

pi-7. クラス, メソッド, オブジェクト生成 (コ ンストラクタ)

トピックス: クラス, class, メソッド, コンストラクタ, new, this (Java Tutor による演習)

URL: <https://www.kkaneko.jp/cc/pi/index.html>

(Java の基本)

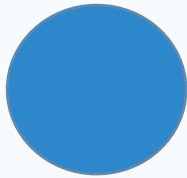
金子邦彦



全体まとめ

クラス Ball

オブジェクト



半径 1, 場所 (8, 10)
色 blue

オブジェクト



半径 3, 場所 (2, 4)
色 green



```
1 class Ball {  
2     double x;  
3     double y;  
4     double r;  
5     String color;  
6     public Ball(double x, double y, double r, String color) {  
7         this.x = x;  
8         this.y = y;  
9         this.r = r;  
10        this.color = color;  
11    }  
12    public void printout() {  
13        System.out.println(this.x);  
14        System.out.println(this.y);  
15        System.out.println(this.r);  
16        System.out.println(this.color);  
17    }  
18 }
```

クラス定義のプログラム

```
Ball a = new Ball(8, 10, 1, "blue");  
Ball b = new Ball(2, 4, 3, "green");
```

オブジェクト生成のプログラム

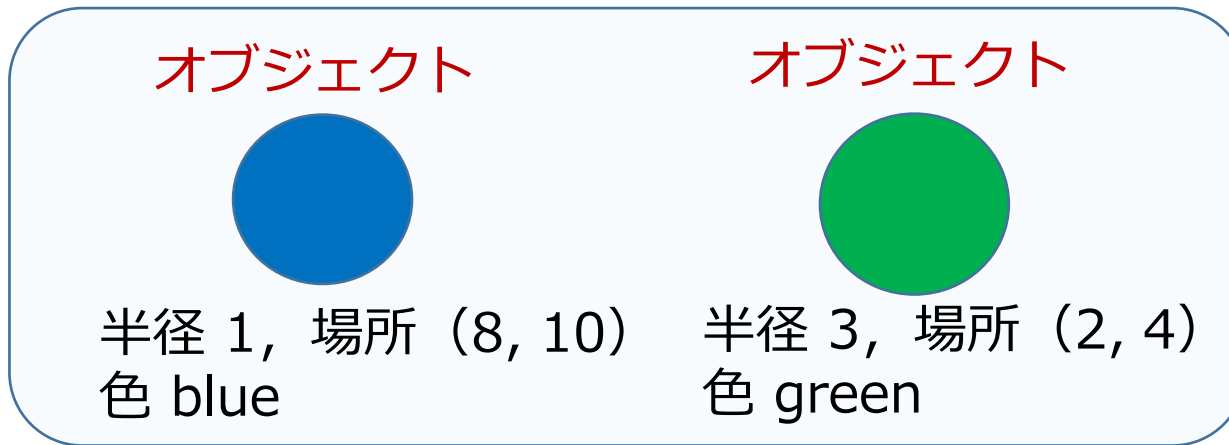
```
a.printout();  
b.printout();
```

メソッドアクセスのプログラム

全体まとめ



- クラスは、同じ種類のオブジェクトの集まりと考えることができる



クラス Ball

- メソッド定義内で、そのメソッドが所属するクラスで定義された属性やメソッドにアクセスするときは this + 「.」

```
public Ball(double x, double y, double r, String color) {  
    this.x = x;  
    this.y = y;  
    this.r = r;  
    this.color = color;  
}
```

	項目
	復習
7-1	クラスとオブジェクト
7-2	クラス定義, <code>class</code> , オブジェクト生成 (コンストラクタ), <code>new</code>
7-3	メソッドアクセス, 属性アクセス, <code>this</code>
7-4	演習

オブジェクトとメソッド



hero.moveDown()

hero **オブジェクト**
moveDown() **メソッド**
間を「.」で区切っている

- **メソッド: オブジェクト**に属する操作や処理.
- **メソッド**呼び出しでは, **引数**を指定することがある. **引数** (ひきすう) は, **メソッド**に渡す値のこと

hero.attack("fence", 36, 26)

Java プログラムの書き方



プログラムの例

```
x = 100  
a = x + 200  
enemy1 = hero.findNearestEnemy()  
hero.attack(enemy1)
```

- **代入** : オブジェクト名 + 「**=**」
+ 式または値またはメソッド呼び出し
- **メソッドアクセス** : オブジェクト名 + 「**.**」
+ **メソッド名** + 「**()**」 (引数を付けることも)

