計算機プログラミングI

第3回　2002年10月17日(木)

- オブジェクト値
- クラス変数・クラスメソッド
- 制御構造
  - 条件分岐
  - 繰返し
- クイズ

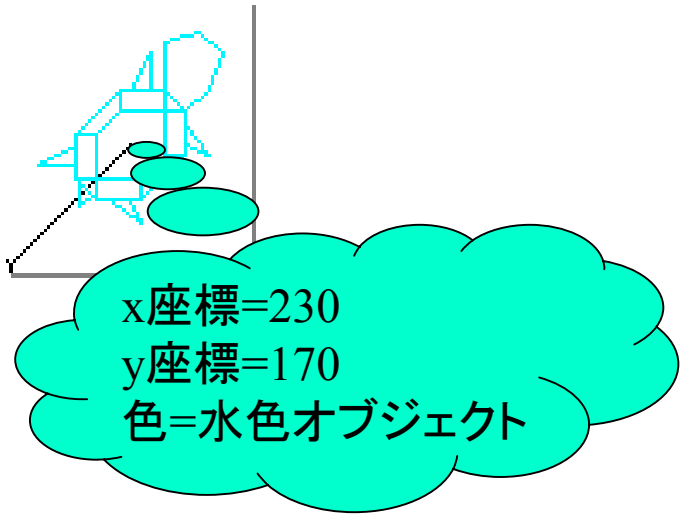
復習

オブジェクトとは? (2.8)

- オブジェクト:
  「もの」を表わす値

(クラスに所属)

  - 内部に状態を持つ (インスタンス変数)
  - メソッドが呼出されると
    - 状態を変化させる
    - 他のオブジェクトのメソッドを呼出す
    - 値を返す
- クラス
  - どんなインスタンス変数を持つか?
  - どんなメソッドを持つか?



オブジェクト値とプリミティブ値 (3.1-2)

2種類の値

- プリミティブ値 (原始的/原子的な値)
  - 整数 (8, 16, 32, 64ビット)
  - 小数 (浮動小数点方式; 32, 64ビット)
  - 真偽値 (1ビット)
  - 文字 (Unicode方式; 16ビット)
- オブジェクト値: 値を組み合わせて作られた値
  - オブジェクト
  - (配列)

値の同一性

- プリミティブ値の場合

  int x = 1;

  int y = 2-1;

  → xとyは同じ値を持つ

- オブジェクトの場合

  作られるごとに違う値ができる

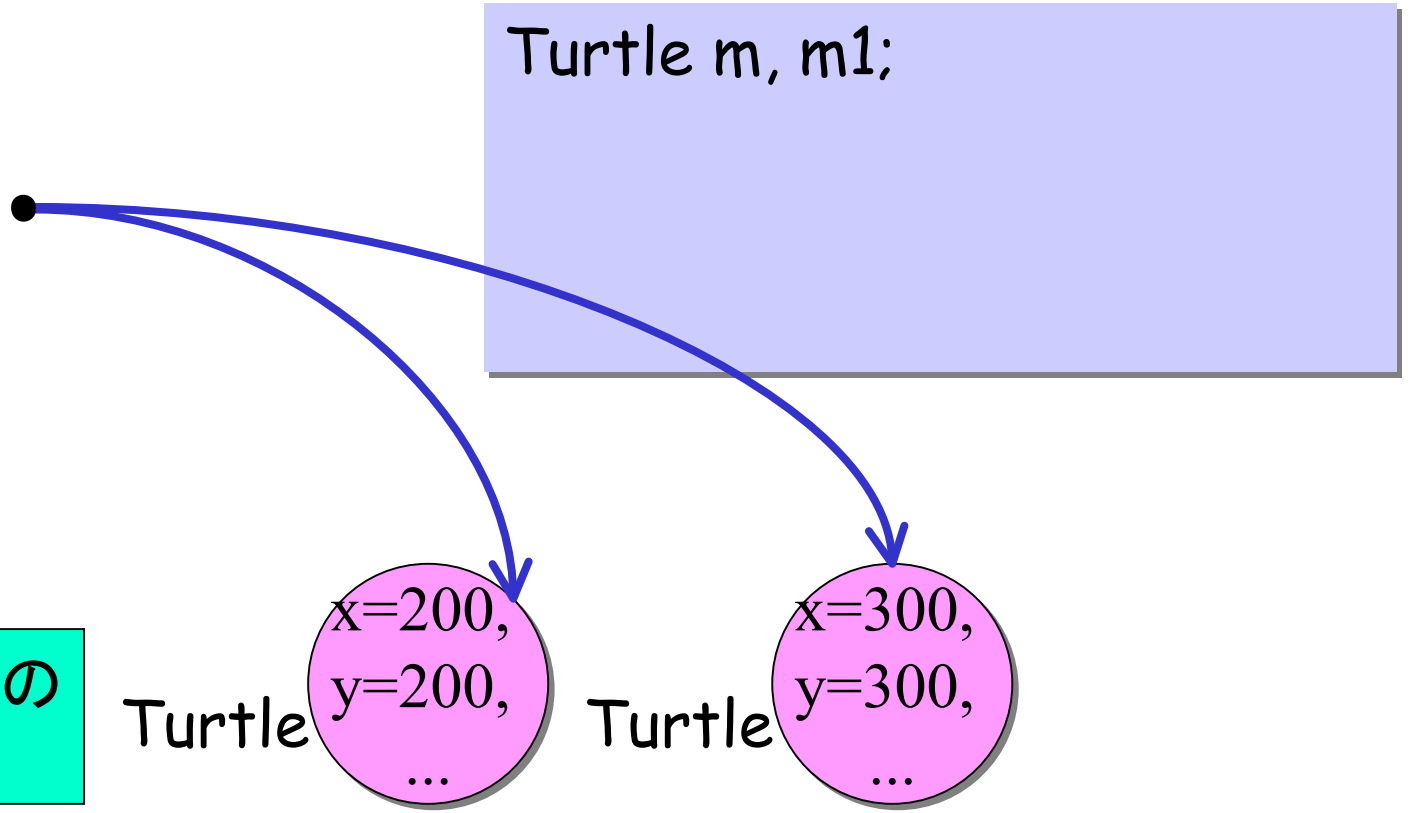
  Turtle m = new Turtle();

