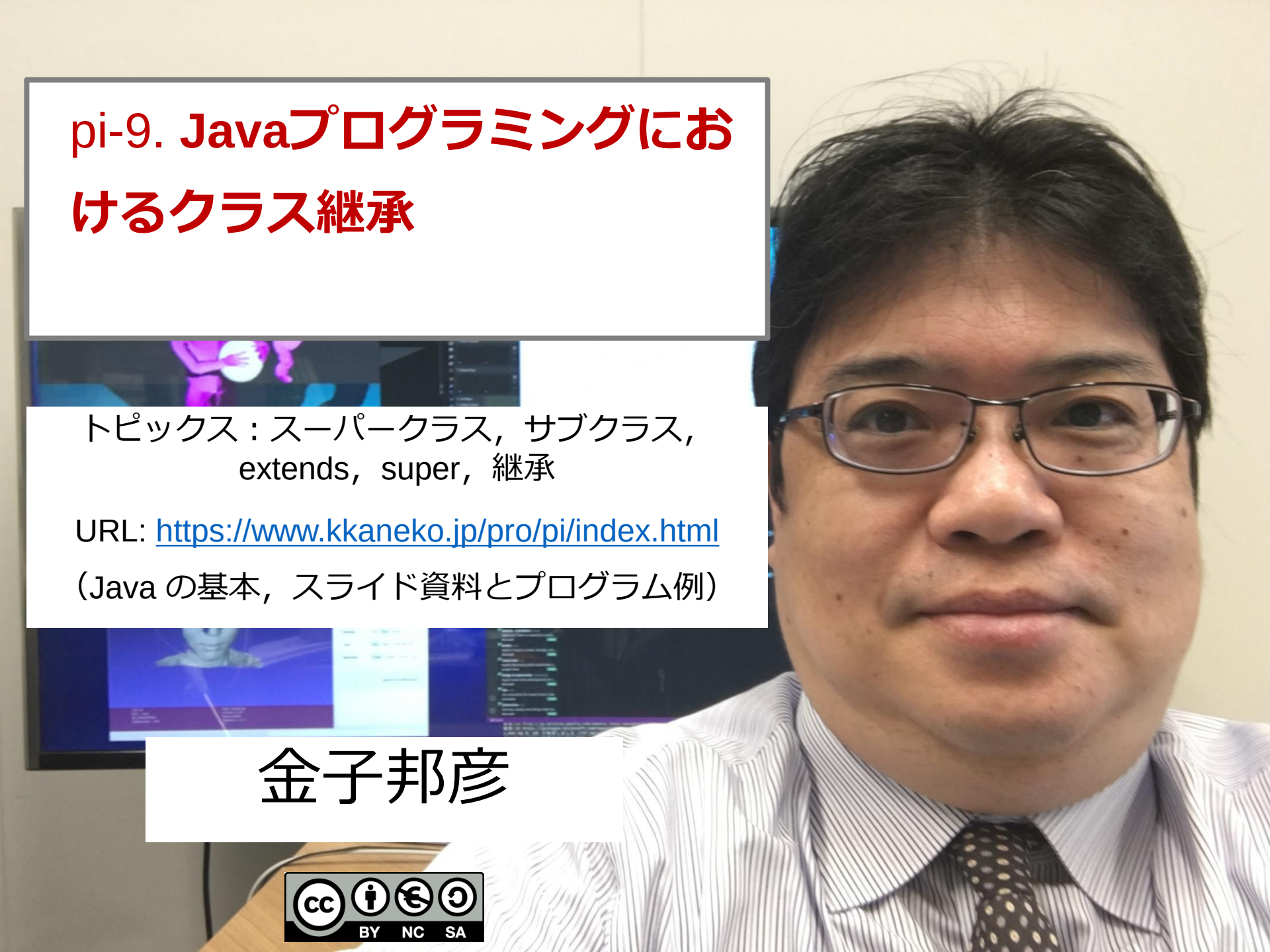
pi-9. Javaプログラミングにおけるクラス継承



トピックス：スーパークラス，サブクラス，
extends，super，継承

URL: https://www.kkaneko.jp/pro/pi/index.html

（Java の基本，スライド資料とプログラム例）

金子邦彦

全体まとめ

- **サブクラス**の**オブジェクト**は，すべて**スーパークラス**に属する
- **継承**：**スーパークラス**の**属性**と**メソッド**を**サブクラス**が**受け継ぐ**

  但し，コンストラクタは**受け継がない**
- コンストラクタ**以外**の**メソッドは，オーバーライドできる**
- **サブクラス**で，**スーパークラス**にない**属性**や**メソッド**を**追加できる**

アウトライン

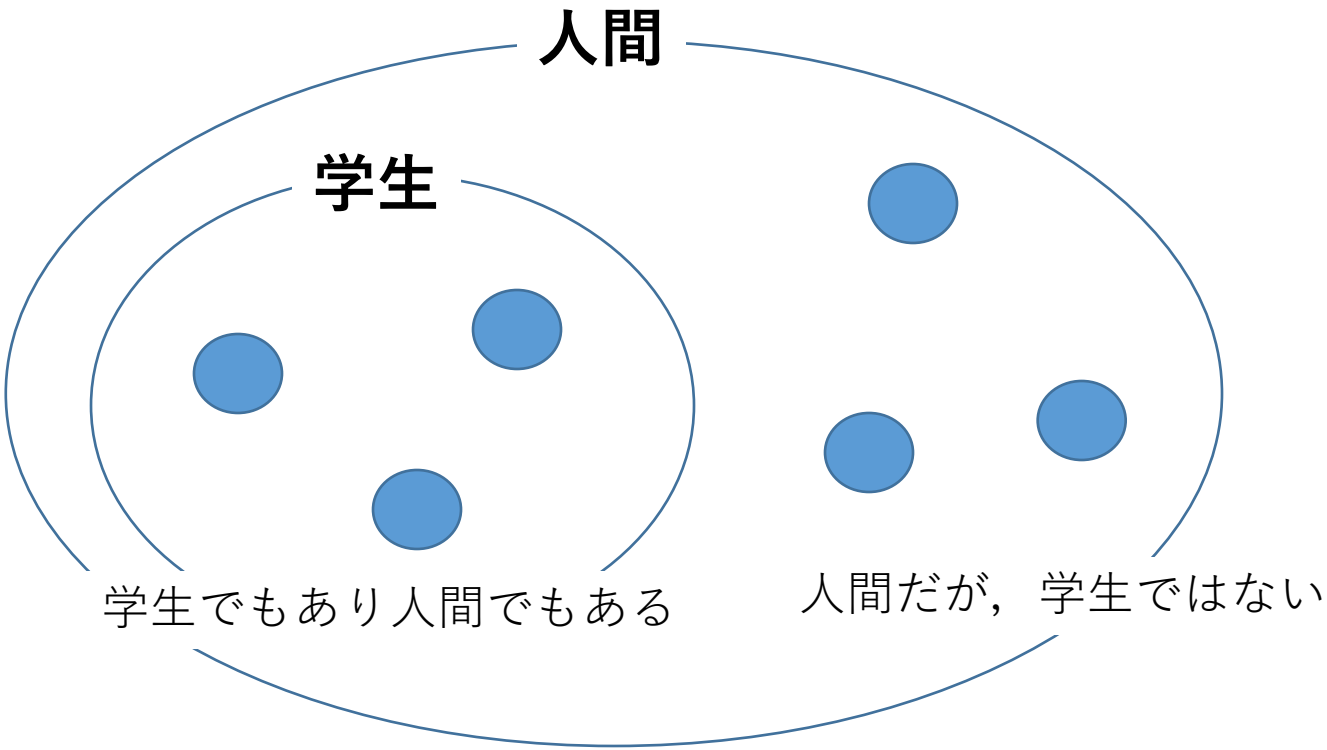
番号	項目
	復習
9-1	スーパークラス，サブクラス
9-2	スーパークラス，サブクラスの Java プログラム，継承
9-3	メソッドのオーバーライド
9-4	サブクラスでの属性の追加

