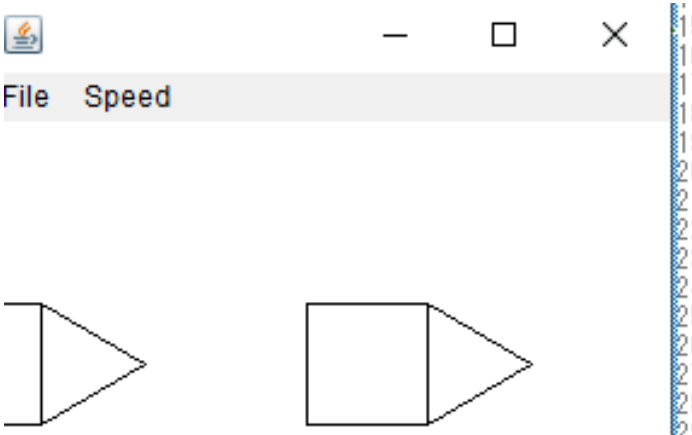
- MoveRightHouse1.javaの家を2軒に増やし，逆方向に動かしなさい。



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

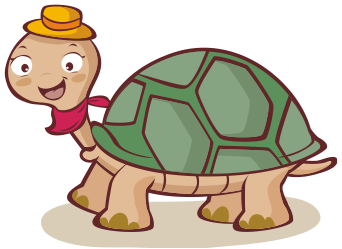



```
15    public void start() {
16
17        hide();//元の亀を消す
18
19        House house = new House(); // 家を生成
20        House house2 = new House(); // 家2を生成
21
22        house.rt(90); // 右に向ける
23        house2.rt(90); // 右に向ける
24
25        // アニメーションループ
26        while (true) {
27
28            // 待つ
29            sleep(0.1); // 0.1秒
30
31            // 処理を行う
32            house.fd(5);
33            house2.bk(5);
34
35            // 再描画する
36            update();
37        }
38
39    }
```

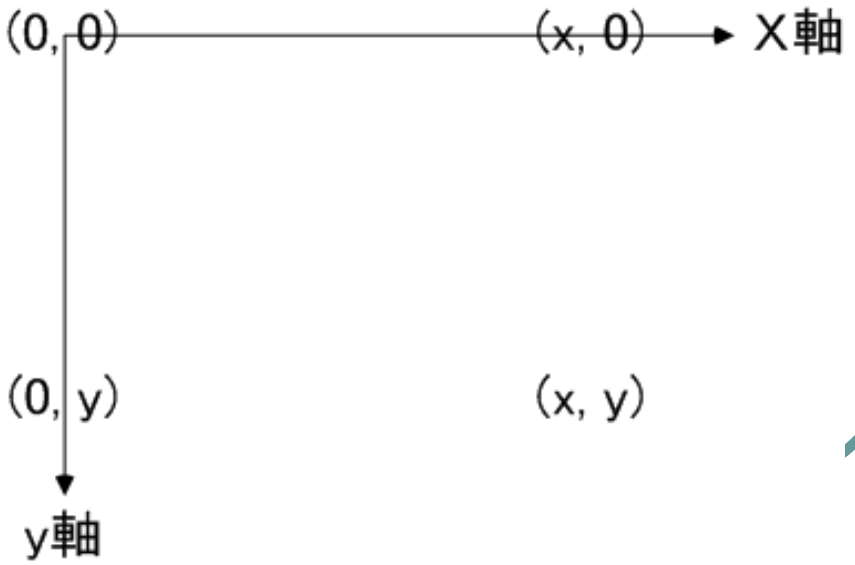
③座標系について

座標系

- タートルを含め一般にコンピュータ環境では数学とは違う以下の座標系が使われる
- ちなみにオブジェクトの初期位置は(100,100)



タートルの座標系



新しい命令ついて

座標に関する命令

- **warp([x座標, y座標])**：

指定されたx座標, y座標の位置にそのオブジェクトを**ワープ（瞬間的に移動）**

- **getX()**：現在のx座標を取得
- **getY()**：現在のy座標を取得

例) MoveRightHouse3.java
MoveRightHouse4.java
を見てみよう！

例について

① MoveRightHouse3.javaを実行してみる
② MoveRightHouse4.javaを実行してみる
③ プログラムをよく読んでみる
④ プログラムの意味を理解する
- 「ワープ」を使ったプログラミング
- 「座標系」を使ったプログラミング
- 図形オブジェクト(House)
- 画面外から戻ってくるかどうかの違い

