

プログラミング入門 2

第5回 インスタンスメソッド

講義資料について

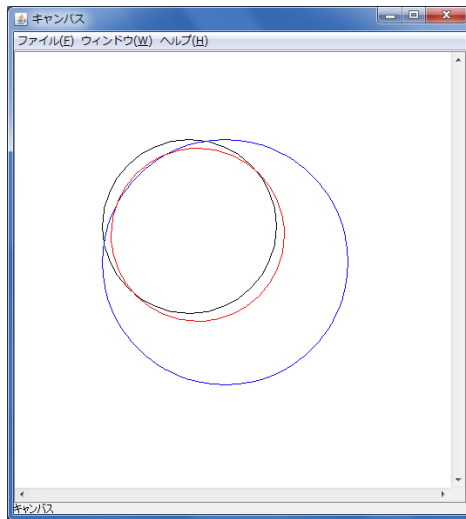
- 本講義から、新しい言語の機能（オブジェクト指向の機構）を連続で学んでいくので、随時参考書などを参照するのがよい。
- 2008年度までの「Java2」の講義資料も参考になる。
 - <http://java.cis.k.hosei.ac.jp/>
 - 今回の範囲、および、これまで（複合データ）の範囲は、上記ページの14回、15回に詳しい。

テーマ：インスタンスメソッド

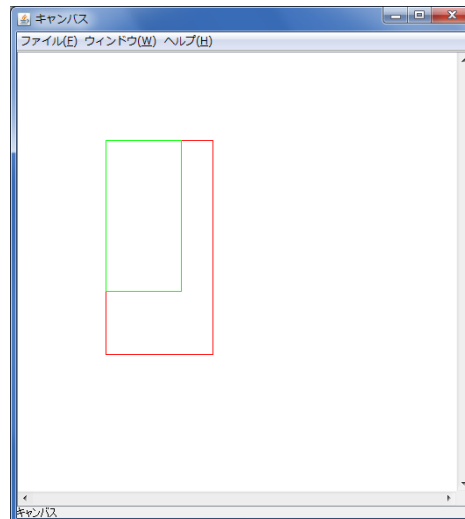
- クラスとインスタンス（復習）
- インスタンスメソッド
 - 定義と考え方
 - インスタンス変数の参照
 - インスタンスメソッドの呼び出し

本日の主な題材

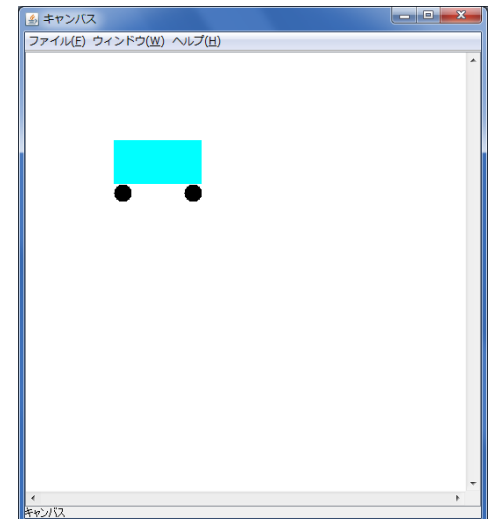
- MyCircle や MyRectangle, MyCar のインスタンスを生成して、インスタンスメソッドを実行する。(例題15, 問題16, 問題25)



MyCircle



MyRectangle



MyCar

クラスとインスタンス(復習)

問題: クラスMyCircleの内容を理解せよ。Sampleのメインメソッドを実行したときにCanvasにどのような円が出力されるかを示せ。

クラスMyCircle

```
public class MyCircle {  
    int x;  
    int y;  
    int diameter;  
}
```

クラスSample

```
public class Sample {  
    public static void main(String[] args) {  
        new Sample().start();  
    }  
  
    void start() {  
        MyCircle c0 = createMyCircle(100, 100, 200);  
        MyCircle c1 = createMyCircle(200, 200, 200);  
  
        Canvas.show();  
        drawMyCircle(c0);  
        drawMyCircle(c1);  
    }  
}
```

```
MyCircle createMyCircle(int x, int y, int diameter) {  
    MyCircle newObj = new MyCircle();  
    newObj.x = x;  
    newObj.y = y;  
    newObj.diameter = diameter;  
    return newObj;  
}
```

```
void drawMyCircle(MyCircle c) {  
    Canvas.drawOval(c.x, c.y, c.diameter, c.diameter);  
}
```

①

②

ヒント: インスタンスの内容(状態)の図 (答えは次スライド)

MyCircle のインスタンスを一つ作成する
MyCircle の別のインスタンスを作成する

クラスSample

```
public class Sample {  
    public static void main(String[] args) {  
        new Sample().start();  
    }  
  
    void start() {  
        MyCircle c0 = createMyCircle(100, 100, 200);  
        MyCircle c1 = createMyCircle(200, 200, 200);  
  
        Canvas.show();  
        drawMyCircle(c0);  
        drawMyCircle(c1);  
    }  
}
```

クラスMyCircle

```
public class MyCircle {  
    int x;  
    int y;  
    int diameter;  
}
```

インスタンスc0

```
x:100  
y:100  
diameter:200
```

インスタンスc1

```
x:200  
y:200  
diameter:200
```

クラスとインスタンス(復習)

答え(右下)

クラスMyCircle

```
public class MyCircle {  
    int x;  
    int y;  
    int diameter;  
}
```

