# ソフトウェア工学実習
# Software Engineering Practice
# (第05回)

## SEP05-002 【実習編】NetBeans-Counter

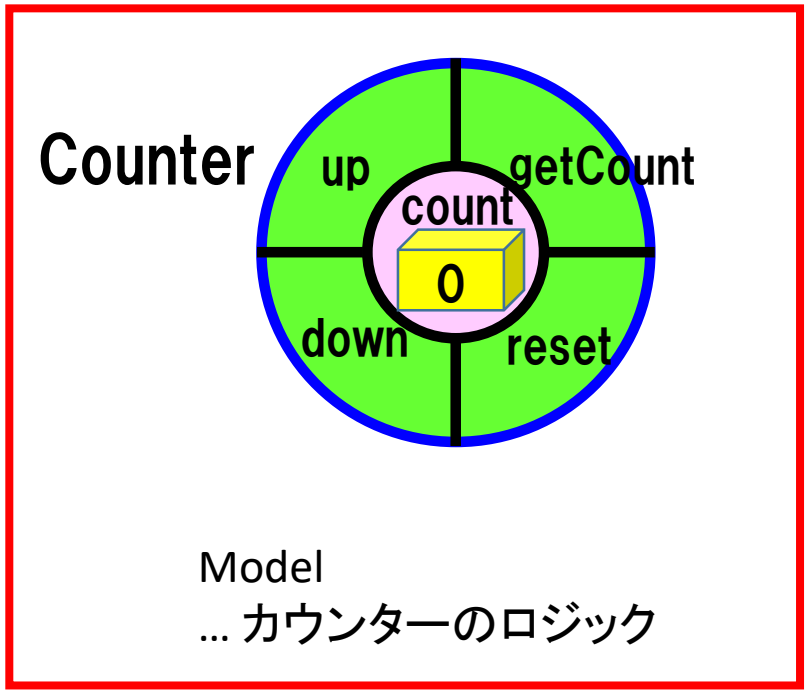
P0305-Counter-005-Inheritance


慶應義塾大学・理工学部・管理工学科
飯島 正
iijima@ae.keio.ac.jp

# モデル-ビュー-コントローラ (Counterの分離)

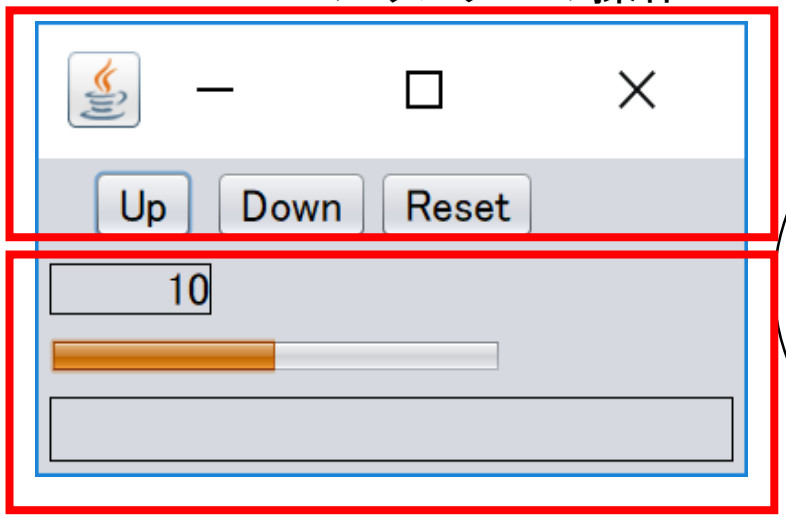


- Model … カウンターのロジック
- View … カウンターの見た目 (表示)
- Controller …カウンターの操作




Model
… カウンターのロジック

Controller …カウンターの操作




View … カウンターの見た目(表示)

一つの
アプリケーション
を
モデル，ビュー，
コントローラに
分離しましょう．

ViewとControllerが一体化していて，分離しにくい
GUI部品もある(たとえば，スライダー)


iijima@ae.keio.ac.jp

# 前回の課題



- 前回の課題 (1)
  - モデルであるカウンタ (Counterクラス) で，エラーを識別するようにします
  - 例外処理機能を使ってエラー処理をしましょう
  - 何をエラーとするか？
    - down () メソッド：カウントがゼロなのに，デクリメントしようとしたとき
    - up () メソッド：上限値も設けて，上限値を越えた場合もエラーとしましょう
    - セッターsetCount () メソッドにも，エラーチェックをつけましょう

- 前回の課題 (2)
  - 多重ビュー：プログレスバーを使いましょう．
    - 前回は，スライダーを使うように伝えていましたが，
      より簡単なプログラスバー (JProgressBar) を使うことにします．

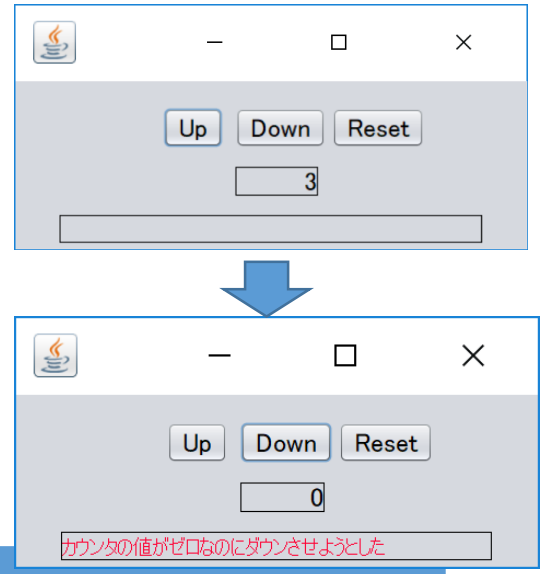


iijima@ae.keio.ac.jp

# 前回の課題（1）



- エラーが仕様した時，例外を発生させてみよう．
  - 内部状態が使用の範囲外
    - throw new IllegalStateException（“メッセージ”）;
  - 引数が仕様の範囲外
    - throw new IllegalArgumentException（“メッセージ”）;
- 呼び出す側のメソッドでは…
  - throws句で更に，呼び出し元に移譲する，か，もしくは，
  - Try { } catch () {} finally {} で例外処理を記述する．




| | プロジェクト名 | | |
|---|---|---|---|
| (1) | P0303_Counter-003-Exception | 例外処理 | エラーコードよりも，例外処理を使う方が，モデルとUIとの分離を高めます． |

