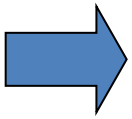
プログラミング入門2

第6回　継承、コンストラクタ

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17回に詳しい。

テーマ：継承、コンストラクタ



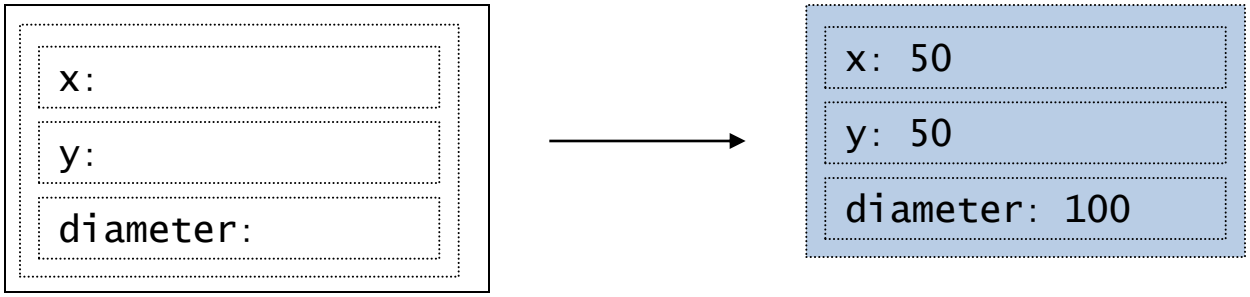
- 継承(inheritance)
  - インスタンス変数の継承
  - メソッドの継承
  - メソッドのオーバーライド
  - Super呼び出し
- コンストラクタ(概要）

テーマ：継承、コンストラクタ

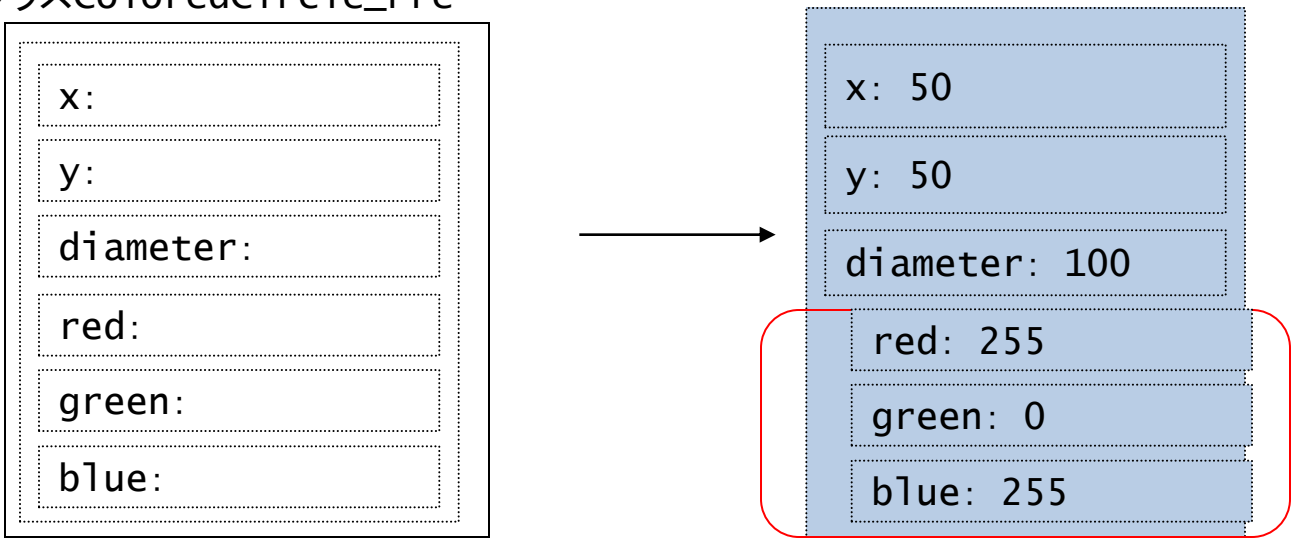
- 継承(inheritance)
  - インスタンス変数の継承
  - メソッドの継承
  - メソッドのオーバーライド
  - Super呼び出し
- コンストラクタ(概要)

継承の基本（動機(1)）

クラスMyCircle



クラスColoredCircle_Pre



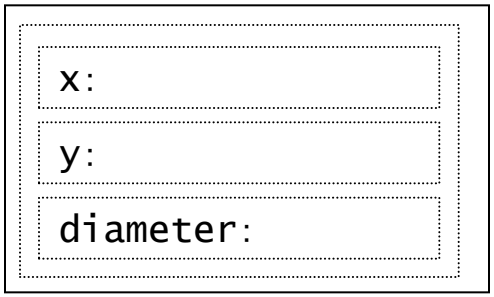
保持するデータのフォーマットが少しだけ異なるようなインスタンスを作成したい。

継承の基本（動機(2)）

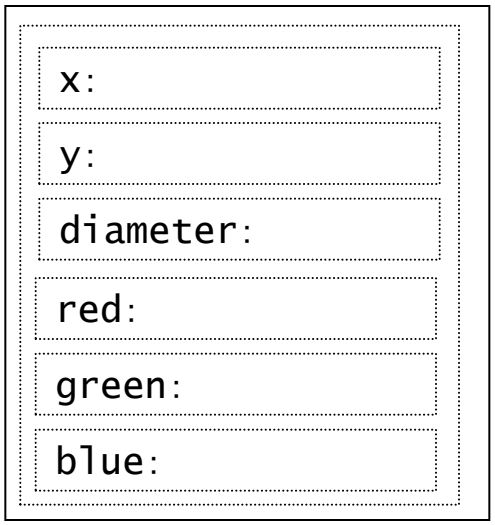
```
public class MyCircle {
  int x = 50;
  int y = 50;
  int diameter = 100;
}
```

```
public class ColoredCircle_Pre {
  int x = 50;
  int y = 50;
  int diameter = 100;
  int red = 255;
  int green = 0;
  int blue = 255;
}
```

