# Inner Classes for Publisher-Subscriber

```
class PublSubsc {

  public abstract class Publisher {

    abstract void expectedChangeOp(Object[] args);

    public void attach(Subscriber sub) {
       subscribers.addElement((Object) sub);
      }
    public void detach(Subscriber sub) {
       subscribers.removeElement((Object) sub);
      }

    public void providedChangeOp(Object[] args) {
       for (int i = 0; i < subscribers.size(); i++) {
            ((Subscriber) subscribers.elementAt(i)).update(this);
          };
       expectedChangeOp(args);
       }

    protected java.util.Vector subscribers = new java.util.Vector();

   }

  public abstract class Subscriber {

    protected Publisher publ;

    abstract void expectedUpdate();

    public void update(Publisher _publ) {
```

```
        publ =  _publ;
        expectedUpdate();
       }
   }
}


// Application

   class Point {
     private int x = 3;
     private int y = 10;

      public Point() {super();}
      public Point(int _x, int _y) {
          x = _x;
          y = _y;
        }

      public int getX() {
         return x;
       }

      public void setX(int _x) {
         x = _x;
       }
      public int getY() {
        return y;
        }

      public void setY(int _y) {
         y = _y;
       }

      public void set(int _x, int _y) {
          x = _x;
          y = _y;
        }
    }

  class ChangePrinter {

     public void output1() {
        System.out.println("CHANGE of ONE KIND!!!!");
```

```
      }
     public void output2() {
        System.out.println("CHANGE of ANOTHER KIND!!!!");
      }

  }

// Adaptation
// ----------
//
// The following adaptation corresponds to the following mappings
//
// connector c {
//    { Publisher.expectedChangeOp = Point.{setX, set};
//      Subscriber.expectedUpdate = ChangePrinter.output1;}
//    { Publisher.expectedChangeOp = Point.{set, steY};
//      Subscriber.expectedUpdate = ChangePrinter.output2;}
//  }
//
// Note, that the same appc gets applied two times to the same
// ccg, as the result of two connection-clauses.
//
// The following code for the class Adaptation can be considered
// as generated by the appc compiler as the result of translating c1.
//
// The adaptation of the application to the aspect happens at the level
// of individual instances (see the client code). Could envisage the
// following extension, to account for class level adaptations.
// To activate/deactivate an aspect at the level of classes, the
// programmer has to enclose the activation scope with a pair of
// sentences: activate(c1)/deactivate(c1).
// We could then, have the compiler change the client code
// between activate/deactivate
// so that adapt(p) is invoked on each new Point instance.

 class Adaptation extends PublSubsc {

     Point adapted_p;
     Subscriber subAd1;
     Subscriber subAd2;

     public Point adapt(final Point _p) {

        class MyPoint extends Point {
```

```
  int x = _p.getX();
  int y = _p.getY();
  public void setX(int _x) {
      Object[] args = {(Object) new Integer(_x)};
      publAd1.providedChangeOp(args);
  }
  public void set(int _x, int _y) {
    Object[] args = {(Object) new Integer(_x), (Object) new Integer(_y)};
    publAd2.providedChangeOp(args);
  }
  public void setY(int _y) {
      Object[] args = {(Object) new Integer(_y)};
      publAd3.providedChangeOp(args);
   }

  private void superSetX(int _x) {super.setX(_x);}
  private void superSetY(int _y) {super.setY(_y);}
  private void superSet(int _x, int _y) {super.set(_x, _y);}

  public Publisher publAd1 = Adaptation.this.new Publisher() {
    void expectedChangeOp(Object[] args) {
       superSetX(((Integer) args[0]).intValue());
    }
  };

  public Publisher publAd2 = Adaptation.this.new Publisher() {
    void expectedChangeOp(Object[] args) {
       superSet(((Integer) args[0]).intValue(),
              ((Integer) args[1]).intValue());
    }
  };

  public Publisher publAd3 = Adaptation.this.new Publisher() {
    void expectedChangeOp(Object[] args) {
       superSetY(((Integer) args[0]).intValue());
    }
  };
}
MyPoint temp = new MyPoint();
temp.publAd1.attach(subAd1);
temp.publAd2.attach(subAd1);
temp.publAd2.attach(subAd2);
temp.publAd3.attach(subAd2);
adapted_p = temp;
```

```
        return adapted_p;
      }

      void attach(final ChangePrinter ch) {
         subAd1 = Adaptation.this.new Subscriber() {
           void expectedUpdate() {ch.output1();}
          };
         subAd2 = Adaptation.this.new Subscriber() {
           void expectedUpdate() {ch.output2();}
          };
      }
}

class Client {

  public static void main(String[] args) {
    Adaptation adapter = new Adaptation();
    Point p = new Point();
    ChangePrinter ch = new ChangePrinter();
    adapter.attach(ch);
    p = adapter.adapt(p);

    p.setX(12);
    System.out.println("X changed = " + p.getX());
    p.setY(10);
    System.out.println("Ychanged  = " + p.getX());

    p.set(22, 20);
    System.out.println("Both changed: X = " + p.getX() + " Y = " + p.getY());

    p.setX(20);
    p.setX(30);

    p.set(100, 7);
    System.out.println("Both changed: X = " + p.getX() + " Y = " + p.getY());
 }
}


// CHANGE of ONE KIND!!!!
// X changed = 12
// CHANGE of ANOTHER KIND!!!!
// changed  = 12
// CHANGE of ONE KIND!!!!
```

```
// CHANGE of ANOTHER KIND!!!!
// Both changed: X = 22 Y = 20
// CHANGE of ONE KIND!!!!
// CHANGE of ONE KIND!!!!
// CHANGE of ONE KIND!!!!
// CHANGE of ANOTHER KIND!!!!
// Both changed: X = 100 Y = 7
```