その他, 属性アクセス, 関数呼び出し, 制御, 「*****」, 「**+**」などの演算子, コマンド, 定義など

Java Tutor の起動



① ウェブブラウザを起動する

② **Java Tutor** を使いたいので, 次の URL を開く
<http://www.javatutor.com/>

③ 「**Java**」をクリック ⇒ **編集画面**が開く

Learn Python, JavaScript, C, C++, and Java

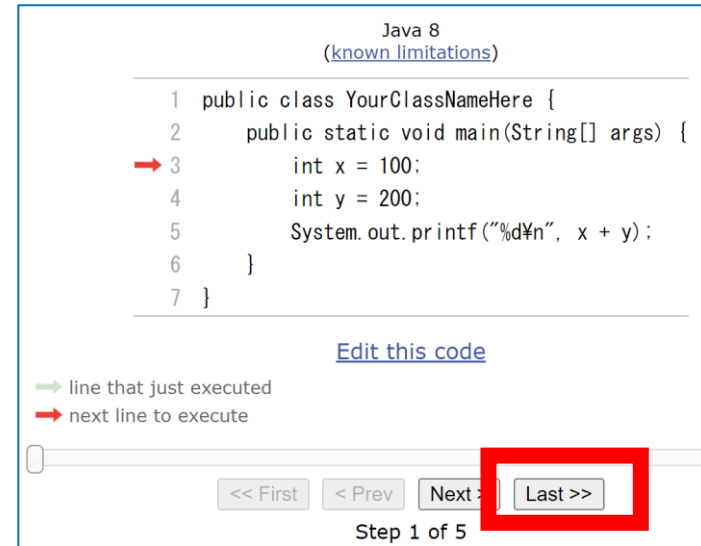
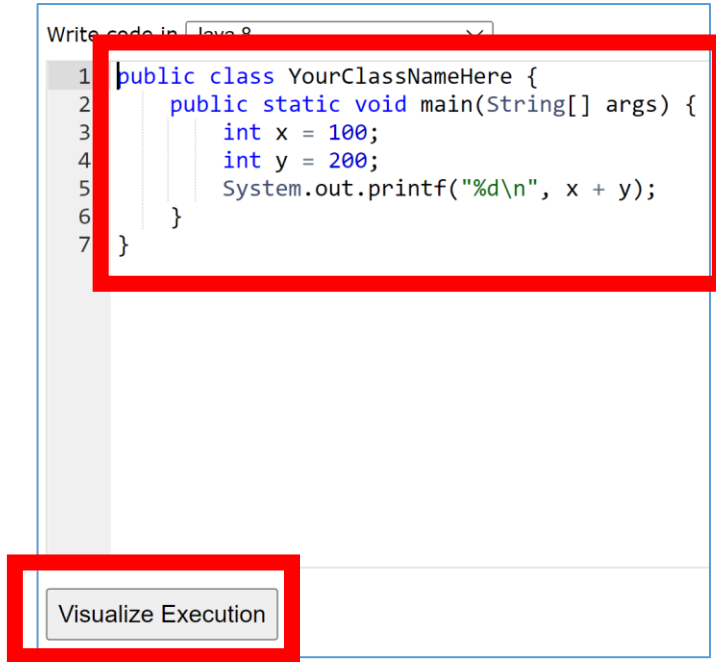
This tool helps you learn Python, JavaScript, C, C++, and Java programming by [visualizing code execution](#). You can use it to debug your homework assignments and as a supplement to online coding tutorials.

Start coding now in [Python](#), [JavaScript](#), [C](#), [C++](#), and [Java](#)

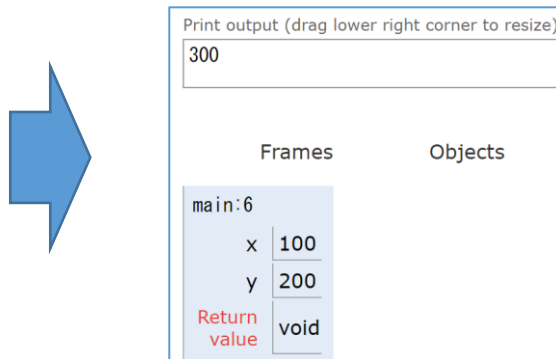
Over 15 million people in more than 180 countries have used Python Tutor to visualize over 200 million pieces of code. It is the most widely-used program visualization tool for computing education.

You can also embed these visualizations into any webpage. Here's an example showing recursion in Python:

Java Tutor でのプログラム実行手順



(1) 「**Visualize Execution**」をクリックして**実行画面**に切り替える



(2) 「**Last**」をクリック。



(3) **実行結果を確認**する。

(4) 「**Edit this code**」をクリックして**編集画面**に戻る

Java Tutor 使用上の注意点①



- 実行画面で、次のような**赤の表示**が出ることがある →
無視してよい

過去の文法ミスに関する確認表示
邪魔なときは「**Close**」

Python Tutor: Visualize code in [Python](#), [JavaScript](#), [C](#), [C++](#), and [Java](#)

The screenshot displays the Python Tutor interface with a Java code snippet. The code is as follows:

```
Java 8  
(known limitations)  
1 public class YourClassNameHere {  
2     public static void main(String[] args) {  
→ 3         int x = 100;  
4     }  
5 }
```

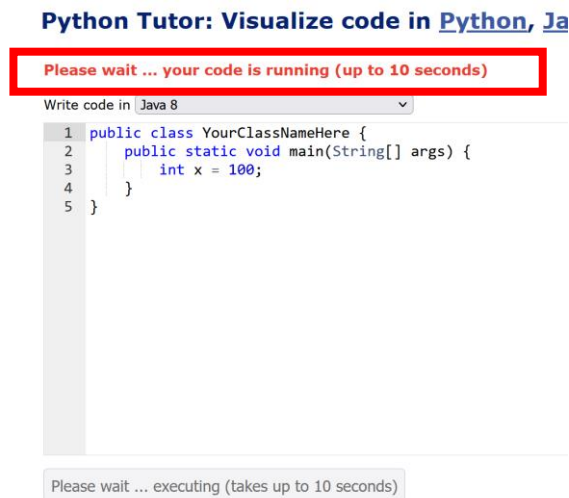
Below the code, there is a legend: a green arrow for "line that just executed" and a red arrow for "next line to execute". Navigation buttons include "<< First", "< Prev", "Next >", and "Last >>". A progress bar shows "Step 1 of 3". A link "Customize visualization" is at the bottom left.

On the right, the "Frames" tab shows "main:3". An error message box is displayed, stating: "You just fixed the following error:" followed by the same code snippet. The error message is "Error: ';' expected". Below the error message, there is a text input field for feedback and two buttons: "Submit" and "Close". The "Close" button is highlighted with a red box. A link "hide all of these pop-ups" is also present.

Java Tutor 使用上の注意点②



「please wait ... executing」のとき， 10秒ほど待つ。



→ 混雑しているときは， 「Server Busy・・・」
というメッセージが出ることがある。

混雑している． 少し（数秒から数十秒）待つと自動で表示が変わる（変わらない場合には，操作をもう一度行ってみる）

Java Tutor でのステップ実行



ステップ実行により, プログラム実行の流れを確認できる

Java 8
([known limitations](#))

```
1 public class YourClassNameHere {  
2     public static void main(String[] args) {  
3         int age = 20;  
4         if (age <= 12) {  
5             System.out.println("500 yen");  
6         } else {  
7             System.out.println("1200 yen");  
8         }  
9     }  
10 }
```

[Edit this code](#)

→ line that just executed

→ next line to execute

Print output (drag lower right corner)