各自、資料を読み返したり、課題に取り組んだりも行う

この授業では、**Java** を用いて基礎を学び、マスターする

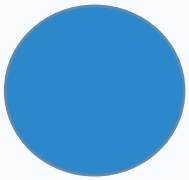
クラスとオブジェクト

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



Java のクラス
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

クラス定義のプログラム

```
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

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");
```

Javaプログラム

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

Java Tutor の起動

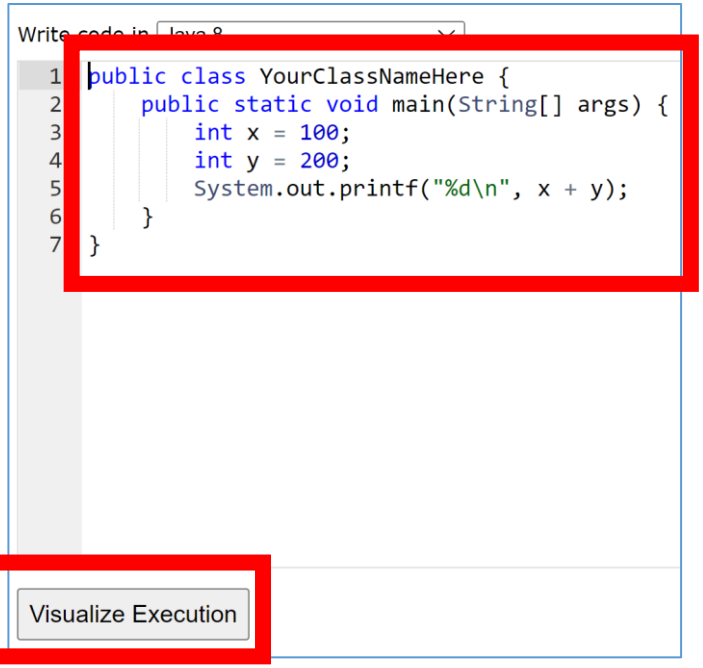
① **ウェブブラウザ**を起動する

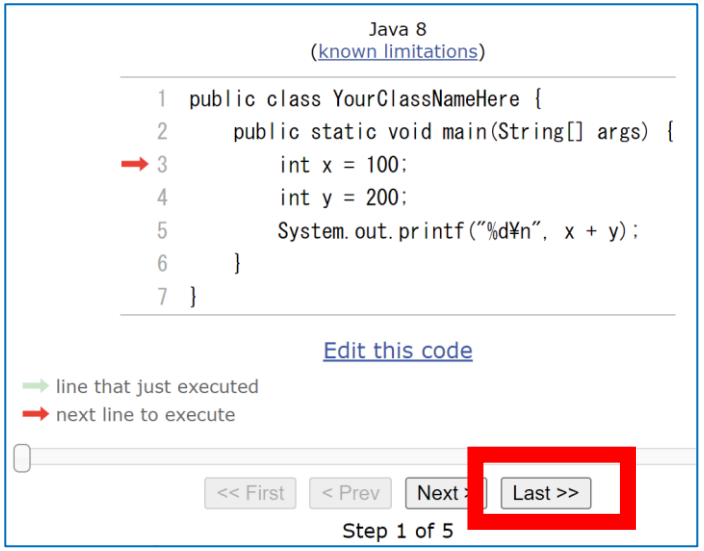
② **Java Tutor** を使いたいので，次の URL を開く

http://www.pythontutor.com/

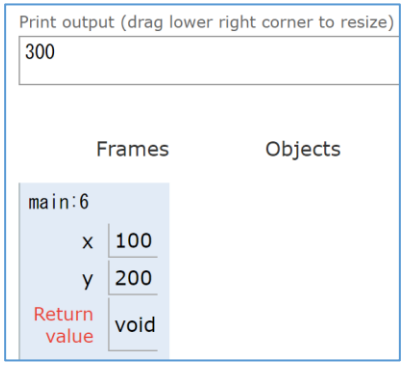
③ 「**Java**」をクリック　⇒ **編集画面**が開く

Java Tutor でのプログラム実行手順



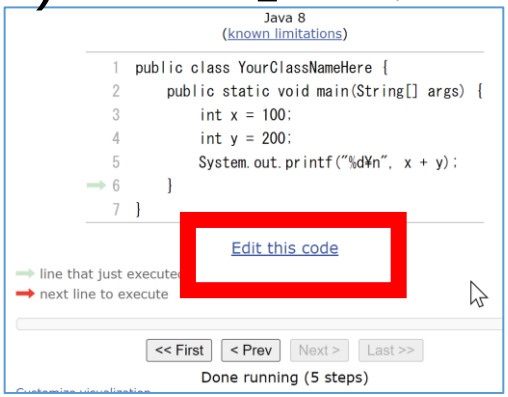


(1)「**Visualize Execution**」をクリックして**実行画面**に切り替える



(3) **<u>実行結果を確認</u>**する.

(2)「**Last**」をクリックする.



