プログラミング入門2

第11回　インスタンスと型

講義資料について

- 新しい言語の機能(オブジェクト指向の機構)については、随時参考書などを参照するのがよい。
- 過去の資料も参考になる。
  - http://java2005.cis.k.hosei.ac.jp/
  - 今回の範囲は、上記ページの18回に詳しい。

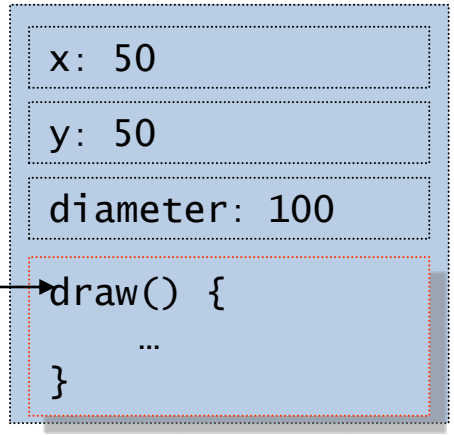
復習（インスタンスメソッドとクラスメソッド）

インスタンスは、インスタンス変数、インスタンスメソッドを保持できる。

```
public class MyCircle {
  int x = 50;
  int y = 50;
  int diameter = 100;

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

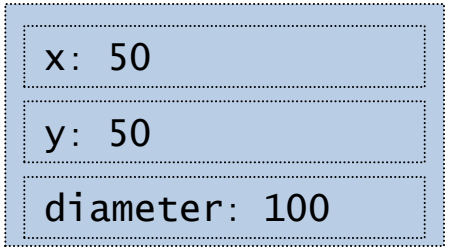
インスタンスメソッド

MyCircleインスタンス



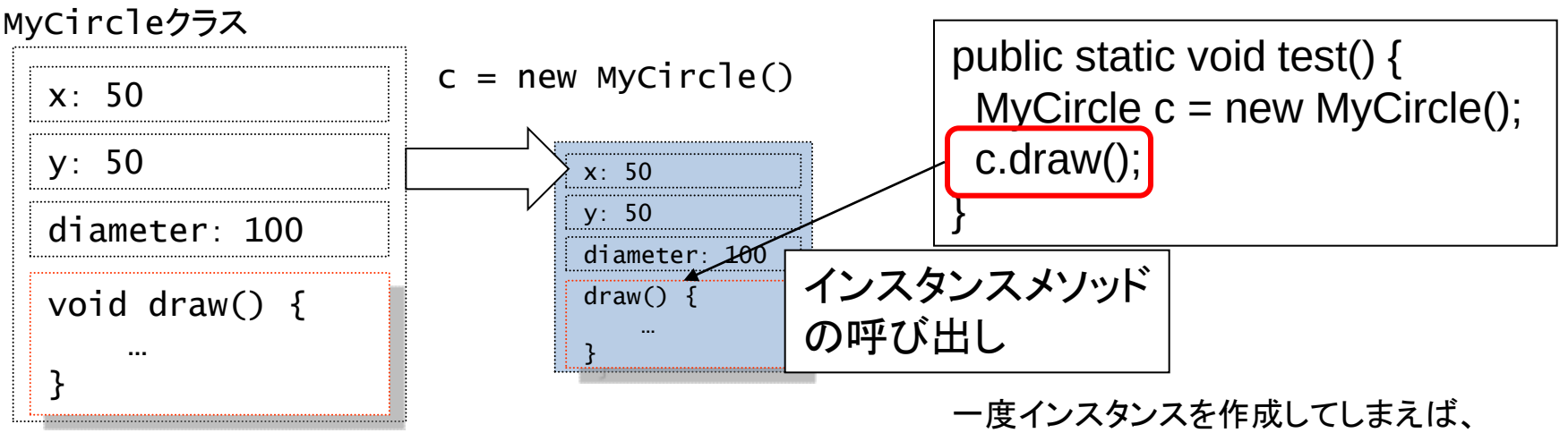

```
public class MyCircle2 {
  int x = 50;
  int y = 50;
  int diameter = 100;

  public static void draw(MyCircle2 c) {
      Canvas.drawOval(c.x, c.y, c.diameter,
                                c.diameter);
  }
}
```

MyCircle2インスタンス

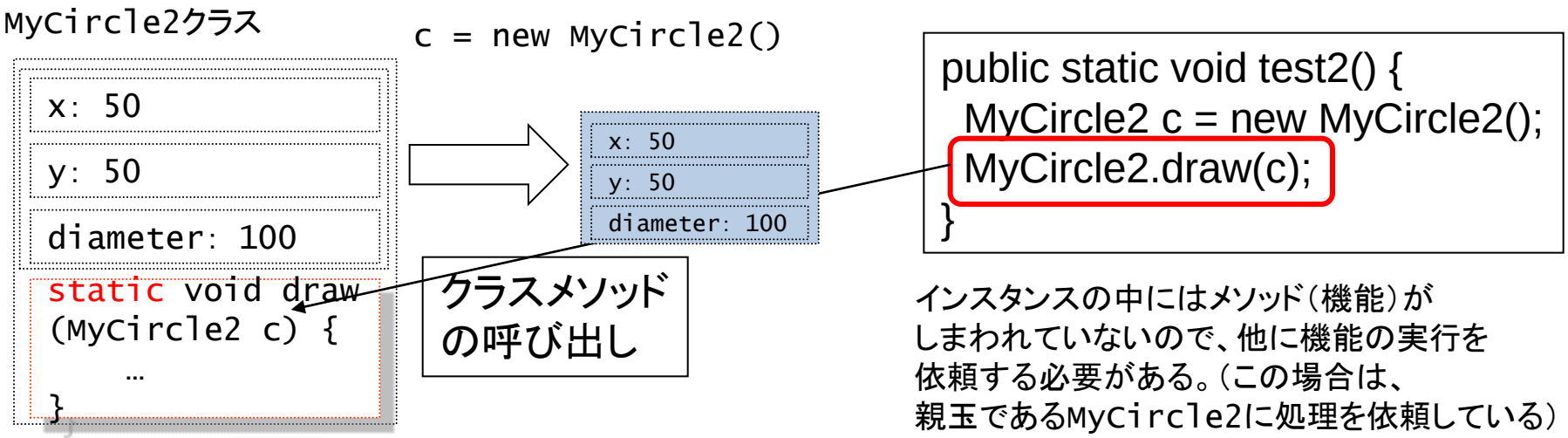


クラスメソッドは、インスタンスには含まれない。

復習（インスタンスメソッドとクラスメソッド）



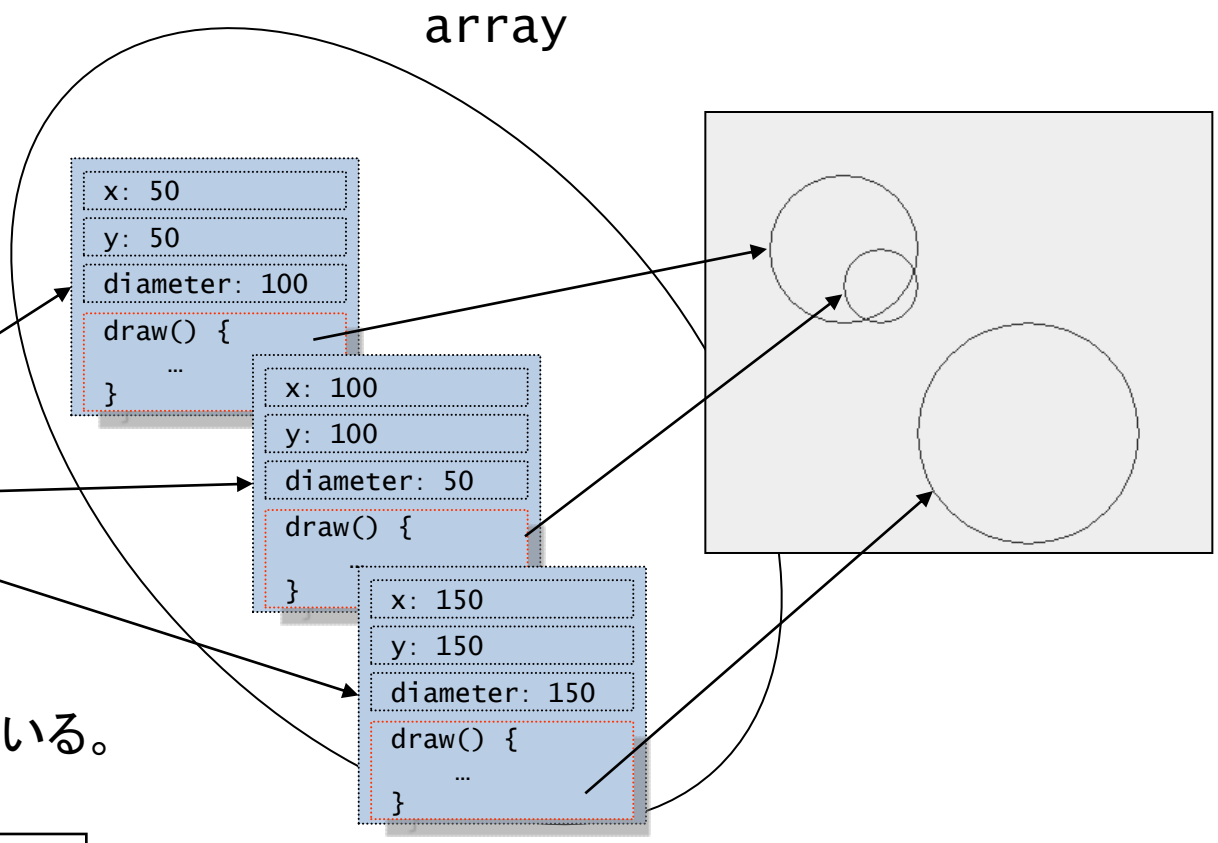
一度インスタンスを作成してしまえば、
インスタンスにメソッド（機能）がしまわれているので、直接的に機能を起動できる。



インスタンスの中にはメソッド（機能）がしまわれていないので、他に機能の実行を依頼する必要がある。（この場合は、親玉であるMyCircle2に処理を依頼している）

インスタンスメソッドの実行（例）