クラスMyCircle

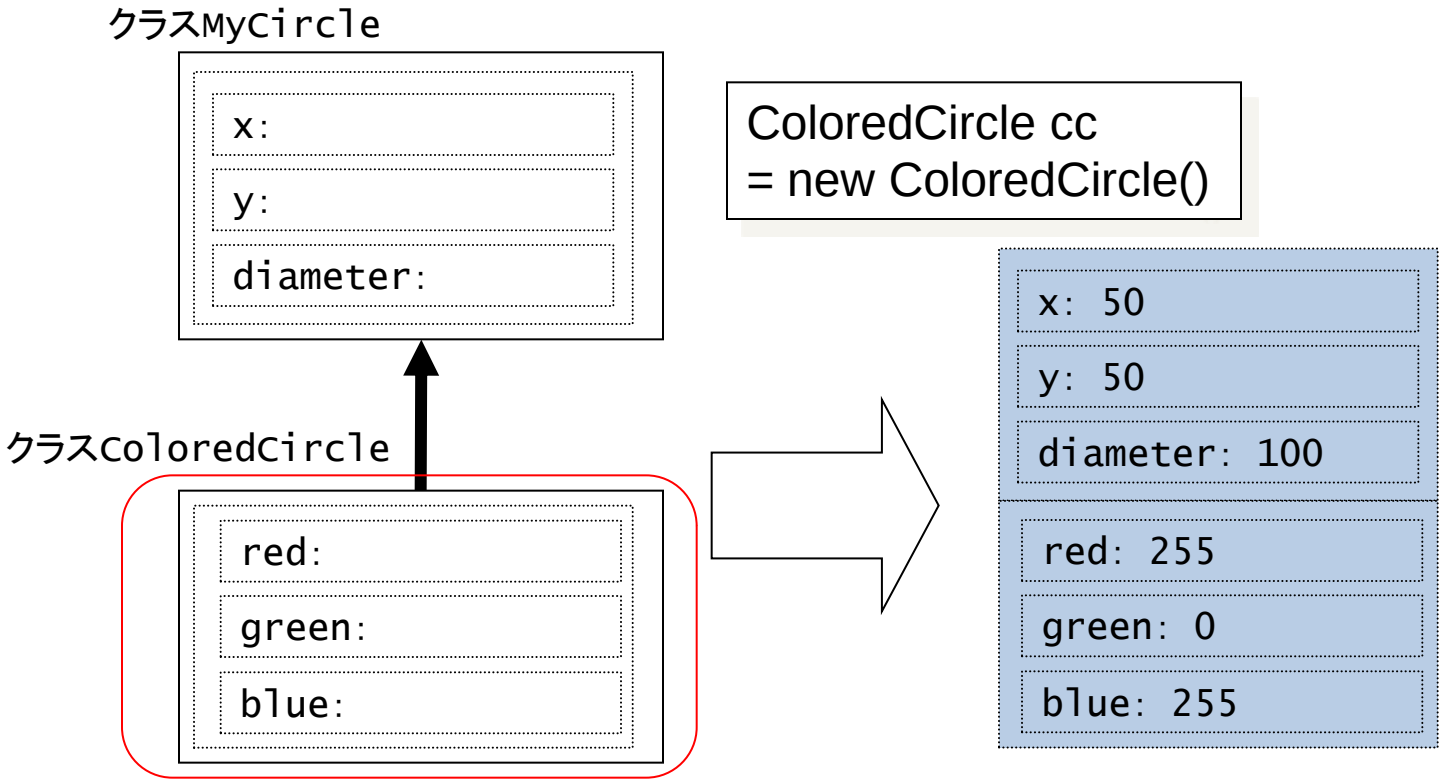


クラスColoredCircle_Pre



素朴な方法では、同じ内容をもう一度、新たに作成するクラスで定義する必要がある。（このやり方は望ましくない）

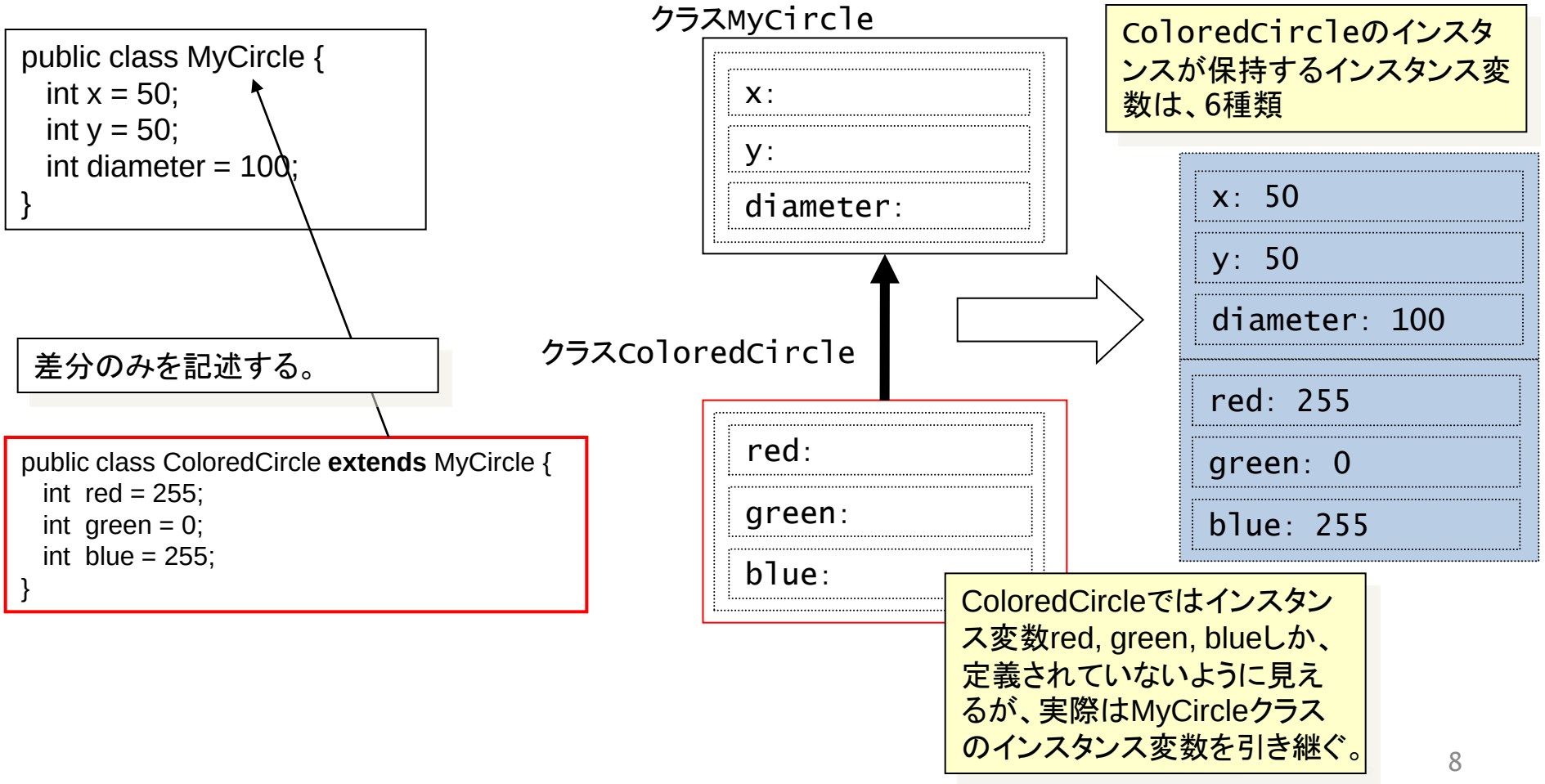
継承の基本(考え方(1))



継承機構を用いると、ゼロから作り直すのではなく、もとのインスタンスを拡張(extends)する形で、異なる部分(差分)のみを定義するだけでよい。

継承の基本(考え方(2))

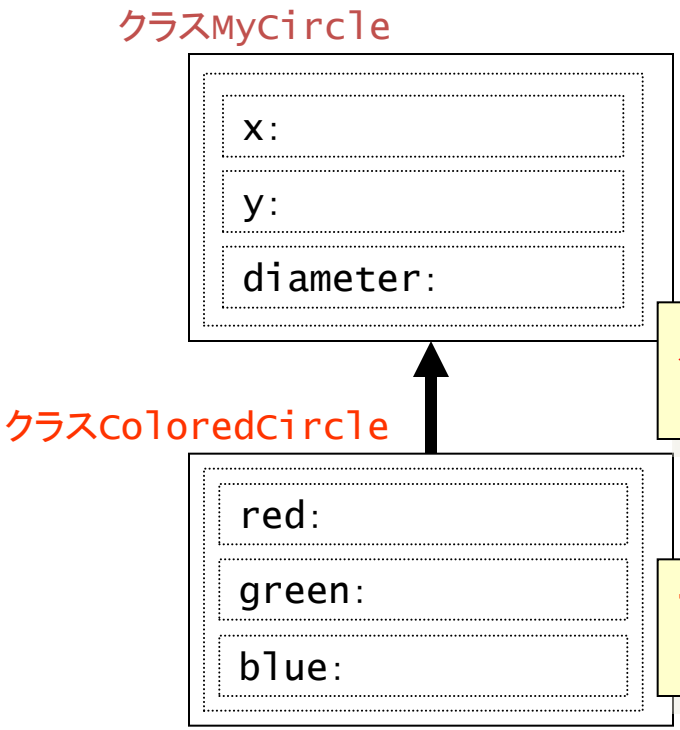
継承機構を用いると、ゼロから作り直すのではなく、もとのインスタンスを拡張(extends)する形で、異なる部分(差分)のみを定義するだけでよい。