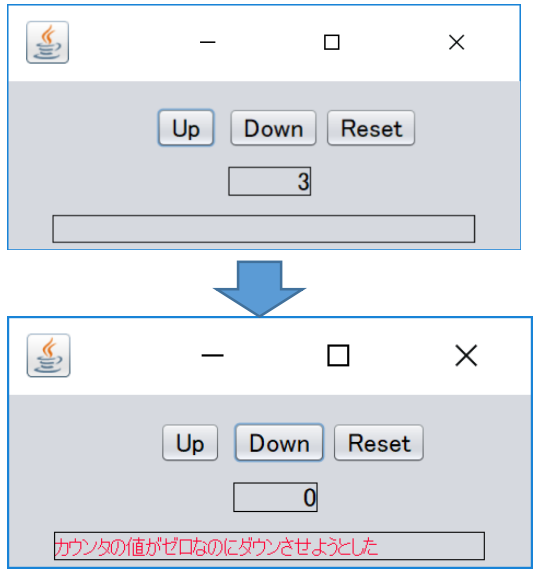
| | プロジェクト名 | パッケージ | メインクラス | その他のクラス |
|---|---|---|---|---|
| (1) | P0303_Counter-003-Exception | counter | Main | CounterFrame(JFrameフォーム)<br>Counter(カウンタのモデル) |



iijima@ae.keio.ac.jp

# 前回の課題（1）：例外処理

- エラーが仕様した時，例外を発生させてみよう．
  - 内部状態が使用の範囲外
    - throw new IllegalStateException（“メッセージ”）;
  - 引数が仕様の範囲外
    - throw new IllegalArgumentException（“メッセージ”）;
- 呼び出す側のメソッドでは…
  - throws句で更に，呼び出し元に移譲する，か，もしくは，
  - Try｛｝catch（）｛｝finally｛｝で例外処理を記述する．

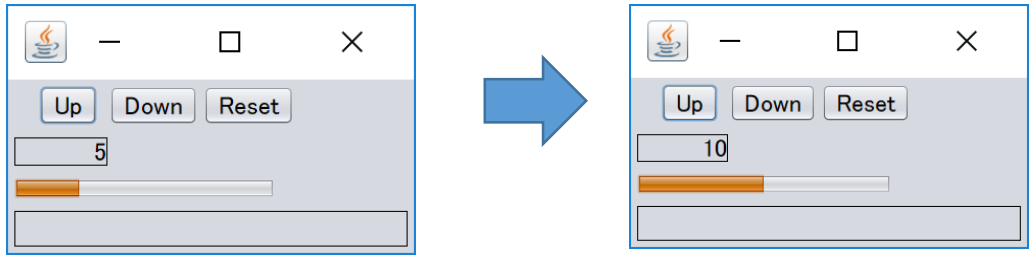


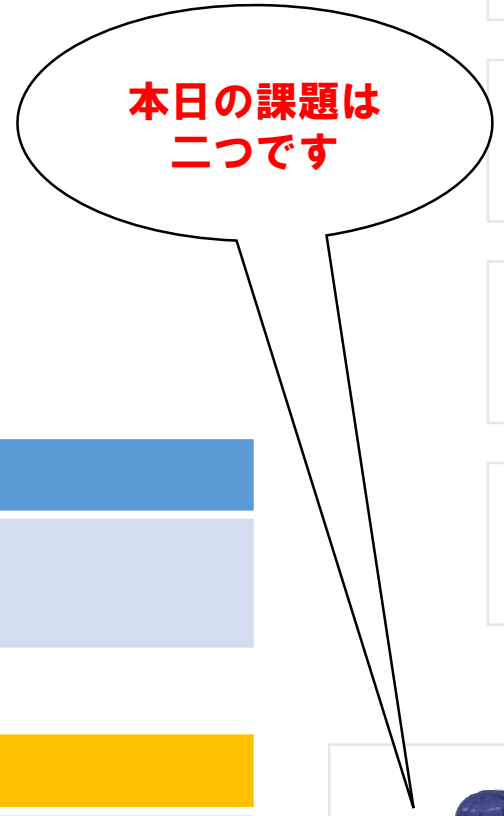


| | プロジェクト名 | パッケージ | メインクラス | その他のクラス |
|---|---|---|---|---|
| (1) | P0303_Counter-003-Exception | counter | Main | CounterFrame(JFrameフォーム)<br>Counter(カウンタのモデル) |

# 前回の課題 (2)

- 前回の課題 (2)
  - カウンタの値を，テキストで数値表示するとともに，プログレスバー（進捗バー）をつかってグラフィカルに表示しましょう




| | プロジェクト名 | | |
|---|---|---|---|
| (1) | P0304_Counter-004-MultiView | 多重ビュー | JProgressBarも併用します. |

| | プロジェクト名 | パッケージ | メインクラス | その他のクラス |
|---|---|---|---|---|
| (1) | P0304_Counter-004-MultiView | counter | Main | CounterFrame(JFrameフォーム)<br>Counter(カウンタのモデル) |

# 本日の課題

- **本日の課題**

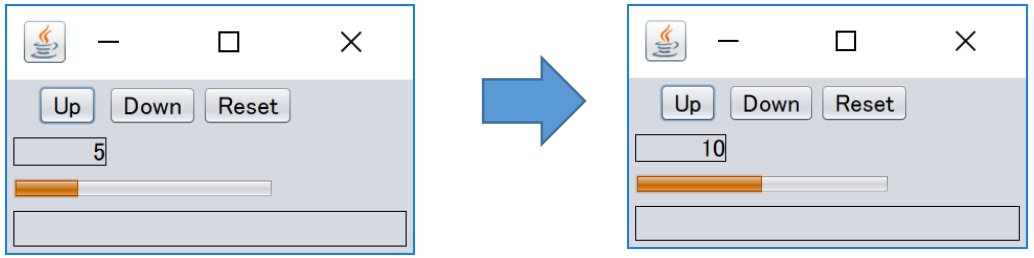
| | プロジェクト名 | |
|---|---|---|
| (1) | P0305a_Counter-005-Inheritance | モデルを拡張する．<br>属性maxとminだけをサブクラスCounterRangedへ |
| (2) | P0305b_Counter-005-Inheritance | サブクラスCounterRangedへにもメソッドupやdownの改訂版を作って，オーバーライドさせます．<br>その際．Counterの属性countがprivateではアクセスできないことを確認してください．publicに替えましょう． |
| (3) | P0305c_Counter-005-Inheritance | サブクラスCounterRangedで再定義されている，<br>up()メソッドやdown()メソッドでは，getCount() / setCount()メソッドで，countにアクセスします． |
| (4) | P0305d_Counter-005-Inheritance | サブクラスCounterRangedでsetCount()メソッドを<br>再定義して，上限値/下限値のチェックを含めましょう．スーパークラスのsetCountメソッドを<br>呼び出すには，super.setCount()と書きます． |
| (5) | P0305e_Counter-005-Inheritance | スーパークラスCounterの中の属性countの<br>アクセスレベルをprotectedにするだけで，簡単に解決します． |



iijima@ae.keio.ac.jp

# 本日の課題 (1)

- 本日の課題 (1)
  - 前回のプロジェクトをコピーして
  - 継承を導入しましょう
    - まず，モデルに下限値（ゼロ）だけでなく
      上限値も導入して，例外チェックさせましょう
    - それに先立って，モデルに，max (デフォールト値は0) とmin (同，20) を導入し，
      対応するgetterとsetterもつけましょう
    - 次にCounterのサブクラスCounterRangedを作り，差分を移動させます．