  Turtle m1 = new Turtle();

  → mとm1は違うオブジェクト

プログラム言語によっては
「状態が同じ」=「同じ値」
とするものもある

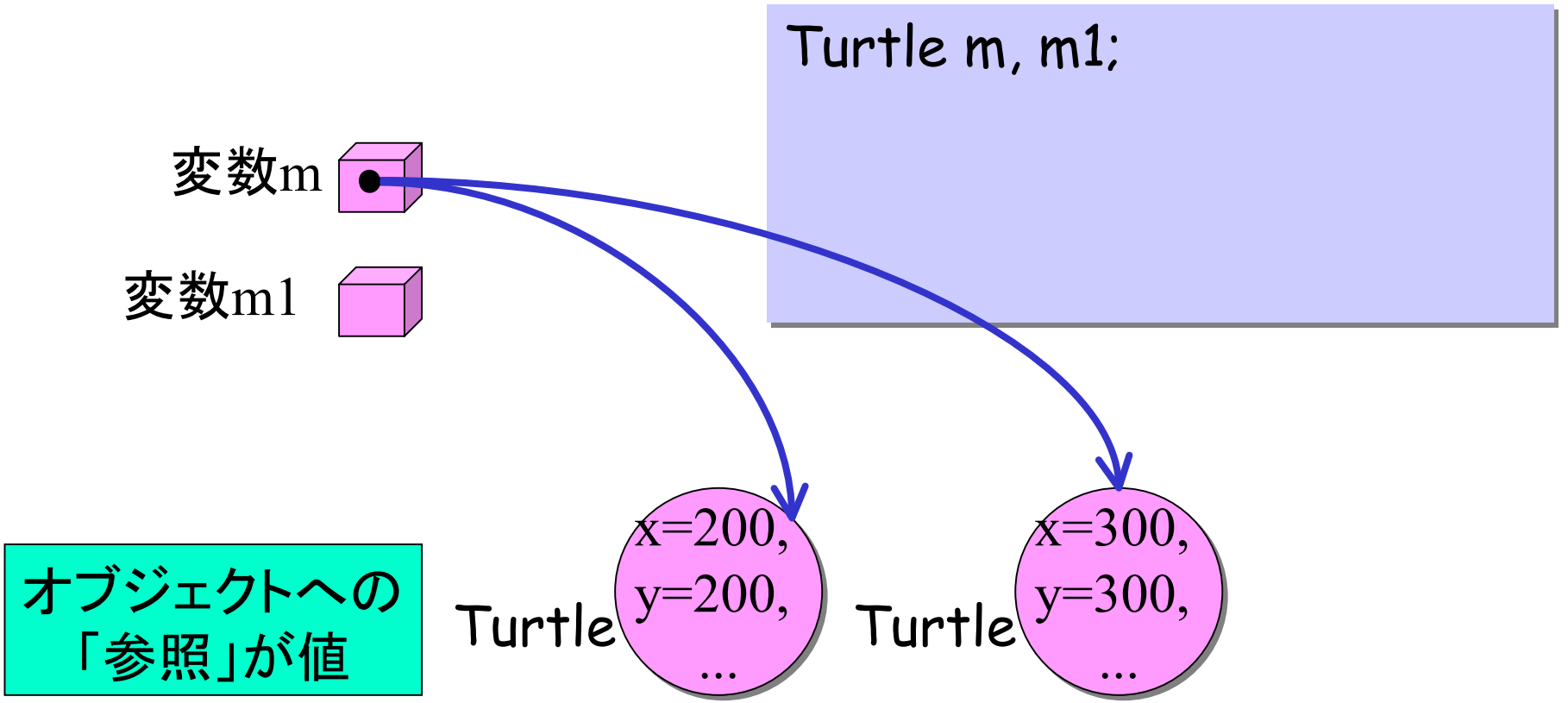
オブジェクト値



オブジェクトへの
「参照」が値

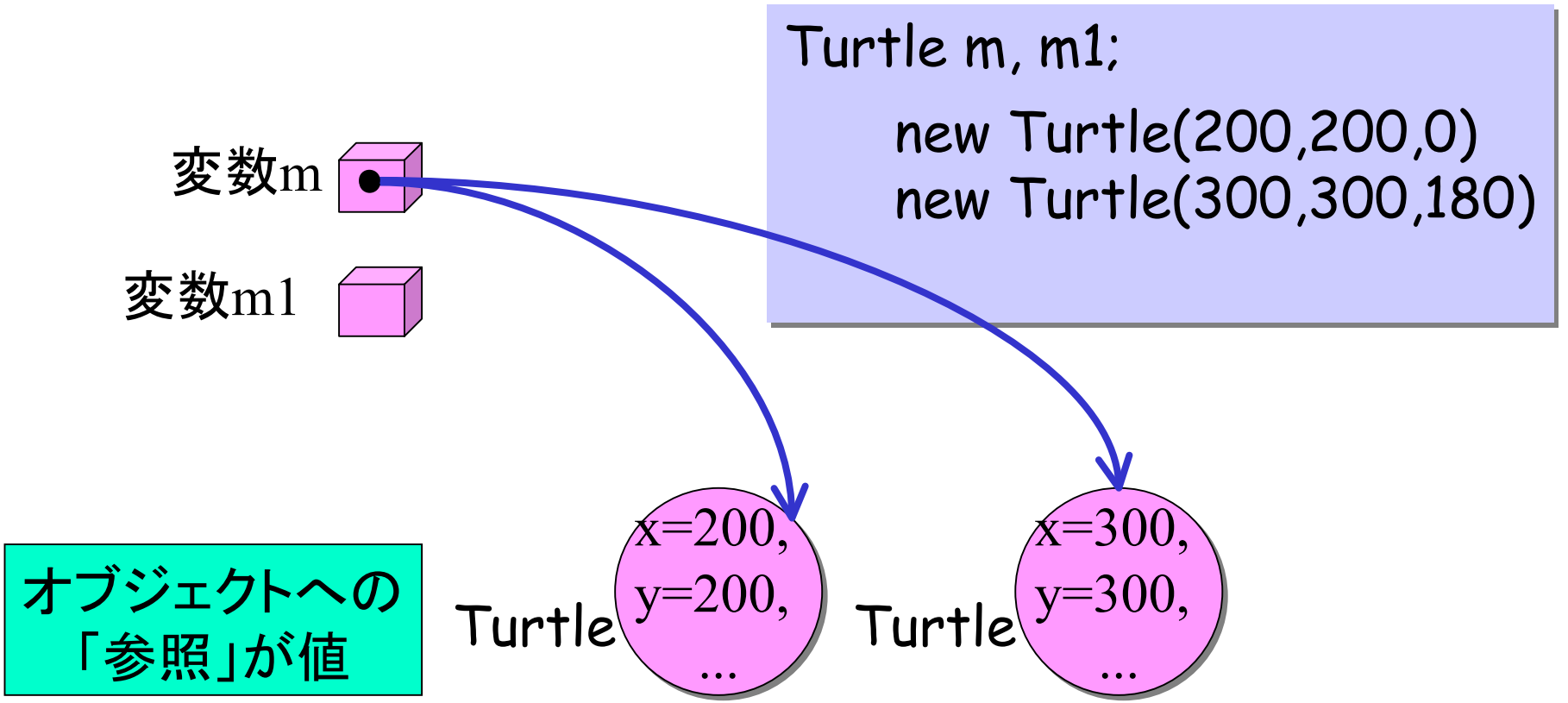
他の言語を知っている人向けの説明:
オブジェクト: メモリ上に置かれたデータ領域
参照: メモリアドレス