Frames

Objects

main:7

age 20

<< First

< Prev

Next >

Last >>

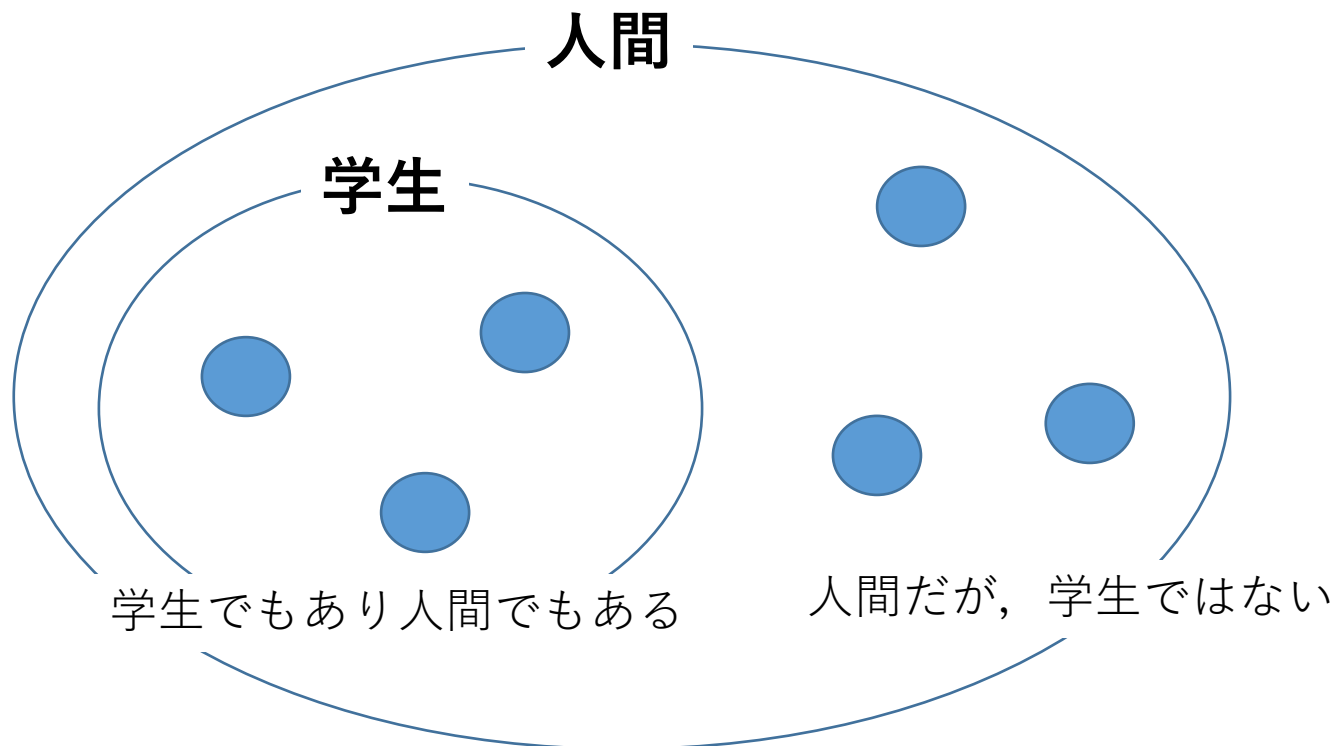
Step 4 of 5

7-1. クラスとオブジェクト

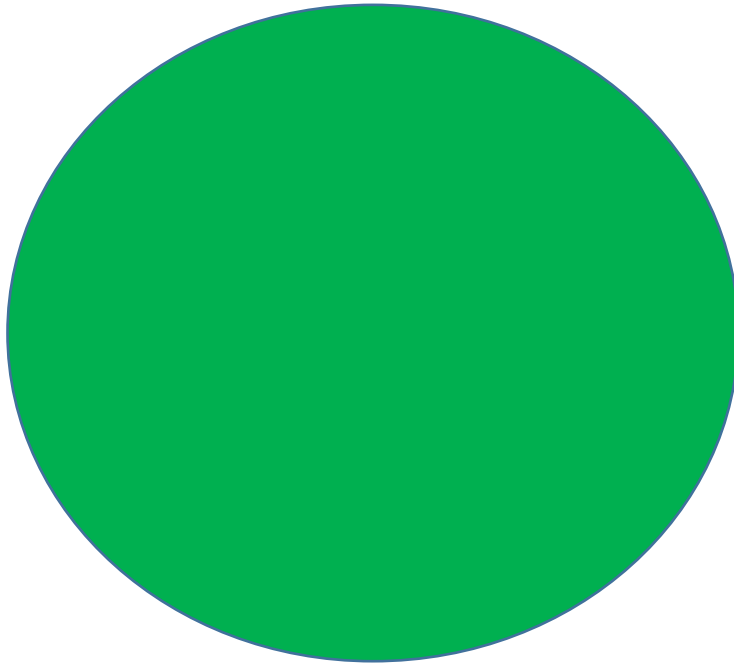
クラスとオブジェクト



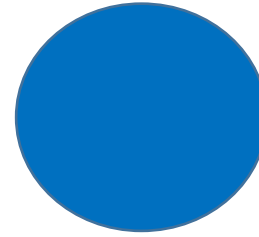
クラスは、同じ種類のオブジェクトの集まりと考えることができる



同じクラスの2つのオブジェクト



半径 3, 場所 (2, 4)
色 green



半径 1, 場所 (8, 10)
色 blue

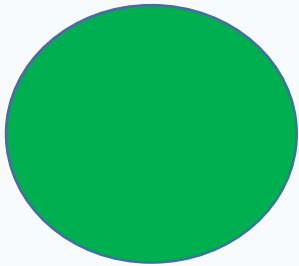
クラス Ball

オブジェクト



半径 1, 場所 (8, 10)
色 blue

オブジェクト



半径 3, 場所 (2, 4)
色 green

- 2つのオブジェクトともに、
同じクラス Ball
と考えることができる
- オブジェクトは属性を持つ。
半径, 場所, 色などの属性

7-2. クラス定義, class, オブジェクト生成 (コンストラク タ), new

クラス定義の例



```
1 class Ball {  
2     double x;  
3     double y;  
4     double r;  
5     String color;  
6     public Ball(double x, double y, double r, String color) {  
7         this.x = x;  
8         this.y = y;  
9         this.r = r;  
10        this.color = color;  
11    }  
12    public void printout() {  
13        System.out.println(this.x);  
14        System.out.println(this.y);  
15        System.out.println(this.r);  
16        System.out.println(this.color);  
17    }  
18 }
```