クラスSample

```
public class Sample {  
    public static void main(String[] args) {  
        new Sample().start();  
    }  
  
    void start() {  
        MyCircle c0 = createMyCircle(100, 100, 200);  
        MyCircle c1 = createMyCircle(200, 200, 200);  
  
        Canvas.show();  
        drawMyCircle(c0);  
        drawMyCircle(c1);  
    }  
}
```

```
MyCircle createMyCircle(int x, int y, int diameter) {  
    MyCircle newObj = new MyCircle();  
    newObj.x = x;  
    newObj.y = y;  
    newObj.diameter = diameter;  
    return newObj;  
}
```

```
void drawMyCircle(MyCircle c) {  
    Canvas.drawOval(c.x, c.y, c.diameter, c.diameter);  
}
```

(100, 100)に直径200の円を描く

(200, 200)に直径200の円を描く

解説: メソッドの呼び出し

答え(右下)

クラスSample

```
public class Sample {  
    public static void main(String[] args) {  
        new Sample().start();  
    }  
  
    void start() {  
        MyCircle c0 = createMyCircle(100, 100, 200);  
        MyCircle c1 = createMyCircle(200, 200, 200);  
  
        Canvas.show();  
        drawMyCircle(c0);  
        drawMyCircle(c1);  
    }  
}
```

クラスMyCircle

```
public class MyCircle {  
    int x;  
    int y;  
    int diameter;  
}
```

```
MyCircle createMyCircle(int x, int y, int diameter) {  
    MyCircle newObj = new MyCircle();  
    newObj.x = x;  
    newObj.y = y;  
    newObj.diameter = diameter;  
    return newObj;  
}
```

```
void drawMyCircle(MyCircle c) {  
    Canvas.drawOval(c.x, c.y, c.diameter, c.diameter);  
}
```

(100, 100)に直径200の円を描く

クラスSampleで定義されているもの

- (1) MyCircle を生成するメソッド
- (2) MyCircle を描くメソッド

クラスMyCircleで定義されているもの

- (1) インスタンス変数 (x, y, diameter)

テーマ：インスタンスメソッド

- クラスとインスタンス（復習）
- インスタンスメソッド
 - 定義と考え方
 - インスタンス変数の参照
 - インスタンスメソッドの呼び出し

インスタンスメソッド(考え方)

(復習) インスタンスは複合データを構成するために、複数のインスタンス変数を保持できる。

```
x:100  
y:100  
diameter:200
```

(新規学習事項) インスタンスはインスタンス変数だけではなく、インスタンスを操作するメソッドを保持することもできる。これをインスタンスメソッドと言う。

```
x:100  
y:100  
diameter:200  
  
void drawMyCircle() {  
    Canvas.drawOval(  
        this.x,  
        this.y,  
        this.diameter,  
        this.diameter);  
}
```

インスタンスメソッドの定義

クラスAのインスタンスに保持させたいメソッドは、クラスAに記述する。
クラスAの**インスタンスを生成した後に**、インスタンスを操作するメソッドだけを
インスタンスメソッドとして定義できる。

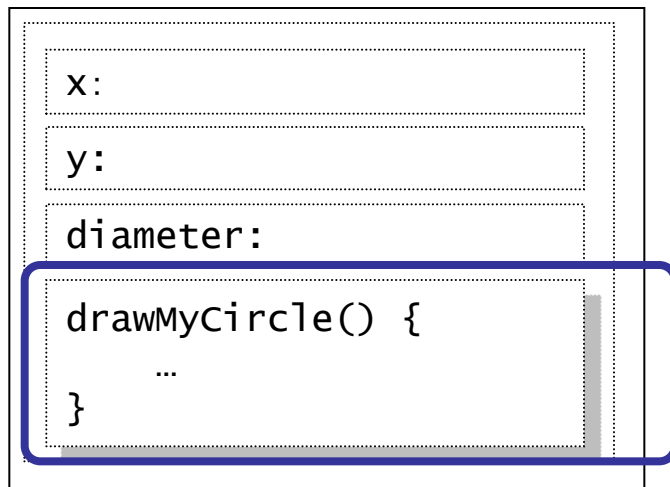
```
public class MyCircle2 {  
    int x;  
    int y;  
    int diameter;  
  
    void drawMyCircle(MyCircle c) {  
        Canvas.drawOval(this.x, this.y, this.diameter,  
                        this.diameter);  
    }  
}
```

```
public class クラス名 {  
    変数の型 インスタンス変数名;  
  
    戻り値の型 インスタンスメソッド名(引数){  
        ...  
    }  
}  
// 戻り値がない時には、戻り値の型はvoid にする
```

createMyCircle メソッドは、**インスタンスを生成する前に呼ばれるメソッド**なので、ここに定義できない!!