インスタンス変数の継承（定義の方法）

```
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の宣言
        …..
}
```

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

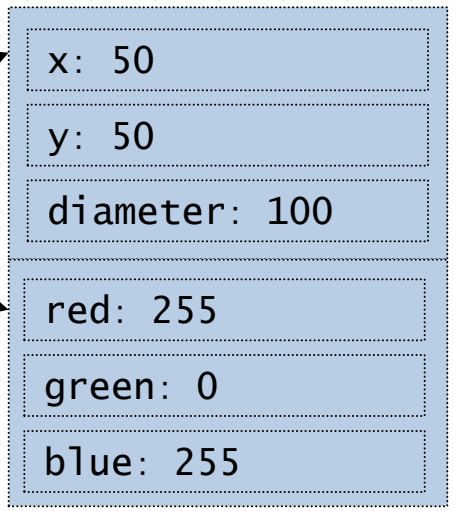


インスタンス変数へのアクセス（例）

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

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

```
ColoredCircle cc = new ColoredCircle();
System.out.println(cc.red);

System.out.println(cc.x);
```



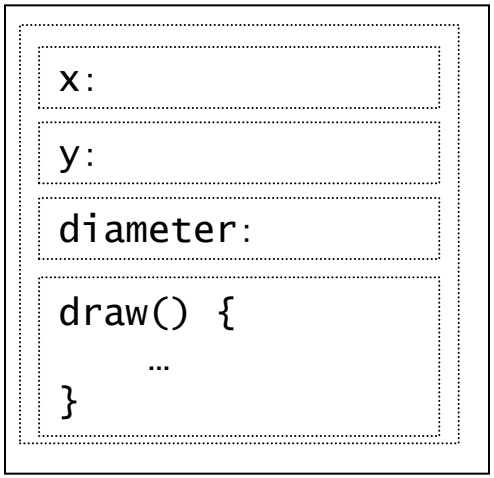
出力

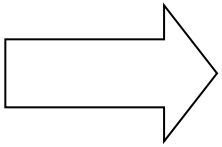
```
255
50
```

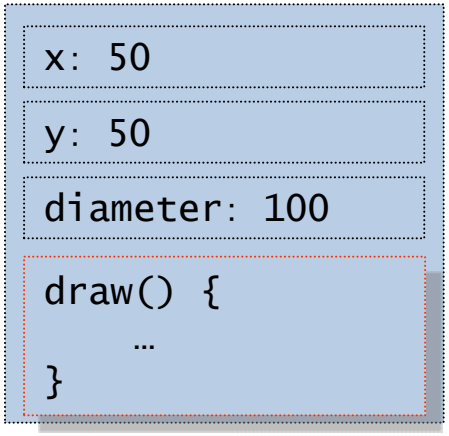
ccのインスタンス変数xは、ColoredCircleクラス内では直接定義はされていないが、スーパークラスであるMyCircleクラス内で定義されている。

メソッド群の継承（例題）

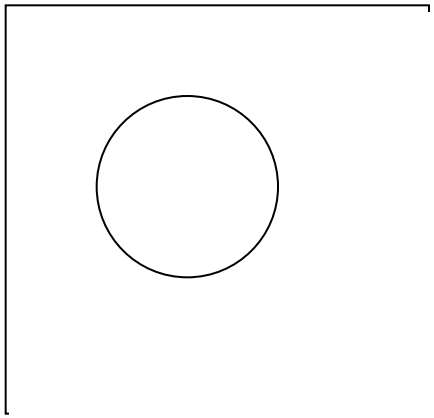
クラスMyCircle







drawメソッドの実行例



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

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

メソッド群の継承（考え方）

クラスMyCircle

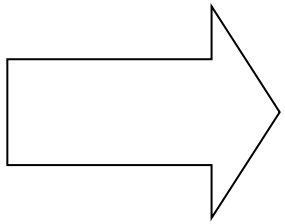
```
x:
y:
diameter:
draw() {
    …
}
```

クラスColoredCircle

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

継承を利用することで、スーパークラスのインスタンス変数をサブクラスに引き継がすことができた。

これと同様にインスタンスメソッドもサブクラスに引き継がすことができる。



```
x: 50
y: 50
diameter: 100
draw() {
    …
}
red: 255
green: 0
blue: 255
fill() {
    …
}
```

ColoredCircleクラスのインスタンスは、MyCircleクラスのインスタンス変数 x, y, diameterとメソッド drawを引き継ぐ

メソッド群の継承（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

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

メソッドの起動

クラスMyCircle

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

クラス
ColoredCircle

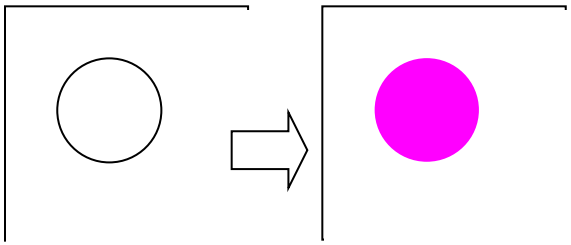
```
red:
green:
blue:
fill() {
    …
}
```

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

```
Canvas.show()
cc.draw();
cc.fill();
```

インスタンスcc

```
x: 50
y: 50
diameter: 100
draw() {
    …
}
red: 255
green: 0
blue: 255
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。

テーマ：継承、コンストラクタ

- 継承(inheritance)
  - インスタンス変数の継承
  - メソッドの継承
  - メソッドのオーバーライド
  - Super呼び出し
- コンストラクタ(概要)

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

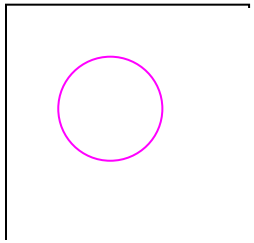
MyCircleクラスのdrawメソッドをオーバーライドしている。
このメソッドは、色付きで円をキャンバスで表示する。

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);
}
```

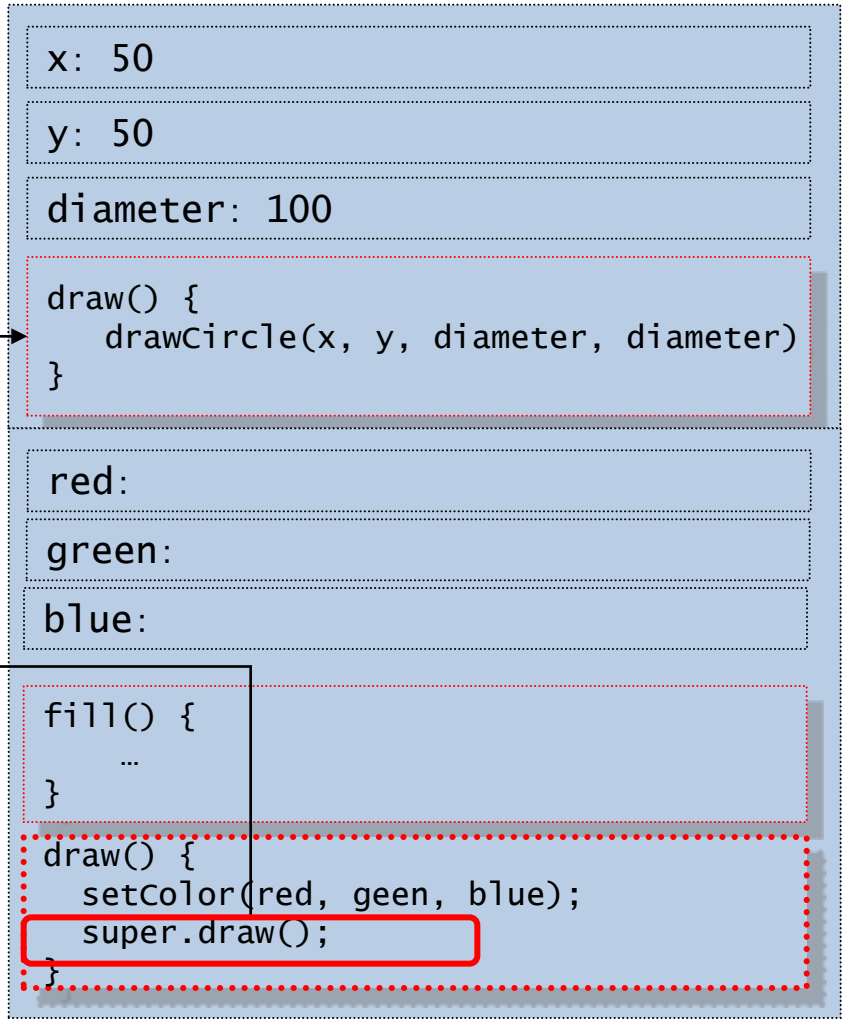
ColoredCircleのdrawインスタンスメソッドが、スーパークラスで定義されているdrawメソッドに代わって起動される。