(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

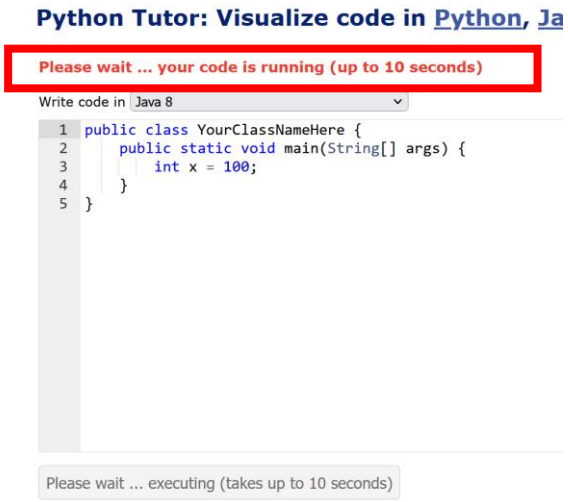
- **実行画面で，次のような赤の表示が出ることがある → 無視してよい**

  過去の文法ミスに関する確認表示

  邪魔なときは「**Close**」

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）

9-1. スーパークラス，サブクラス

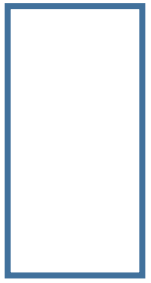
スーパークラスとサブクラス

正方形は，**長方形**である

（**正方形**ではない**長方形**もある）

長方形：　スーパークラス
正方形：　サブクラス

長方形と正方形

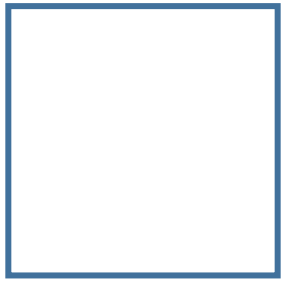


幅 1，高さ 2，
場所（4, 8）

幅 2，高さ 1，
場所（8, 10）

幅 1，高さ 1，
場所（0, 3）

幅 3，高さ 1，
場所（7, 5）



幅 2，高さ 2，
場所（1, 1）

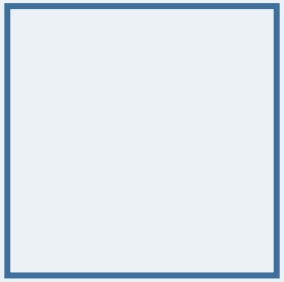
長方形と正方形

長方形であり，
正方形でもある

幅 1，高さ 2，
場所（4, 8）

幅 2，高さ 1，
場所（8, 10）

幅 1，高さ 1，
場所（0, 3）



幅 2，高さ 2，
場所（1, 1）

幅 3，高さ 1，
場所（7, 5）

長方形であるが，
正方形ではない

スーパークラスとサブクラス

「**長方形**」のクラス

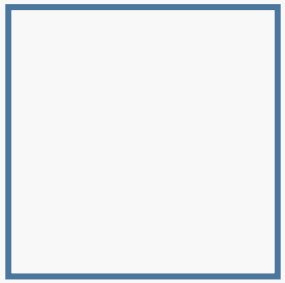
幅 2，高さ 1，
場所（8, 10）

幅 1，高さ 2，
場所（4, 8）

「**正方形**」のクラス

幅 1，高さ 1，
場所（0, 3）

幅 3，高さ 1



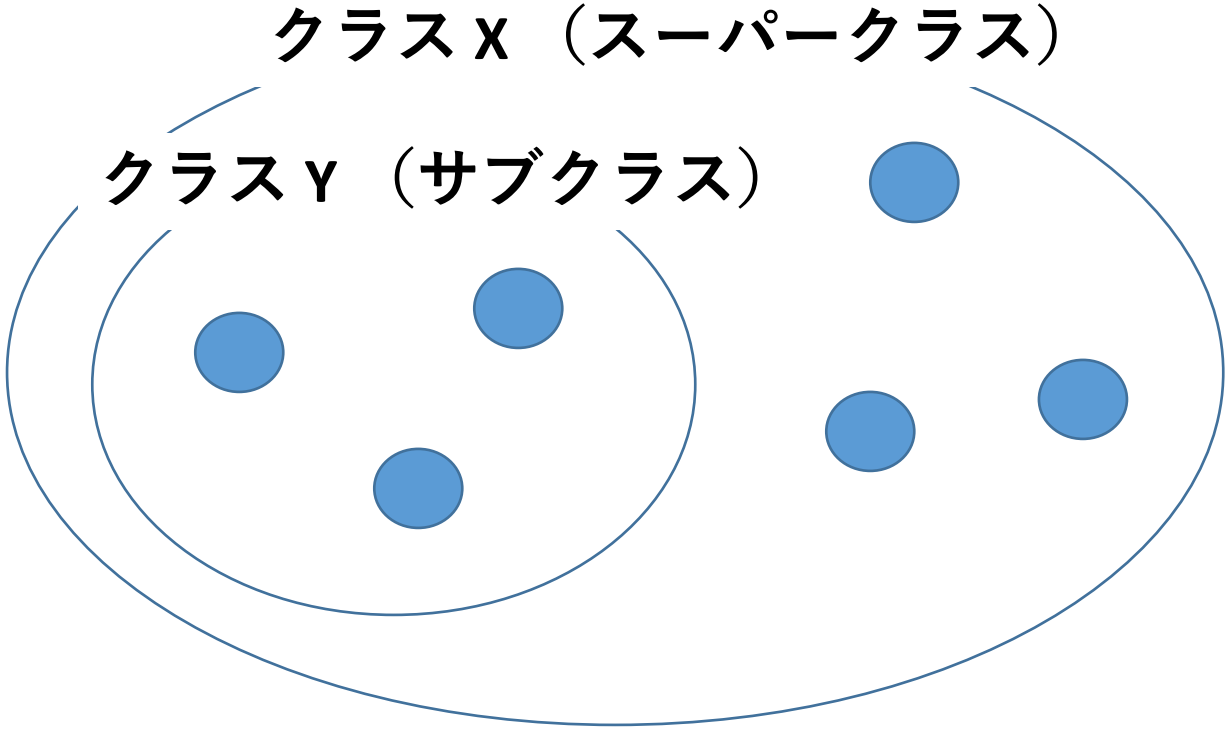
幅 2，高さ 2，
場所（1, 1）

「**正方形**」**ではない**長方形もある

「**正方形**」はすべて長方形である

スーパークラス，サブクラス

サブクラスの**オブジェクト**は，すべて**スーパークラス**に属する



スーパークラス，サブクラスの例

例①

くだもの	りんご
スーパークラス	サブクラス

例②

住民	世帯主
スーパークラス	サブクラス

例③

図形	円
スーパークラス	サブクラス

9-2. スーパークラス，サブクラスの Java プログラム，継承

Java でのクラス定義

- class **Y** **extends X** : クラス Y の**スーパークラス**はX であることを指定
- **super** : **スーパークラス**のコンストラクタの呼び出し

```
class X {
   ・・・
   public X(・・・) {
      ・・・
   }
    ・・・
}
```

クラス定義

```
class Y extends X {
  ・・・
   public Y(・・・) {
     super(・・・)
      ・・・
   }
    ・・・
}
```