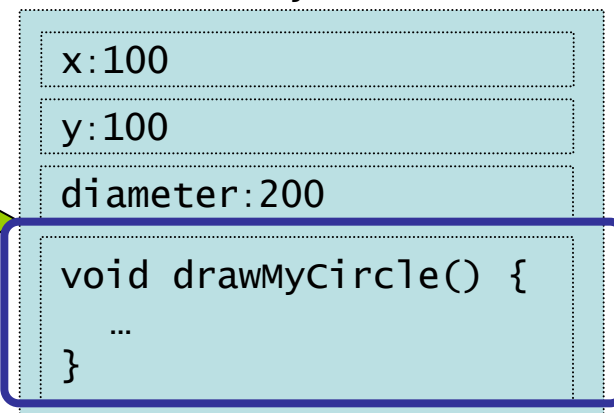
インスタンスの生成とインスタンスメソッド

クラスMyCircle2



```
MyCircle2 c0 = new MyCircle2();
```

インスタンス MyCircle2



インスタンスメソッドがクラスに書かれていると、そのクラスから生成されたインスタンスは、インスタンスメソッドを保持する。

上の図の場合、生成されたインスタンス `c0` がインスタンスメソッド `drawMyCircle` を保持する。

(クラスに書かれていたものがインスタンスにコピーされるイメージ。ただし、実際にコピーされるわけではない。)

インスタンスメソッドの呼び出し(起動)

インスタンスが保持するメソッドを、どのようにして起動するのか？

→メソッド名の前に、対象となるインスタンスを指定する。

インスタンス変数の参照と似た形である。

インスタンス.メソッド名(引数)

の形で行う。

インスタンスc0

x:100

y:100

diameter:200

```
void drawMyCircle() {  
    Canvas.drawOval(  
        this.x,  
        this.y,  
        this.diameter,  
        this.diameter);  
}
```

```
public class Sample {  
    public static void main(String[] args) {  
        new Sample().start();  
    }  
  
    void start() {  
        MyCircle c0 = createMyCircle(100, 100, 200);  
  
        Canvas.show();  
        c0.drawMyCircle();  
    }  
}
```

インスタンスc0が保持するdrawMyCircleメソッドを起動する。

(100, 100)に直径200の円を描く

インスタンスメソッドの呼び出し(考え方)

```
public class Sample2 {  
    public static void main(String[] args) {  
        new Sample2().start();  
    }  
  
    void start() {  
        MyCircle2 c0 = createMyCircle(100, 100, 200);  
        MyCircle2 c1 = createMyCircle(200, 200, 200);  
  
        Canvas.show();  
        c0.drawMyCircle();  
        c1.drawMyCircle();  
    }  
  
    // MyCircle2 createMyCircle(...) ...  
}
```

インスタンスc0

x:100

y:100

diameter:200

```
drawMyCircle() {  
    ...  
}
```

インスタンスc0が保持する
drawMyCircleメソッドを起動する。

(100,100)に直径200の
円を描く。

インスタンスc1

x:200

y:200

diameter:200

```
drawMyCircle() {  
    ...  
}
```

インスタンスc1が保持する
drawMyCircleメソッドを起動する。

(200,200)に直径200の
円を描く

インスタンスメソッド内からの インスタンス変数へのアクセス(1)

呼び出し

```
c0.drawMyCircle();
```

インスタンスc0

x:100

y:100

diameter:200

```
void drawMyCircle() {  
    Canvas.drawOval(  
        this.x,  
        this.y,  
        this.diameter,  
        this.diameter);  
}
```

thisは特別な変数で、インスタンスが自分自身を参照するために用いる。

this.x は、自分自身のインスタンス変数 x であるから、その内容は100である。

this.y は、100 である。

this.diameter は、200 である。

(100, 100)に直径200の円を描く

インスタンスメソッドdrawMyCircleの説明

インスタンスメソッド内からの インスタンス変数へのアクセス(2)

呼び出し

```
c0.drawMyCircle();
```

インスタンスc0

x:100

y:100

diameter:200

```
void drawMyCircle() {  
    Canvas.drawOval(  
        x,  
        y,  
        diameter,  
        diameter);  
}
```

thisと書かなくとも、
インスタンス変数
を参照することは可能。

問題: インスタンス c1 の状態に注目して 空欄を埋めよ

呼び出し

```
c1.drawMyCircle();
```

インスタンスc1

x:200

y:200

diameter:200

```
void drawMyCircle() {  
  Canvas.drawOval(  
    this.x,  
    this.y,  
    this.diameter,  
    this.diameter);  
}
```

thisは特別な変数で、インスタンスが自分自身を参照するために用いる。

this.x の内容は？

this.y の内容は？

this.diameter の内容は？

どんな円を描く？

インスタンスが自分自身が保持している インスタンスメソッドを呼ぶ(1)

インスタンス

```
public class MyCircle2 {  
    int x;  
    int y;  
    int diameter;  
  
    void drawMyCircle(MyCircle c) {  
        Canvas.drawOval(this.x, this.y, this.diameter,  
                        this.diameter);  
    }  
  
    // 追加したメソッド  
    void drawAndClear() {  
        this.drawMyCircle();  
        Canvas.waitForCountdown(2000);  
        Canvas.clear();  
    }  
}
```

x:100

y:100

diameter:200

```
void drawMyCircle() {  
    Canvas.drawOval(  
        this.x,  
        this.y,  
        this.diameter,  
        this.diameter);  
}
```

```
void drawAndClear() {  
    this.drawMyCircle();  
    Canvas.waitForCountdown(2000);  
    Canvas.clear();  
}
```

drawAndClearメソッド
が呼び出されると、
自分が保持している
drawMyCircleメソッド
を呼び出す

インスタンスが自分自身が保持している インスタンスメソッドを呼ぶ(2)