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。

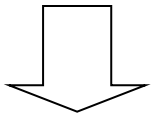


```
ColoredCircle cc = new ColoredCircle();
cc.draw();
```

スーパークラスで定義されたメソッドを呼ぶ
Super呼び出し (Super Call)




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



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

オーバーライドされたメソッドは、
消える訳ではなく、隠されているだけである。
インスタンスメソッド内で「super.メソッド名（引数）」の形で、起動することができる。

テーマ：継承、コンストラクタ

- 継承(inheritance)
  - インスタンス変数の継承
  - メソッドの継承
  - メソッドのオーバーライド
  - Super呼び出し
- コンストラクタ(概要)

コンストラクタ

これまで、インスタンスを生成し初期値を与えるまでの一連の流れを呼び出す側のクラスのメソッドによって単純化してきた。

```
MyCircle c = Ex00.create(50, 50, 100);
```

以下では、「コンストラクタ」の概要を説明する。一番単純なコンストラクタは、既に下記の形で利用してきた。

```
MyCircle c = new MyCircle();
```

Java言語におけるコンストラクタは、インスタンスの生成時に初期値を指定するなどの処理を行わせるための記述である。

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

初期値x=50, y=50, diameter=100
でMyCircleインスタンスを生成

コンストラクタ（定義と呼び出し）

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

```
public class Ex00{
  MyCircle create(int x, int y, int d) {
    MyCircle newObj = new MyCircle();
    newObj.x = x;
    newObj.y = y;
    newObj.diameter = d;
    return newObj;
  }
}
```

メソッド（ファクトリメソッド）によるインスタンス生成と初期値の指定

```
MyCircle c = Ex00.create(50, 50, 100)
```

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

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

実際には、1行目（this.x = x）の
直前にインスタンスが生成されている。
なお、return文でインスタンスを返す必要はない。

コンストラクタを利用して初期値を指定

```
MyCircle c = new MyCircle(50, 50, 100)
```

クラス名(引数の宣言) {
　インスタンスを生成した直後に
　おこなう処理。
　（通常、初期値の代入などの処理を
　ここに記述する。）
}

コンストラクタ（super呼び出し）

サブクラスのコンストラクタから、スーパークラスのコンストラクタを呼び出すことができる。

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

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

なお、同じクラスでオーバーロード（引数の数や型などが違う）されているコンストラクタは、
`this(実引数)`
の形で呼ぶことができる。（詳細は省略）

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

  public ColoredCircle(int x, int y, int d,
                       int red, int green, int blue){
    super(x, y, d);
    this.red = red;
    this.green = green;
    this.blue = glue;
  }
}
```

ColoredCircle cc
= new ColoredCircle(50, 50, 100, 255, 0, 255)

super(実引数）
の形

このsuper呼び出しでは、ColoredCircleクラスのスーパークラスのMyCircleクラスのコンストラクタを呼び出す。

コンストラクタから、他のコンストラクタを呼び出す際には、先頭で呼び出しを行わなければならない。

デフォルトコンストラクタ（概要のみ）

デフォルトコンストラクタ

クラス定義に、コンストラクタが定義されていない場合は、自動的に

クラス名（） {

}

の形のコンストラクタが自動的に加えられる。これをデフォルトコンストラクタと言う。

一方で、明示的にコンストラクタを定義した場合は、デフォルトコンストラクタは加えられない。

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

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

左の例では、コンストラクタが明示的に定義されているので、デフォルトコンストラクタ

```
MyCircle() {
}
```

は追加されない。したがって、

```
MyCircle c = new MyCircle();
```

の形でインスタンスを生成する場合は、
明示的にコンストラクタを定義する必要がある。

まとめ：継承、コンストラクタ

- 継承(inheritance)
  - インスタンス変数の継承
  - メソッドの継承
  - メソッドのオーバーライド
  - Super呼び出し
- コンストラクタ(概要)

本日の例題と問題

- MyCircle , ColoredCircle を用いた継承、コンストラクタの演習
  - Ex11, Ex12, Ex13, Ex14, Ex15, Ex16, Ex17
- BorderedCircle を用いた課題
  - Q11, Q12, Q13, Q14
- Book を用いた継承とオーバーライド
  - Ex21, Ex22, Q21*

(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.lesson06」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

返り値の型	メソッド名(引数)	機能
void	`draw()`	キャンバスに円を描画する。
void	**showFields(int row)**	MyCircleインスタンスの内容をSpreadsheetのrow行表示する。 ※実行例参考
void	**header()**	Spreadsheetのヘッダを表示する

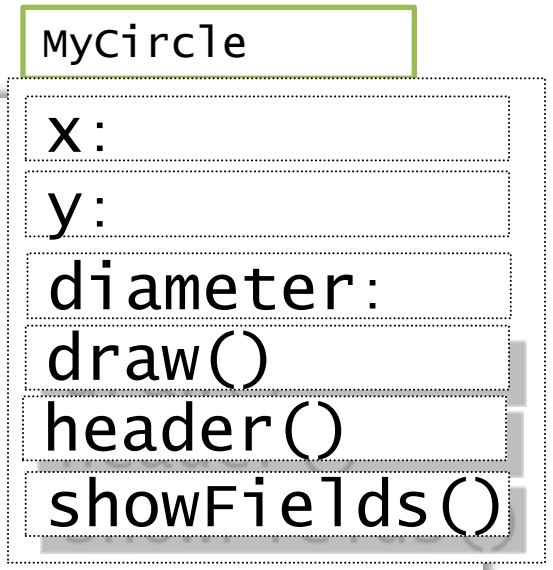
Ex11DrawCircle のメソッド

返り値の型	メソッド名(引数)	機能
MyCircle	`create(int x, int y, int d)`	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し、そのインスタンスを返す。

(クラス名： MyCircle)

動作確認クラス:Ex11DrawMyCircle

例題11(MyCircle)




```java
package j2.lesson06;

import gpjava.Canvas;
import gpjava.Spreadsheet;

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

    MyCircle(){}

    //Ex11
    void draw() {
        Canvas.drawOval(this.x, this.y, this.diameter, this.diameter);
    }
    //Ex11
    void header() {
        Spreadsheet.setString(0, 0, "円のX座標");
        Spreadsheet.setString(0, 1, "円のY座標");
        Spreadsheet.setString(0, 2, "円の直径");
    }
    //Ex11
    void showFields(int row){
        Spreadsheet.setInt(row, 0, this.x);
        Spreadsheet.setInt(row, 1, this.y);
        Spreadsheet.setInt(row, 2, this.diameter);
    }
}
```

例題11(Ex11DrawMyCircle)

```java
package j2.lesson06;

import gpjava.Canvas;
import gpjava.Spreadsheet;

public class Ex11DrawMyCircle {

    public static void main(String[] args) {
        new Ex11DrawMyCircle().start();
    }

    void start() {
        Canvas.show();
        Spreadsheet.show();