クラス名: **Ball**

メソッド: **Ball, printout**

属性: **x, y, r, color**

※ Ball は**コンストラクタ**

(オブジェクト生成のためのメソッド)

クラス定義の例

```
1 class Ball { クラス名: Ball
2     double x;
3     double y;
4     double r;
5     String color;
6     public Ball(double x, double y, double r, String color) {
7         this.x = x;
8         this.y = y;
9         this.r = r;
10        this.color = color;
11    }
12    public void printout() {
13        System.out.println(this.x);
14        System.out.println(this.y);
15        System.out.println(this.r);
16        System.out.println(this.color);
17    }
18 }
```

4つの属性

メソッド: Ball

メソッド: printout

このクラス定義を使用した, **オブジェクト**の生成

a	8	10	1	"red"
b	2	4	3	"green"
	x	y	r	color

```
Ball a = new Ball(8, 10, 1, "blue");
Ball b = new Ball(2, 4, 3, "green");
```

メソッドアクセス



```
a.printout();  
b.printout();
```

Javaプログラム

a や b **オブジェクト**
printout() **メソッド**
間を「.」で区切っている

演習

資料 : 21 ~ 27

【トピックス】

- ・ クラス定義
- ・ class
- ・ オブジェクト生成 (コンストラクタ)
- ・ メソッドアクセス

① Java Tutor のエディタで次のプログラムを入れる



クラス定義, オブジェクト生成, メソッドアクセス

```
1  class Ball {  
2      double x;  
3      double y;  
4      double r;  
5      String color;  
6      public Ball(double x, double y, double r, String color) {  
7          this.x = x;  
8          this.y = y;  
9          this.r = r;  
10         this.color = color;  
11     }  
12     public void printout() {  
13         System.out.println(this.x);  
14         System.out.println(this.y);  
15         System.out.println(this.r);  
16         System.out.println(this.color);  
17     }  
18 }
```

クラス定義

字下げにより, プログラムが見やすくなる

```
20 public class YourClassNameHere {  
21     public static void main(String[] args) {  
22         Ball a = new Ball(8, 10, 1, "blue");  
23         Ball b = new Ball(2, 4, 3, "green");  
24         a.printout();  
25         b.printout();  
26     }  
27 }
```

オブジェクト生成

メソッドアクセス

字下げにより，プログラムが見やすくなる

② 実行し，結果を確認する.

```

Java 8
(known limitations)
 8      this.y = y;
 9      this.r = r;
10      this.color = color;
11  }
12  public void printout() {
13      System.out.println(this.x);
14      System.out.println(this.y);
15      System.out.println(this.r);
16      System.out.println(this.color);
17  }
18  }
19
20  public class YourClassNameHere {
21      public static void main(String[] args) {
22          Ball a = new Ball(8, 10, 1, "blue");
23          Ball b = new Ball(2, 4, 3, "green");
24          a.printout();
25          b.printout();
26      }
27  }

```

Print output (drag lower right corner to resize)

```

8.0
10.0
1.0
blue
2.0
4.0
3.0
green

```

メソッド printout
による表示

Frames

main:26	
a	
b	
Return value	void

Objects

x	8.0
y	10.0
r	1.0
color	"blue"

x	2.0
y	4.0
r	3.0
color	"green"

右下をドラッグして
表示枠を広げる

オブジェクト a, b

「**Visual Execution**」をクリック，そして「**Last**」をクリック，結果を確認。
「**Edit this code**」をクリックすると，エディタの画面に戻る

③ 「First」 をクリックして, プログラム実行を先頭に戻す

```

Java 8
(known limitations)
8      this.y = y;
9      this.r = r;
10     this.color = color;
11   }
12   public void printout() {
13     System.out.println(this.x);
14     System.out.println(this.y);
15     System.out.println(this.r);
16     System.out.println(this.color);
17   }
18 }
19
20 public class YourClassNameHere {
21   public static void main(String[] args) {
22     Ball a = new Ball(8, 10, 1, "blue");
23     Ball b = new Ball(2, 4, 3, "green");
24     a.printout();
25     b.printout();
26   }
27 }

```

[Edit this code](#)

→ line that just executed
 → next line to execute

<< First < Prev Next > Last >>

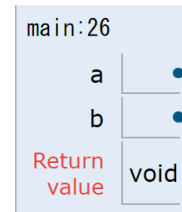
Print output (drag lower right corner to resize)

```

8.0
10.0
1.0
blue
2.0
4.0
3.0
green

```

Frames



Objects

Ball instance

x	8.0
y	10.0
r	1.0
color	"blue"

Ball instance

x	2.0
y	4.0
r	3.0
color	"green"

④ 「**Step 1 of 38**」 と表示されているので、
全部で、ステップ数は **38** あることが分かる
(ステップ数と、プログラムの行数は**違うもの**)

Java 8
(known limitations)

```
8      this.y = y;
9      this.r = r;
10     this.color = color;
11 }
12 public void printout() {
13     System.out.println(this.x);
14     System.out.println(this.y);
15     System.out.println(this.r);
16     System.out.println(this.color);
17 }
18 }
19
20 public class YourClassNameHere {
21     public static void main(String[] args) {
22         Ball a = new Ball(8, 10, 1, "blue");
23         Ball b = new Ball(2, 4, 3, "green");
24         a.printout();
25         b.printout();
26     }
27 }
```

Edit this code

→ line that just executed

→ next line to execute

<< First

Step 1 of 38

Next >>

Print output (drag lower right corner to resize)