インスタンス

```
public class MyCircle2 {  
    int x;  
    int y;  
    int diameter;  
  
    void drawMyCircle(MyCircle c) {  
        Canvas.drawOval(this.x, this.y, this.diameter,  
                        this.diameter);  
    }  
  
    // 追加したメソッド  
    void drawAndClear() {  
        this.drawMyCircle();  
        Canvas.waitForCountdown(2000);  
        Canvas.clear();  
    }  
}
```

x:100

y:100

diameter:200

```
void drawMyCircle() {  
    Canvas.drawOval(  
        this.x,  
        this.y,  
        this.diameter,  
        this.diameter);  
}
```

```
void drawAndClear() {  
    drawMyCircle();  
    Canvas.waitForCountdown(2000);  
    Canvas.clear();  
}
```

自分自身のインスタンスの
インスタンス変数へのアクセスと
同様に、thisを書かなくても良い。

まとめ：インスタンスメソッド

- クラス定義（復習）
- インスタンスメソッド
 - 定義と考え方
 - インスタンス変数の参照
 - インスタンスメソッドの呼び出し

本日の例題と問題

- MyCircle2 のインスタンスメソッド作成例
 - Ex01, Ex02
- MyCircle3 のインスタンスメソッド作成例
 - Ex11, Ex12, Ex13, Ex14, Ex15
- MyRectangle のインスタンスメソッド作成例
 - Q11, Q12, Q13, Q14, Q15, Q16
- MyCar のインスタンスメソッド作成例
 - Q21, Q22, Q23, Q24, Q25
- MovingBall のインスタンスメソッド作成例
 - (Q31*), (Q32), (Q41*), (Q42)

(Ex:例題, Q:問題, *は少し手間のかかる問題)

各自に適した順番で解けばよいが、上記の順番が自然な流れとなるよう構成されている。

例題集

パッケージ「j2.lesson5」を作成する。

パッケージやクラスの作成, 実行の仕方の説明は省略する。
作り方を忘れた場合は過去のスライドや
<http://java2010.cis.k.hosei.ac.jp/01/material-01/>
を参考にせよ

■例題01

問題：次のクラスMyCircle2を作成し、動作を確認せよ。

インスタンス変数

変数の型と名前	初期値	説明
int x	無し	円のx座標
int y	無し	円のy座標
int diameter	無し	円の直径

インスタンスメソッド

返り値の型	メソッド名(引数)	機能
void	drawMyCircle()	円をインスタンス変数の値に従って描画する

(クラス名 : MyCircle2)
動作確認クラス:Sample2

例題01 (MyCircle2)

クラス
MyCircle2

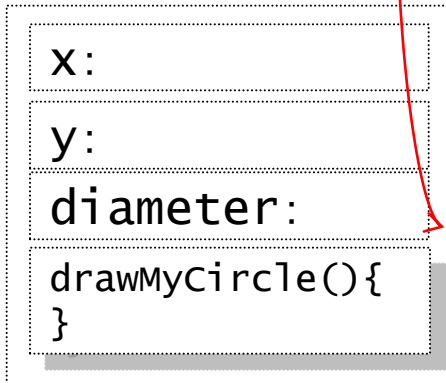
x:
y:
diameter:
drawMyCircle(){ }

```
1 package j2.lesson05;
2
3 import gpjava.Canvas;
4
5 public class MyCircle2 {
6     int x;
7     int y;
8     int diameter;
9
10    void drawMyCircle() {
11        Canvas.drawOval(this.x, this.y, this.diameter, this.diameter);
12    }
13 }
```

例題01 (Sample2)

```
1 package j2.lesson05;
2
3 import gpjava.Canvas;
4
5 public class Sample2 {
6     public static void main(String[] args) {
7         new Sample2().start();
8     }
9
10    void start() {
11        MyCircle2 c0 = createMyCircle(100, 100, 200);
12        MyCircle2 c1 = createMyCircle(200, 200, 200);
13
14        Canvas.show();
15        c0.drawMyCircle();
16        c1.drawMyCircle();
17    }
18 }
```

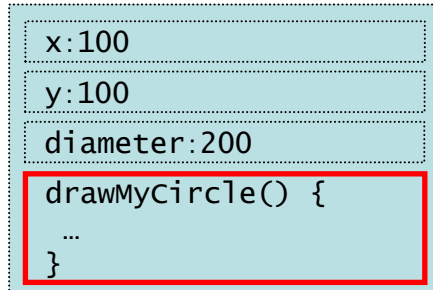
クラス
MyCircle2



createMyCircle
(100,100,200)

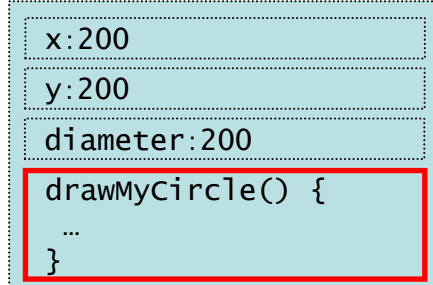


インスタンスc0

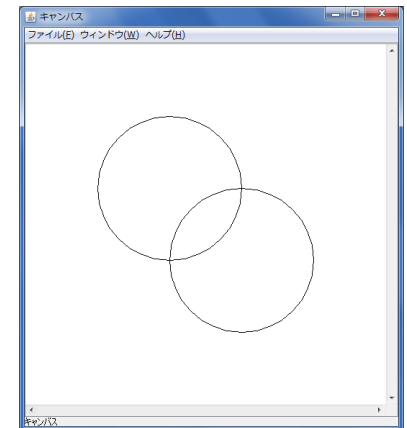


createMyCircle
(200,200,200)

インスタンスc1



c0.drawMyCircle();
c1.drawMyCircle();