```
public static void test3() {
  MyCircle [] array = {
       new MyCircle(50, 50, 100),
       new MyCircle(100,100,50),
       new MyCircle(150,150,150)
  };
  for (int i = 0; i < array.length; i++){
    MyCircle aFigure = array[i];
    aFigure.draw();
  }
}
```



(1)それぞれのインスタンスに drawメソッドの実行を依頼している。

インスタンスaFigureにdraw()というメッセージ（すなわち「drawの機能を実行して下さい」という依頼）を送っていると考えられる。（メッセージ送信）

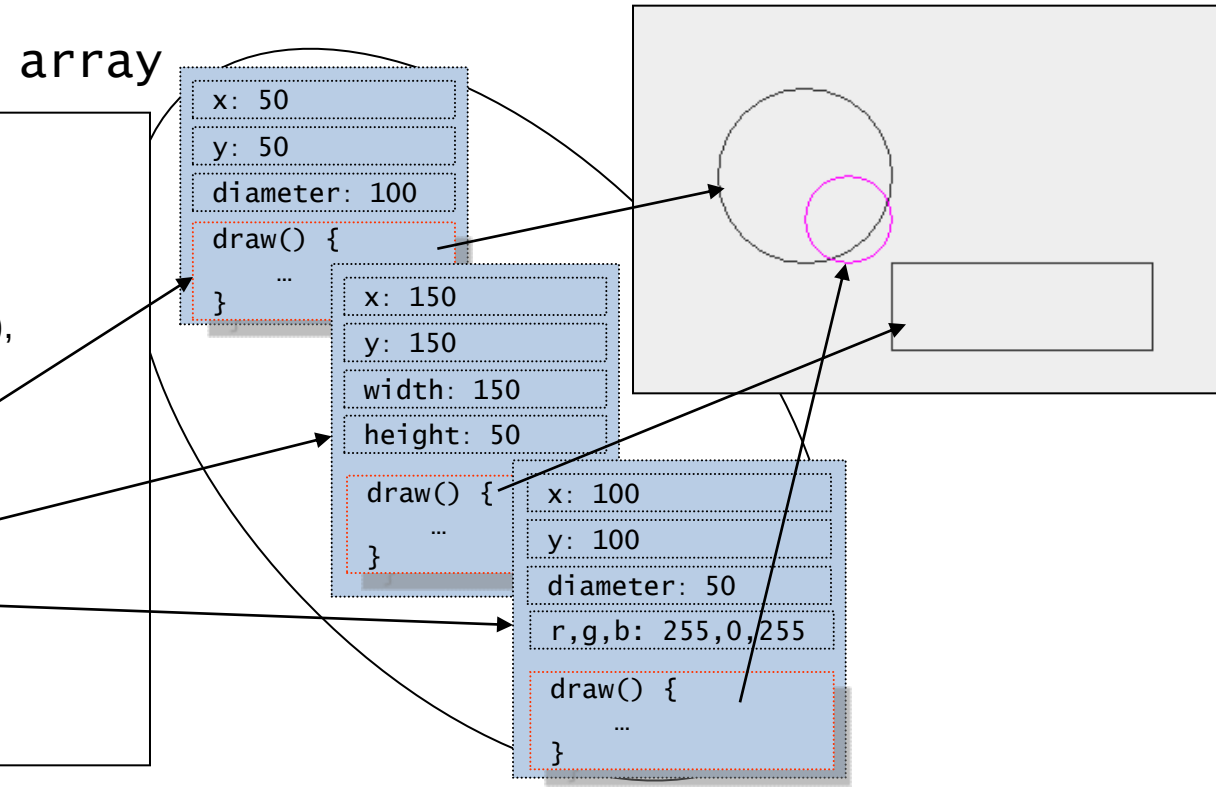
(2)各インスタンスは、drawメソッドを実行し、自分自身（円）をキャンバス上に表示させる。

テーマ：インスタンスと型

- インスタンスと型
- 継承の利用
- 抽象クラス（概要）

インスタンスメソッドの活用（本日の概要）

```
test () {
// これは擬似コード
  array = {
       new MyCircle(50, 50, 100),
       new MyRectangle(150,150,150,50),
       new ColoredCircle(100,100,50,
                                255, 0, 255);
  };

  for (int i = 0; i < array.length; i++){
    aFigure = array[i];
     aFigure.draw();
  }
}
```



(1)それぞれのインスタンスに
drawメソッドの実行を依頼している。

(2)各インスタンスは、自分が保持する
drawメソッドを実行し、自分自身（図形）を
キャンバス上に表示させる。

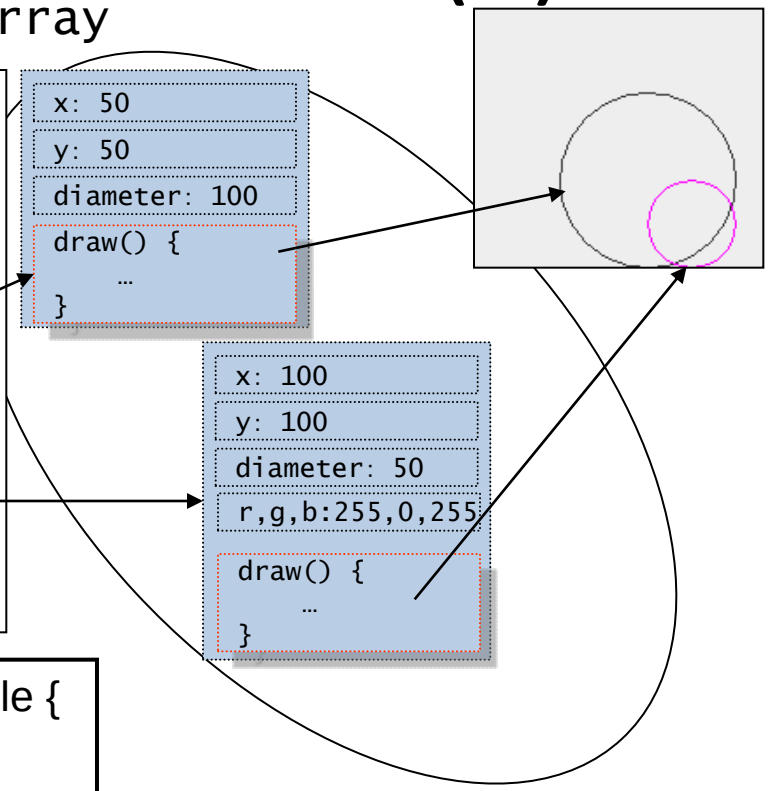
インスタンスメソッドを用いると、種類の違うインスタンスに対して同じメソッド名（ここではdraw）で機能を呼び出すことが可能となる。
このような性質をポリモーフィズムと呼ぶ。Javaでは、継承を利用する。
（インタフェースという機構を利用する方法もあるが、今回は触れない）

テーマ：インスタンスと型

- インスタンスと型
- 継承の利用
- 抽象クラス（概要）

インスタンスと型(1)

```
public static void test4() {
// Xの型を設定しなければ動作しない
 X [] array = {
      new MyCircle(50, 50, 100),
      new ColoredCircle(100,100,50, 255,0,255);
 };

 for (int i = 0; i < array.length; i++){
  X aFigure = array[i];
   aFigure.draw();
 }
}
```




```
public class ColoredCircle extends MyCircle {
   int  red = 255;
   int  green = 0;
   int  blue = 255;
   public void draw() {
      Canvas.setColor(red, green, blue);
      super.draw();
   }
}
```

（コンストラクタは省略）

Xはどういう型にすべきか？

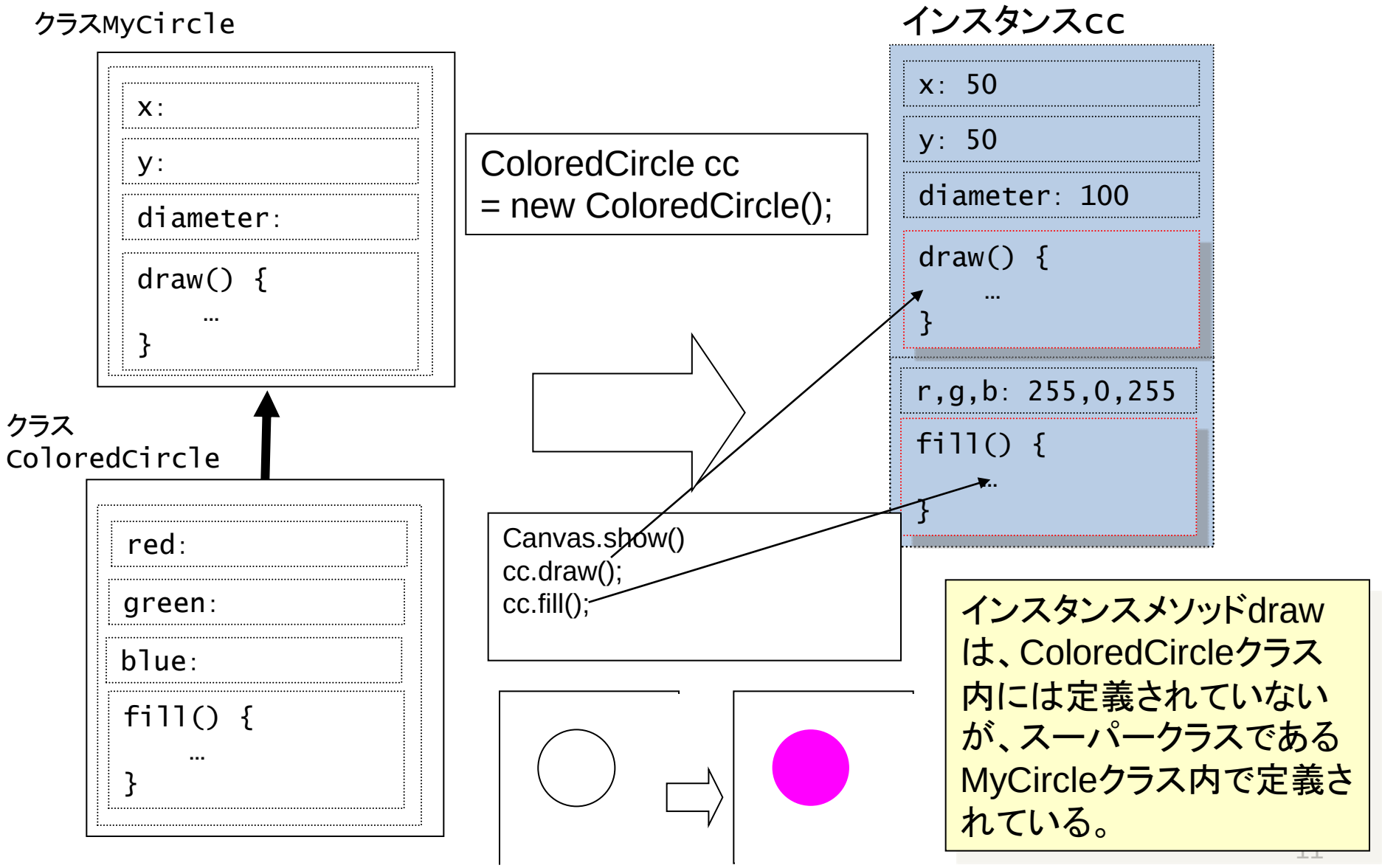
ここで復習：継承（定義の例）