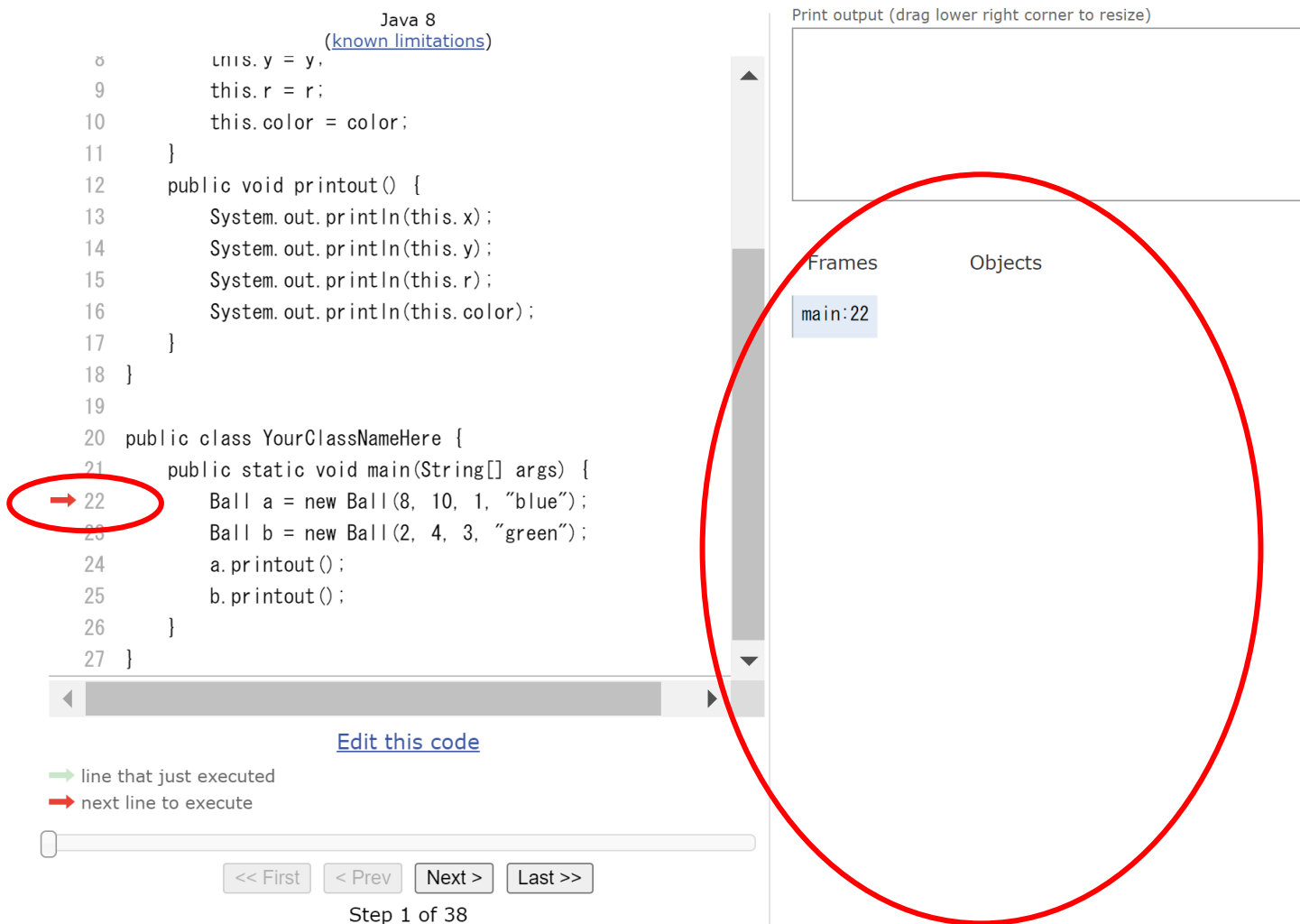
Frames

Objects

main:22

⑤ 先頭に戻したので

- すべての**オブジェクト**は消えている
- **赤い矢印**は main メソッドの**先頭**に戻っている



Java 8
(known limitations)

```
8      this.y = y;  
9      this.r = r;  
10     this.color = color;  
11 }  
12 public void printout() {  
13     System.out.println(this.x);  
14     System.out.println(this.y);  
15     System.out.println(this.r);  
16     System.out.println(this.color);  
17 }  
18 }  
19  
20 public class YourClassNameHere {  
21     public static void main(String[] args) {  
22         Ball a = new Ball(8, 10, 1, "blue");  
23         Ball b = new Ball(2, 4, 3, "green");  
24         a.printout();  
25         b.printout();  
26     }  
27 }
```

Print output (drag lower right corner to resize)

Frames Objects

main: 22

[Edit this code](#)

→ line that just executed
→ next line to execute

<< First < Prev Next > Last >>

Step 1 of 38

⑥ **ステップ実行**したいので、「**Next**」をクリックしながら、**矢印の動きを確認**しなさい。



※「**Next**」ボタンを何度か押し、それ以上進めなくなったら終了

あとで使うので、プログラムを消さずに残しておくこと

ジャンプ
している

Java 8 (known limitations)

```
double x;  
String color;  
public Ball(double x, double y, double r, String color) {  
    this.x = x;  
    this.y = y;  
    this.r = r;  
    this.color = color;  
}  
public void printout() {  
    System.out.println(this.x);  
    System.out.println(this.y);  
    System.out.println(this.r);  
    System.out.println(this.color);  
}  
public class YourClassNameHere {  
    public static void main(String[] args) {  
        Ball a = new Ball(8, 10, 1, "blue");  
        Ball b = new Ball(2, 4, 3, "green");  
    }  
}
```

Print output (drag lower right corner to resize)

Frames

main:22

<init>:6

this

x 8.0

y 10.0

r 1.0

color "blue"

Objects

Ball instance

x	0.0
y	0.0
r	0.0
color	null

Edit this code

→ line that just executed

→ next line to execute

Step 3 of 38

実行が進むと、
オブジェクトが更新される

まとめ



- **クラス定義**では、**クラス名の指定**と、**メソッド定義**を行う。

```
1 class Ball { クラス名: Ball
2     double x;
3     double y;
4     double r;
5     String color;
6     public Ball(double x, double y, double r, String color) {
7         this.x = x;
8         this.y = y;
9         this.r = r;
10        this.color = color;
11    }
12    public void printout() {
13        System.out.println(this.x);
14        System.out.println(this.y);
15        System.out.println(this.r);
16        System.out.println(this.color);
17    }
18 }
```