クラス定義
Y のスーパークラスが
X である場合

「長方形」のクラス定義の例

クラス名: Rectangle

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

4つの属性

メソッド: Rectangle

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

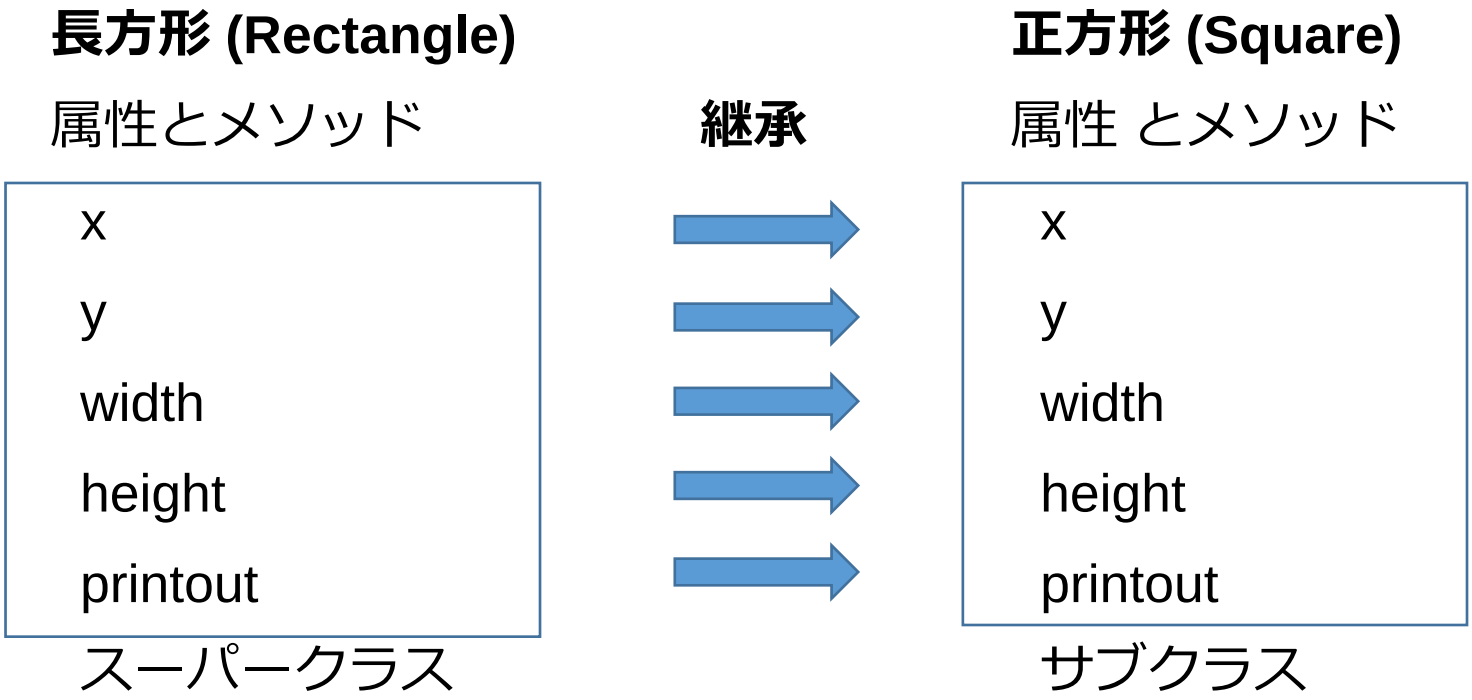
	x	y	width	height
a	4	8	1	2
b	8	10	2	1

```
Rectangle a = new Rectangle(4, 8, 1, 2);
Rectangle b = new Rectangle(8, 10, 2, 1);
```

Javaプログラム

継承

- **継承**：**スーパークラス**の**属性**と**メソッド**を**サブクラス**が**受け継ぐ**
- 但し，コンストラクタは**受け継がない**
  （コンストラクタは必ず定義する必要がある）



「正方形」のクラス定義の例

クラス名: Square　　スーパークラス: Rectangle

メソッド: Square

```
19  class Square extends Rectangle {
20      public Square(double x, double y, double size) {
21          super(x, y, size, size);
22      }
23  }
```

このクラス定義を使用した，**オブジェクト**の生成

	x	y	width	height
c	0	3	1	1
d	1	1	2	2

```
Square c = new Square(0, 3, 1);
Square d = new Square(1, 1, 2);
```

Javaプログラム

キーワード

extends　**スーパークラス**の指定
super　**スーパークラス**の**コンストラクタ**の呼び出し

演習

資料：**26 ～ 28**

【トピックス】

- **スーパークラス**
- **サブクラス**
- **extends**
- **super**

① Java Tutor のエディタで次のプログラムを入れる

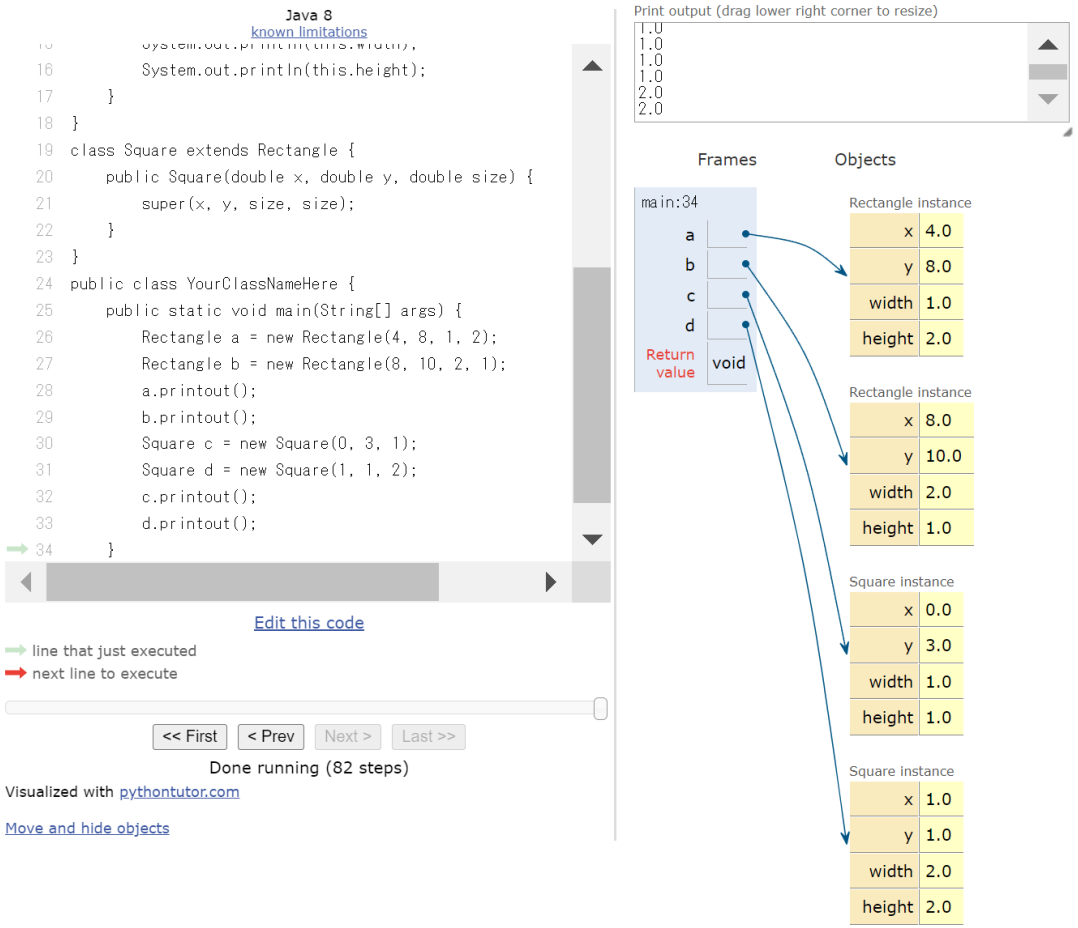

```
1  class Rectangle {
2      double x;
3      double y;
4      double width;
5      double height;
6      public Rectangle(double x, double y, double width, double height) {
7          this.x = x;
8          this.y = y;
9          this.width = width;
10         this.height = height;
11     }
12  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

② 引き続き，次のプログラムを入れる