オブジェクト値



他の言語を知っている人向けの説明:
オブジェクト: メモリ上に置かれたデータ領域
参照: メモリアドレス

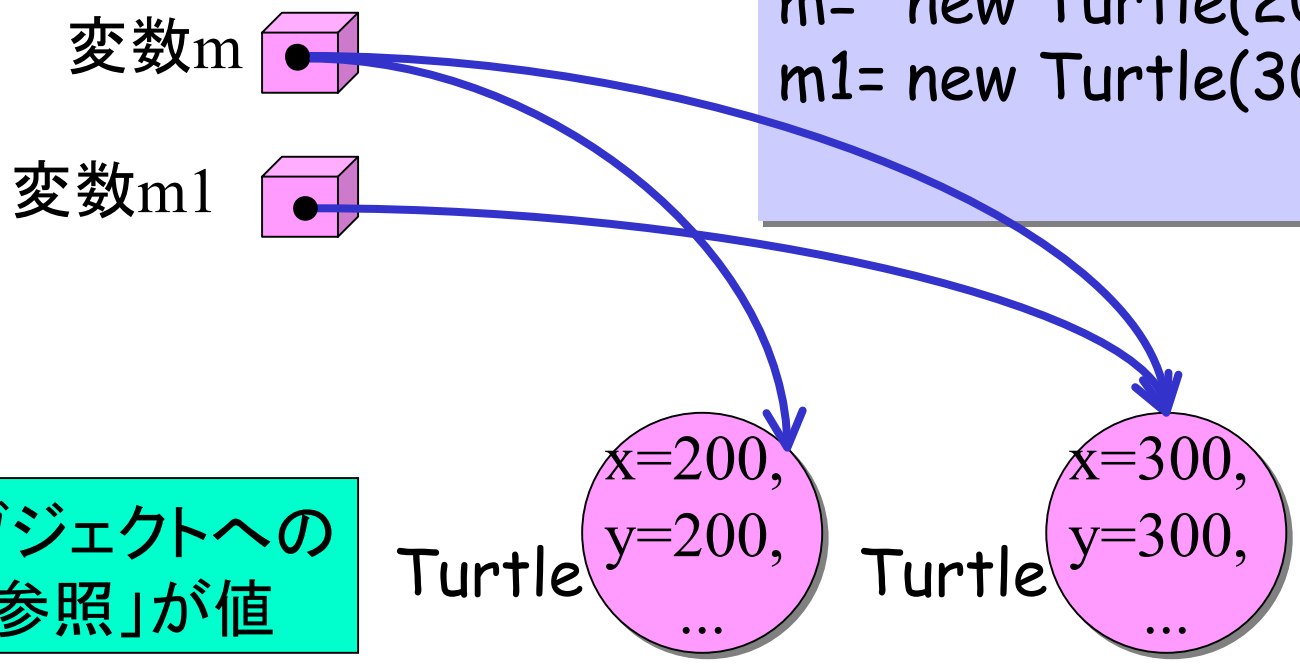
オブジェクト値



他の言語を知っている人向けの説明:
オブジェクト: メモリ上に置かれたデータ領域
参照: メモリアドレス

オブジェクト値

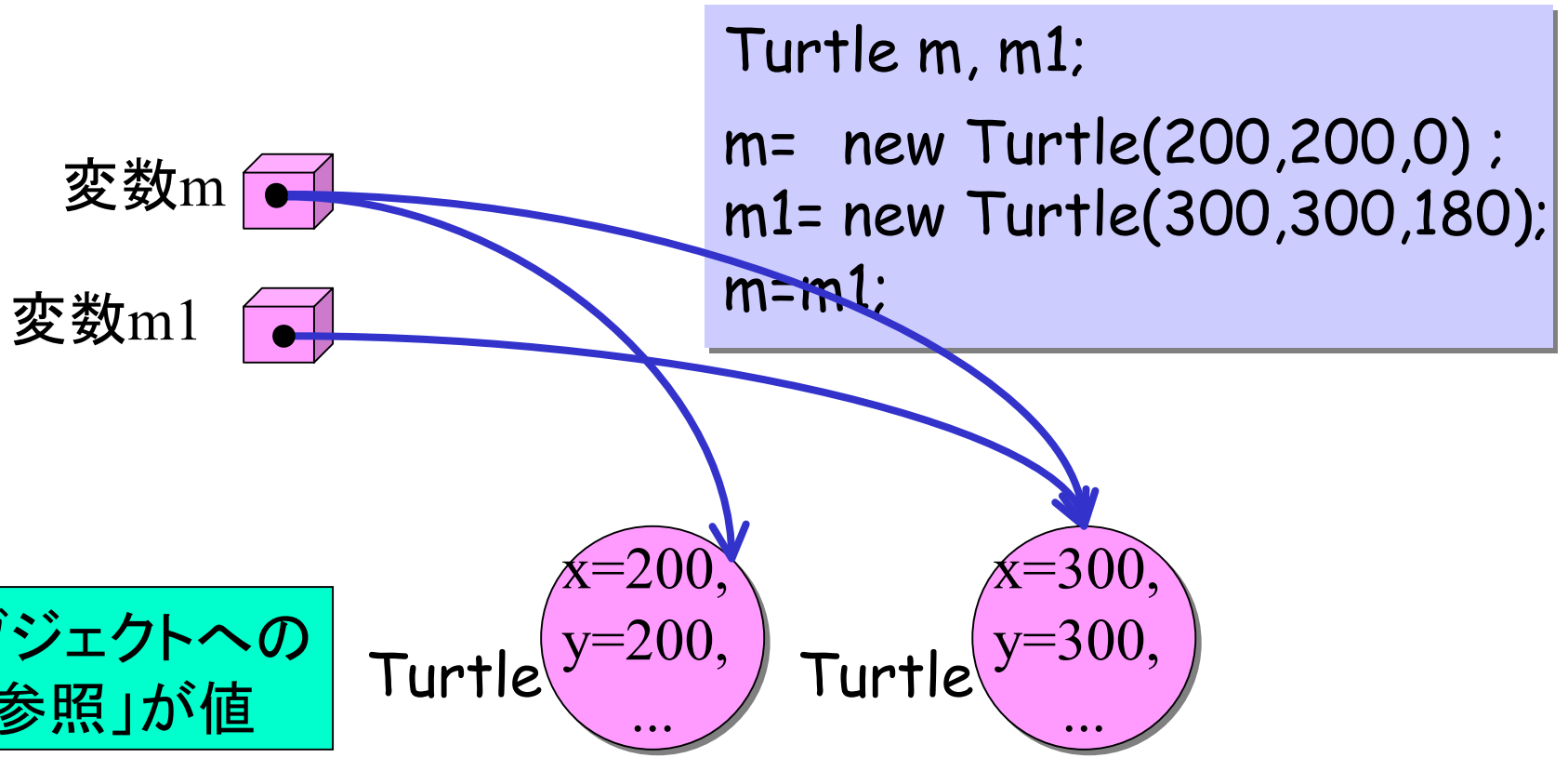
```
Turtle m, m1;
m=  new Turtle(200,200,0) ;
m1= new Turtle(300,300,180);
```



オブジェクトへの「参照」が値

他の言語を知っている人向けの説明:
オブジェクト: メモリ上に置かれたデータ領域
参照: メモリアドレス

オブジェクト値



オブジェクトへの「参照」が値

他の言語を知っている人向けの説明:
オブジェクト: メモリ上に置かれたデータ領域
参照: メモリアドレス

クラス変数・クラスメソッド (3.3)

- オブジェクトのインスタンス変数・インスタンスメソッド:
  オブジェクト毎の状態・操作
  - オブジェクト指向: ほとんどの「もの」をオブジェクトとして表現

- オブジェクトに関係ない状態・操作は?
  → クラス変数・クラスメソッド
  クラス毎に用意された変数・手続き
  - 定数 (円周率)
  - 数学関数 (三角関数・乱数)
  - プログラムの実行環境に関する情報
    (ターミナルウィンドウの入出力)

(非オブジェクト指向言語の)大域変数・手続きに相当

クラス変数・クラスメソッド (3.4)

- Mathクラス —— 数学定数・関数など
  - static final double PI　— 円周率の値
  - static double sin(double r)　— rの正弦値を返す
  - static double random() — 乱数値を返す
- Systemクラス —— 計算機に関するもの
  - static final java.io.PrintStream out
    — ターミナルウィンドウに文字を出す「ロ」
  - static void exit(int status)
    — プログラムを強制終了させる

クラス変数

返り値の型の前に“static”
→ クラス変数

フィールドの概要	
int[][]	kame
java.awt.Color	kameColor
int[][]	kameL
int[][]	kameR
double	kameScale
static boolean	withKameAll もし、false なら、個々のタートルの withKame の値に関わらず、全てのタートルに高速な描画を行う。

“クラス名．変数名”で
値をとり出す・代入

```
import java.awt.Color;
public class T23 {
  public static void main(String[] args){
    int d = 100, x, y, a;
    Turtle.withKameAll = false;
    TurtleFrame f = new TurtleFrame();
    Turtle m = new Turtle(200,300,0);
    f.add(m);
    m.fd(d);
    x = m.getX();                                    // m.
```

クラスメソッド

```
import java.awt.Color;
public class T23 {
  public static void main(String[] args){
    int d = 100, x, y, a;
    TurtleFrame f = new TurtleFrame();
    int a = (int)(Math.random()*360);