本日の課題は
継承です

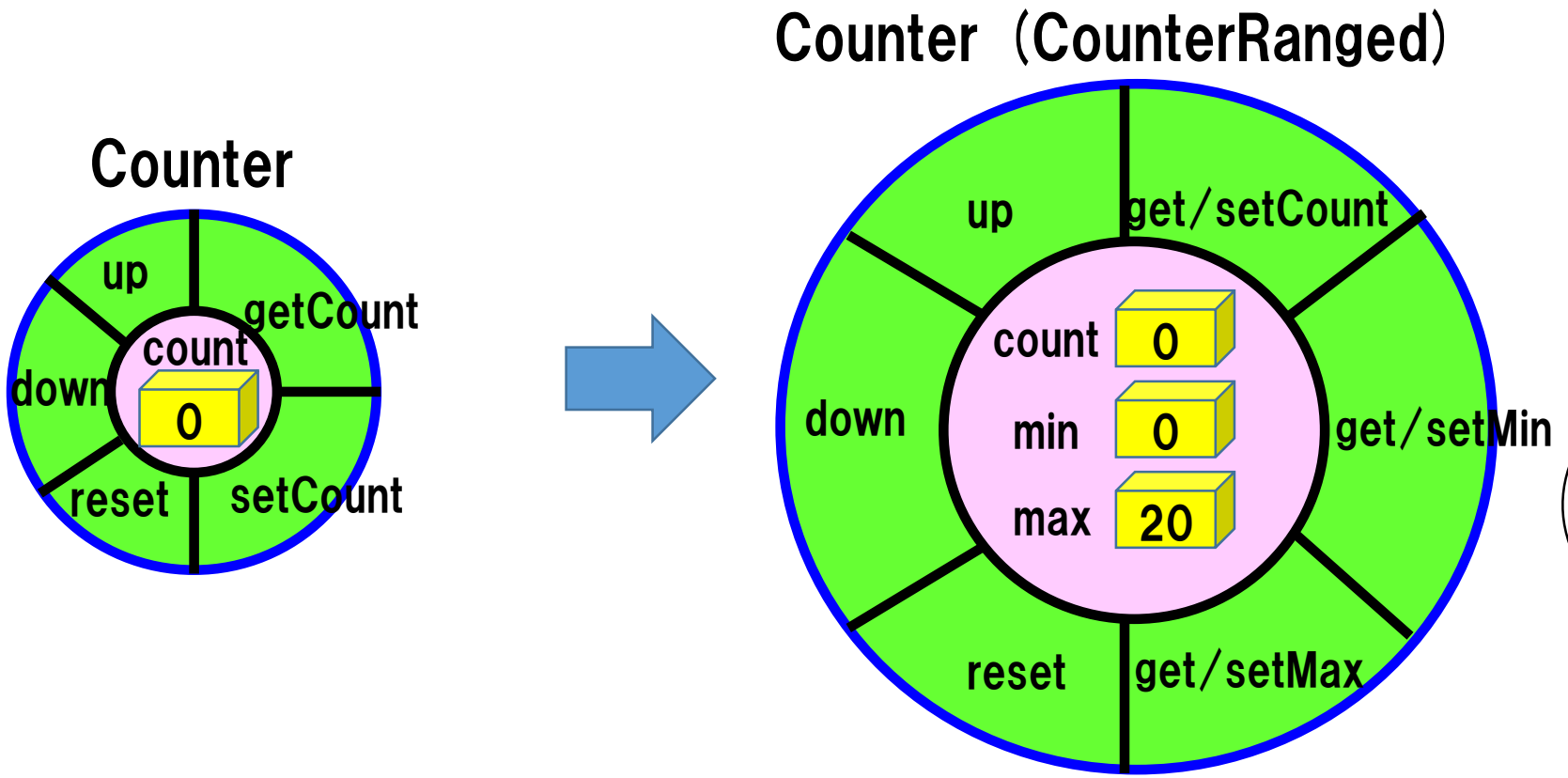
| | プロジェクト名 | | |
|---|---|---|---|
| (1) | P0305a_Counter-005-Inheritance | 継承 | モデルを拡張する.<br>属性maxとminだけをサブクラスCounterRangedへ |

| | プロジェクト名 | パッケージ | メインクラス | その他のクラス |
|---|---|---|---|---|
| (1) | P0305_Counter-005-Inheritance | counter | Main | CounterFrame(JFrameフォーム)<br>Counter(カウンタのモデル) |