```
19  class Square extends Rectangle {
20      public Square(double x, double y, double size) {
21          super(x, y, size, size);
22      }
23  }
24  public class YourClassNameHere {
25      public static void main(String[] args) {
26          Rectangle a = new Rectangle(4, 8, 1, 2);
27          Rectangle b = new Rectangle(8, 10, 2, 1);
28          a.printout();
29          b.printout();
30          Square c = new Square(0, 3, 1);
31          Square d = new Square(1, 1, 2);
32          c.printout();
33          d.printout();
34      }
35  }
```

③ 実行し，結果を確認する

（あとで使うので、プログラムを消さないこと）



４つのオブジェクト
a, b, c, d が生成される

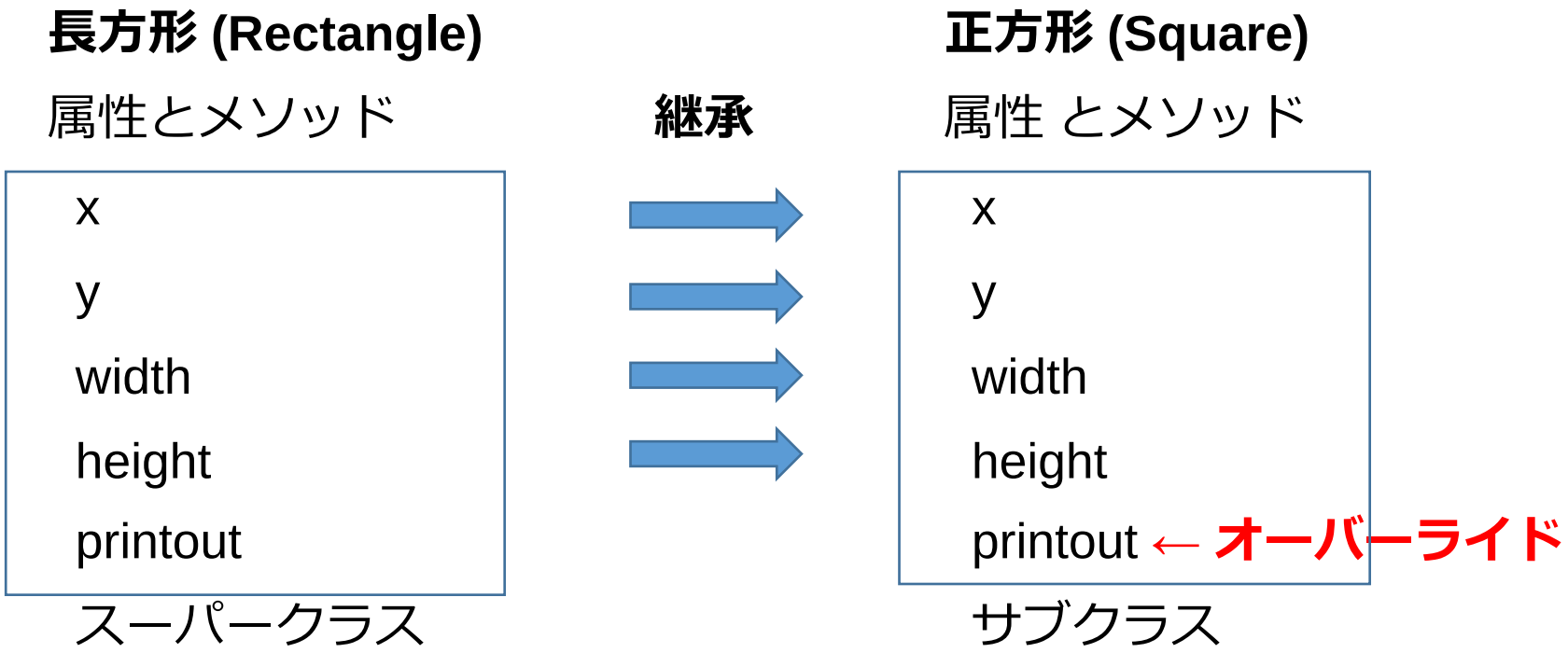
「**Visual Execution**」をクリック．そして「**Last**」をクリック．結果を確認．
「**Edit this code**」をクリックすると，エディタの画面に戻る

9-3. メソッドのオーバーライド

メソッドのオーバーライド

- コンストラクタ**以外**の**メソッド**は，**受け継がずに**
  オーバーライドできる

  オーバーライドは，別の定義を行うこと



「正方形」のクラス定義の例

クラス名: Square　　スーパークラス: Rectangle

メソッド: Square