    Turtle m = new Turtle(200,300,a);
    f.add(m);
    m.fd(d);
    x = m.getX()                                  // m のX 座標とり出し
    y = m.getY();                                 // m のY 座標とり出し
    a = m.getAngle() - 45;                        // m の角度とり出し
    Turtle m1 = new Turtle(x, y, a);              //m1 の作成
    f.add(m1);
    m.kameColor = new Color(0,255,255);           // m の亀の色を水色変える
    m.kameScale = m.kameScale * 2;                // m の亀を現在の 2 倍の大きさにする
    m.rt(45);
    d = d / 2;
    m.fd(d);
    m1.fd(d);
    }
}
```

クラスメソッド

```
import java.awt.Color;
public class T23 {
  public static void main(String[] args){
    int d = 100, x, y, a;
    TurtleFrame f = new TurtleFrame();
    int a = (int)(Math.random()*360);
    Turtle m = new Turtle(200,300,a);
    f.add(m);
    m.fd(d);
    x = m.getX()
    y = m.getY();
    a = m.getAngle() - 45;
    Turtle m1 = new Turtle(x, y, a);
    f.add(m1);
    m.kameColor = new Color(0,255,255);
    m.kameScale = m.kameScale * 2;
    m.rt(45);
    d = d / 2;
    m.fd(d);
    m1.fd(d);
    }
}
```

クラス名．メソッド名(引数, ...)
クラスメソッドの呼出し

//m1 の作成

// m の亀の色を水色変える
// m の亀を現在の 2 倍の大きさにする

クラスメソッド

```
import java.awt.Color;
public class T23 {
  public static void main(String[] args){
    int d = 100, x, y, a;
    TurtleFrame f = new TurtleFrame();
    int a = (int)(Math.random()*360);
    Turtle m = new Turtle(200,300,a);
    f.add(m);
    m.fd(d);
    x = m.getX()
    y = m.getY();
    a = m.getAngle() - 45;
    Tur
    f.ad
    m.k                                    を水色変える
    m.k                                    在の 2 倍の大きさにする
    m.r
    d =
    m.fd(d);
    m1.fd(d);
    }
}
```

クラス名．メソッド名(引数, ...)
クラスメソッドの呼出し

randomメソッドの返り値は
double(浮動小数点数)型
→int(整数)型に変換する必要がある

クラス vs. インスタンス

- クラスメソッド呼出し

  クラス名.メソッド名(式,...)