# 発展:上限値と下限値はモデルが持つべきもの

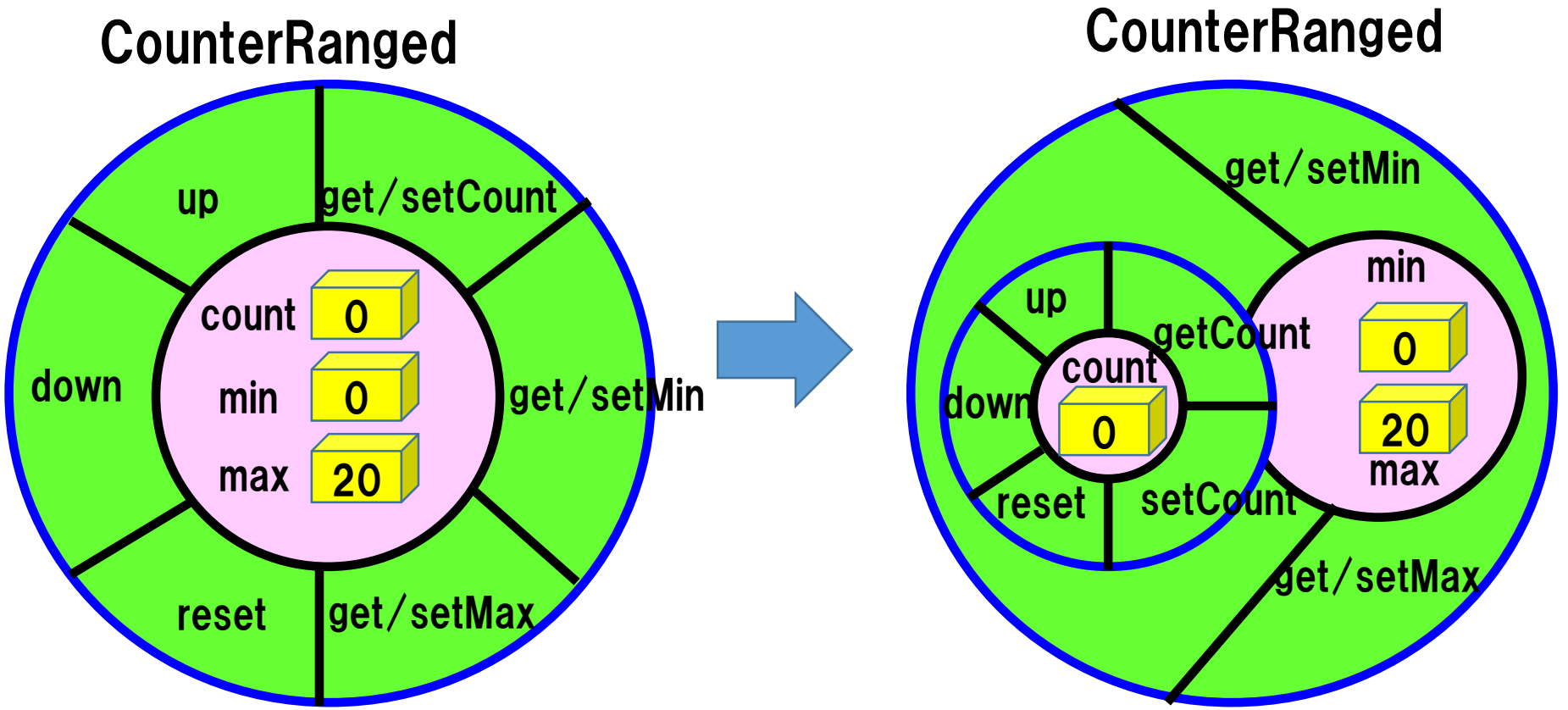


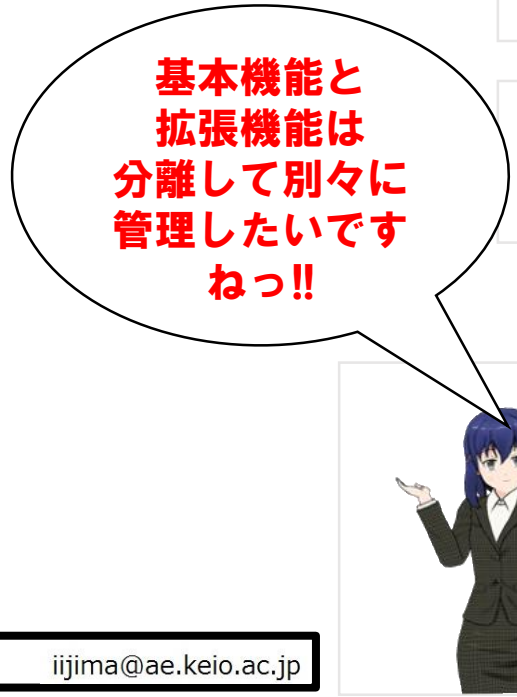
- 上限値maxと下限値minをCounterクラスに持たせる



# 発展: 基本機能に拡張機能を追加したもの

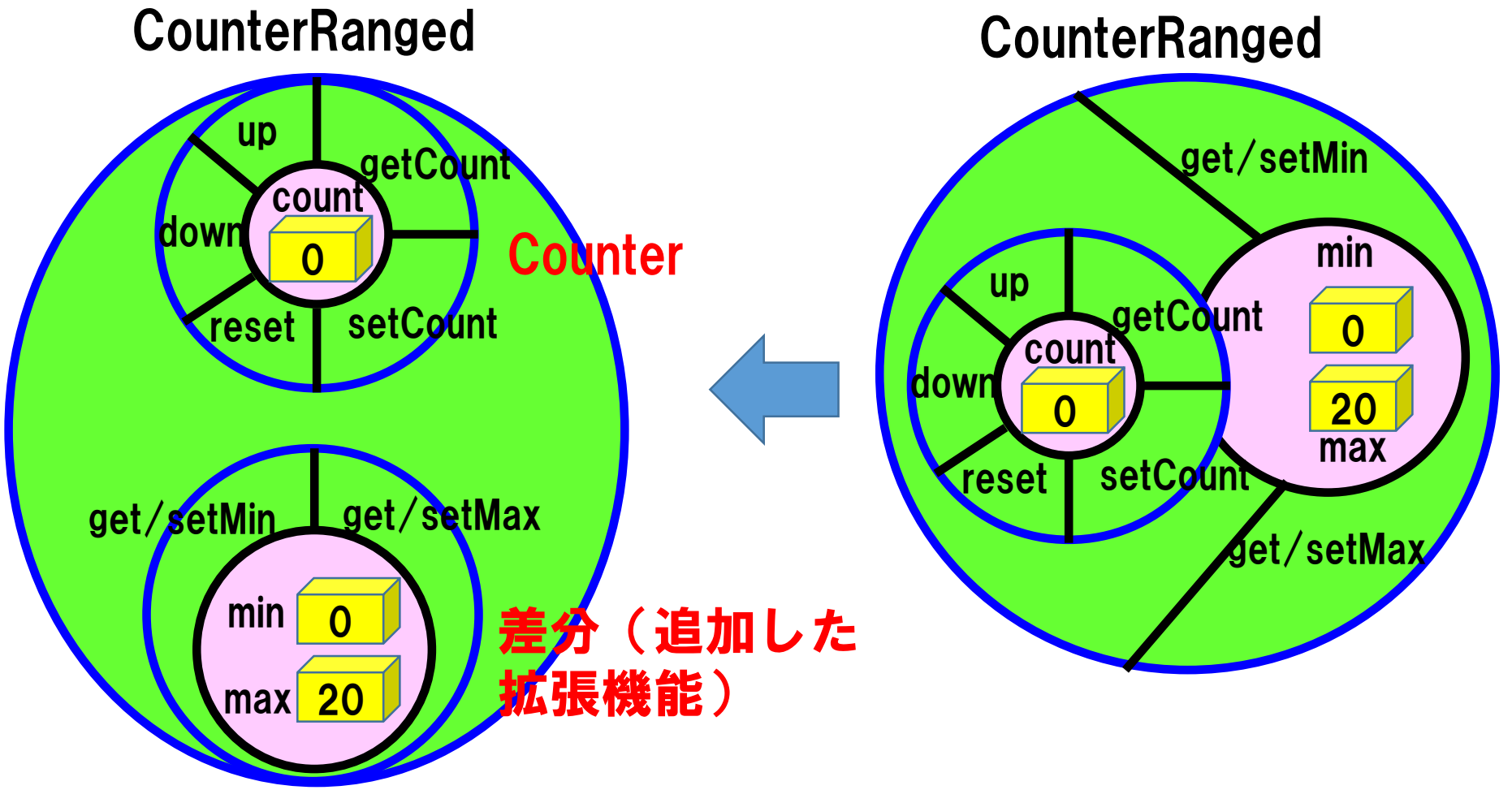
- 基本機能と拡張機能を混在させないで，管理できるか？



# 今回の発展:　継承を使った差分プログラミング

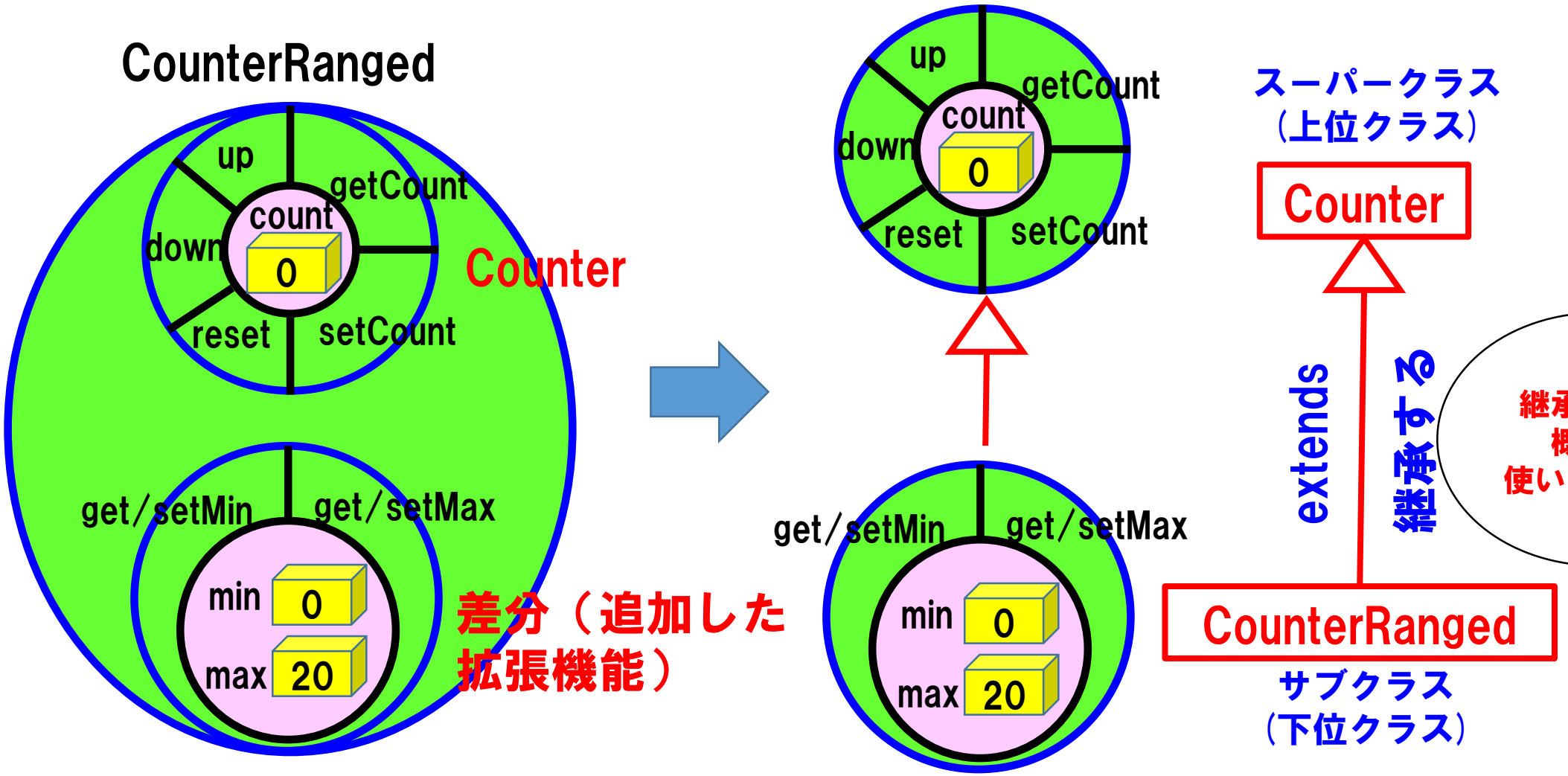
- 上限値maxと下限値minをCounterクラスに持たせる





# 今回の発展:　継承を使った差分プログラミング

- 上限値maxと下限値minという差分を，サブクラスCounterRangedに



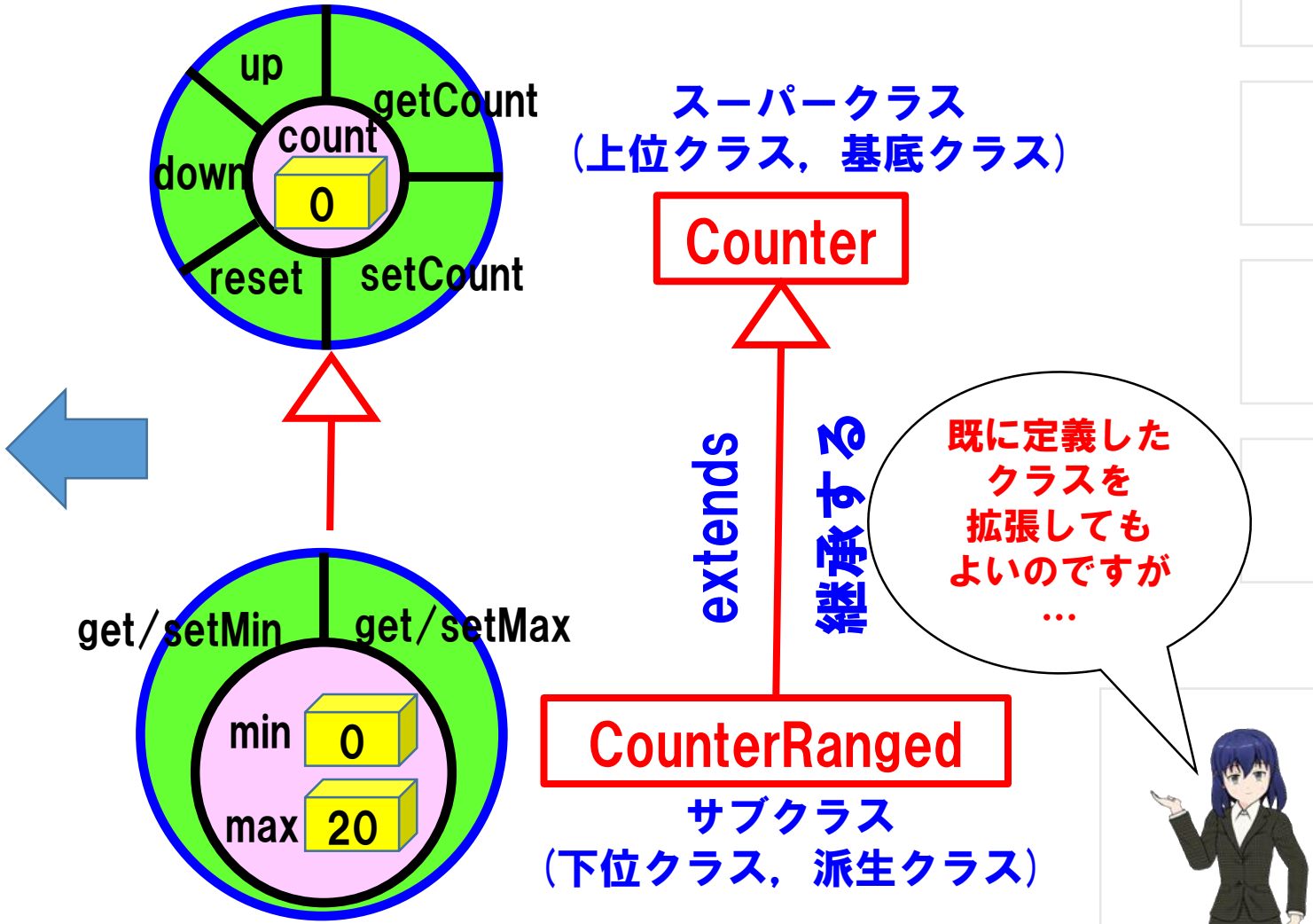


iijima@ae.keio.ac.jp

# 継承を使った差分プログラミング

- サブクラスは，
  (差分を定義することで)
  スーパークラスを拡張する
- サブクラスは，
  スーパークラスから
  性質(属性，メソッド)を
  継承する(受け継ぐ)

```
public class CounterRanged
                    extends Counter {
    int min  = 0;
    int max = 20;

    …
}
```



# 継承を使った差分プログラミング



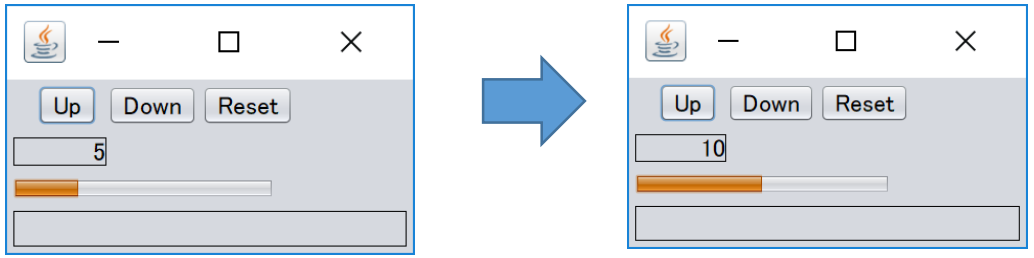
```
package counter;

public class CounterRanged extends Counter  {