```
public class 新しいクラス名 extends もとにするクラス {
   新たに加わるインスタンス変数、あるいは、メソッドの定義
        …..
}
```

```
public class ColoredCircle extends MyCircle {
  int red = 255;
  int green = 0;
  int blue = 255;

  public void fill() {
    Canvas.setColor(red, green, blue);
    Canvas.fillOval(x, y, diameter, diameter);
  }
}
```

継承を利用することで、似た形、機能を持つインスタンスを簡単に作成できるようになる。

復習：メソッドの起動


クラスMyCircle
x:
y:
diameter:
draw() {
…
}
クラス
ColoredCircle
red:
green:
blue:
fill() {
…
}
ColoredCircle cc
= new ColoredCircle();
インスタンスcc
x: 50
y: 50
diameter: 100
draw() {
…
}
r,g,b: 255,0,255
fill() {
…
}
Canvas.show()
cc.draw();
cc.fill();
インスタンスメソッドdrawは、ColoredCircleクラス内には定義されていないが、スーパークラスであるMyCircleクラス内で定義されている。

復習: メソッドのオーバーライド

```
public class MyCircle {
  int x = 50;
  int y = 50;
  int diameter = 100;

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

スーパークラスで定義されているものと同じ名前、引数の数、型を持つメソッドを、サブクラスで定義することを、スーパークラスのメソッドを「**オーバーライド**する」と言う。

これを用いることで、スーパークラスで実装されている機能を、新たにサブクラスで実装し直すことができる。

```
public class ColoredCircle extends MyCircle {
  int red = 255;
  int green = 0;
  int blue = 255;
  void fill() {
    Canvas.setColor(red, green, blue);
    Canvas.fillOval(x, y, diameter, diameter);
  }
  void draw() {
      Canvas.setColor(red, green, blue);
      Canvas.drawOval(x, y, diameter, diameter);
  }
}
```

MyCircleクラスのdrawメソッドをオーバーライドしている。
このメソッドは、色付きで円をキャンバスで表示する。

復習: オーバーライドしたメソッドの起動

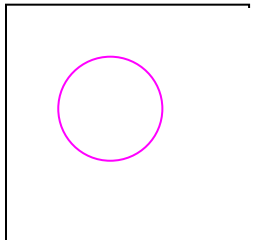
```
x: 50
y: 50
diameter: 100
draw() {
    drawOval(x, y, diameter, diameter)
}
```

```
red:
green:
blue:
fill() {
    …
}
```

```
draw() {
  setColor(red, green, blue);
  drawOval(x, y, diameter, diameter);
}
```

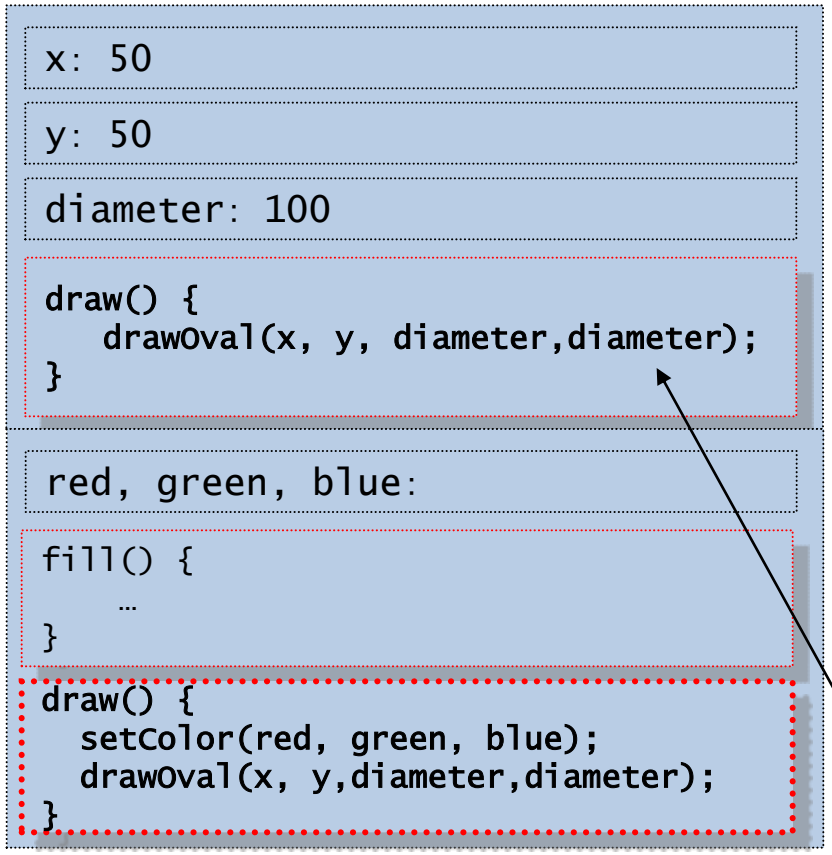
ColoredCircleのdrawインスタンスメソッドが、スーパークラスで定義されているdrawメソッドに代わって起動される。

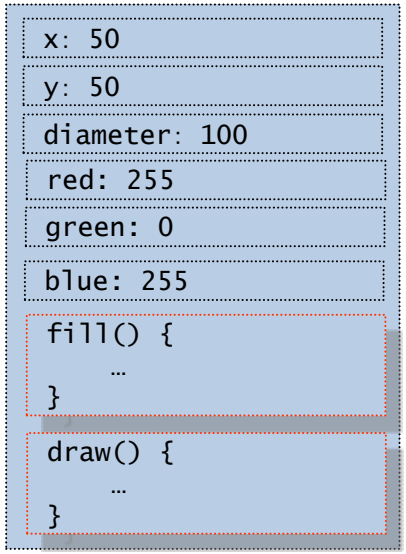
ColoredCircleクラスのdrawメソッドでは、色を指定できるよう実装し直されている。



```
ColoredCircle cc = new ColoredCircle();
cc.draw();
```

復習はここまで



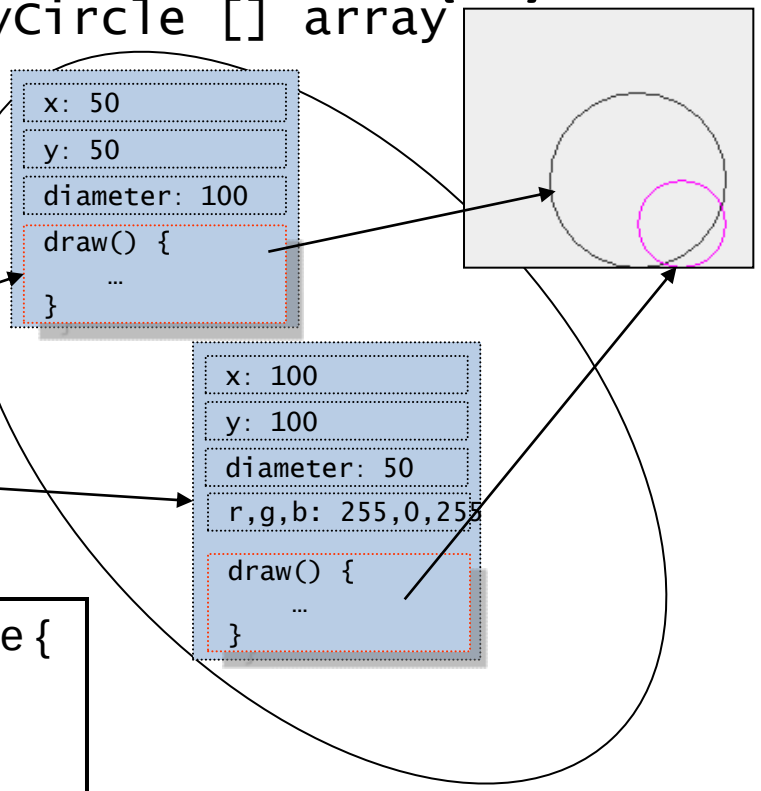


以降のインスタンスの図では、オーバーライドされたメソッドは省略する。

右の図は左の図にあるdraw()を省略した形になっている。

インスタンスと型(2)

```
public static void test4() {
  MyCircle [] array = {
      new MyCircle(50, 50, 100),
      new ColoredCircle(100,100,50,
              255, 0, 255)
  };
  for (int i = 0; i < array.length; i++){
    MyCircle aFigure = array[i];
    aFigure.draw();
  }
}
```




```
public class ColoredCircle extends MyCircle {
   int red = 255;
   int green = 0;
   int blue = 255;
   public void draw() {
      Canvas.setColor(red, green, blue);
      super.draw();  // super呼び出し(前回内容)
   }
}
```

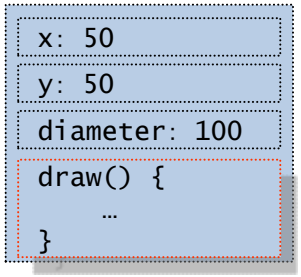
（コンストラクタは省略）

X = MyCircleとする。

MyCircleの配列として宣言されたarrayに、ColoredCircleインスタンスが格納されているが、このプログラムは正しく動作する。

インスタンスと型(3)

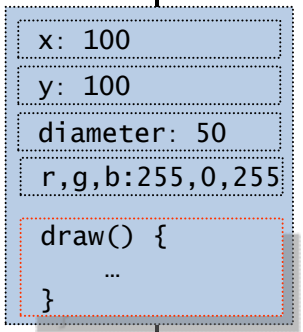
```
public class MyCircle {
   int x;
   int y;
   int diameter;

   public void draw(){
   …
  }
}
```



MyCircleのインスタンスは、
MyCircle型を持つ

ポイント:ColoredCircleのインスタンスは、ColoredCircle型だけではなく、スーパークラスの型であるMyCircle型も同時に持つ。

```
public class ColoredCircle extends MyCircle {
   int red = 255;
   int green = 0;
   int bue = 255;
   public void draw() {
      Canvas.setColor(red,green, blue);
      super.draw();
   }
}
```