  例: Math.sin(3.141592)

- クラス変数

  クラス名.変数名

  例: Math.PI

プログラム全体に渡る・・・

- インスタンスメソッド呼出し

  式.メソッド名(式,...)

  例: m.fd(100)

- インスタンス変数

  式.変数名

  例: m.KameScale = 0.1;

個々のオブジェクトに対する・・・

式と文 (3.7)

- 式: 値を表わす表現
  - 数値式　　180
  - 変数式　　x
  - 計算式　　d / 2
  - コンストラクタ呼出し式　　new Turtle()
  - …
- 文: 命令を表わす表現
  - 代入文　　m=new Turtle();
  - 変数宣言文　　TurtleFrame f;
  - …

注意!
これは不正確

式と文

- 式: 値を表わす表現
  - 数値式　　180
  - 変数式　　x
  - 計算式　　d / 2
  - コンストラクタ呼出し式　　new Turtle()
  - 代入式　　m=new Turtle()
  - メソッド呼出し式　　m.getX()
  - …
- 文: 命令を表わす表現
  - 変数宣言文　　TurtleFrame f;
  - 式文　　代入式;
メソッド呼出式;
  - ブロック　　{ 文; 文; … }

式と文

- 式: 値を表わす表現
  - 数値式　180
  - 変数式　x
  - 計算式　d / 2
  - コンストラクタ呼出し式　new Turtle()
  - 代入式　m=new Turtle()
  - メソッド呼出し式　m.getX()
  - …
- 文: 命令を表わす表現
  - 変数宣言文　TurtleFrame f;
  - 式文　代入式;
メソッド呼出式;
  - ブロック　{ 文; 文; … }

代入された値が値になる
f.add(m=new Turtle());

式と文

- 式: 値を表わす表現
  - 数値式
    - 180
    - x
    - d / 2
    - new Turtle()
    - m=new Turtle()
    - m.getX()
  - …
- 文: 命令を表わす表現
  - 変数宣言文 — TurtleFrame f;
  - 式文 — 代入式; / メソッド呼出式;
  - ブロック — { 文; 文; … }

ある種の式は「;」をつけると
文(命令)になる
例: m.fd(100);
式の値は捨てられる

代入された値が値になる
f.add(m=new Turtle());

演算子 (3.8)

- “c / 2”, “-x”, …
- 算術演算子: +, -, *, /, …
- 変数の値を変化させるもの: “++x”
- ビット演算子: &, |, >>, <<
- キャスト演算子: “(int)3.14”
- 代入演算子: “x=3”
- 関係演算子: “x==3”
- 論理演算子: “&&” “||”

練習問題

3.1	クラス変数・クラスメソッドの利用
3.2～5	標準クラスライブラリ中の・・・
3.3, 3.4	〃 (文字列とメッセージ表示)
3.6	演算子

制御構造

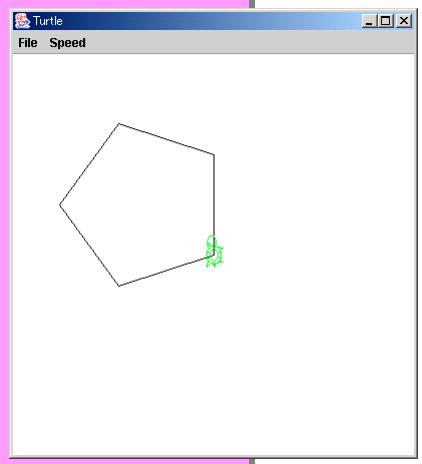
- (これまで)プログラムの実行:
  上から下へと順に命令を1つずつ実行

- もっと複雑な動きを!
  - 同じ命令を ***n*** 回繰返して実行する
  - 同構命令を～になるまで実行する
  - ～のときは～を実行する

→ 構造制御 (for文, while文, if文)

繰返し T41 (4.1)

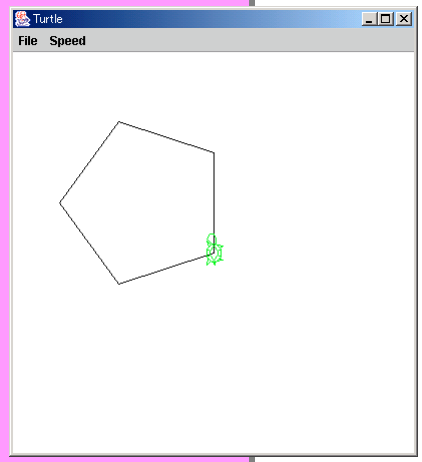
```
/** 正五角形を描くプログラム */
public class T20Pentagon {
  public static void main(String[] args){
    TurtleFrame f;              //変数 f の型宣言
    f =  new TurtleFrame();     //TurtleFrameを作成しfに代入
    Turtle m = new Turtle();    //Turtle を作成し，m の初期値として代入
    f.add(m);                   //f に m を追加
    m.fd(100);                  //m よ前に 100 進め
    m.lt(72);                   //m よ右に 72 度回れ
    m.fd(100);                  //m よ前に 100 進め
    m.lt(72);                   //m よ右に 72 度回れ
    m.fd(100);                  //m よ前に 100 進め
    m.lt(72);                   //m よ右に 72 度回れ
    m.fd(100);                  //m よ前に 100 進め
    m.lt(72);                   //m よ右に 72 度回れ
    m.fd(100);                  //m よ前に 100 進め
    m.lt(72);                   //m よ右に 72 度回れ
  }
}
```