    private int min = 0;
    private int max = 20;

    public CounterRanged ()  {  min = 0;     max = 20;  }

    public int getMin ()  {  return  (min) ;  }
    public int getMax ()  {    return  (max) ;    }
    public void setMin (int min)  { this.min = min; }
    public void setMax (int max)  { this.max = max; }
}
```

差分を
サブクラスへ
移動します


iijima@ae.keio.ac.jp

# 本日の課題 (2)

- 本日の課題 (2)
  - 課題 (1) プロジェクトをコピーして
  - さらに発展させましょう.
    - CounterのサブクラスCounterRangedを作って差分を移動させただけでは…
    - それだけでは，サブクラスの変数max/minを，upやdownで使えません.
    - サブクラスにもメソッドupやdownの改訂版を作り，オーバーライド (上書き) させます.




| | プロジェクト名 | | |
|---|---|---|---|
| (1) | P0305a_Counter-005-Inheritance | 継承 | モデルを拡張する.<br>属性maxとminだけをサブクラスCounterRanged |

| | プロジェクト名 | | |
|---|---|---|---|
| (2) | P0305b_Counter-005-Inheritance | 継承 | サブクラスCounterRangedへにもメソッドupやdownの改訂版を作って，オーバーライドさせます.<br>その際. Counterの属性countがprivateではアクセスできないことを確認してください. publicに替えましょう. |

メソッドの
オーバーライド
（上書き）


iijima@ae.keio.ac.jp

# 継承を使った差分プログラミング



```
package counter;

public class CounterRanged extends Counter {
        …
    @Override
    public void up() {
        if (count < max) {
            count++;
        } else {
            throw new IllegalStateException(
                    String.format("カウントが上限値%dなのにアップさせようとした", max));
        }
    }
        …
}
```

# 継承を使った差分プログラミング



```
package counter;

public class CounterRanged extends Counter {
        …
    @Override
    public void down() {
        if (count > min) {
            count--;
        } else {
            throw new IllegalStateException(
                    String.format("カウントが下限値%dなのにダウンさせようとした", min));
        }
    }
        …
}
```