ColoredCircleのインスタンスは、
ColoredCircle型および
MyCircle型を持つ

例えば次の両者が可能

```
ColoredCircle c0 = new ColoredCircle(100,100,50,255,0,255);
```

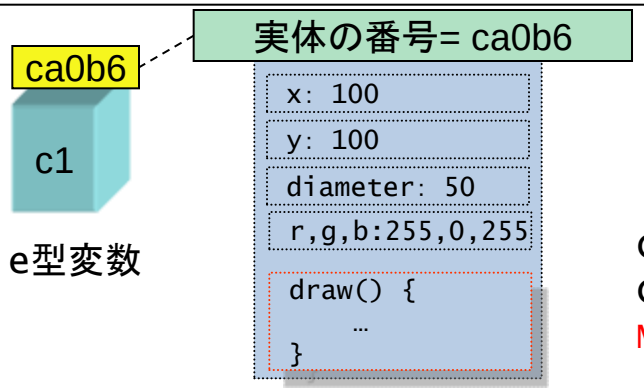
```
MyCircle c1 = new ColoredCircle(100,100,50,255,0,255);
```

インスタンスと型(4)

```
MyCircle c1 = new ColoredCircle(100,100,50,255,0,255);
c1 = new MyCircle(50, 50, 100);
```

同じ変数c1に異なる
種類のインスタンス
を割り当てることができる
点に注目

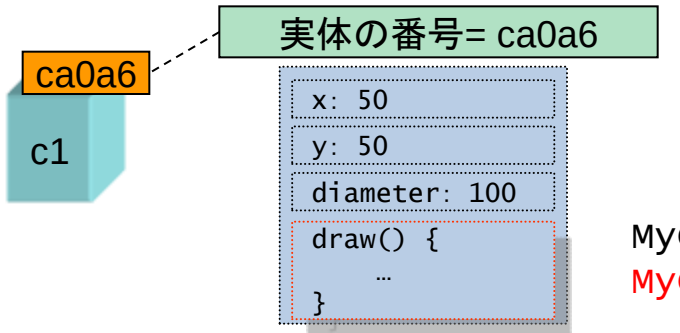
```
MyCircle c1 = new ColoredCircle(100,100,50,255,0,255);
```



MyCircle型変数

ColoredCircleのインスタンスは、
ColoredCircle型および
MyCircle型を持つ

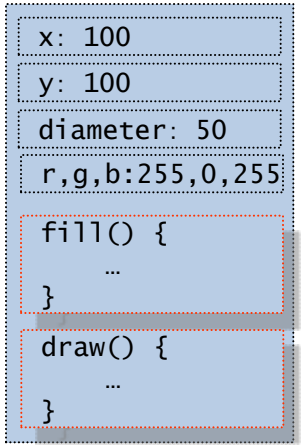
```
c1 = new MyCircle(50, 50, 100);
```



MyCircleのインスタンスは、
MyCircle型を持つ

インスタンスと型(注意)

```
public class ColoredCircle extends MyCircle {
   int red = 155;
   int greenn = 0;
   int blue = 255;
   public void draw() {
      Canvas.setColor(red, green, blue);
      super.draw();
   }
  public void fill() {
    Canvas.setColor(255, 0, 255);
    Canvas.fillOval(x, y, diameter, diameter);
  }
}
```



新規のメソッド(オーバーライドの形ではない)fill

```
ColoredCircle c0 = new ColoredCircle(100,100,50,255,0,255);
c0.fill(); // OK
```

```
MyCircle c1 = new ColoredCircle(100,100,50,255,0,255);
c1.fill(); // NG …
```

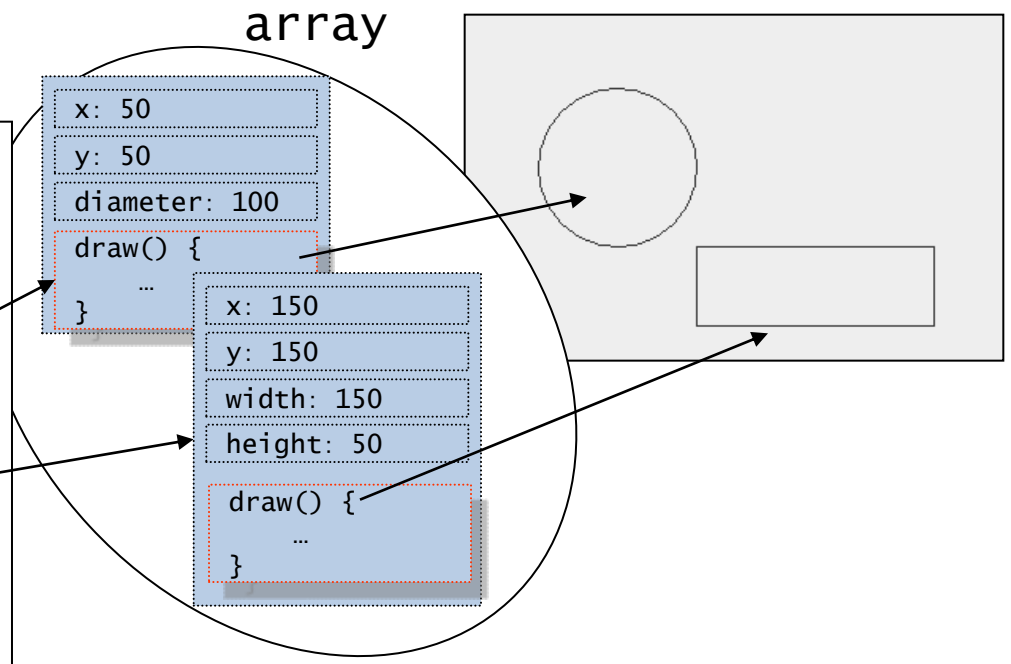
下記の場合MyCircleクラスのインスタンスメソッドとしてはfillが定義されていないので、この呼び出しは認められない。
可能なように見えるが、Java(のコンパイラ)は，c1には、もしかするとMyCircleのインスタンスが割り当てられるかもしれない、と考える。

テーマ：インスタンスと型

- インスタンスと型
- 継承の利用
- 抽象クラス（概要）

継承の利用（共通のスーパークラス）

```
test() {
// これは擬似コード
X [] array = {
      new MyCircle(50, 50, 100),
      new MyRectangle(150,150,150,50);
  };

  for (int i = 0; i < array.length; i++){
    X aFigure = array[i];
    aFigure.draw();
  }
}
```



Xはどういう型にすべきか？

MyCircleとMyRectangleは継承関係にはないので、スライド15の方法でそれぞれのインスタンスを同じ配列に格納することは出来ない。

継承の利用（共通のスーパークラス）

```
public class Figure {
  public void draw() {
    System.out.println("描画できません。");
  }
}
```

共通のスーパークラスを設定する。

```
public class MyCircle extends Figure {
  int x;
  int y;
  int diameter;

  public void draw(){
  …
 }
}
```

```
x: 50
y: 50
diameter: 100
draw() {
    …
}
```

MyCircleのインスタンスは、
MyCircle型および
Figure型を持つ

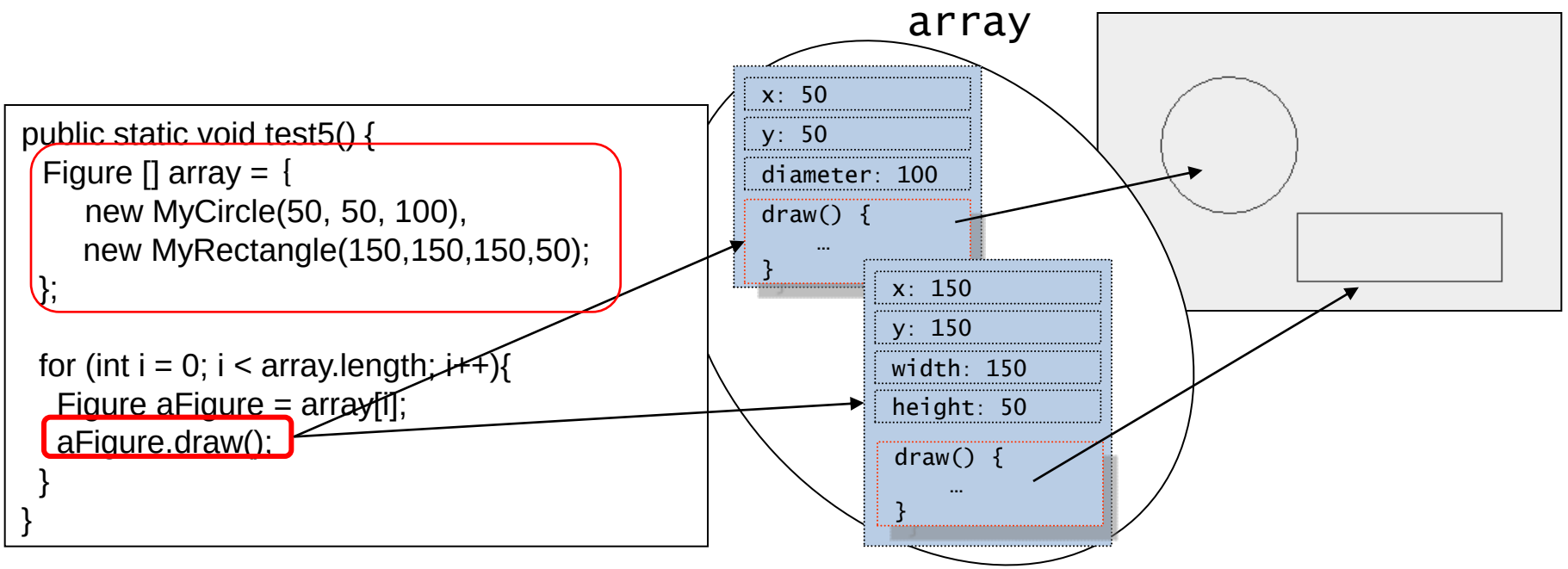
```
public class MyRectangle extends Figure {
  int x;
  int y;
  int width;
  int height;
  public void draw(){
  …
 }
}
```

```
x: 150
y: 150
width: 150
height: 50
draw() {
    …
}
```

MyRectangleのインスタンスは、
MyRectangle型および
Figure型を持つ

両方とも共通の型Figureを持つ

継承の利用(共通のスーパークラス)



MyCircleおよびMyRectangleのインスタンスは、ともに共通の型Figureを持つので、Figure型の配列に格納できる。

```
public class Figure {
  public void draw() {
     System.out.println("描画できません。");
   }
}
```

```
public class MyCircle extends Figure {
   int x;
   int y;
   int diameter;

   public void draw(){
   …
  }
}
```

```
x: 50
y: 50
diameter: 100
draw() {
    …
}
```

```
public class MyRectangle extends Figure {
   int x;
   int y;
   int width;
   int height;
   public void draw(){
   …
  }
}
```

```
x: 150
y: 150
width: 150
height: 50
draw() {
    …
}
```