繰返し T41 (4.1)

```
/** 正五角形を描くプログラム */
public class T20Pentagon {
  public static void main(String[] args){
    TurtleFrame f;              //変数 f の型宣言
    f =  new TurtleFrame();     //TurtleFrameを作
    Turtle m = new Turtle();    //Turtle を作成し,
    f.add(m);                   //f に m を追加
    m.fd(100);                  //m よ前に 100 進め
    m.lt(72);                   //m よ右に 72 度回れ
    m.fd(100);                  //m よ前に 100 進め
    m.lt(72);                   //m よ右に 72 度回れ
    m.fd(100);                  //m よ前に 100 進め
    m.lt(72);                   //m よ右に 72 度回れ
    m.fd(100);                  //m よ前に 100 進め
    m.lt(72);                   //m よ右に 72 度回れ
    m.fd(100);                  //m よ前に 100 進め
    m.lt(72);                   //m よ右に 72 度回れ
  }
}
```

辺の長さ・角度を
変えるのは大変
↓
変数を使って抽象化



繰返し T41 (4.1)

```
/** 正五角形を描くプログラム (変数を使って抽象化) */
public class T20Pentagon2 {
  public static void main(String[] args){
    int length = 100;                    // 一辺の長さ
    int numberOfEdges = 5;               // 辺の数
    int angle = 360/numberOfEdges;       // 頂点で曲がる角度

    TurtleFrame f;                       //変数 f の型宣言
    f =  new TurtleFrame();              //TurtleFrameを作成しfに代入
    Turtle m = new Turtle();             //Turtle を作成し，m の初期値として代入
    f.add(m);                            //f に m を追加
    m.fd(length);                        //一辺を描く
    m.lt(angle);                         //頂点で曲がる
    m.fd(length);                        //一辺を描く
    m.lt(angle);                         //頂点で曲がる
    m.fd(length);                        //一辺を描く
    m.lt(angle);                         //頂点で曲がる
    m.fd(length);                        //一辺を描く
    m.lt(angle);                         //頂点で曲がる
    m.fd(length);                        //一辺を描く
    m.lt(angle);                         //頂点で曲がる
  }
}
```

変数を使って抽象化

繰返し T41 (4.1)

```
/** 正五角形を描くプログラム (変数を使って抽象化) */
public class T20Pentagon2 {
  public static void main(String[] args){
    int length = 100;                    // 一辺の長さ
    int numberOfEdges = 5;               // 辺の数
    int angle = 360/numberOfEdges;       // 頂点で曲がる角度

    TurtleFrame f;                       //変数 f の型宣言
    f =  new TurtleFrame();              //TurtleFrameを作成しfに代入
    Turtle m = new Turtle();             //Turtle を作成し，m の初期値として代入
    f.add(m);                            //f に m を追加
    m.fd(length);                        //一辺を描く
    m.lt(angle);                         //頂点で曲がる
    m.fd(length);                        //一辺を描く
    m.lt(angle);                         //頂点で曲がる
    m.fd(length);                        //一辺を描く
    m.lt(angle);                         //頂点で曲がる
    m.fd(length);                        //一辺を描く
    m.lt(angle);                         //頂点で曲がる
    m.fd(length);                        //一辺を描く
    m.lt(angle);                         //頂点で曲がる
  }
}
```

変数を使って抽象化

辺の数は固定!
↓
《繰返し》で
回数を制御

繰返し T41 (4.1)

```
public class T41 {
  public static void main(String[] args){
    TurtleFrame f = new TurtleFrame();
    Turtle m = new Turtle();
    f.add(m);
    int i;
    for(i = 0; i < 5; i++){
       m.fd(100);
       m.rt(72);
    }
  }
}
```

繰返し T41 (4.1)

前進(fd)と右回転(rt)を5回繰り返す
↓
回数を制御

```
public class T41 {
  public static void mai
    TurtleFrame f = nev
    Turtle m = new Turtle();
    f.add(m);
    int i;
    for(i = 0; i < 5; i++){
       m.fd(100);
       m.rt(72);
    }
  }
}
```

繰返し T41 (4.1)

```
public class T41 {
  public static void mai
    TurtleFrame f = nev
    Turtle m = new Turtle();
    f.add(m);
    int i;
    for(i = 0; i < 5; i++){
       m.fd(100);
       m.rt(72);
    }
  }
}
```

前進(fd)と右回転(rt)を5回繰り返す
↓
回数を制御

直感的な意味:

```
int変数;
for(変数=0; 変数<上界; 変数++) {
  文;
  文;
   …
}
```

上界回、文を繰り返す

繰返し T41 (4.1)

```
public class T41 {
  public static void mai
    TurtleFrame f = nev
    Turtle m = new Turtle();
    f.add(m);
    int i;
    for(i = 0; i < 5; i++){
       m.fd(100);
       m.rt(72);
    }
  }
}
```

前進(fd)と右回転(rt)を5回繰り返す
↓
回数を制御

厳密な意味:

```
for(式1; 式2; 式3) {
  文;
  文;
   …
}
```

i) 式1を計算
ii) 式2の値がtrueなら
iii)文を実行
iv)式3を計算
v) ii以下を繰り返す

繰返し T41 (4.1)

i) i=0を計算
ii) i<5の値がtrueなら
iii)fd,rtを実行
iv)i++を計算
v) ii以下を繰り返す

前進(fd)と右回転(rt)を5回繰り返す
↓
回数を制御

```
tle();
    int i;
    for(i = 0; i < 5; i++){
        m.fd(100);
        m.rt(72);
    }
  }
}
```

厳密な意味:

```
for(式1; 式2; 式3) {
  文;
  文;
    …
}
```

i) 式1を計算
ii) 式2の値がtrueなら
iii)文を実行
iv)式3を計算
v) ii以下を繰り返す

繰返し T41 (4.1)

i) iを0にする
ii) iが5未満なら
iii)fd,rtを実行
iv)iを1増やす
v) ii以下を繰り返す

前進(fd)と右回転(rt)を5回繰り返す
↓
回数を制御

```
tle();
    int i;
    for(i = 0; i < 5; i++){
        m.fd(100);
        m.rt(72);
    }
  }
}
```