        MyCircle c = create(50, 50, 100);
        c.draw();
        c.header();
        c.showFields(1);
    }

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

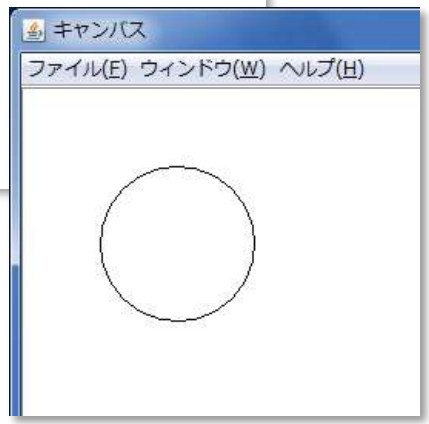
create(50,50,100)

MyCircle c

- x:50
- y:50
- d:100
- draw()
- showFields()
- header()

c.draw()

c.header()
c.showFields()



■例題12

問題：次のColoredCircleクラスを作成せよ。

スーパークラス: MyCircle

インスタンス変数

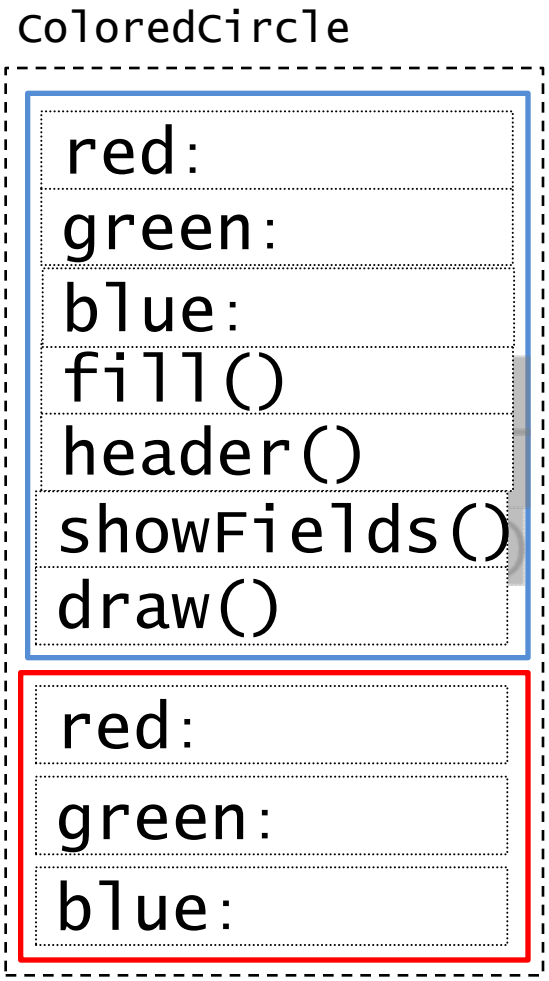
変数の型と名前	初期値	説明
int red	無し	色の赤成分
int blue	無し	色の緑成分
int green	無し	色の青成分

Ex12DrawCircle のメソッド

返り値の型	メソッド名(引数)	機能
ColoredCircle	create(int x, int y, int d, 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返す。

（クラス名：ColoredCircle）
動作確認クラス：Ex12DrawColoredCircle

例題12
(ColoredCircle)




```
1  package j2.lesson06;
2
3  public class ColoredCircle extends MyCircle {
4      int red;
5      int green;
6      int blue;
7
8      ColoredCircle(){}
9  }
```

例題12
(Ex12DrawColoredCircle)

```
1  package j2.lesson06;
2
3  public class Ex12DrawColoredCircle {
4
5      public static void main(String args[]) {
6          new Ex12DrawColoredCircle().start();
7      }
8
9      void start() {
10         ColoredCircle c = create(50, 50, 100, 255, 0, 0);
11         System.out.println(c.toString());
12     }
13
14     ColoredCircle create(int x, int y, int d, int r, int g, int b) {
15         ColoredCircle newObj = new ColoredCircle();
16         newObj.x = x;
17         newObj.y = y;
18         newObj.diameter = d;
19         newObj.red = r;
20         newObj.green = g;
21         newObj.blue = b;
22         return newObj;
23     }
24 }
```

■例題13

問題：次のColoredCircleインスタンスメソッドfillを作成し動作を確認せよ。

ColoredCircle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	fill()	キャンバスに円を描画する。但し、円の内部がインスタンス変数colorに対する色で塗り潰されているものとする。

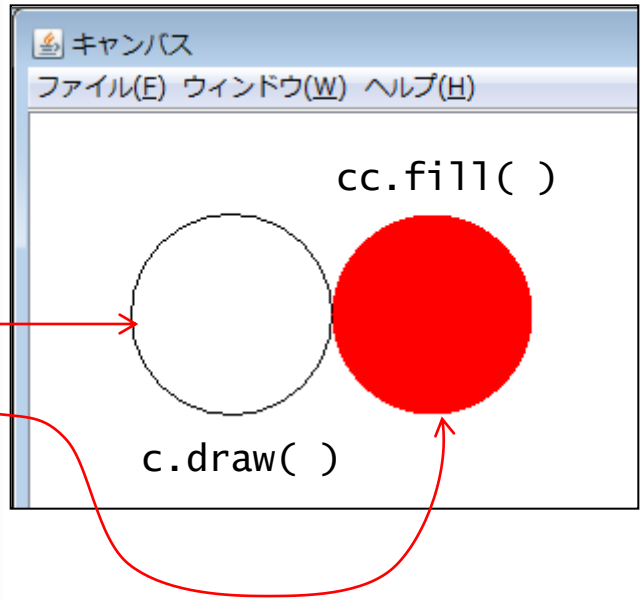
※既存のColoredCircleクラスを編集する。
動作確認クラス：Ex13DrawColoredCircle

例題13
(ColoredCircle)

```
1  package j2.lesson06;
2
3  import gpjava.Canvas;
4
5  public class ColoredCircle extends MyCircle {
6      int red;
7      int green;
8      int blue;
9
10     ColoredCircle(){}
11
12     // Ex13
13     void fill() {
14         Canvas.setColor(red, green, blue);
15         Canvas.fillOval(x, y, diameter, diameter);
16     }
17 }
```

例題13(Ex13DrawColoredCircle)

```java
package j2.lesson06;

import gpjava.Canvas;

public class Ex13DrawColoredCircle {
    public static void main(String[]args){
        new Ex13DrawColoredCircle().start();
    }

    void start() {
        Canvas.show();

        MyCircle c = create(50, 50, 100);
        c.draw();

        ColoredCircle cc = create(150, 50, 100, 255, 0, 0);
        cc.fill();
    }

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

    ColoredCircle create(int x, int y, int d, int r, int g, int b) {
        ColoredCircle newObj = new ColoredCircle();
        newObj.x = x;
        newObj.y = y;
        newObj.diameter = d;
        newObj.red = r;
        newObj.green = g;
        newObj.blue = b;
        return newObj;
    }
}
```



■例題14

問題: ColoredCircleインスタンスメソッドshowFields、headerを作成し動作を確認せよ。(メソッドのオーバーライド)

新規ColoredCircle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)	ColoredCircleインスタンスの文字列表現をSpreadsheet の row 行に表示する
void	header()	Spreadsheetのヘッダを表示する。red, green, blue まで表示する点が追加される。
既存ColoredCircleインスタンスメソッド		
void fill()		