```
19  class Square extends Rectangle {
20      public Square(double x, double y, double size) {
21          super(x, y, size, size);
22      }
23      public void printout() {
24          System.out.println(this.x);
25          System.out.println(this.y);
26          System.out.println(this.width);
27      }
28  }
```

メソッド: printout

- メソッド printout のオーバーライド
- スーパークラスのメソッド printout とは別の定義を行う。

演習

資料：**33 ～ 34**

【トピックス】

・メソッドのオーバーライド

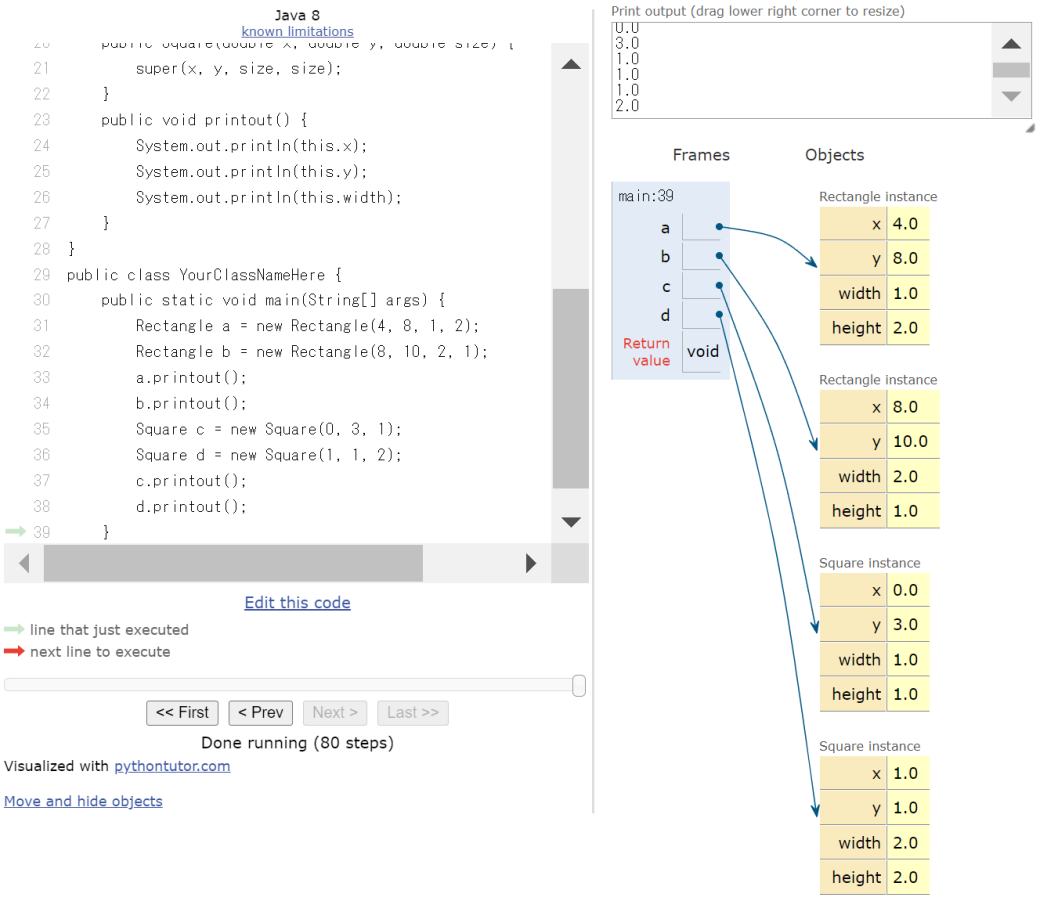
① Java Tutor のエディタで次のプログラムを加える


```java
class Rectangle {
    double x;
    double y;
    double width;
    double height;
    public Rectangle(double x, double y, double width, double height) {
        this.x = x;
        this.y = y;
        this.width = width;
        this.height = height;
    }
    public void printout() {
        System.out.println(this.x);
        System.out.println(this.y);
        System.out.println(this.width);
        System.out.println(this.height);
    }
}
class Square extends Rectangle {
    public Square(double x, double y, double size) {
        super(x, y, size, size);
    }
    public void printout() {
        System.out.println(this.x);
        System.out.println(this.y);
        System.out.println(this.width);
    }
}
public class YourClassNameHere {
    public static void main(String[] args) {
        Rectangle a = new Rectangle(4, 8, 1, 2);
        Rectangle b = new Rectangle(8, 10, 2, 1);
        a.printout();
        b.printout();
        Square c = new Square(0, 3, 1);
        Square d = new Square(1, 1, 2);
        c.printout();
        d.printout();
    }
}
```

```java
public void printout() {
    System.out.println(this.x);
    System.out.println(this.y);
    System.out.println(this.width);
}
```

② 実行し，結果を確認する

（あとで使うので、プログラムを消さないこと）



表示が変わる

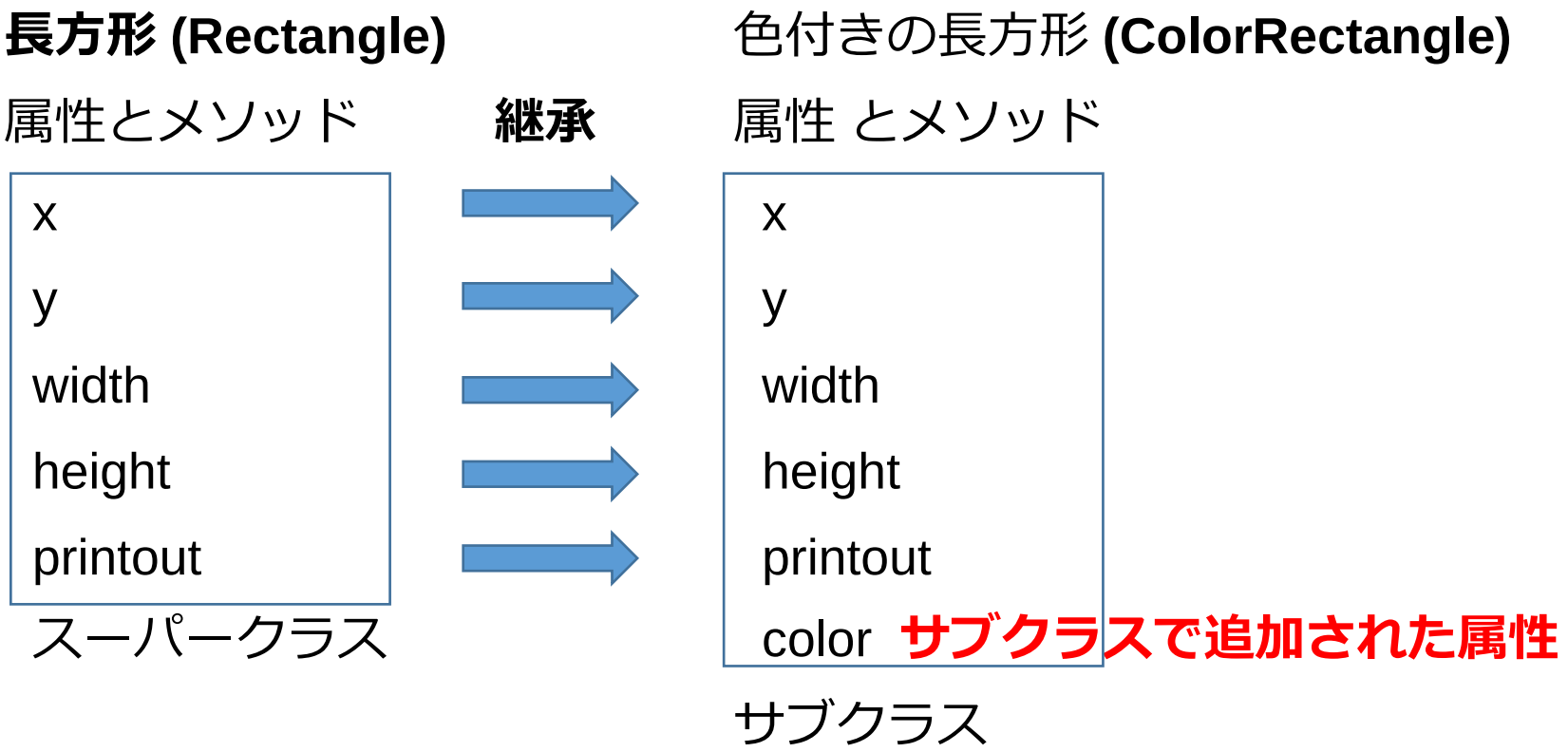
４つのオブジェクト
a, b, c, d が生成される

「**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

9-4. サブクラスでの属性の追加

サブクラスでの属性，メソッドの追加

- **サブクラス**で，**スーパークラス**にない**属性**や**メソッド**を**追加できる**



「色付きの長方形」のクラス定義の例

クラス名: ColorRectangle　スーパークラス: Rectangle

メソッド: Square

