Solution – Classification of Classes

Given:

```java
import java.util.Collection;
import java.util.LinkedList;

abstract class Shape {
    // ... defined
}

class Title extends Shape {
    String title;
    Shape s;
    ScrollBar(Shape S, String T) {
        s = S;
        title = T;
    }
    // ... defined
}

class Circle extends Shape {
    // ... defined
}

class RedBorder extends Shape {
    RedBorder(Shape s) {
    }
    // ... defined
}

class Rectangle extends Shape {
    // ... defined
}

class ScrollBar extends Shape {
    ScrollBar(Shape s) {
    }
    // ... defined
}

class Panel extends Shape implements Collection<Shape> {
    LinkedList<Shape> list;
    Panel() { list = new LinkedList<>(); }
    void add(Shape s) { list.add(s); }
    // ... defined
}
```

The solution table is shown below, where column2 can have 1 or more terms in {base, decorator, adaptor, decoraptor, composite}:

<table>
<thead>
<tr>
<th>Class</th>
<th>What Kind Is It?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>decorator</td>
</tr>
<tr>
<td>Circle</td>
<td>base</td>
</tr>
<tr>
<td>RedBorder</td>
<td>decorator</td>
</tr>
<tr>
<td>Rectangle</td>
<td>base</td>
</tr>
<tr>
<td>ScrollBar</td>
<td>decorator</td>
</tr>
<tr>
<td>Panel</td>
<td>composite, adaptor</td>
</tr>
</tbody>
</table>

Note: there are no decoraptors in this example. There is a “composite adaptor”, for which there is no cute name known. A decorator is a class C whose superclass is S and C has a constructor with a single parameter of type S. An adaptor is a class C that implements interface S (or whose superclass is S) and whose methods translate to a different class W. A decoraptor is a class C that implements interface S (or whose superclass is S), whose constructor has a parameter of type S, and that implements another interface I (different from C).