■ 例題02

問題：インスタンスメソッドdrawAndClear()を作成し、動作を確認せよ。

新規MyCircle2インスタンスメソッド		
返り値の型	メソッド名(引数)	機能
void	drawAndClear()	drawMyCircle() を呼び、描画後2秒経って画面を消去する
既存インスタンスメソッド(MyCircle2内)		
void drawMyCircle()		
既存インスタンス生成メソッド(Sample2内)		
MyCircle2 createMyCircle(int x, int y, int diameter)		

※既存のMyCircle2クラスを編集する。
動作確認クラス: Sample2

例題02 (MyCircle2)

クラス
MyCircle2

x:

y:

diameter:

drawMyCircle(){
}

drawAndClear (){
}

```
1 package j2.lesson05;
2
3 import gpjava.Canvas;
4
5 public class MyCircle2 {
6     int x;
7     int y;
8     int diameter;
9
10    void drawMyCircle() {
11        Canvas.drawOval(this.x, this.y, this.diameter, this.diameter);
12    }
13
14    void drawAndClear() {
15        this.drawMyCircle();
16        Canvas.waitForCountdown(2000);
17        Canvas.clear();
18    }
19 }
```

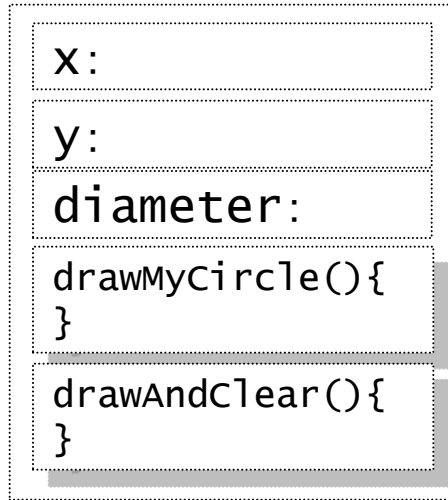
例題02(Sample3)

```
1 package j2.lesson05;
2
3 import gpjava.Canvas;
4
5 public class Sample3 {
6     public static void main(String[] args) {
7         new Sample3().start();
8     }
9
10    void start() {
11        MyCircle2 c0 = createMyCircle(100, 100, 200);
12        MyCircle2 c1 = createMyCircle(200, 200, 200);
13
14        Canvas.show();
15        c0.drawMyCircle();
16        c1.drawAndClear();
17        c1.drawMyCircle();
18    }
19
20    MyCircle2 createMyCircle(int x, int y, int diameter) {
21        MyCircle2 newObj = new MyCircle2();
22        newObj.x = x;
23        newObj.y = y;
24        newObj.diameter = diameter;
25        return newObj;
26    }
27 }
```

例題02の動作確認

クラス

MyCircle2

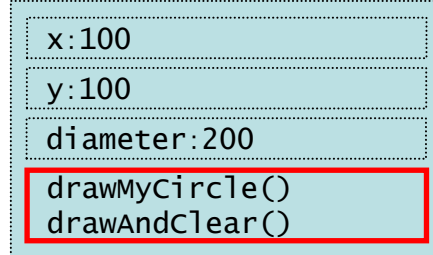


createMyCircle
(100,100,200)

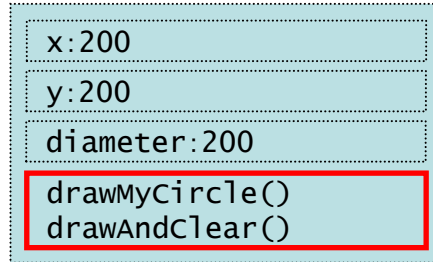
createMyCircle
(200,200,200)



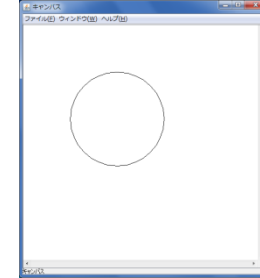
インスタンスc0



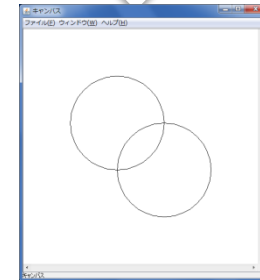
インスタンスc1



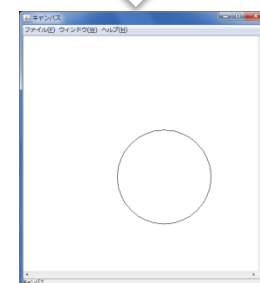
c0.drawMyCircle()



c1.drawAndClear()



c1.drawMyCircle()



■ 例題11

問題: 次のクラスMyCircle3を作成し動作を確認せよ。

インスタンス変数

変数の型と名前	初期値	説明
Int x	無し	X座標
int y	無し	Y座標
Int diameter	無し	円の直径

インスタンスメソッド

戻り値の型	メソッド名(引数)	機能
void	showFields(int row)	インスタンス変数の内容を、Spreadsheet の row 行に表示する。
void	header()	Spreadsheet の先頭行に、ヘッダとして、X座標、Y座標、直径という String を表示する。

Ex1DrawCircles クラス

戻り値の型	メソッド名(引数)	機能
MyCircle3	<code>createMyCircle(int x, int y, int diameter)</code>	MyCircle3クラスのインスタンスを作成し、各インスタンス変数にそれぞれ引数の値を代入し、そのインスタンスを返す。

(クラス名: MyCircle3)
動作確認クラス: Ex11DrawCircles

例題11

(MyCircle3)