④画像オブジェクトとアニメーション

画像オブジェクト

- 好きな画像に名前を付け、画像オブジェクトとして使うことが可能

- ①オブジェクトの生成（⇒②命令を出すは省略）
  - **オブジェクトの種類**にImageTurtleを記述
  - **右辺の( )に”画像のファイル名.拡張子”**を記述
    - (例)star.pngを使いたい場合

ImageTurtle 名前を付ける =

```
new ImageTurtle("star.png");
```

画像オブジェクトの場合

画像オブジェクト

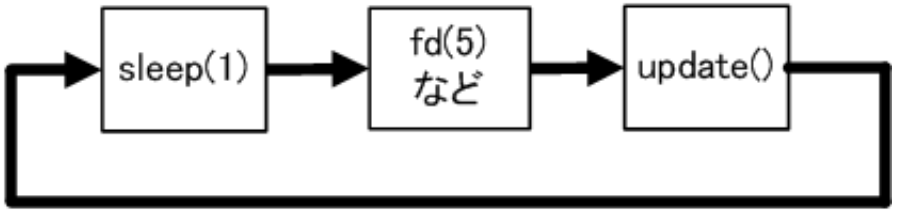
- 使いたい画像のファイル(star.pngなど)はプログラミング中のファイルの一つ上のフォルダに入れる

練習問題を解く際にはこちらで準備しているため
特に気にしないで大丈夫です

※ 使える画像の種類（拡張子）は「.jpg」「.jpeg」「.png」「.gif（アニメNG）」のどれか。

例) RotateStar.java
を見てみよう！

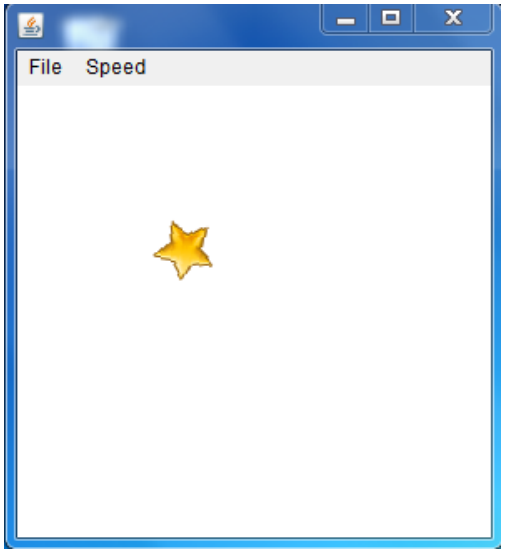
アニメーションの基本構造(具体的な処理)



while(true)による
無限の繰り返し

例について

① RotateStar.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
- 「画像」を使ったプログラミング
- 画像オブジェクト(House)



画像のファイル名を
car.gifに変更してみよう

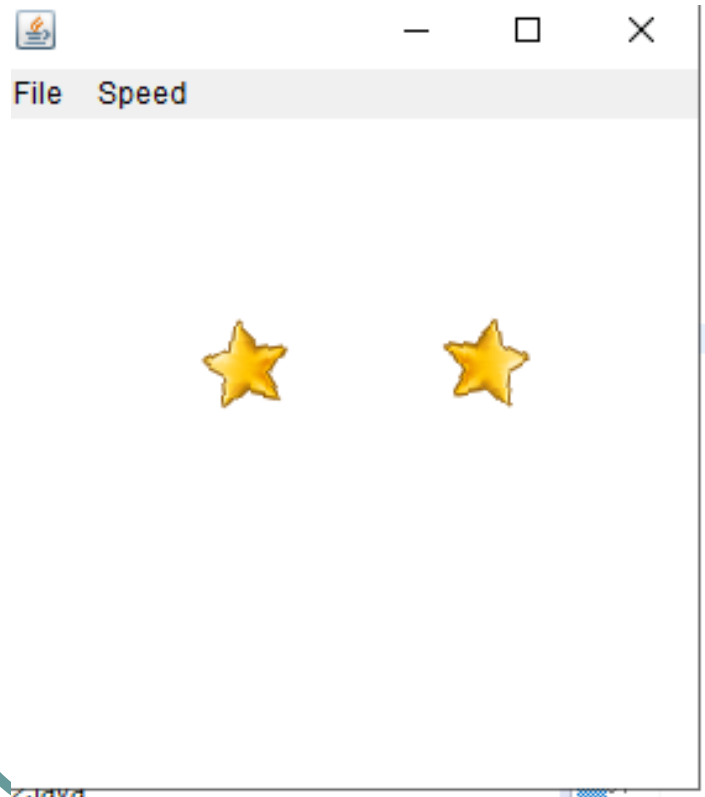
例題3:

例題：3

- RotateStar.javaの星を2個に増やし，逆回転に動かしなさい

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
public void start() {

    // カメを隠す
    hide();

    // 星を生成する
    ImageTurtle star = new ImageTurtle("star.png");
    // 星2を生成する
    ImageTurtle star2 = new ImageTurtle("star.png");

    //星2の初期位置を変更
    star2.warp(200, 100);

    // アニメーションループ
    while (true) {

        // 待つ
        sleep(0.1);

        // 処理
        star.rt(5);
        star2.lt(5);

        // 再描画
        update();
    }
}
```

【おまけ】
これまでの命令まとめ

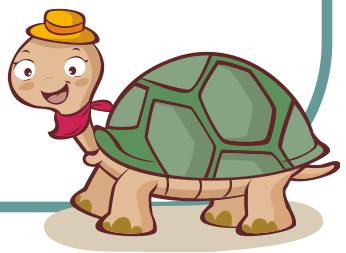
タートル(オブジェクト)への命令

- **基本命令**
  - rt(〇〇)　タートル（オブジェクト）を〇〇度右に回らせる
  - lt (〇〇)　タートル（オブジェクト）を〇〇度左に回らせる
  - fd(〇〇)　タートル（オブジェクト）を〇〇歩前に進ませる
  - bk(〇〇) タートル（オブジェクト）を〇〇歩後に進ませる

  - up()　　ペンを上げる(軌跡を書かない)
  - down()　ペンを下げる(軌跡を書く)

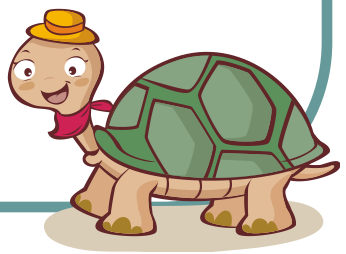
  - color(java.awt.Color.**色の指定**) ペンの色を変える

【参考】black（黒） blue（青） green（緑） red（赤）
white（白） yellow（黄色） cyan（水色） gray（灰色）
pink（ピンク） orange（オレンジ）magenta（赤紫）
darkGray（濃い灰色）lightGray（明るい灰色）



その他の基本命令

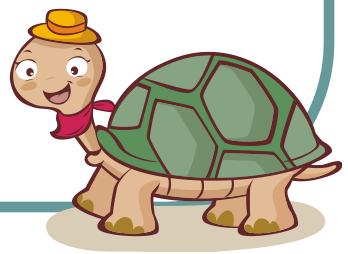
- **その他の基本命令**
  - input()
    - ユーザからの入力を受け付ける
  - print(“出力する文字列”や変数)
    - 文字列等をコンソールに出力する
  - random(数)
    - 指定した数の乱数を発生（「0～指定した数-1」まで）
  - System.exit(0)
    - プログラムの最後に書いてウィンドウ画面を閉じる
  - show() タートル（オブジェクト）を出現させる
  - hide()　タートル（オブジェクト）を隠す