厳密な意味:

```
for(式1; 式2; 式3) {
  文;
  文;
    …
}
```

i) 式1を計算
ii) 式2の値がtrueなら
iii)文を実行
iv)式3を計算
v) ii以下を繰り返す

繰返し T41 (4.1)

```
    int i;
    for(i = 0; i < 5; i++){
      m.fd(100);
      m.rt(72);
    }
  }
}
```

i) iを0にする
ii) iが5未満なら
iii)fd,rtを実行
iv)iを1増やす
v) ii以下を繰り返す

直感的な意味:
iを0, 1, ... と順に1ずつ増やし、
i が 5未満である間、
fd,rtを実行

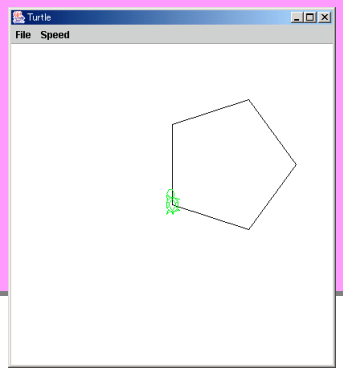
厳密な意味:

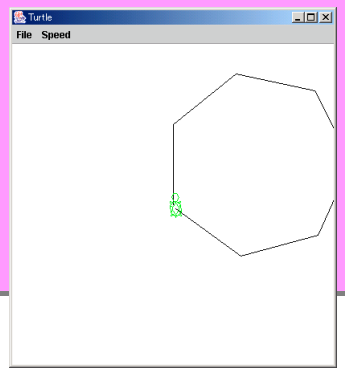
```
for(式1; 式2; 式3) {
  文;
  文;
   ...
}
```

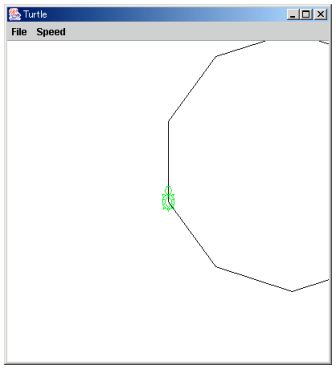
i) 式1を計算
ii) 式2の値がtrueなら
iii)文を実行
iv)式3を計算
v) ii以下を繰り返す

繰返し T41 (4.1)

```
/** 正多角形を描くプログラム */
public class T41Polygon {
  public static void main(String[] args){
    int length = 100, numEdges = 5, angle = 360/numEdges;
    TurtleFrame f = new TurtleFrame();
    Turtle m = new Turtle();
    f.add(m);
    int i;
    for(i = 0; i < numEdges; i++){
      m.fd(length);
      m.rt(angle);
    }
  }
}
```



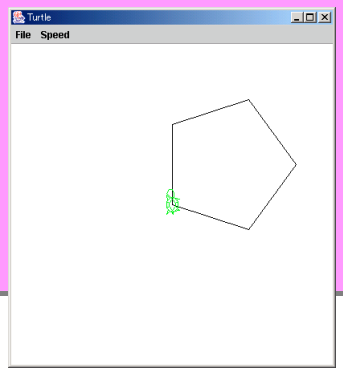


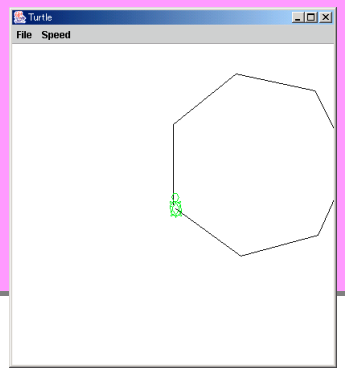


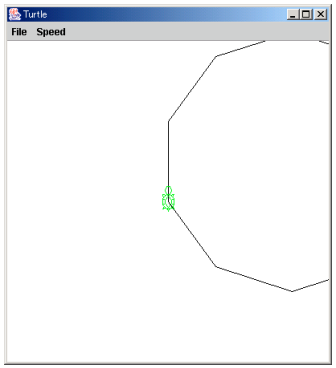
繰返し T41 (4.1)

```
/** 正多角形を描くプログラム */
public class T41Polygon {
  public static void main(String[] args){
    int length = 100, numEdges = 5, angle = 360/numEdges;
    TurtleFrame f = new TurtleFrame();
    Turtle m = new Turtle();
    f.add(m);
    int i;
    for(i = 0; i < numEdges; i++){
      m.fd(length);
      m.rt(angle);
    }
  }
}
```

繰返し回数を
抽象化できた!







繰返しのネスト T45 (4.2)

```
public class T41 {
  public static void main(String[] args){
    TurtleFrame f = new TurtleFrame();
    Turtle m = new Turtle();
    f.add(m);
    int i;
    for(i = 0; i < 5; i++){
       m.fd(100);
       m.rt(72);
    }
  }
}
```

直感的な意味:
五角形を描く

繰返しのネスト T45 (4.2)

```
public class T41 {
  public static void ma
    TurtleFrame f = ne
    Turtle m = new Tur
    f.add(m);
    int i;
    for(i = 0; i < 5; i++){
       m.fd(100);
       m.rt(72);
    }
  }
}
```

直感的な意味:
五角形を描く

繰返しのネスト T45 (4.2)

```
public class T41 {
  public static void ma
    TurtleFrame f = ne
    Turtle m = new Tur
    f.add(m);
    int i;
    for(i = 0; i < 5; i++){
       m.fd(100);
       m.rt(72);
    }
  }
}
```

- for文全体は1つの文として扱える

直感的な意味:
五角形を描く

繰返しのネスト T45 (4.2)

```
public class T41 {
  public static void ma
    TurtleFrame f = ne
    Turtle m = new Tur
    f.add(m);
    int i;
    for(i = 0; i < 5; i++){
       m.fd(100);
       m.rt(72);
    }
  }
}
```

- for文全体は1つの文として扱える
- for文の{ }内は文がいくつも書ける

直感的な意味:
五角形を描く

繰返しのネスト T45 (4.2)

```
public class T41 {
  public static void ma
    TurtleFrame f = ne
    Turtle m = new Tur
    f.add(m);
    int i;
    for(i = 0; i < 5; i++){
       m.fd(100);
       m.rt(72);
    }
  }
}
```