クラス
MyCircle3

x:
y:
diameter:
showFields(row)
header()

```
1 package j2.lesson05;
2
3 import gpjava.Spreadsheet;
4
5 public class MyCircle3 {
6     int x;
7     int y;
8     int diameter;
9
10    // Ex11
11    void showFields(int row) {
12        Spreadsheet.setInt(row, 0, x);
13        Spreadsheet.setInt(row, 1, y);
14        Spreadsheet.setInt(row, 2, diameter);
15    }
16
17    void header() {
18        Spreadsheet.setString(0, 0, "X座標");
19        Spreadsheet.setString(0, 1, "Y座標");
20        Spreadsheet.setString(0, 2, "直径");
21    }
22 }
```

例題11(Ex11DrawCircles)

x:
y:
diameter:
showFields(row)
header()

createMyCircle
(100,100,200)



インスタンスc0

x:100
y:100
diameter:200

c0.header();
c0.showFields();



```
1 package j2.lesson05;
2
3 import gpjava.Spreadsheet;
4
5 public class Ex11DrawCircles {
6     public static void main(String[] args) {
7         new Ex11DrawCircles().start();
8     }
9
10    void start() {
11        MyCircle3 c0 = createMyCircle(100, 100, 200);
12        Spreadsheet.show();
13        c0.header();
14        c0.showFields(1);
15    }
16
17    MyCircle3 createMyCircle(int x, int y, int diameter) {
18        MyCircle3 newObj = new MyCircle3();
19        newObj.x = x;
20        newObj.y = y;
21        newObj.diameter = diameter;
22
23        return newObj;
24    }
25 }
```

スプレッドシート		
ファイル(E) ウィンドウ(W) ヘルプ(H)		
A	B	C
X座標	Y座標	直径
100	100	200

■ 例題12

問題: MyCircle3にインスタンスメソッドshowAreaを作成し動作を確認せよ。

MyCircleインスタンスメソッド

戻り値の型	メソッド名(引数)	機能
void	showArea(int row)	円の面積を SpreadSheet の row行に表示する。
既存クラスメソッド(MyCircle3内)		
void showFields(int row)		
void header() 面積欄を追加		

x:

y:

diameter:

showArea(row)

showFields(row)

header()

※既存のMyCircle3クラスを編集する。
動作確認クラス: Ex12DrawCircles

例題12 (MyCircle3)

円の面積 = $r^2 \times \pi$

※r:半径

x:

y:

diameter:

showArea(row)

showFields(row)

header()

```
1 package j2.lesson05;
2
3 import gpjava.Spreadsheet;
4
5 public class MyCircle3 {
6     int x;
7     int y;
8     int diameter;
9
10    // Ex11
11    void showFields(int row) {
12        Spreadsheet.setInt(row, 0, x);
13        Spreadsheet.setInt(row, 1, y);
14        Spreadsheet.setInt(row, 2, diameter);
15    }
16
17    void header() {
18        Spreadsheet.setString(0, 0, "X座標");
19        Spreadsheet.setString(0, 1, "Y座標");
20        Spreadsheet.setString(0, 2, "直径");
21        Spreadsheet.setString(0, 3, "面積");
22    }
23
24    //Ex12
25    void showArea(int row) {
26        double r = this.diameter / 2;
27        double area = Math.PI * r * r;
28        Spreadsheet.setDouble(row, 3, area);
29    }
30 }
```

Math.PIは円周率を表す

例題12 (Ex12DrawCircles)

```
1 package j2.lesson05;
2
3 import gpjava.Spreadsheet;
4
5 public class Ex12DrawCircles {
6     public static void main(String[] args) {
7         new Ex12DrawCircles().start();
8     }
9
10    void start() {
11        MyCircle3 c0 = createMyCircle(100, 100, 200);
12        Spreadsheet.show();
13        c0.header();
14        c0.showFields(1);
15        c0.showArea(1);
16    }
17
18    MyCircle3 createMyCircle(int x, int y, int diameter) {
19        MyCircle3 newObj = new MyCircle3();
20        newObj.x = x;
21        newObj.y = y;
22        newObj.diameter = diameter;
23
24        return newObj;
25    }
26 }
```

インスタンスc0

x:100

y:100

diameter:200

showArea() {
 ...
}

showFields() {}

header() {}

c0.showFields(1)
c0.showArea(1)

スプレッドシート				
ファイル(E) ウィンドウ(W) ヘルプ(H)				
A	B	C	D	E
X座標	Y座標	直径	面積	
100	100	200	31415.92653589...	

■ 例題13

問題：次のMyCircle3のインスタンスメソッド draw(), area() を作成し、Canvas, Spreadsheet を使って動作を確認せよ。

MyCircle3インスタンスメソッド

戻り値の型	メソッド名(引数)	機能
void	draw()	キャンバスに円を描画する。
double	area()	円の面積を計算し、返す
既存MyCircleインスタンスメソッド		
void showArea(int row)		
void showFields(int row)		
void header()		

※既存のMyCircle3クラスを編集する。
動作確認クラス: Ex13DrawCircles

例題13

(MyCircle3)