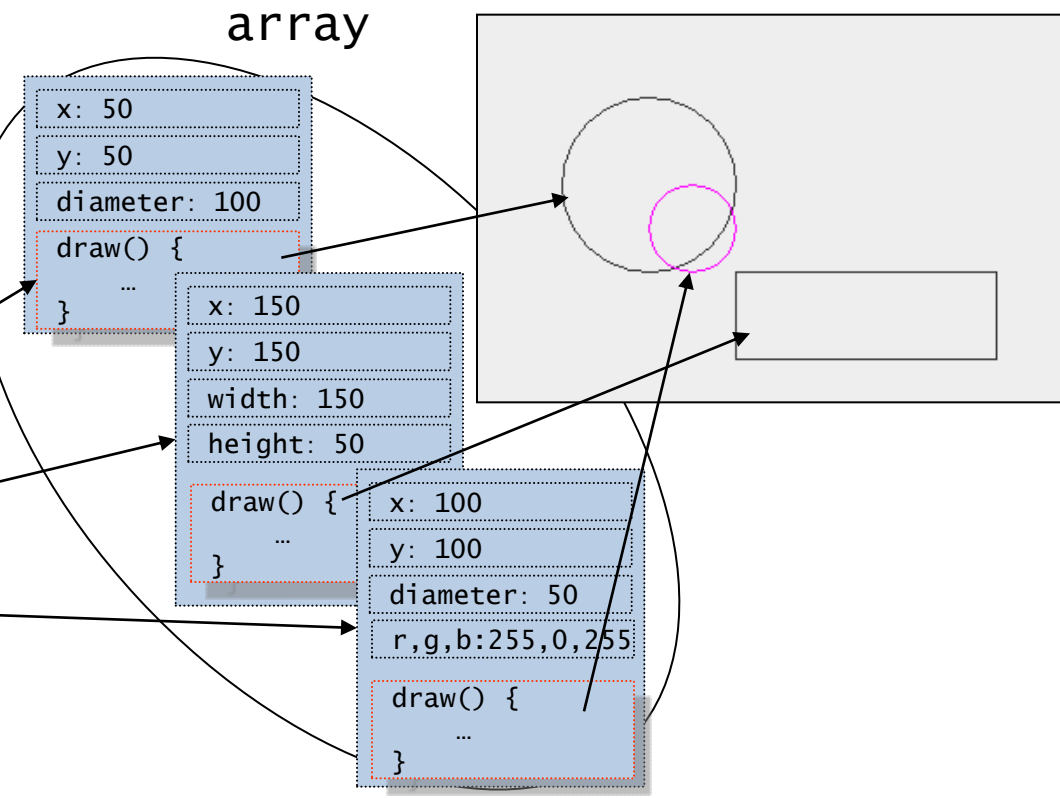
```
public class ColoredCircle extends MyCircle {
   int r;
   int g;
   int b;
    public void draw() {
      …
  }
}
```

```
x: 100
y: 100
diameter: 50
r,g,b:255,0,255
draw() {
    …
}
```

ColoredCircleのインスタンスは、
ColoredCircle型および
MyCircle型および
Figure型を持つ

最終形

```
public static void test6() {

Figure [] array = {
      new MyCircle(50, 50, 100),
      new MyRectangle(150,150,150,50),
      new ColoredCircle(100,100,50,255,0,255)
  };

  for (int i = 0; i < array.length; i++){
    Figure aFigure = array[i];
    aFigure.draw();
  }
}
```



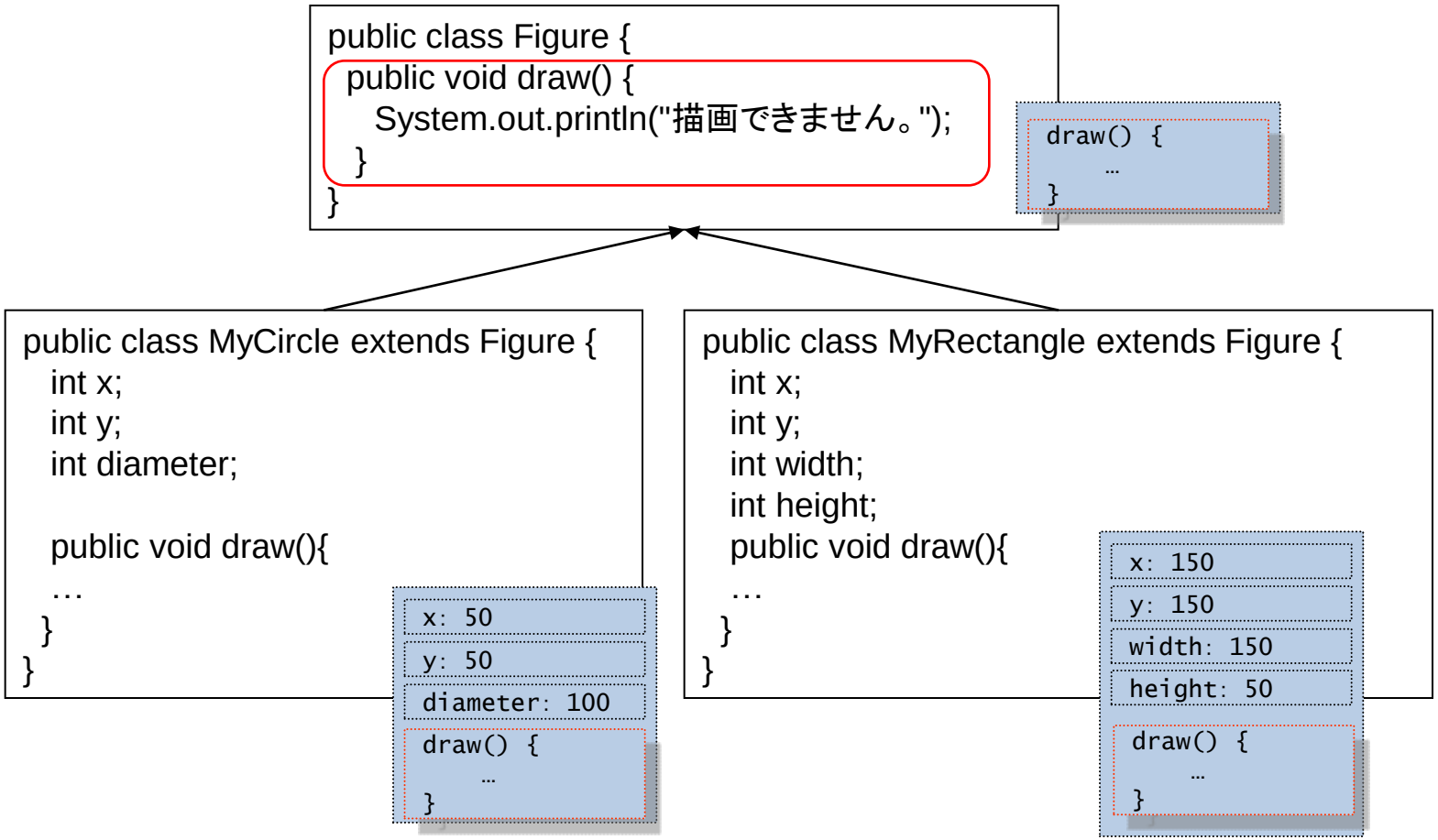
(1)それぞれのインスタンスに
drawメソッドの実行を依頼している。

(2)各インスタンスは、自分が保持する
drawメソッドを実行し、自分自身(図形)を
キャンバス上に表示させる。

テーマ：インスタンスと型

- インスタンスと型
- 継承の利用
- 抽象クラス（概要のみ）

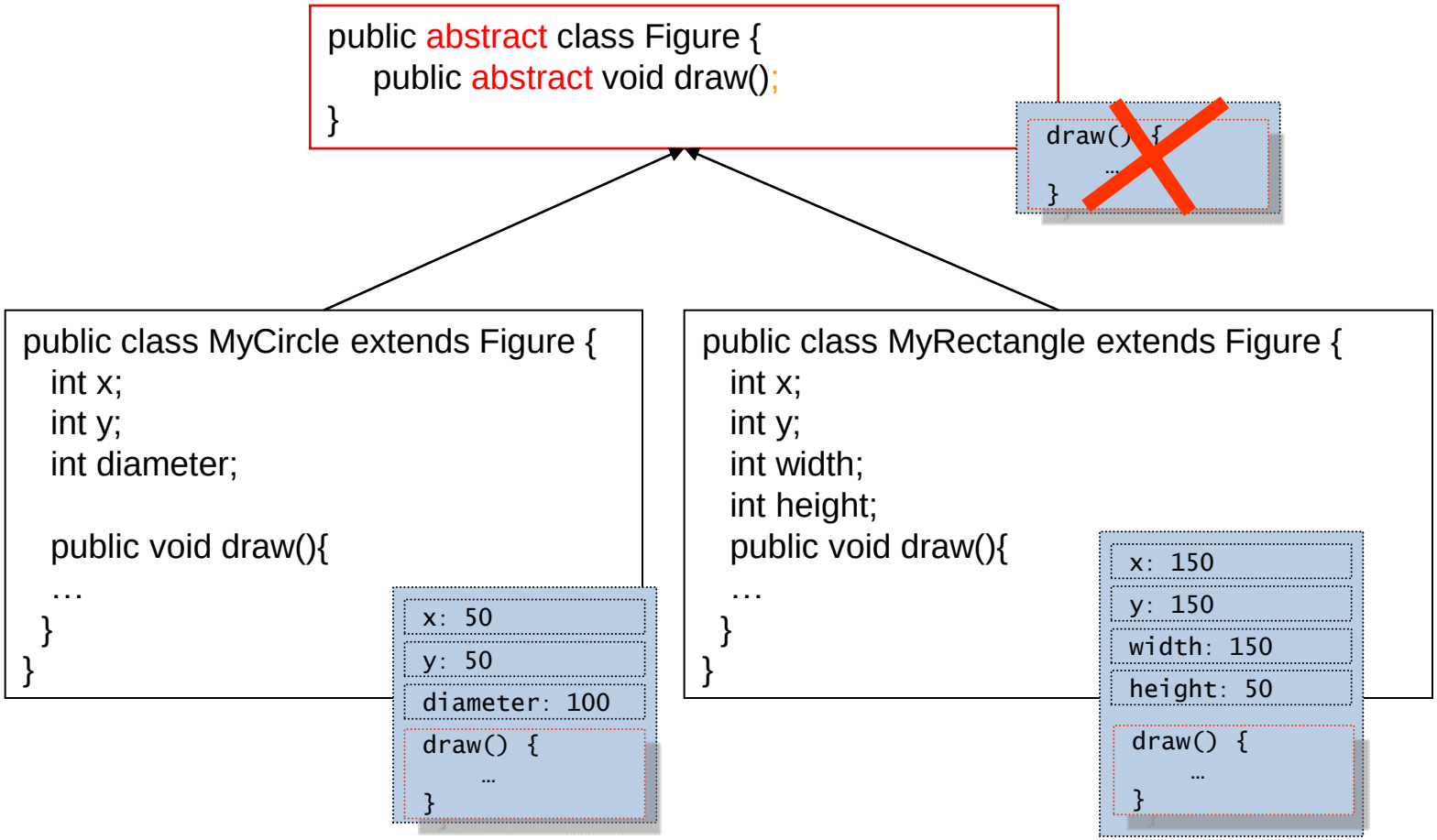
抽象クラス（概要）(1)