```
29  class ColorRectangle extends Rectangle {
30      String color;
31      public ColorRectangle(double x, double y, double width, double heigh, String color) {
32          super(x, y, width, height);
33          this.color = color;
34      }
35      public void printout() {
36          System.out.println(this.x);
37          System.out.println(this.y);
38          System.out.println(this.width);
39          System.out.println(this.height);
40          System.out.println(this.color);
41      }
42  }
```

メソッド: printout

- サブクラスで，属性 color を追加

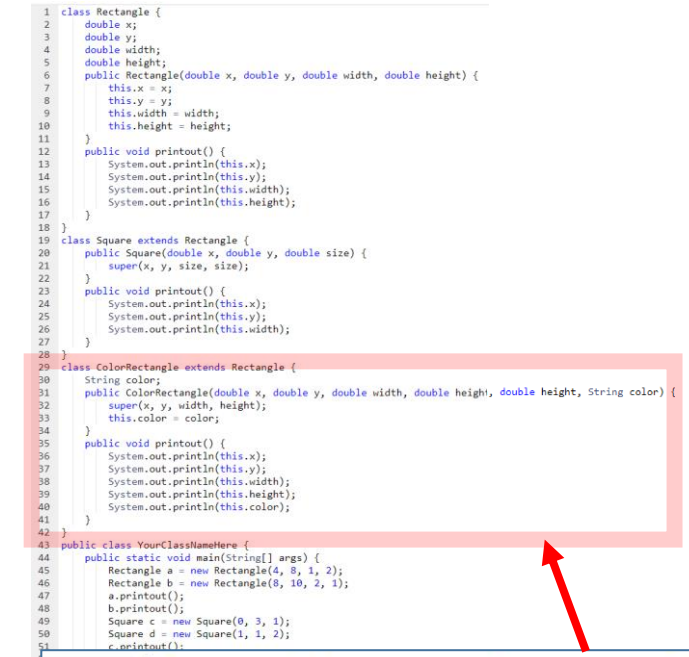
演習

資料：**39 ～ 41**

【トピックス】

- **サブクラスでの属性の追加**

① Java Tutor のエディタで次のプログラムを**加える**



```
class Rectangle {
    double x;
    double y;
    double width;
    double height;
    public Rectangle(double x, double y, double width, double height) {
        this.x = x;
        this.y = y;
        this.width = width;
        this.height = height;
    }
    public void printout() {
        System.out.println(this.x);
        System.out.println(this.y);
        System.out.println(this.width);
        System.out.println(this.height);
    }
}
class Square extends Rectangle {
    public Square(double x, double y, double size) {
        super(x, y, size, size);
    }
    public void printout() {
        System.out.println(this.x);
        System.out.println(this.y);
        System.out.println(this.width);
    }
}
class ColorRectangle extends Rectangle {
    String color;
    public ColorRectangle(double x, double y, double width, double heigh!, double height, String color) {
        super(x, y, width, height);
        this.color = color;
    }
    public void printout() {
        System.out.println(this.x);
        System.out.println(this.y);
        System.out.println(this.width);
        System.out.println(this.height);
        System.out.println(this.color);
    }
}
public class YourClassNameHere {
    public static void main(String[] args) {
        Rectangle a = new Rectangle(4, 8, 1, 2);
        Rectangle b = new Rectangle(8, 10, 2, 1);
        a.printout();
        b.printout();
        Square c = new Square(0, 3, 1);
        Square d = new Square(1, 1, 2);
        c.printout();
```

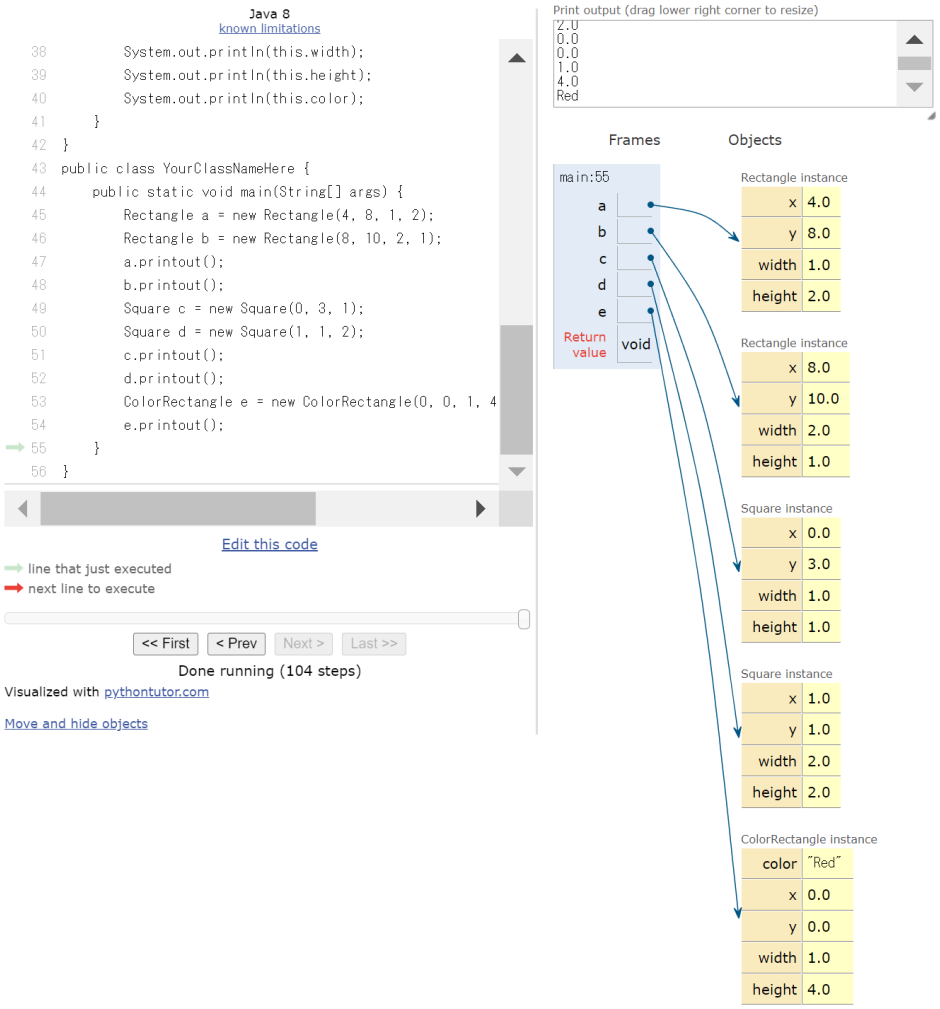
```
29  class ColorRectangle extends Rectangle {
30      String color;
31      public ColorRectangle(double x, double y, double width, double heigh, String color) {
32          super(x, y, width, height);
33          this.color = color;
34      }
35      public void printout() {
36          System.out.println(this.x);
37          System.out.println(this.y);
38          System.out.println(this.width);
39          System.out.println(this.height);
40          System.out.println(this.color);
41      }
42  }
```

② 引き続き，次のプログラムを**加える**