- for文全体は1つの文として扱える
- for文の{ }内は文がいくつも書ける
- for文の中にfor文を書ける

直感的な意味:
五角形を描く

繰返しのネスト T45 (4.2)

```
public class T41 {
  public static void ma
    TurtleFrame f = ne
    Turtle m = new Tur
    f.add(m);
    int i;
    for(i = 0; i < 5; i++){
       m.fd(100);
       m.rt(72);
    }
  }
}
```

- for文全体は1つの文として扱える
- for文の{ }内は文がいくつも書ける
- for文の中にfor文を書ける
- 外側の繰返しの各回で
  内側の繰返しが起きる

直感的な意味:
五角形を描く

繰返しのネスト T45 (4.2)

```
public class T45 {
  public static void main(String[] args){
    TurtleFrame f = new TurtleFrame();
    Turtle m = new Turtle();
    f.add(m);
    for(int j = 0; j < 8; j++){




      m.fd(50);
      m.rt(45);
    }
  }
}
```

直感的な意味
八角形を描く

繰返しのネスト T45 (4.2)

```
public class T45 {
  public static void main(String[] args){
    TurtleFrame f = new TurtleFrame();
    Turtle m = new Turtle();
    f.add(m);
    for(int j = 0; j < 8; j++){



      m.fd(50);
      m.rt(45);
    }
  }
}
```

直感的な意味:
五角形を描く

直感的な意味
八角形を描く

繰返しのネスト T45 (4.2)

```
public class T45 {
  public static void main(String[] args){
    TurtleFrame f = new TurtleFrame();
    Turtle m = new Turtle();
    f.add(m);
    for(int j = 0; j < 8; j++){




      m.fd(50);
      m.rt(45);
    }
  }
}
```

直感的な意味:
五角形を描く

直感的な意味:
八角形を描く

繰返しのネスト T45 (4.2)

```
public class T45 {
  public static void main(String[] args){
    TurtleFrame f = new TurtleFrame();
    Turtle m = new Turtle();
    f.add(m);
    for(int j = 0; j < 8; j++){




      m.fd(50);
      m.rt(45);
    }
  }
}
```

直感的な意味:
五角形を描く

直感的な意味:
八角形を描く

合わせた意味:
八角形の各頂点で
五角形を描く

while文 (4.3)

```
public class T48 {
  public static void main(String[] args){
    TurtleFrame f = new TurtleFrame();
    Turtle m = new Turtle();
    f.add(m);
    int i = 1;
    while(m.getX()>=100) {
      m.fd(i*10);
      m.rt(72);
      i++;
    }
  }
}
```

直感的な意味:

```
while(式) {
  文;
  文;
   …
}
```

式がtrueである間、文を繰り返す

while文 (4.3)

```
public class T48 {
  public static void main(String[] args){
    TurtleFrame f = new TurtleFrame();
    Turtle m = new Turtle();
    f.add(m);
    int i = 1;
    while(m.getX()>=100) {
      m.fd(i*10);
      m.rt(72);
      i++;
    }
  }
}
```

回数があらかじめ
分からない繰返し

直感的な意味:

```
while(式) {
  文;
  文;
   …
}
```

式がtrueである間、
文を繰り返す

while文 (4.3)

```
public class T48 {
  public static void main(String[] args){
    TurtleFrame f = new TurtleFrame();
    Turtle m = new Turtle();
    f.add(m);
    int i = 1;
    while(m.getX()>=100) {
      m.fd(i*10);
      m.rt(72);
      i++;
    }
  }
}
```

厳密な意味:

```
while(式) {
  文;
  文;
   …
}
```

i) 式を計算
ii) もしtrueなら
iii) 文を実行
iv) iから繰返す

while文 (4.3)

```
publi
  pub                                    rgs){
    Tu                                   me();
    Tu
    f.add(m);
    int i = 1;
    while(m.getX()>=100) {
      m.fd(i*10);
      m.rt(72);
      i++;
    }
  }
}
```

i) m.getX()>=0を計算
ii) もしtrueなら
iii) fd;rt;i++を実行
iv) iから繰返す

厳密な意味:

```
while(式) {
  文;
  文;
   …
}
```

i) 式を計算
ii) もしtrueなら
iii) 文を実行
iv) iから繰返す

while文 (4.3)

```
publi
  pub
    Tu
    Tu
    f.add(m);
    int i = 1;
    while(m.getX()>=100) {
      m.fd(i*10);
      m.rt(72);
      i++;
    }
  }
}
```

i) m.getX()>=0を計算
ii) もしtrueなら
iii) fd;rt;i++を実行
iv) iから繰返す

mのx座標が正である間
fd;rt;i++を繰返す

厳密な意味:

```
while(式) {
  文;
  文;
   …
}
```

i) 式を計算
ii) もしtrueなら
iii) 文を実行
iv) iから繰返す

条件分岐 (4.4)

```
public class T493 {
  public static void main(String[] args){
    TurtleFrame f = new TurtleFrame();
    Turtle m = new Turtle();
    f.add(m);
    for(int i = 0; i < 12; i++){
      if(i % 3 == 0){              // i が 3 の倍数のとき
        m.setColor(java.awt.Color.red);
      }else if (i % 3 == 1) {    // i が 3 で割って余り1のとき
        m.setColor(java.awt.Color.green);
      }else{                       // それ以外（3 で割って余り2）のとき
        m.setColor(java.awt.Color.yellow);
      }
      m.lt(30);
      m.fd(50);
    }
  }
}
```

条件分岐 (4.4)

```
public class T493 {
  public static void main(String[] args){
    TurtleFrame f = new TurtleFrame();
    Turtle m = new Turtle();
    f.add(m);
    for(int i = 0; i < 12; i++){
      if(i % 3 == 0){              // i が 3 の倍数のとき
        m.setColor(java.awt.Color.red);
      }else if (i % 3 == 1) {    // i が 3 で割って余り
        m.setColor(java.awt.Color.green);
      }else{                       // それ以外（3 で割って余り2）のとき
        m.setColor(java.awt.Color.yellow);
      }
      m.lt(30);
      m.fd(50);
    }
  }
}
```

```
if (式) {
   文;
} else {
   文;
}
```

練習

4.1～2	for文による繰返し
4.3～4	二重のfor文
4.5～6	while文による繰返し
4.7	if文

クイズ