Figureクラスのインスタンスは、（draw()メソッドには意味がないので）作成しても機能をなさない。このクラスは「図形」という抽象的な概念のみを表し、他の具体的な図形のクラスとそのインスタンスを整理するためのみに用いたい。

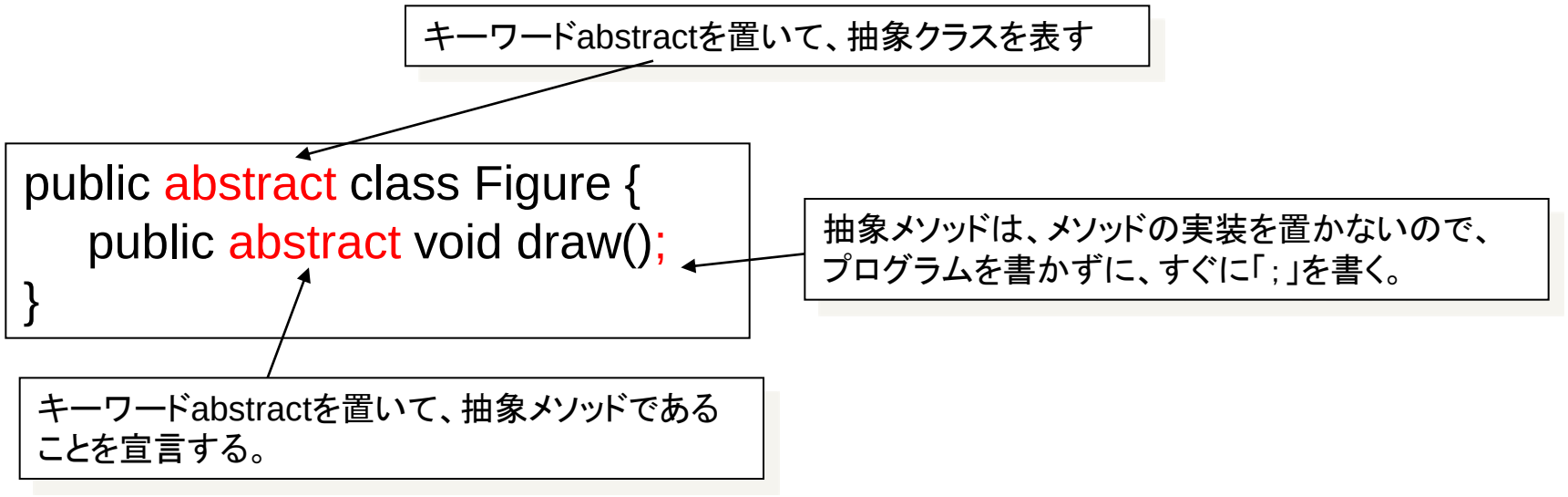
→インスタンスを生成させないクラス（抽象クラス）を利用する。

抽象クラス（概要）(2)



意味を持たないdrawメソッドは、実装（実行可能なコード）を定義しない。このようなメソッドを抽象メソッドという。抽象メソッドを一つ以上もつクラスは、インスタンスの生成が不可能な抽象クラスとしなければならない。（例題3を参照）

抽象クラス（概念）(3) 説明



抽象（abstract）クラスは、インスタンスを生成させないことを前提としたクラスである。
なお「抽象クラス」と対比させ、インスタンスを生成できる通常のクラスは、
具象（concrete）クラスと呼ばれる。

抽象メソッドAをもった抽象クラスを継承して作成される具象クラスでは、
そのインスタンスが実行可能なメソッドAを必ず持つようにしなければならない。
（この場合、Figureがスーパークラスに含まれるような具象クラスでは、
そのインスタンスがdraw()メソッドが実行できるようになっている必要がある。
一度、MyCircleクラスのdrawメソッドを消してみよ。Eclipse上で表示される
エラーメッセージに着目せよ。）

まとめ：インスタンスと型

- インスタンスと型
- 継承の利用
- 抽象クラス（概要のみ）

例題集

パッケージ「j2.lesson11」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11

問題:次のクラスMyCircleを作成せよ。

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

返り値の型	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	draw()	キャンバスに円を描画する。

クラスメソッド

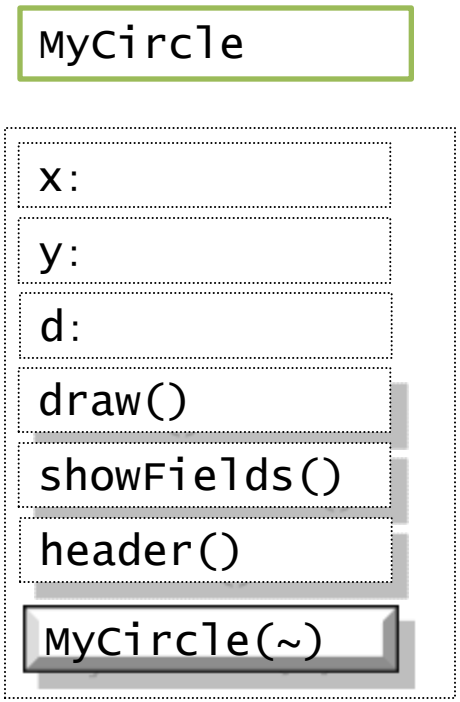
返り値の型	メソッド名(引数)	機能
void	header()	Spreadsheet の先頭行に、ヘッダとして、X座標、Y座標、半径という String を表示する。

コンストラクタ

コンストラクタ名(引数)	機能
`MyCircle(int x, int y, int diameter)`	MyCircle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する。

(クラス名： MyCircle)

例題11 (MyCircle)




```java
package j2.lesson11;

import gpjava.Spreadsheet;
import gpjava.Canvas;

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

    public MyCircle(int x, int y, int diameter) {
        this.x = x;
        this.y = y;
        this.diameter = diameter;
    }

    public static void header() {
        Spreadsheet.setString(0, 0, "X座標");
        Spreadsheet.setString(0, 1, "Y座標");
        Spreadsheet.setString(0, 2, "直徑");
        Spreadsheet.setString(0, 3, "面積");
    }

    public void showFields(int row) {
        Spreadsheet.setInt(row, 0, x);
        Spreadsheet.setInt(row, 1, y);
        Spreadsheet.setInt(row, 2, diameter);
    }

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

■例題12

問題：次のColoredCircleクラスを作成し、ColoredCircleとMyCircleの動作を確認せよ。

スーパークラス: MyCircle

インスタンス変数

変数の型と名前	初期値	説明
int red, green, blue	無し	色の赤成分, 緑成分、青成分

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	draw()	インスタンスの内容をもとに色付きの円をキャンバス上に描画する。
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	fill()	塗りつぶした円をキャンバス上に描画する。

クラスメソッド

返り値の型	メソッド名(引数)	機能
void	header()	Spreadsheet の先頭行に、ヘッダとして、X座標、Y座標、半径, 赤成分、緑成分、青成分という String を表示する。

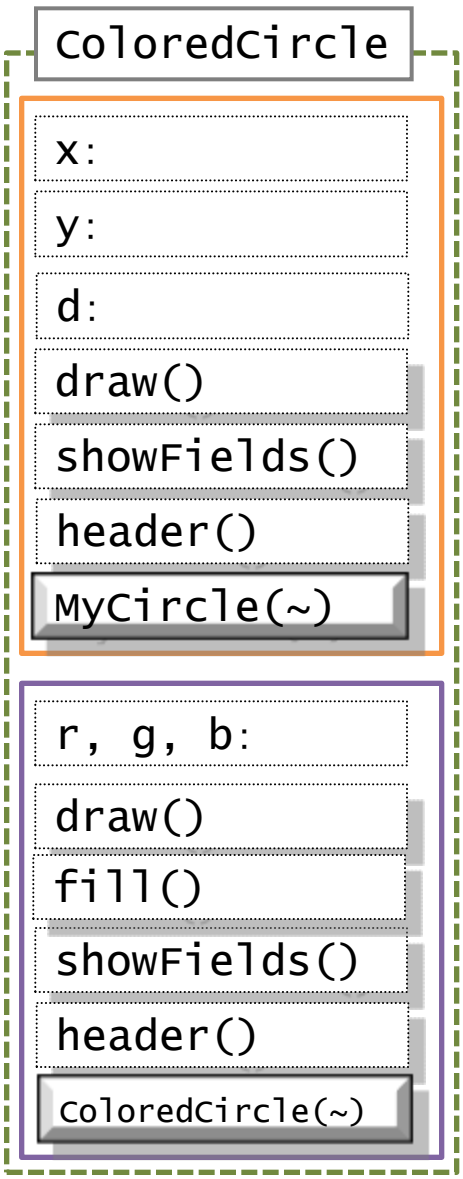
コンストラクタ

コンストラクタ名(引数)	機能
`MyCircle(int x, int y, int diameter, int red, int green, int blue)`	ColoredCircle 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作成する。

(クラス名：ColoredCircle)

動作確認クラス：Ex12DrawCircles

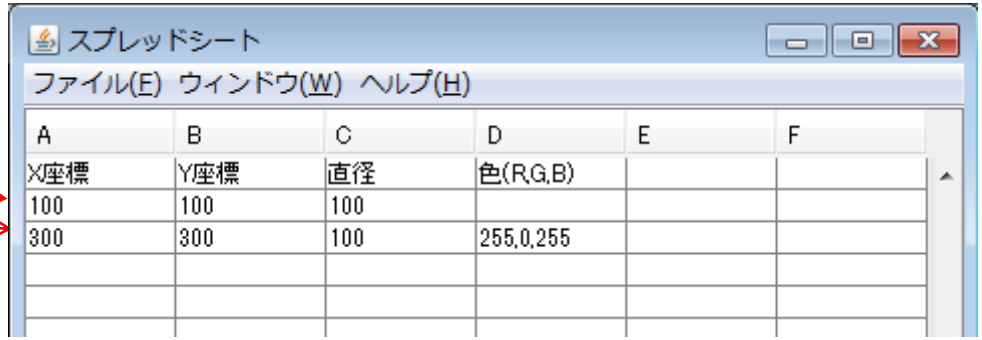
例題12 (ColoredCircle)