サブクラスで
再定義して
オーバーライ
ドさせます


iijima@ae.keio.ac.jp

# 継承を使った差分プログラミング



```
package counter;


public class Counter  {
        …
    public int count = 0;
        …
}
```

countをpublic
に替えます

# 本日の課題 (3)

- 本日の課題 (3)
  - 課題 (2) プロジェクトをコピーして，さらに発展させます．
    - 属性countのアクセスレベルをpublicにしてしまうとカプセル化を破壊してしまいます．
    - アクセス修飾子をprivateに戻して，アクセッサでアクセスしてみます．

| | プロジェクト名 | | |
|---|---|---|---|
| (2) | P0305b_Counter-005-Inheritance | 継承 | サブクラスCounterRangedへにもメソッドupやdownの改訂版を作って，オーバーライドさせます．その際．Counterの属性countがprivateではアクセスできないことを確認してください．publicに替えましょう． |

| | プロジェクト名 | | |
|---|---|---|---|
| (3) | P0305c_Counter-005-Inheritance | 継承 | サブクラスCounterRangedで再定義されている，up()メソッドやdown()メソッドでは，getCount() / setCount()メソッドで，countにアクセスします． |



iijima@ae.keio.ac.jp

# 継承を使った差分プログラミング



```
package counter;


public class Counter  {
        …
    private int count = 0;
        …
}
```

# 継承を使った差分プログラミング



```
package counter;

public class CounterRanged extends Counter {
        ...
    @Override
    public void up() {

      if (count < max) { //  <---------------------- エラーになる
          count++;       //  <---------------------- エラーになる

      } else {
          throw new IllegalStateException(
                  String.format("カウントが上限値%dなのにアップさせようとした", max));
      }
    }
        ...
}
```



iijima@ae.keio.ac.jp

# 継承を使った差分プログラミング



```
package counter;

public class CounterRanged extends Counter {
        ...
    @Override
    public void up() {
      int count = getCount(); //  <---------------------- 追加する
      if (count < max) { //  <---------------------- エラーになる
          count++;         //  <---------------------- エラーになる
          setCount( count ); //  <---------------------- 追加する
      } else {
          throw new IllegalStateException(
                  String.format("カウントが上限値%dなのにアップさせようとした", max));
      }
    }
        ...
}
```



iijima@ae.keio.ac.jp

# 継承を使った差分プログラミング



```
package counter;

public class CounterRanged extends Counter {
        ...
    @Override
    public void down() {
      int count = getCount(); //  <---------------------- 追加する
      if (count > min) {  //  <---------------------- エラーになる
          count--;        //  <---------------------- エラーになる
          setCount( count );  //  <---------------------- 追加する
      } else {
          throw new IllegalStateException(
                  String.format("カウントが下限値%dなのにダウンさせようとした", min));
      }
    }
        ...
}
```

down() メソッドも同様です.


iijima@ae.keio.ac.jp

# 継承を使った差分プログラミング



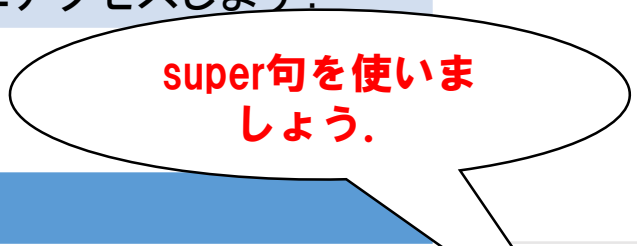
```
package counter;

public class CounterRanged extends Counter {
        ...
    @Override
    public void down() {

      if (count > min) { //  <---------------------- エラーになる
          count--;       //  <---------------------- エラーになる

      } else {
          throw new IllegalStateException(
                  String.format("カウントが下限値%dなのにダウンさせようとした", min));
      }
    }
        ...
}
```

# 本日の課題 (4)

- 本日の課題 (4)
  - 課題 (3) プロジェクトをコピーして，さらに発展させます．
    - サブクラスCounterRangedでsetCountをオーバーライドするにあたり，スーパークラスのsetCountを使えるときれいに書けます．
    - super.setCont () と書くと，スーパークラスのsetCountを呼び出せます

| | プロジェクト名 | | |
|---|---|---|---|
| (3) | P0305c_Counter-005-Inheritance | 継承 | サブクラスCounterRangedで再定義されている，up()メソッドやdown()メソッドでは，getCount() / setCount()メソッドで，countにアクセスします． |

| | プロジェクト名 | | |
|---|---|---|---|
| (4) | P0305d_Counter-005-Inheritance | 継承 | サブクラスCounterRangedでsetCount()メソッドを再定義して，上限値/下限値のチェックを含めましょう．スーパークラスのsetCountメソッドを呼び出すには，super.setCount()と書きます． |





iijima@ae.keio.ac.jp

# 継承を使った差分プログラミング



```
package counter;

public class CounterRanged extends Counter {
        …
    @Override
    public void setCount(int count) {
        if (count >= min && count <= max) {
            this.count = count; //  <---------------------- エラーになる

        } else {
            throw new IllegalStateException(
                    String.format("カウンタに不正な値%dをセットしようとした", count));
        }
    }        …
}
```



iijima@ae.keio.ac.jp

# 継承を使った差分プログラミング



```
package counter;

public class CounterRanged extends Counter {
        …
    @Override
    public void setCount(int count) {
      if (count >= min && count <= max) {
          // this.count = count; //  <---------------------- エラーになる
          super.setCount(count); //  <---------------------- 追加する
      } else {
          throw new IllegalStateException(
                  String.format("カウンタに不正な値%dをセットしようとした", count));
      }
    }        …
}
```

superを使って，スーパークラスのセッターを呼び出します．


iijima@ae.keio.ac.jp

# 本日の課題 (5)

- 本日の課題 (5)
  - 課題 (4) プロジェクトをコピーして，さらに発展させます．
    - いろいろ苦労しましたが，実は，スーパークラスCounterの中の属性countのアクセスレベルをprotectedにするだけで，簡単に解決します．
    - そのことを確認してください．

| | プロジェクト名 | | |
|---|---|---|---|
| (4) | P0305d_Counter-005-Inheritance | 継承 | サブクラスCounterRangedでsetCount()メソッドを再定義して，上限値/下限値のチェックを含めましょう．スーパークラスのsetCountメソッドを呼び出すには，super.setCount()と書きます． |

| | プロジェクト名 | | |
|---|---|---|---|
| (5) | P0305e_Counter-005-Inheritance | 継承 | スーパークラスCounterの中の属性countのアクセスレベルをprotectedにするだけで，簡単に解決します． |