4つの属性

メソッド: Ball

メソッド: printout

- キーワード

class **クラス**

7-3. メソッドアクセス, 属性アクセス, this

メソッドと属性



- **メソッド**や**属性**は, **クラス**に属する
- **メソッド**内のプログラムは, その**メソッド**が所属する**クラス**の**属性**や**メソッド**への**アクセス権がある**

this による属性アクセス, メソッドアクセス



- メソッド定義内で, そのメソッドが所属するクラスで定義された属性やメソッドにアクセスするときは this + 「.」

```
public Ball(double x, double y, double r, String color) {  
    this.x = x;  
    this.y = y;  
    this.r = r;  
    this.color = color;  
}
```

- メソッド外では「オブジェクト名」 + 「.」

```
a.printout();  
b.printout();
```

演習

資料 : 33 ~ 34

【トピックス】

- ・ this によるアクセス

① Java Tutor のエディタで次のプログラムを追加



```
1 class Ball {
2     double x;
3     double y;
4     double r;
5     String color;
6     public Ball(double x, double y, double r, String color) {
7         this.x = x;
8         this.y = y;
9         this.r = r;
10        this.color = color;
11    }
12    public void printout() {
13        System.out.println(this.x);
14        System.out.println(this.y);
15        System.out.println(this.r);
16        System.out.println(this.color);
17    }
18    public double dist() {
19        return this.x + this.y;
20    }
21 }
```

【追加】
メソッド定義内の属性アクセス

```
public double dist() {
    return this.x + this.y;
}
```

```
22
23 public class YourClassNameHere {
24     public static void main(String[] args) {
25         Ball a = new Ball(8, 10, 1, "blue");
26         Ball b = new Ball(2, 4, 3, "green");
27         a.printout();
28         b.printout();
29         System.out.println(a.dist());
30     }
31 }
```

【追加】
メソッド外のメソッドアクセス

```
System.out.println(a.dist());
```

② 実行し，結果を確認する.



あとで使うので，プログラムを消さずに残しておくこと

Java 8
([known limitations](#))

```
12 public void printout() {  
13     System.out.println(this.x);  
14     System.out.println(this.y);  
15     System.out.println(this.r);  
16     System.out.println(this.color);  
17 }  
18 public double dist() {  
19     return this.x + this.y;  
20 }  
21 }  
22  
23 public class YourClassNameHere {  
24     public static void main(String[] args) {  
25         Ball a = new Ball(8, 10, 1, "blue");  
26         Ball b = new Ball(2, 4, 3, "green");  
27         a.printout();  
28         b.printout();  
29         System.out.println(a.dist());  
30     }  
31 }
```

Print output (drag lower right corner to resize)

```
8.0  
10.0  
1.0  
blue  
2.0  
4.0  
3.0  
green  
18.0
```

右下をドラッグして
表示枠を広げる

表示が増える

Frames

main:30

a

b

Return value

void

Objects

Ball instance

x	8.0
y	10.0
r	1.0
color	"blue"

Ball instance

x	2.0
y	4.0
r	3.0
color	"green"

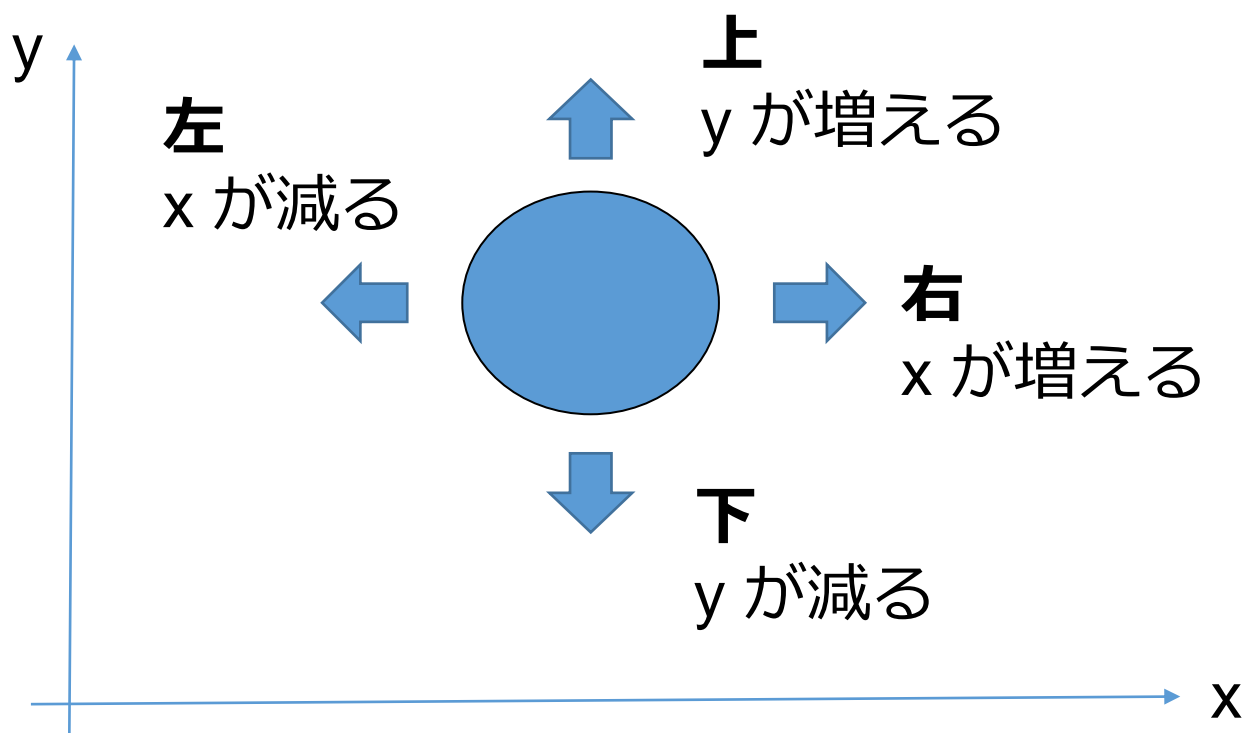
[Edit this code](#)

「**Visual Execution**」をクリック，そして「**Last**」をクリック，結果を確認。
「**Edit this code**」をクリックすると，エディタの画面に戻る