```java
package j2.lesson11;

import gpjava.Canvas;
import gpjava.Spreadsheet;

public class ColoredCircle extends MyCircle {
    int red;
    int green;
    int blue;

    public ColoredCircle(int x, int y, int d, int r, int g, int b){
        super(x, y, d);
        this.red = r;
        this.green = g;
        this.blue = b;
    }

    public static void header() {
        MyCircle.header();
        Spreadsheet.setString(0, 3, "色(R,G,B)");
    }

    public void showFields(int row){
        super.showFields(row);
        Spreadsheet.setString(row, 3,
                this.red + "," + this.green + "," + this.blue);
    }

    public void draw() {
        Canvas.setColor(red, green, blue);
        super.draw();
    }

    public void fill() {
        Canvas.setColor(red, green, blue);
        Canvas.fillOval(x, y, diameter, diameter);
    }
}
```

例題12(Ex12DrawCircles)

```
1  package j2.lesson11;
2
3  import gpjava.Spreadsheet;
4
5  public class Ex12DrawCircles {
6      public static void main(String[] args) {
7          MyCircle c0 = new MyCircle(100, 100, 100);
8          ColoredCircle c1 = new ColoredCircle(300, 300, 100, 255, 0, 255);
9
10         MyCircle aCircle;
11
12
13         Spreadsheet.show(7, 6);
14         ColoredCircle.header();
15
16         aCircle = c0;
17         aCircle.showFields(1);
18
19         aCircle = c1;
20         aCircle.showFields(2);
21     }
22 }
```



■例題13

問題：MyCircleとColoredCircleの内容を交互に表示するプログラムを作成せよ。

A	B	C	D	E	F
X座標	Y座標	直径	色(R,G,B)		
100	100	100			
300	300	100	255,0,255		
100	100	100			
300	300	100	255,0,255		
100	100	100			
300	300	100	255,0,255		
100	100	100			
300	300	100	255,0,255		
100	100	100			
300	300	100	255,0,255		

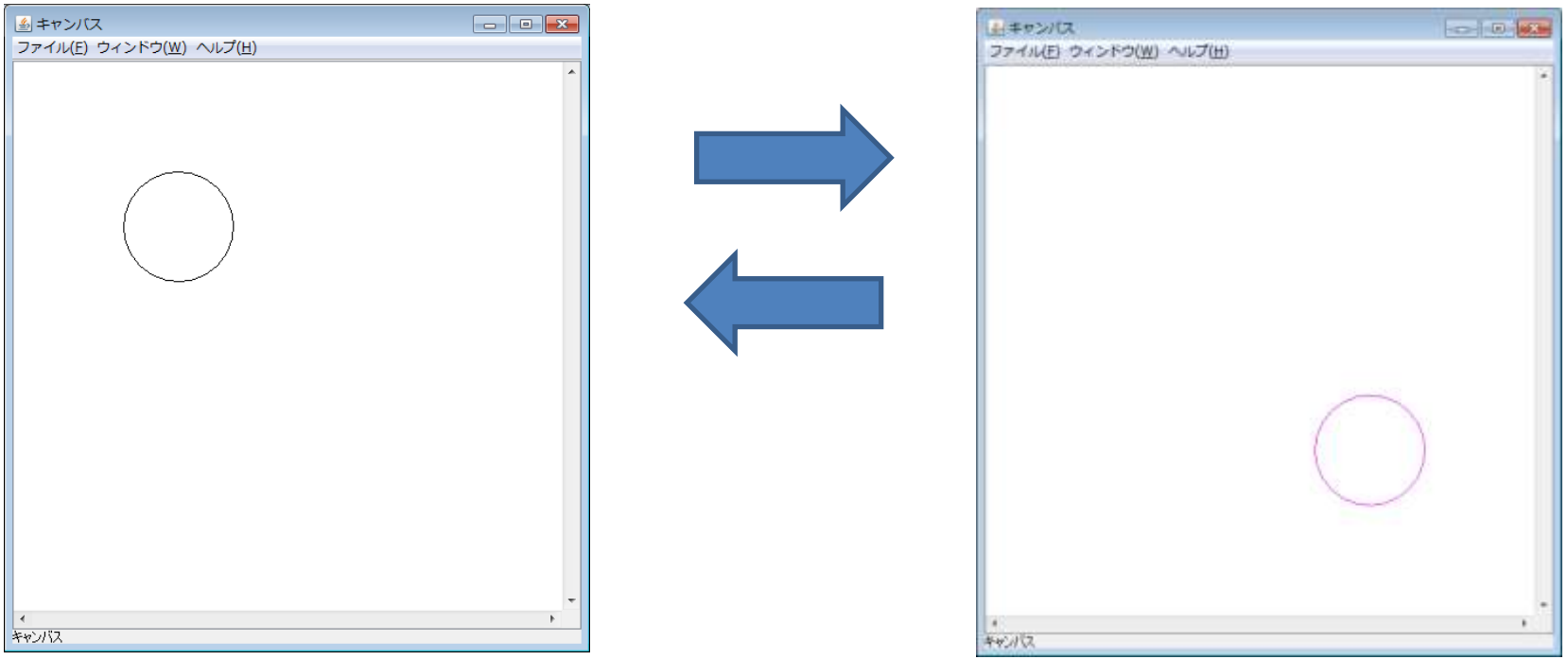
Ex13DrawCircles

例題13 (Ex13DrawCircles)

```
1  package j2.lesson11;
2
3  import gpjava.Spreadsheet;
4
5  public class Ex13DrawCircles {
6      public static void main(String[] args) {
7          MyCircle c0 = new MyCircle(100, 100, 100);
8          ColoredCircle c1 = new ColoredCircle(300, 300, 100, 255, 0, 255);
9
10         MyCircle aCircle;
11
12         Spreadsheet.show(11, 6);
13         ColoredCircle.header();
14
15         for(int i = 0; i < 10; i++) {
16             if(i % 2 == 0) {
17                 aCircle = c0;
18             } else {
19                 aCircle = c1;
20             }
21             aCircle.showFields(i + 1);
22         }
23     }
24 }
```

■例題14

問題：MyCircleとColoredCircleのインスタンスの内容を交互に
　キャンバス上に描画するプログラムを作成せよ。



Ex14DrawCircles

例題14
(Ex14DrawCirecles)

```
1  package j2.lesson11;
2
3  import gpjava.Canvas;
4
5  public class Ex14DrawCircles {
6      public static void main(String[] args) {
7          MyCircle c0 = new MyCircle(100, 100, 100);
8          ColoredCircle c1 = new ColoredCircle(300, 300, 100, 255, 0, 255);
9
10         MyCircle aCircle;
11
12         Canvas.show();
13
14         for(int i = 0; i < 10; i++) {
15             if(i % 2 == 0) {
16                 aCircle = c0;
17             } else {
18                 aCircle = c1;
19             }
20             aCircle.draw();
21             Canvas.waitForCountdown(500);
22             Canvas.clear();
23         }
24     }
25 }
```

■例題15

問題: MyCircle型の配列に、MyCircleとColoredCircleのインスタンスをいくつか格納せよ。次に配列に格納した内容をすべて表示させよ。

Ex15DrawCircles

例題15(Ex15DrawCircles)

```
1  package j2.lesson11;
2
3  import gpjava.Spreadsheet;
4
5  public class Ex15DrawCircles {
6      public static void main(String[] args) {
7          MyCircle[] m = {
8                  new MyCircle(20, 20, 100),
9                  new ColoredCircle(10, 20, 30, 255, 0, 0),
10                 new MyCircle(300, 300, 20),
11         };
12
13         Spreadsheet.show(11, 6);
14
15         ColoredCircle.header();
16         for(int i = 0; i < m.length; i++) {
17             MyCircle c = m[i];
18             c.showFields(i + 1);
19         }
20     }
21 }
```

A	B	C	D	E	F
X座標	Y座標	直径	色(R,G,B)		
20	20	100			
10	20	30	255,0,0		
300	300	20			

■例題21

問題: 次のクラスMyRectangleを作成せよ。

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 width	無し	長方形の横の長さ
int height	無し	長方形の縦の長さ

インスタンスメソッド

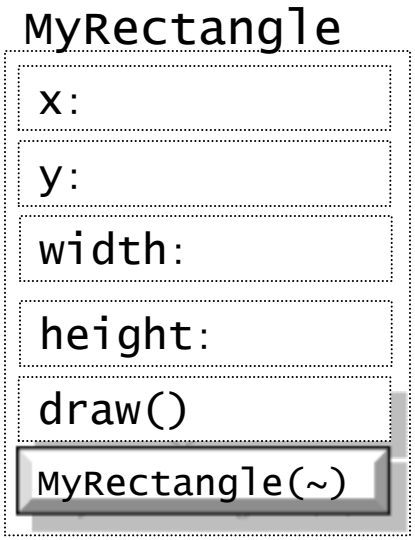
返り値の型	メソッド名(引数)	機能
void	**draw()**	インスタンスの内容をもとにキャンバスに長方形を描画する。

コンストラクタ

コンストラクタ名(引数)	機能
MyRectangle(int x, int y, int width, int height)	MyRectangle 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する。

（クラス名： MyRectangle）

例題21(MyRectangle)




```java
package j2.lesson11;

import gpjava.Canvas;

public class MyRectangle {
    int x;
    int y;
    int width;
    int height;

    public MyRectangle(int x, int y, int width, int height) {
        this.x = x;
        this.y = y;
        this.width = width;
        this.height = height;
    }

    public void draw() {
        Canvas.drawRect(x, y, width, height);
    }
}
```

■例題22

問題: MyRectangleとMyCircleの共通のスーパークラスとして Figureを作成し、MyRectangle、MyCircleに継承させよ。さらに動作を確認せよ。

新規クラスFigure 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	**draw()**	メッセージ「描画できません」をコンソールに出力

MyCircleおよび、MyRectangleのスーパークラスをFigureに設定

（クラス名： Figure）
既存(例題11)のMyCircleを編集する。
既存のMyRectangleを編集する。
動作確認クラス:Ex22DrawFigures

例題22(Figure)