オブジェクトの生成に関する命令

- **オブジェクトの生成に関する命令**
  - Turtle [名前] = new Turtle()
    - タートル(オブジェクト)を生成し、名付ける
  - [図形のファイル名(クラス)] [名前] = new [図形のファイル名]()
    - 図形オブジェクトを生成し、名付ける
  - ImageTurtle [名前] = new ImageTurtle("[画像のファイル名]")
    - 指定の画像を初期値とする画像オブジェクトを生成し、名付ける
  - TextTurtle [名前] = new ImageTurtle("[文字列]")
    - 指定の文字列を初期値とする文字オブジェクトを生成し、名付ける

次回



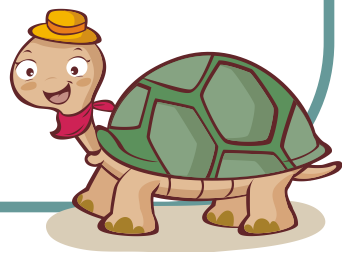
アニメーションのための命令

- アニメーションのための命令
  - **sleep([秒の指定]) ;**
    指定された秒数だけ止まる
  - **update() ;**
    画面を書き換える

（※ 画面を書き換えないと、アニメーションが動かない！）

  - **break;**
    アニメーションループから脱出する

次回



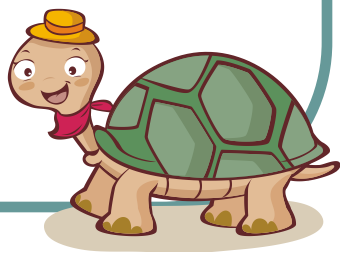
座標に関する命令

- **座標に関する命令**
  - **warp([x座標, y座標])**

指定された(x, y)座標にオブジェクトを**ワープ（瞬間移動）**

【参考】ウィンドウ画面の表示位置を変更する
window.warp(数値, 数値);

- **getX()** (オブジェクトの)現在のx座標を取得
- **getY()** (オブジェクトの)現在のy座標を取得



演習課題

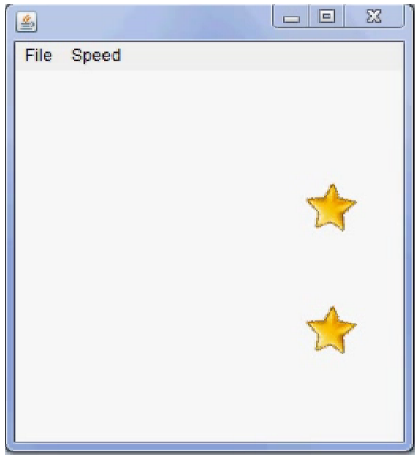
演習課題

- ShootingStarEX.java
  - 流れ星を表示するプログラムを作りなさい

ポイント

①星を回してみる
②座標を使って星を右に動かす
③一番右端にいったら、左端に戻るようにする
④星を下方向にも動かす
⑤星を増やす（2個でよい）

完成イメージは
Webページに上がっています



演習課題

- FlashStar.java（発展問題）
  - 星を点滅するプログラムを作りなさい

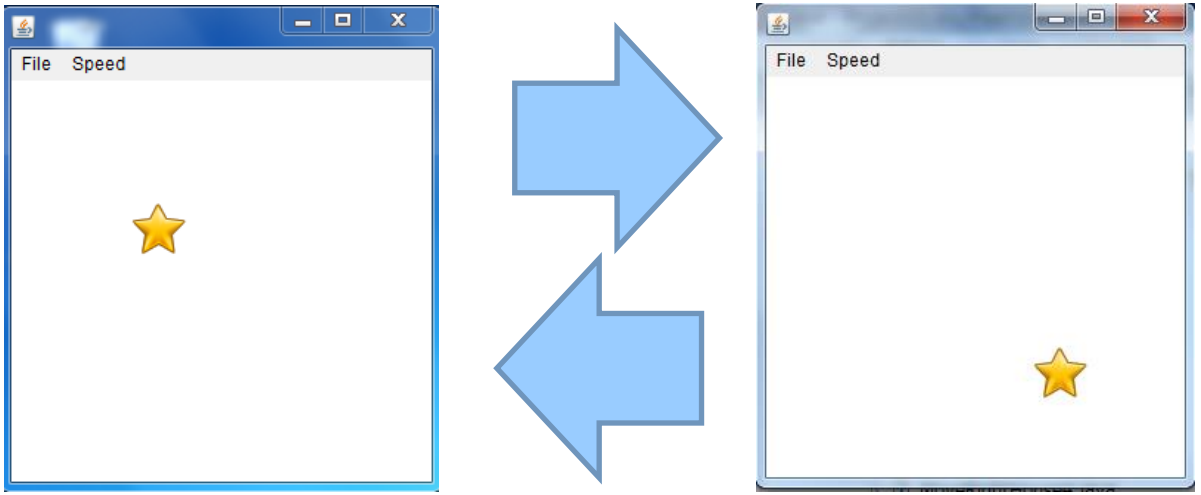
ポイント

①星を表示する

②座標を上手く使って星を点滅させる

③星を増やす（2個でよい）

完成イメージは
Webページに上がっています



次回予告

次回予告

- Project.5「アニメーション入門(複数オブジェクト)」
- Project.6「作品制作プロジェクト①」

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！！