7-4. 演習

メソッド

- 上下左右の移動を考える
- **オブジェクト**の属性 x , y を増減
- そのためのメソッド `move` を定義



演習

資料 : 38 ~ 41

【トピックス】

- メソッド定義
- 「.」を用いたメソッドへのアクセス

① Java Tutor のエディタで次のプログラムを追加



```
1 class Ball {
2     double x;
3     double y;
4     double r;
5     String color;
6     public Ball(double x, double y, double r, String color) {
7         this.x = x;
8         this.y = y;
9         this.r = r;
10        this.color = color;
11    }
12    public void printout() {
13        System.out.println(this.x);
14        System.out.println(this.y);
15        System.out.println(this.r);
16        System.out.println(this.color);
17    }
18    public double dist() {
19        return this.x + this.y;
20    }
21    public void move(double xx, double yy) {
22        this.x = this.x + xx;
23        this.y = this.y + yy;
24    }
25 }
```

```
27 public class YourClassNameHere {
28     public static void main(String[] args) {
29         Ball a = new Ball(8, 10, 1, "blue");
30         Ball b = new Ball(2, 4, 3, "green");
31         a.move(5, 5);
32         b.move(10, 10);
33         a.printout();
34         b.printout();
35         System.out.println(a.dist());
36     }
37 }
```

メソッド定義

```
public void move(double xx, double yy) {
    this.x = this.x + xx;
    this.y = this.y + yy;
}
```

「.」でメソッドにアクセス

```
a.move(5, 5);
b.move(10, 10);
```

② 実行し，結果を確認する.



あとで使うので，プログラムを消さずに残しておくこと

```
Java 8
(known limitations)
18 public double dist() {
19     return this.x + this.y;
20 }
21 public void move(double xx, double yy) {
22     this.x = this.x + xx;
23     this.y = this.y + yy;
24 }
25 }
26
27 public class YourClassNameHere {
28     public static void main(String[] args) {
29         Ball a = new Ball(8, 10, 1, "blue");
30         Ball b = new Ball(2, 4, 3, "green");
31         a.move(5, 5);
32         b.move(10, 10);
33         a.printout();
34         b.printout();
35         System.out.println(a.dist());
36     }
37 }
```

[Edit this code](#)

→ line that just executed

Print output (drag lower right corner to resize)

13.0
15.0
1.0
blue
12.0
14.0
3.0
green
28.0

右下をドラッグして
表示枠を広げる

表示が変わる

Frames

main:36	
a	
b	
Return value	void

Objects

Ball instance

x	13.0
y	15.0
r	1.0
color	"blue"

表示が変わる

Ball instance

x	12.0
y	14.0
r	3.0
color	"green"

表示が変わる

「**Visual Execution**」をクリック，そして「**Last**」をクリック，結果を確認。
「**Edit this code**」をクリックすると，エディタの画面に戻る

演習問題



- 右に動かすためのメソッド `right` を定義
- 左に動かすためのメソッド `left` を定義
- `right` を使って, オブジェクト `a` を右に 5 動かす
- `Left` を使って, オブジェクト `b` を左に 10 動かす

演習問題の解答例



右に移動するメソッド right, 左に移動するメソッド left

```
1 class Ball {
2     double x;
3     double y;
4     double r;
5     String color;
6     public Ball(double x, double y, double r, String color) {
7         this.x = x;
8         this.y = y;
9         this.r = r;
10        this.color = color;
11    }
12    public void printout() {
13        System.out.println(this.x);
14        System.out.println(this.y);
15        System.out.println(this.r);
16        System.out.println(this.color);
17    }
18    public double dist() {
19        return this.x + this.y;
20    }
21    public void move(double xx, double yy) {
22        this.x = this.x + xx;
23        this.y = this.y + yy;
24    }
25    public void right(double xx) {
26        this.move(xx, 0);
27    }
28    public void left(double xx) {
29        this.move(-xx, 0);
30    }
31 }
32
33 public class YourClassNameHere {
34     public static void main(String[] args) {
35         Ball a = new Ball(8, 10, 1, "blue");
36         Ball b = new Ball(2, 4, 3, "green");
37         a.move(5, 5);
38         b.move(10, 10);
39         a.left(5);
40         b.right(10);
41         a.printout();
42         b.printout();
43         System.out.println(a.dist());
44     }
45 }
```

メソッド定義

```
public void right(double xx) {
    this.move(xx, 0);
}

public void left(double xx) {
    this.move(-xx, 0);
}
```

「.」でメソッドにアクセス

```
a.left(5);
b.right(10);
```

実行し，結果を確認してみる



Java 8
([known limitations](#))

```
25 public void right(double xx) {
26     this.move(xx, 0);
27 }
28 public void left(double xx) {
29     this.move(-xx, 0);
30 }
31 }
32
33 public class YourClassNameHere {
34     public static void main(String[] args) {
35         Ball a = new Ball(8, 10, 1, "blue");
36         Ball b = new Ball(2, 4, 3, "green");
37         a.move(5, 5);
38         b.move(10, 10);
39         a.left(5);
40         b.right(10);
41         a.printout();
42         b.printout();
43         System.out.println(a.dist());
```

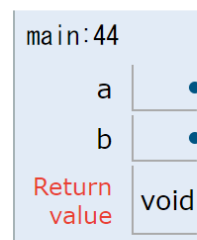
[Edit this code](#)

→ line that just executed
→ next line to execute

Print output (drag lower right corner to resize)

```
8.0
15.0
1.0
blue
22.0
14.0
3.0
green
23.0
```

Frames



Objects

Ball instance

x	8.0
y	15.0
r	1.0
color	"blue"

Ball instance

x	22.0
y	14.0
r	3.0
color	"green"

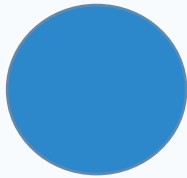
「**Visual Execution**」をクリック。そして「**Last**」をクリック。結果を確認。
「**Edit this code**」をクリックすると，エディタの画面に戻る

上や下に動かすためのメソッドを、
メソッド left, right を参考に作ってみなさい。

全体まとめ

クラス Ball

オブジェクト



半径 1, 場所 (8, 10)
色 blue

オブジェクト



半径 3, 場所 (2, 4)
色 green



```
1 class Ball {  
2     double x;  
3     double y;  
4     double r;  
5     String color;  
6     public Ball(double x, double y, double r, String color) {  
7         this.x = x;  
8         this.y = y;  
9         this.r = r;  
10        this.color = color;  
11    }  
12    public void printout() {  
13        System.out.println(this.x);  
14        System.out.println(this.y);  
15        System.out.println(this.r);  
16        System.out.println(this.color);  
17    }  
18 }
```