※既存のColredCircleクラスを編集する。
動作確認クラス:Ex14DrawColoredCircle

例題14 (ColoredCircle)

```java
package j2.lesson06;

import gpjava.Canvas;
import gpjava.Spreadsheet;

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

    ColoredCircle(){}

    // Ex13
    void fill() {
        Canvas.setColor(red, green, blue);
        Canvas.fillOval(x, y, diameter, diameter);
    }


    //Ex14
    void header() {
        Spreadsheet.setString(0, 0, "円のX座標");
        Spreadsheet.setString(0, 1, "円のY座標");
        Spreadsheet.setString(0, 2, "円の直径");
        Spreadsheet.setString(0, 3, "色の赤成分");
        Spreadsheet.setString(0, 4, "色の緑成分");
        Spreadsheet.setString(0, 5, "色の青成分");
    }
    //Ex14
    void showFields(int row){
        Spreadsheet.setInt(row, 0, this.x);
        Spreadsheet.setInt(row, 1, this.y);
        Spreadsheet.setInt(row, 2, this.diameter);
        Spreadsheet.setInt(row, 3, this.red);
        Spreadsheet.setInt(row, 4, this.green);
        Spreadsheet.setInt(row, 5, this.blue);
    }
}
```

例題14(Ex1DrawColoredCircle)

```java
package j2.lesson06;

import gpjava.Spreadsheet;

public class Ex14DrawColoredCircle {
    public static void main(String[]args){
        new Ex14DrawColoredCircle().start();
    }

    void start() {
        Spreadsheet.show(4, 6);

        ColoredCircle c = create(50, 50, 100, 0, 255, 0);
        c.header();
        c.showFields(1);
    }

    ColoredCircle create(int x, int y, int d, int r, int g, int b){
        ColoredCircle newObj = new ColoredCircle();
        newObj.x = x;
        newObj.y = y;
        newObj.diameter = d;
        newObj.red = r;
        newObj.green = g;
        newObj.blue = b;
        return newObj;
    }
}
```

■例題15

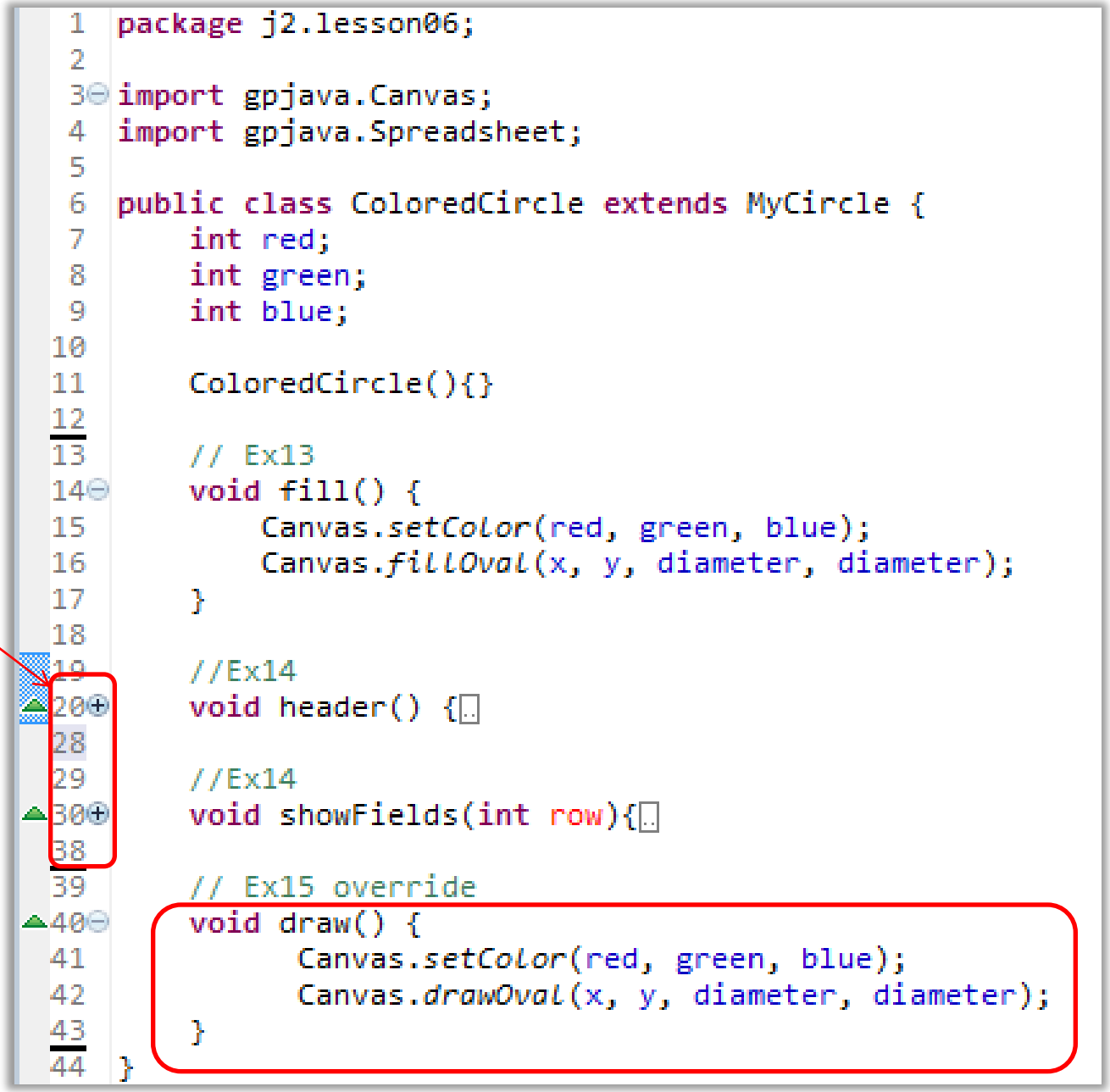
問題: ColoredCircleインスタンスメソッドdrawを作成し動作を確認せよ。(メソッドのオーバーライド)

新規ColoredCircle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()	キャンバスに円を描画する。
既存ColoredCircleインスタンスメソッド		
void fill()		
void showFields(int row)		
void header()		

※既存のColoredCircleクラスを編集する。
動作確認クラス:Ex15DrawColoredCircle

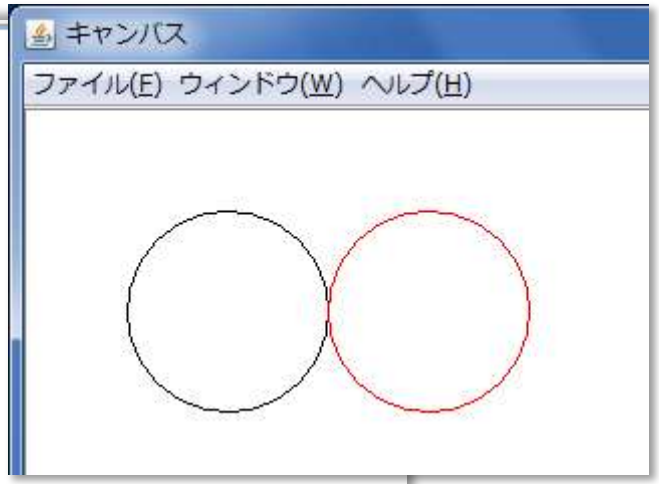
例題15 (ColoredCircle)



```
1  package j2.lesson06;
2
3  import gpjava.Canvas;
4  import gpjava.Spreadsheet;
5
6  public class ColoredCircle extends MyCircle {
7      int red;
8      int green;
9      int blue;
10
11     ColoredCircle(){}
12
13     // Ex13
14     void fill() {
15         Canvas.setColor(red, green, blue);
16         Canvas.fillOval(x, y, diameter, diameter);
17     }
18
19     //Ex14
20     void header() {..
28
29     //Ex14
30     void showFields(int row){..
38
39     // Ex15 override
40     void draw() {
41         Canvas.setColor(red, green, blue);
42         Canvas.drawOval(x, y, diameter, diameter);
43     }
44 }
```