```
1  package j2.lesson11;
2
3  public class Figure {
4      public void draw() {
5          System.out.println("描画できません");
6      }
7  }
```

例題22(MyCircle)

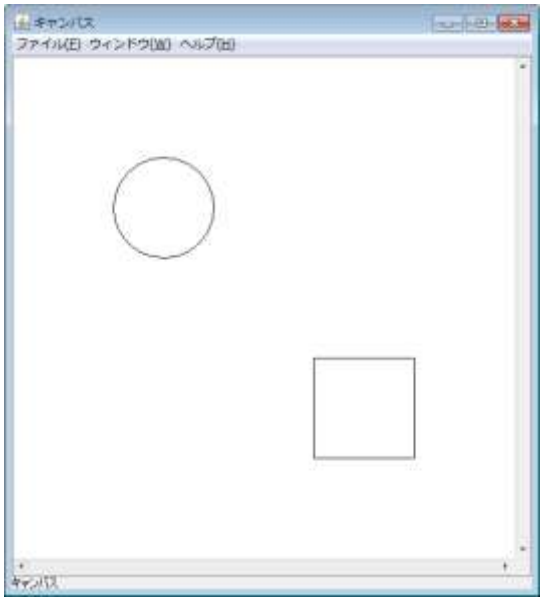
```
5
6  public class MyCircle extends Figure {
7      int x;
8      int y;
```

例題22(MyRectangle)

```
4
5  public class MyRectangle extends Figure {
6      int x;
7      int y;
```

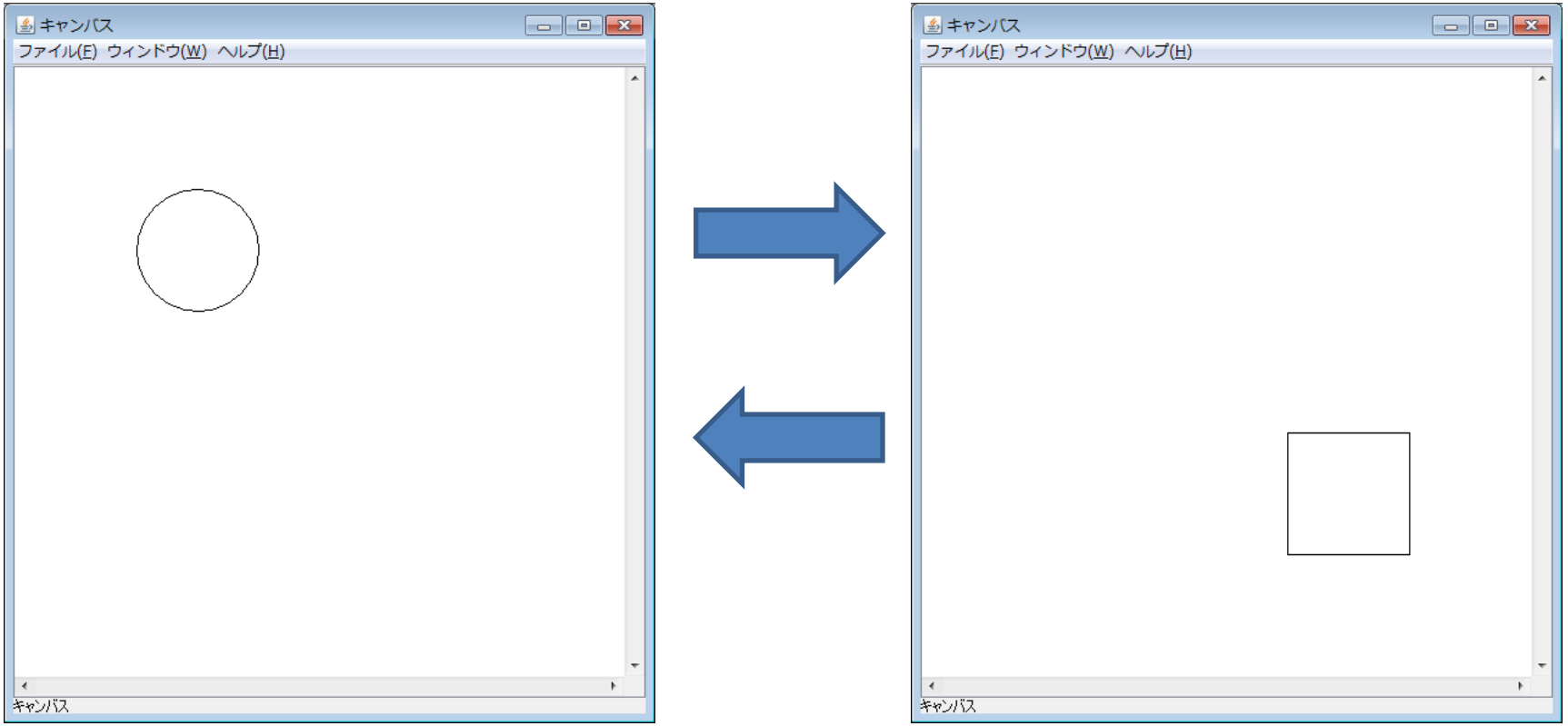
例題22(Ex22DrawFigures)

```java
package j2.lesson11;

import gpjava.Canvas;

public class Ex22DrawFigures {
    public static void main(String[] args) {
        MyCircle c = new MyCircle(100, 100, 100);
        MyRectangle r = new MyRectangle(300, 300, 100, 100);

        Canvas.show();

        Figure aFigure;

        aFigure = c;
        aFigure.draw();

        aFigure = r;
        aFigure.draw();
    }
}
```



■例題23

問題: 円と長方形を交互に表示させるプログラムを作成せよ



既存のEx23DrawFiguresを編集する

例題23(Ex23DrawFigures)

```
1  package j2.lesson11;
2
3  import gpjava.Canvas;
4
5  public class Ex23DrawFigures {
6      public static void main(String[] args) {
7          MyCircle c = new MyCircle(100, 100, 100);
8          MyRectangle r = new MyRectangle(300, 300, 100, 100);
9
10         Canvas.show();
11
12         Figure aFigure;
13
14         for(int i = 0; i < 10; i++) {
15             if(i % 2 == 0) {
16                 aFigure = c;
17             } else {
18                 aFigure = r;
19             }
20             aFigure.draw();
21             Canvas.waitForCountdown(500);
22             Canvas.clear();
23         }
24     }
25 }
```

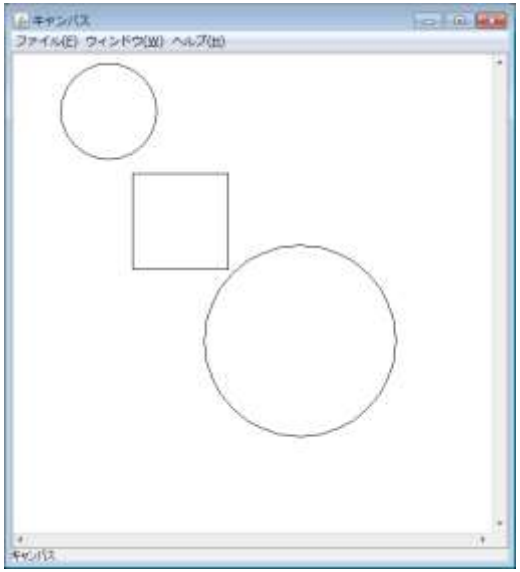
■例題24

問題: Figureの配列に、MyCircleとMyRectangleのインスタンスをいくつか格納せよ。次に配列に格納した図形をキャンバスにすべて表示させよ。

Ex24DrawFigures

例題24(Ex24DrawFigures)

```java
package j2.lesson11;

import gpjava.Canvas;

public class Ex24DrawFigures {
    public static void main(String[] args) {
        Figure[] m = {
                new MyCircle(50, 10, 100),
                new MyRectangle(125, 125, 100, 100),
                new MyCircle(200, 200, 200),
        };

        Canvas.show();
        for(int i = 0; i < m.length; i++) {
            Figure fig = m[i];
            fig.draw();
        }
    }
}
```



■例題25

問題: 図形の面積を求めるarea()メソッドを作成し、例題24の配列に格納された各図形の面積と、その面積の合計を表示せよ。

area()メソッドを、どのクラスに作成すべきかまず考えよ。

```
<終了> Ex25DrawFigures [Java アプリケーション]
7853.981633974483
10000.0
31415.926535897932
sumArea = 49269.90816987242
```

Ex24DrawFigures

■例題25

新規Figureインスタンスメソッド

返り値の型	メソッド名(引数)	機能
double	**area()**	0.0を返す。

新規MyCircleインスタンスメソッド

返り値の型	メソッド名(引数)	機能
double	**area()**	MyCircleインスタンスの面積を返す。

新規MyRectangleインスタンスメソッド

返り値の型	メソッド名(引数)	機能
double	**area()**	MyRectangleインスタンスの面積を返す。

```
<終了> Ex25DrawFigures [Java アプリケーション]
7853.981633974483
10000.0
31415.926535897932
sumArea = 49269.90816987242
```

Ex25DrawFigures

既存のFigure, MyCircle, MyRectangleを編集する。

例題25(Figure)

```java
package j2.lesson11;

public class Figure {
    public void draw() {
        System.out.println("描画できません");
    }

    // Ex25
    public double area() {
        return 0.0;
    }
}
```

例題25(MyRectangle)

```java
package j2.lesson11;

import gpjava.Canvas;

public class MyRectangle extends Figure {
    int x;
    int y;
    int width;
    int height;

    public MyRectangle(int x, int y, int width, int height) {...

    public void draw() {...

    public double area() {
        return width * height;
    }
}
```

例題25(MyCircle)

```
 1  package j2.lesson11;
 2
 3  import gpjava.Spreadsheet;
 4  import gpjava.Canvas;
 5
 6  public class MyCircle extends Figure {
 7      int x;
 8      int y;
 9      int diameter;
10
11      public MyCircle(int x, int y, int diameter) {..
16
17      public static void header() {..
23
24      public void showFields(int row) {..
29
30      public void draw() {..
33
34      public double area() {
35          double r = (double)diameter / 2;
36          return Math.PI * r * r;
37      }
38  }
```

例題25(Ex25DrawFigures)

```java
package j2.lesson11;

public class Ex25DrawFigures {
    public static void main(String[] args) {
        Figure[] m = {
                new MyCircle(50, 10, 100),
                new MyRectangle(125, 125, 100, 100),
                new MyCircle(200, 200, 200),
        };

        double sumArea = 0;

        for(int i = 0; i < m.length; i++) {
            Figure fig = m[i];
            sumArea += fig.area();
            System.out.println(fig.area());
        }
        System.out.println("sumArea = " + sumArea);
    }
}
```

```
<終了> Ex25DrawFigures [Java アプリケーション]
7853.981633974483
10000.0
31415.926535897932
sumArea = 49269.90816987242
```

■例題31

問題:Figureを抽象クラスに変更せよ。

抽象メソッド
public abstract void draw();
public abstract double area();

例題31(Figure)

```
1  package j2.lesson11;
2
3  public abstract class Figure {
4      public abstract void draw();
5      public abstract double area();
6  }
```