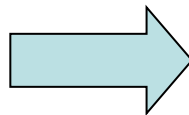
Ex11, 12
と同じ

```
1 package j2.lesson05;
2
3 import gpjava.Spreadsheet;
4 import gpjava.Canvas;
5
6 public class MyCircle3 {
7     int x;
8     int y;
9     int diameter;
10
11     // Ex11
12     void showFields(int row) {...}
17
18     void header() {...}
24
25     //Ex12
26     void showArea(int row) {...}
31
32     //Ex13
33     double area() {
34         double r = this.diameter / 2;
35         double area = Math.PI * r * r;
36         return area;
37     }
38
39     void draw() {
40         Canvas.drawOval(x, y, diameter, diameter);
41     }
42 }
```

補足：例題12で作成したshowAreaメソッドは、areaメソッドを利用することができる。修正して、同じ動作になるか確認せよ。

showAreaメソッド

```
//Ex12
void showArea(int row) {
    double r = this.diameter / 2;
    double area = Math.PI * r * r;
    Spreadsheet.setDouble(row, 3, area);
}
```



```
void showArea(int row) {
    double area = this.area();
    Spreadsheet.setDouble(row, 3, area);
}
```

あるいは

```
void showArea(int row) {
    double area = area();
    Spreadsheet.setDouble(row, 3, area);
}
```

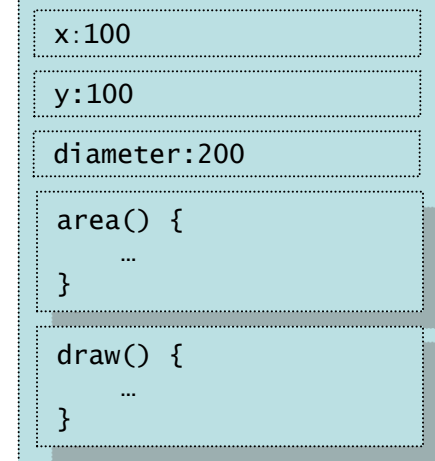
areaメソッド

```
//Ex13
double area() {
    double r = this.diameter / 2;
    double area = Math.PI * r * r;
    return area;
}
```

例題13 (Ex13DrawCircles)

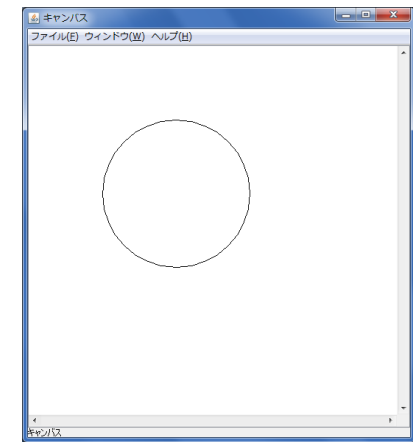
```
1 package j2.lesson05;
2
3 import gpjava.Canvas;
4 import gpjava.Spreadsheet;
5
6 public class Ex13DrawCircles {
7     public static void main(String[] args) {
8         new Ex13DrawCircles().start();
9     }
10
11 void start() {
12     MyCircle3 c0 = createMyCircle(100, 100, 200);
13     Spreadsheet.show();
14
15     c0.header();
16     c0.showFields(1);
17
18     double area = c0.area();
19     Spreadsheet.setDouble(1, 3, area);
20
21     Canvas.show();
22     c0.draw();
23 }
24
25 MyCircle3 createMyCircle(int x, int y, int diameter) {
26     MyCircle3 newObj = new MyCircle3();
27     newObj.x = x;
28     newObj.y = y;
29     newObj.diameter = diameter;
30
31     return newObj;
32 }
33 }
```

インスタンスc0



The screenshot shows the Spreadsheet application window with the following data:

A	B	C	D	E
X座標	Y座標	直径	面積	
100	100	200	31415.92653589...	



■ 例題14

問題:円の面積を変えるMyCircle3のインスタンスメソッド
setArea(double area)を作成し動作を確認せよ。

新規MyCircle3インスタンスメソッド

返り値の型	メソッド名(引数)	機能
void	setArea(double area)	MyCircle3オブジェクトの面積の値がareaになるようにする。 (つまりdiameterの値を変える)
既存MyCircleインスタンスメソッド		
void draw()		
double area()		
void showArea(int row)		
void showFields(int row)		
void header()		

※既存のMyCircle3クラスを編集する。
動作確認クラス: Ex14DrawCircles

例題14 (MyCircle3)

Ex11, 12, 13
と同じ

円の面積を100にしたければ
面積100に対応する直径の値
を新たなインスタンス変数の値
として代入する。

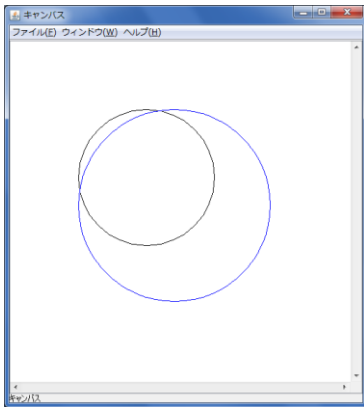
$$r = \sqrt{\frac{area}{\pi}}$$

※rは半径

```
1 package j2.lesson05;
2
3 import gpjava.Spreadsheet;
4 import gpjava.Canvas;
5
6 public class MyCircle3 {
7     int x;
8     int y;
9     int diameter;
10
11     // Ex11
12     void showFields(int row) {
13
14
15
16
17
18     void header() {
19
20
21
22
23
24
25     //Ex12
26     void showArea(int row) {
27
28
29
30
31
32     //Ex13
33     double area() {
34
35
36
37
38
39     void draw() {
40
41
42
43     //Ex14
44     void setArea(double area) {
45         double r = Math.sqrt(area / Math.PI);
46         this.diameter = (int)(r * 2);
47     }
48 }
```