```
class Rectangle {
    double x;
    double y;
    double width;
    double height;
    public Rectangle(double x, double y, double width, double height) {
        this.x = x;
        this.y = y;
        this.width = width;
        this.height = height;
    }
    public void printout() {
        System.out.println(this.x);
        System.out.println(this.y);
        System.out.println(this.width);
        System.out.println(this.height);
    }
}
class Square extends Rectangle {
    public Square(double x, double y, double size) {
        super(x, y, size, size);
    }
    public void printout() {
        System.out.println(this.x);
        System.out.println(this.y);
        System.out.println(this.width);
    }
}
class ColorRectangle extends Rectangle {
    String color;
    public ColorRectangle(double x, double y, double width, double height, double height, String color) {
        super(x, y, width, height);
        this.color = color;
    }
    public void printout() {
        System.out.println(this.x);
        System.out.println(this.y);
        System.out.println(this.width);
        System.out.println(this.height);
        System.out.println(this.color);
    }
}
public class YourClassNameHere {
    public static void main(String[] args) {
        Rectangle a = new Rectangle(4, 8, 1, 2);
        Rectangle b = new Rectangle(8, 10, 2, 1);
        a.printout();
        b.printout();
        Square c = new Square(0, 3, 1);
        Square d = new Square(1, 1, 2);
        c.printout();
        d.printout();
        ColorRectangle e = new ColorRectangle(0, 0, 1, 4, "Red");
        e.printout();
    }
}
```

```
ColorRectangle e = new ColorRectangle(0, 0, 1, 4, "Red");
e.printout();
```

③ 実行し，結果を確認する



5つのオブジェクト
a, b, c, d, e が生成される

「**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

全体まとめ

- **サブクラス**の**オブジェクト**は，すべて**スーパークラス**に属する
- **継承**：**スーパークラス**の**属性**と**メソッド**を**サブクラス**が**受け継ぐ**

  但し，コンストラクタは**受け継がない**
- コンストラクタ**以外**の**メソッドは，オーバーライドできる**
- **サブクラス**で，**スーパークラス**にない**属性**や**メソッド**を**追加できる**

関連ページ

- **Java プログラミング入門**

GDB online を使用

https://www.kkaneko.jp/pro/ji/index.html

- **Java の基本**

Java Tutor, GDB online を使用

https://www.kkaneko.jp/pro/pi/index.html

- **Java プログラム例**

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

ソースコード (9-2)


```
class Rectangle {
    double x;
    double y;
    double width;
    double height;
    public Rectangle(double x, double y, double width, double height) {
        this.x = x;
        this.y = y;
        this.width = width;
        this.height = height;
    }
    public void printout() {
        System.out.println(this.x);
        System.out.println(this.y);
        System.out.println(this.width);
        System.out.println(this.height);
    }
}
class Square extends Rectangle {
    public Square(double x, double y, double size) {
        super(x, y, size, size);
    }
}
public class YourClassNameHere {
    public static void main(String[] args) {
        Rectangle a = new Rectangle(4, 8, 1, 2);
        Rectangle b = new Rectangle(8, 10, 2, 1);
        a.printout();
        b.printout();
        Square c = new Square(0, 3, 1);
        Square d = new Square(1, 1, 2);
        c.printout();
        d.printout();
    }
}
```

ソースコード (9-3)

```
class Rectangle {
    double x;
    double y;
    double width;
    double height;
    public Rectangle(double x, double y, double width, double height) {
        this.x = x;
        this.y = y;
        this.width = width;
        this.height = height;
    }
    public void printout() {
        System.out.println(this.x);
        System.out.println(this.y);
        System.out.println(this.width);
        System.out.println(this.height);
    }
}
class Square extends Rectangle {
    public Square(double x, double y, double size) {
        super(x, y, size, size);
    }
    public void printout() {
        System.out.println(this.x);
        System.out.println(this.y);
        System.out.println(this.width);
    }
}
public class YourClassNameHere {
    public static void main(String[] args) {
        Rectangle a = new Rectangle(4, 8, 1, 2);
        Rectangle b = new Rectangle(8, 10, 2, 1);
        a.printout();
        b.printout();
        Square c = new Square(0, 3, 1);
        Square d = new Square(1, 1, 2);
        c.printout();
        d.printout();
    }
}
```

ソースコード (9-4)


```
class Rectangle {
    double x;
    double y;
    double width;
    double height;
    public Rectangle(double x, double y, double width, double height) {
        this.x = x;
        this.y = y;
        this.width = width;
        this.height = height;
    }
    public void printout() {
        System.out.println(this.x);
        System.out.println(this.y);
        System.out.println(this.width);
        System.out.println(this.height);
    }
}
class Square extends Rectangle {
    public Square(double x, double y, double size) {
        super(x, y, size, size);
    }
    public void printout() {
        System.out.println(this.x);
        System.out.println(this.y);
        System.out.println(this.width);
    }
}
class ColorRectangle extends Rectangle {
    String color;
    public ColorRectangle(double x, double y, double width, double height, String color) {
        super(x, y, width, height);
        this.color = color;
    }
    public void printout() {
        System.out.println(this.x);
        System.out.println(this.y);
        System.out.println(this.width);
        System.out.println(this.height);
        System.out.println(this.color);
    }
}
public class YourClassNameHere {
    public static void main(String[] args) {
        Rectangle a = new Rectangle(4, 8, 1, 2);
        Rectangle b = new Rectangle(8, 10, 2, 1);
        a.printout();
        b.printout();
        Square c = new Square(0, 3, 1);
        Square d = new Square(1, 1, 2);
        c.printout();
        d.printout();
        ColorRectangle e = new ColorRectangle(0, 0, 1, 4, "Red");
        e.printout();
    }
}
```