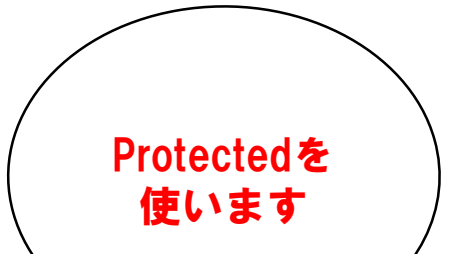


iijima@ae.keio.ac.jp

# 継承を使った差分プログラミング



```
package counter;


public class Counter  {
        …
    protected int count = 0;
        …
}
```



# 継承を使った差分プログラミング



```
package counter;

public class CounterRanged extends Counter {
        ...
    @Override
    public void up() {

      if (count < max) { //  <---------------------- エラーになりません
          count++;       //  <---------------------- エラーになりません

      } else {
          throw new IllegalStateException(
                  String.format("カウントが上限値%dなのにアップさせようとした", max));
      }
    }
        ...
}
```

# 継承を使った差分プログラミング



```
package counter;

public class CounterRanged extends Counter {
        ...
    @Override
    public void down() {

      if (count > min) { //  <---------------------- エラーになりません
          count--;       //  <---------------------- エラーになりません

      } else {
          throw new IllegalStateException(
                  String.format("カウントが下限値%dなのにダウンさせようとした", min));
      }
    }
        ...
}
```

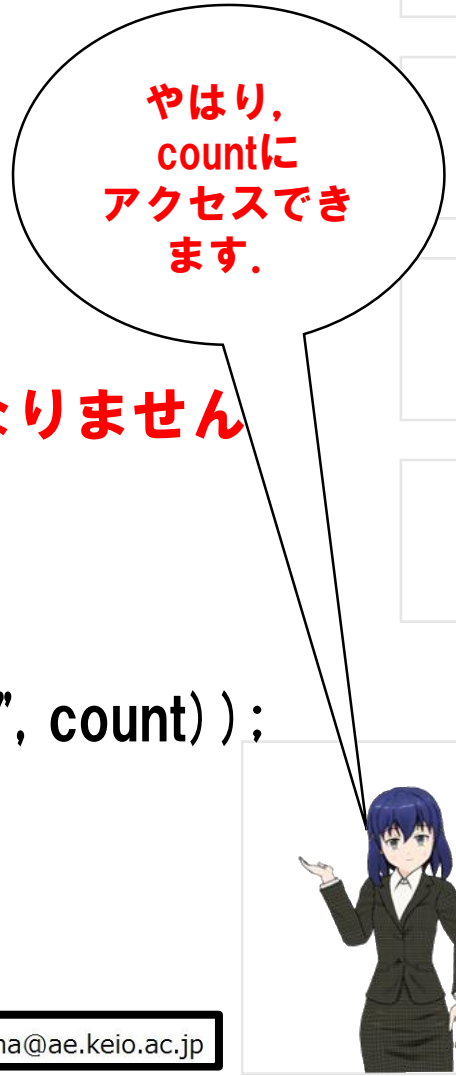
down() メソッドも同様です.


iijima@ae.keio.ac.jp

# 継承を使った差分プログラミング



```
package counter;

public class CounterRanged extends Counter {
        …
    @Override
    public void setCount(int count) {
        if (count >= min && count <= max) {
            this.count = count; //  <---------------------- エラーになりません

        } else {
            throw new IllegalStateException(
                    String.format("カウンタに不正な値%dをセットしようとした", count));
        }
    }        …
}
```



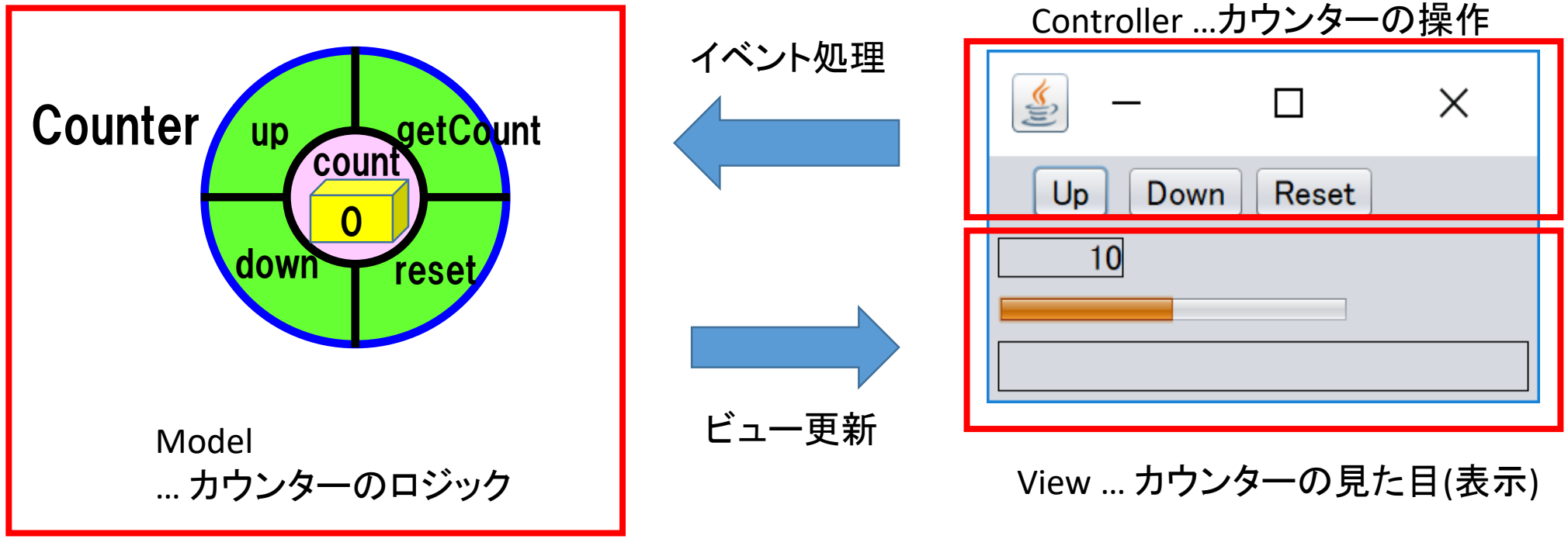


iijima@ae.keio.ac.jp

# 次回: モデル-ビュー-コントローラ (Counterの分離)



- Model … カウンターのロジック
- View … カウンターの見た目 (表示)
- Controller …カウンターの操作





# 次回: Observer/Observableパターンの導入



- Model … カウンターのロジック
- View … カウンターの見た目(表示)
- Controller …カウンターの操作