例題15(Ex15DrawColoredCircle)

```java
package j2.lesson06;

import gpjava.Canvas;

public class Ex15DrawColoredCircle {
    public static void main(String[]args){
        new Ex15DrawColoredCircle().start();
    }

    void start() {
        Canvas.show();

        MyCircle c = create(50, 50, 100);
        c.draw();

        ColoredCircle cc = create(150, 50, 100, 255, 0, 0);
        cc.draw();
    }

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

    ColoredCircle create(int x, int y, int d, int r, int g, int b){
        ColoredCircle newObj = new ColoredCircle();
        newObj.x = x;
        newObj.y = y;
        newObj.diameter = d;
        newObj.red = r;
        newObj.green = g;
        newObj.blue = b;
        return newObj;
    }
}
```



■例題16

問題: ColoredCircleインスタンスメソッドshowFields, header, drawをsuperを用いたメソッド呼び出しを用いて、作成しなおし動作を確認せよ

新規ColoredCircleインスタンスメソッド

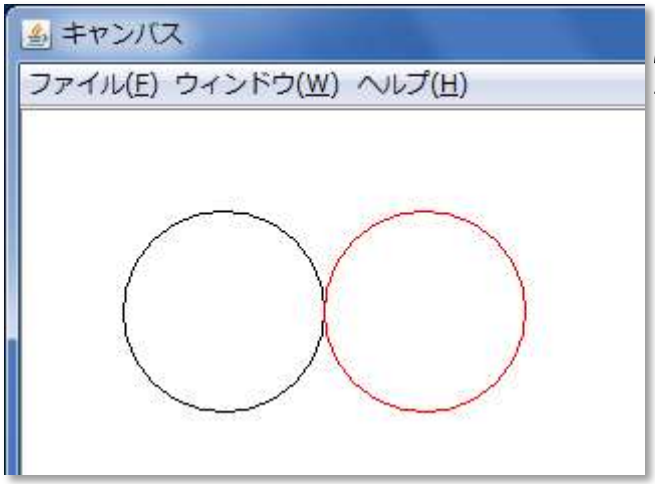
返り値の型	メソッド名(引数)	機能
void	draw()	superを用いて作成し直す。
void	showFields(int row)	superを用いて作成し直す。
void	header()	superを用いて作成し直す。
既存ColoredCircleインスタンスメソッド		
void fill()		

※既存のColoredCircleクラスを編集する。
動作確認クラス:Ex16DrawColoredCircle

例題16(ColoredCircle)

```
1  package j2.lesson06;
2
3  import gpjava.Canvas;
4  import gpjava.Spreadsheet;
5
6  public class ColoredCircle extends MyCircle {
7      int red;
8      int green;
9      int blue;
10
11     ColoredCircle(){}
12
13     // Ex13
14     void fill() {
15         Canvas.setColor(red, green, blue);
16         Canvas.fillOval(x, y, diameter, diameter);
17     }
18
19     //Ex16
20     void header() {
21         super.header();
22         Spreadsheet.setString(0, 3, "色の赤成分");
23         Spreadsheet.setString(0, 4, "色の緑成分");
24         Spreadsheet.setString(0, 5, "色の青成分");
25     }
26
27     //Ex16
28     void showFields(int row){
29         super.showFields(row);
30         Spreadsheet.setInt(row, 3, this.red);
31         Spreadsheet.setInt(row, 4, this.green);
32         Spreadsheet.setInt(row, 5, this.blue);
33     }
34
35     void draw() {
36         Canvas.setColor(red, green, blue);
37         super.draw();
38     }
39 }
```

例題16
(Ex16DrawColoredCircle)




```java
package j2.lesson06;

import gpjava.Canvas;
import gpjava.Spreadsheet;

public class Ex16DrawColoredCircle {
    public static void main(String[]args){
        new Ex16DrawColoredCircle().start();
    }

    void start() {
        Canvas.show();
        Spreadsheet.show(4, 6);

        MyCircle c = create(50, 50, 100);
        c.header();
        c.showFields(1);
        c.draw();

        ColoredCircle cc = create(150, 50, 100, 255, 0, 0);
        cc.header();
        cc.showFields(2);
        cc.draw();
    }

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

    ColoredCircle create(int x, int y, int d, int r, int g, int b){
        ColoredCircle newObj = new ColoredCircle();
        newObj.x = x;
        newObj.y = y;
        newObj.diameter = d;
        newObj.red = r;
        newObj.green = g;
        newObj.blue = b;
        return newObj;
    }
}
```

■例題17

問題: MyCircleおよびColoredCircleのコンストラクタを作成し、動作を確認せよ(既存のcreateメソッドと同等の処理を行う)

コンストラクタ(MyCircle)
MyCircle(int x, int y, int d)

コンストラクタ(ColoredCircle)
ColoredCircle(int x, int y, int d, int r, int g, int b)

※既存のMyCircleクラスを編集する。
※既存のColoredCircleクラスを編集する。
動作確認クラス:Ex17DrawColoredCircle

例題17(MyCircle)

```
1  package j2.lesson06;
2
3  import gpjava.Canvas;
4  import gpjava.Spreadsheet;
5
6  public class MyCircle {
7      int x;
8      int y;
9      int diameter;
10
11     //Ex17
12     MyCircle(int x, int y, int d) {
13         this.x = x;
14         this.y = y;
15         this.diameter = d;
16     }
17
18     MyCircle(){}
19
20     //Ex11
21     void draw() {..
24
25     //Ex11
26     void header() {..
31
32     //Ex11
33     void showFields(int row){..
38 }
```

createメソッドと置き換える。

Ex11と同じ

例題17(ColoredCircle)

```
1  package j2.lesson06;
2
3  import gpjava.Canvas;
4  import gpjava.Spreadsheet;
5
6  public class ColoredCircle extends MyCircle {
7      int red;
8      int green;
9      int blue;
10
11     // Ex17
12     public ColoredCircle(int x, int y, int d, int r, int g, int b){
13         super(x, y, d);
14         this.red = r;
15         this.green = g;
16         this.blue = b;
17     }
18
19     ColoredCircle(){}
20
21     // Ex13
22     void fill() {..
26
27     //Ex16
28     void header() {..
34
35     //Ex16
36     void showFields(int row){..
42
43     // Ex16 override using super
44     void draw() {..
48 }
```

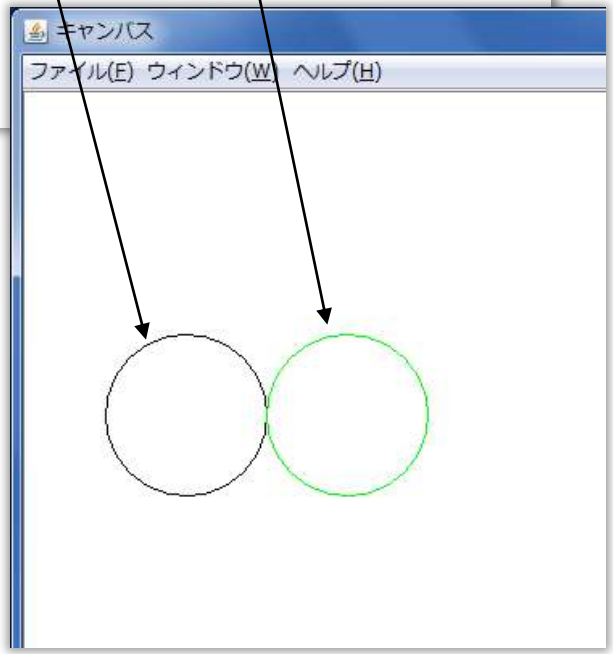
createメソッドと置き換える。

Ex13
Ex16
と同じ

例題17(Ex17DrawColoredCircle)

```
1  package j2.lesson06;
2
3  import gpjava.Canvas;
4
5  public class Ex17DrawColoredCircle {
6      public static void main(String[]args){
7          new Ex17DrawColoredCircle().start();
8      }
9
10     void start() {
11         Canvas.show();
12
13         MyCircle c1 = new MyCircle(50,150,100);
14         c1.draw();
15
16         ColoredCircle c2 = new ColoredCircle(150, 150, 100, 0, 255, 0);
17         c2.draw();
18     }
19
20 }
```

createメソッドの呼び出しをコンストラクタの呼び出しに変更



■例題21

問題：Bookクラスを作成し、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
String title	無し	本のタイトル
Int pages	無し	本のページ数