例題14 (Ex14DrawCircles)

黒い円: 直径200の円
青い円: 黒い円の面積2倍の円
(`c0.setArea(area*2)`適用)



スプレッドシート				
ファイル(E) ウィンドウ(W) ヘルプ(H)				
A	B	C	D	E
X座標	Y座標	直径	面積	
100	100	200	<u>31415.926535897932</u>	
100	100	<u>282</u>	<u>62458.00354601867</u>	

```
1 package j2.lesson05;
2
3 import gpjava.Canvas;
4 import gpjava.Spreadsheet;
5
6 public class Ex14DrawCircles {
7     public static void main(String[] args) {
8         new Ex14DrawCircles().start();
9     }
10
11     void start() {
12         MyCircle3 c0 = createMyCircle(100, 100, 200);
13         Spreadsheet.show();
14         c0.header();
15         c0.showFields(1);
16         c0.showArea(1);
17
18         Canvas.show();
19         c0.draw();
20         Canvas.waitForCountdown(2000);
21
22         double area = c0.area();
23         c0.setArea(area * 2);
24         c0.showFields(2);
25         c0.showArea(2);
26
27         Canvas.setColor(0, 0, 255);
28         c0.draw();
29     }
30
31     MyCircle3 createMyCircle(int x, int y, int diameter) {
32         MyCircle3 newObj = new MyCircle3();
33         newObj.x = x;
34         newObj.y = y;
35         newObj.diameter = diameter;
36         return newObj;
37     }
38 }
```

■ 例題15

問題:面積を(ほぼ)factor倍するメソッドexpandAreaBy(double factor)を作成し動作を確認せよ。

新規MyCircle3インスタンスメソッド

返り値の型	メソッド名(引数)	機能
void	expandBy(double factor)	MyCircle3オブジェクトの面積がfactor倍になるようにdiameterの値を算出し、代入する。
既存MyCircleインスタンスメソッド		
void setArea(double area)		
void draw()		
double area()		
void showArea(int row)		
void showFields(int row)		
void header()		

※既存のMyCircle3クラスを編集する。
動作確認クラス: Ex15DrawCircles

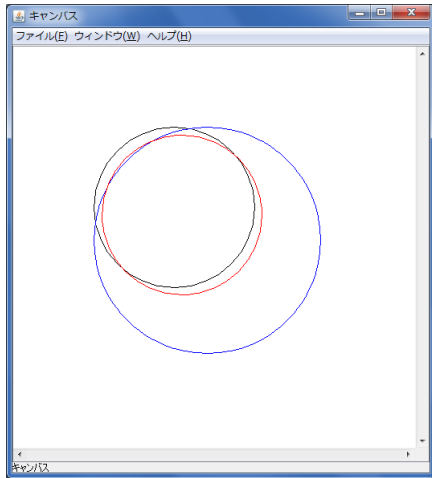
例題15 (MyCircle3)

Ex11, 12, 13, 14
と同じ

円の面積をn倍するた
めには直径を何倍にす
ればいいのか？

```
1 package j2.lesson05;
2
3 import gpjava.Spreadsheet;
4 import gpjava.Canvas;
5
6 public class MyCircle3 {
7     int x;
8     int y;
9     int diameter;
10
11     // Ex11
12     void showFields(int row) {}
13
14     void header() {}
15
16     //Ex12
17     void showArea(int row) {}
18
19     //Ex13
20     double area() {}
21
22     void draw() {}
23
24     //Ex14
25     void setArea(double area) {}
26
27     // Ex15
28     void expandBy(double factor) {
29         this.diameter = (int)(this.diameter * Math.sqrt(factor));
30     }
31 }
```

例題15 (Ex15DrawCircles)



黒い円: 直径200の円
青い円: 黒い円の面積2倍の円
(`c0.setArea(area*2)`適用)
赤い円: 青い円の面積0.5倍の円
(`c0.expandBy(0.5)`適用)

スプレッドシート			
ファイル(E) ウィンドウ(W) ヘルプ(H)			
A	B	C	D
X座標	Y座標	直径	面積
100	100	<u>200</u>	31415.926535897932
100	100	<u>282</u>	62458.00354601867
110	110	<u>199</u>	30790.749597833565

```
1 package j2.lesson05;
2
3 import gpjava.Canvas;
4 import gpjava.Spreadsheet;
5
6 public class Ex15DrawCircles {
7     public static void main(String[] args) {
8         new Ex15DrawCircles().start();
9     }
10
11     void start() {
12         MyCircle3 c0 = createMyCircle(100, 100, 200);
13         Spreadsheet.show();
14         c0.header();
15         c0.showFields(1);
16         c0.showArea(1);
17
18         Canvas.show();
19         c0.draw();
20         Canvas.waitForCountdown(2000);
21
22         double area = c0.area();
23         c0.setArea(area * 2);
24         c0.showFields(2);
25         c0.showArea(2);
26         Canvas.setColor(0, 0, 255);
27         c0.draw();
28
29         c0.expandBy(0.5);
30         c0.x += 10;
31         c0.y += 10;
32         c0.showFields(3);
33         c0.showArea(3);
34
35         Canvas.setColor(255, 0, 0);
36         c0.draw();
37     }
38
39     MyCircle3 createMyCircle(int x, int y, int diameter) {
```

Ex14と同じ