クラス定義のプログラム

```
Ball a = new Ball(8, 10, 1, "blue");  
Ball b = new Ball(2, 4, 3, "green");
```

オブジェクト生成のプログラム

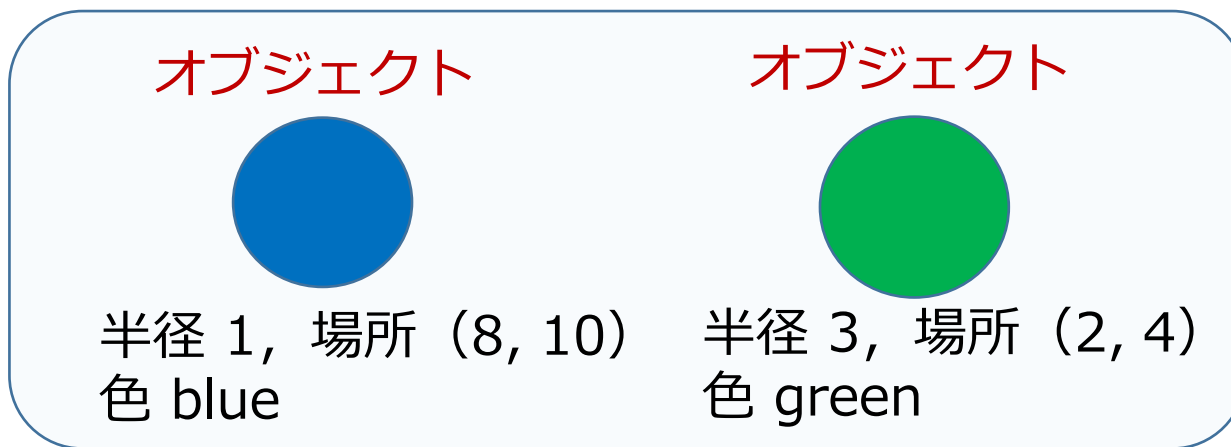
```
a.printout();  
b.printout();
```

メソッドアクセスのプログラム

全体まとめ



- **クラス**は、同じ種類のオブジェクトの集まりと考えることができる



クラス Ball

- メソッド定義内で、その**メソッド**が所属する**クラス**で定義された**属性**や**メソッド**にアクセスするときは **this + 「.」**

```
public Ball(double x, double y, double r, String color) {  
    this.x = x;  
    this.y = y;  
    this.r = r;  
    this.color = color;  
}
```

関連ページ

- **Java プログラミング入門**

GDB online を使用

<https://www.kkaneko.jp/cc/ji/index.html>

- **Java の基本**

Java Tutor, GDB online を使用

<https://www.kkaneko.jp/cc/pi/index.html>

- **Java プログラム例**

<https://www.kkaneko.jp/pro/java/index.html>

```
class Ball {  
    double x;  
    double y;  
    double r;  
    String color;  
    public Ball(double x, double y, double r, String color) {  
        this.x = x;  
        this.y = y;  
        this.r = r;  
        this.color = color;  
    }  
    public void printout() {  
        System.out.println(this.x);  
        System.out.println(this.y);  
        System.out.println(this.r);  
        System.out.println(this.color);  
    }  
}
```

```
public class YourClassNameHere {  
    public static void main(String[] args) {  
        Ball a = new Ball(8, 10, 1, "blue");  
        Ball b = new Ball(2, 4, 3, "green");  
        a.printout();  
        b.printout();  
    }  
}
```

資料中のソースコード 7-3



```
class Ball {
    double x;
    double y;
    double r;
    String color;
    public Ball(double x, double y, double r, String color) {
        this.x = x;
        this.y = y;
        this.r = r;
        this.color = color;
    }
    public void printout() {
        System.out.println(this.x);
        System.out.println(this.y);
        System.out.println(this.r);
        System.out.println(this.color);
    }
    public double dist() {
        return this.x + this.y;
    }
}

public class YourClassNameHere {
    public static void main(String[] args) {
        Ball a = new Ball(8, 10, 1, "blue");
        Ball b = new Ball(2, 4, 3, "green");
        a.printout();
        b.printout();
        System.out.println(a.dist());
    }
}
```


資料中のソースコード 7-4



```
class Ball {
    double x;
    double y;
    double r;
    String color;
    public Ball(double x, double y, double r, String color) {
        this.x = x;
        this.y = y;
        this.r = r;
        this.color = color;
    }
    public void printout() {
        System.out.println(this.x);
        System.out.println(this.y);
        System.out.println(this.r);
        System.out.println(this.color);
    }
    public double dist() {
        return this.x + this.y;
    }
    public void move(double xx, double yy) {
        this.x = this.x + xx;
        this.y = this.y + yy;
    }
}

public class YourClassNameHere {
    public static void main(String[] args) {
        Ball a = new Ball(8, 10, 1, "blue");
        Ball b = new Ball(2, 4, 3, "green");
        a.move(5, 5);
        b.move(10, 10);
        a.printout();
        b.printout();
        System.out.println(a.dist());
    }
}
```

資料中のソースコード 7-4



```
class Ball {
    double x;
    double y;
    double r;
    String color;
    public Ball(double x, double y, double r, String color) {
        this.x = x;
        this.y = y;
        this.r = r;
        this.color = color;
    }
    public void printout() {
        System.out.println(this.x);
        System.out.println(this.y);
        System.out.println(this.r);
        System.out.println(this.color);
    }
    public double dist() {
        return this.x + this.y;
    }
    public void move(double xx, double yy) {
        this.x = this.x + xx;
        this.y = this.y + yy;
    }
    public void right(double xx) {
        this.move(xx, 0);
    }
    public void left(double xx) {
        this.move(-xx, 0);
    }
}

public class YourClassNameHere {
    public static void main(String[] args) {
        Ball a = new Ball(8, 10, 1, "blue");
        Ball b = new Ball(2, 4, 3, "green");
        a.move(5, 5);
        b.move(10, 10);
        a.left(5);
        b.right(10);
        a.printout();
        b.printout();
        System.out.println(a.dist());
    }
}
```