コンストラクタ
Book(String title, int pages) タイトルとページ数を保存する

インスタンスメソッド

返り値の型	メソッド名(引数)	機能
String	getTitle()	本のタイトルを返す。
int	getPages()	本のページ数を返す。
void	showFields(int row)	本のタイトルとページをSpreadsheet の row 行に表示する。
void	header()	Spreadsheet にヘッダ行を表示する

(クラス名： Book)動作確認クラス:Ex21BookTest

例題21 (Book)

クラス

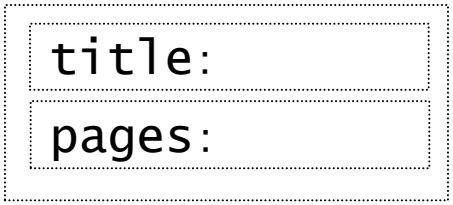
Book


```java
package j2.lesson06;

import gpjava.Spreadsheet;

public class Book {

    // この本のタイトル
    String title;

    // この本のページ数
    int pages;

    // コンストラクタ
    Book(String title, int pages) {
        this.title = title;
        this.pages = pages;
    }

    // タイトルを取得
    String getTitle() {
        return this.title;
    }

    // ページ数を取得
    int getPages() {
        return this.pages;
    }

    // この本の文字列表現を返す
    void showFields(int row) {
        Spreadsheet.setString(row, 0, getTitle());
        Spreadsheet.setInt(row, 1, getPages());
    }

    // この本のためのヘッダを表示する
    void header() {
        Spreadsheet.setString(0, 0, "本のタイトル");
        Spreadsheet.setString(0, 1, "本のページ数");
    }
}

```

行: 27

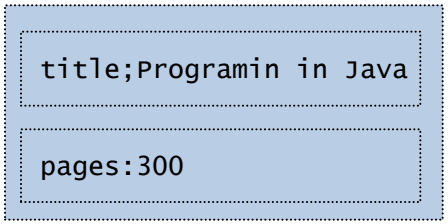
例題21 (Ex21BookTest)

クラスBook



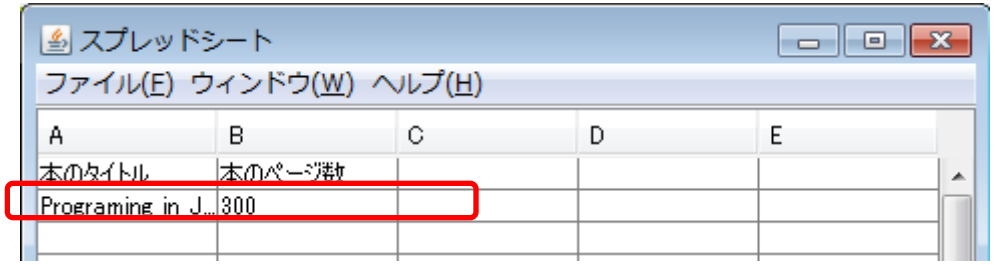
new Book("Programing in Java ", 300);

インスタンスb1




```java
package j2.lesson06;

import gpjava.Spreadsheet;

public class Ex21BookTest {

    public static void main(String[] args) {
        Spreadsheet.show();

        Book b1 = new Book("Programing in Java ", 300);
        b1.header();
        b1.showFields(1);
    }
}
```

b1.tshowFields()



■例題22

問題：Dictionaryクラスを作成し動作を確認せよ。

スーパークラス:　クラスBook

インスタンス変数

変数の型と名前	初期値	説明
int definitions	無し	本の収録語数

コンストラクタ
Dictionary(String title, int pages, int definitions) タイトルとページ数を保存する

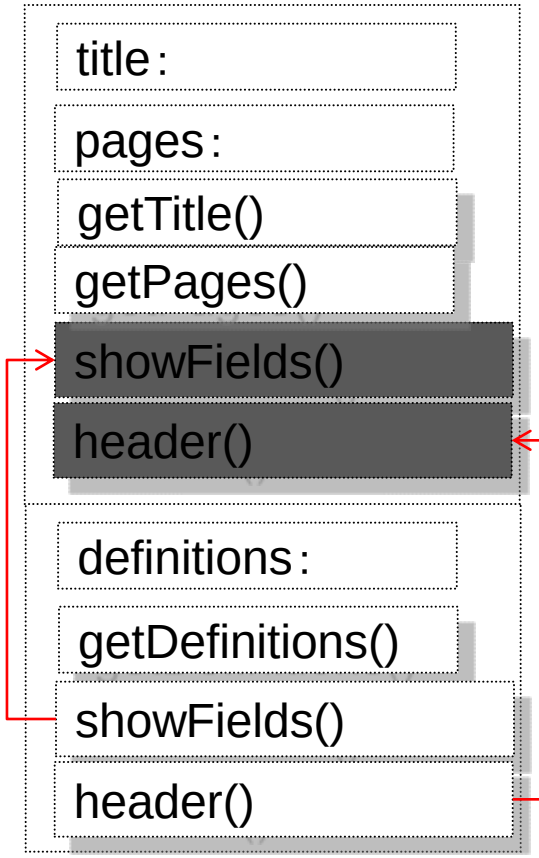
インスタンスメソッド

返り値の型	メソッド名(引数)	機能
int	getDefinitions()	本の収録語数を返す。
void	showFields(int row)	辞書のタイトル、ページ数、収録語数を Spreadsheet の row行に表示する。
void	header()	Spreadsheet のヘッダを表示する

(クラス名：Dictionary) 動作確認クラス:Ex21BookTest

例題22 (Dictionary)

クラス
Dictionary (extends Book)



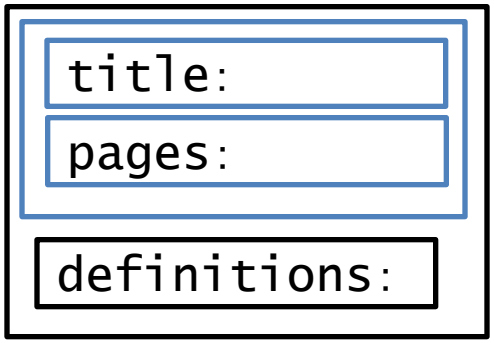

```java
package j2.lesson06;

import gpjava.Spreadsheet;

public class Dictionary extends Book {
    int definitions;

    Dictionary(String title, int pages, int definitions) {
        super(title, pages);
        this.definitions = definitions;
    }

    int getDefinitions() {
        return this.definitions;
    }

    // この本の文字列表現を返す
    void showFields(int row) {
        super.showFields(row);
        Spreadsheet.setInt(row, 2, getDefinitions());
    }

    // この本のためのヘッダを表示する
    void header() {
        super.header();
        Spreadsheet.setString(0, 2, "収録語数");
    }
}
```

オーバーライド

例題22 (Ex21BookTest)

```java
package j2.lesson06;

import gpjava.Spreadsheet;

public class Ex21BookTest {

    public static void main(String[] args) {
        //例題21, 22
        Book b1 = new Book("Programing in Java ", 300);

        Spreadsheet.show();

        // book
        b1.header();
        b1.showFields(1);

        // dictionary
        Dictionary d1 = new Dictionary("国語辞典", 500, 10000);
        d1.header();
        d1.showFields(2);
    }
}
```

クラスDictionary



Dictionary d1 = new Dictionary("国語辞典", 500, 10000);

インスタンス　d1

d1.showFields()

A	B	C	D
本のタイトル	本のページ数	収録語数	
Programing in Java	300		
国語辞典	500	10000	