A **Butterfly** (initially in an **Embryo** state) goes through four states (from **Embryo** to **Larval** to **Pupa** to **Adult**). Suppose a **Season** is a **Visitor** to a Butterfly. Each time it visits, it changes the Butterfly’s state to the next stage of life. Two methods that depend on the state of development: **public boolean edible()** and **public String toString()** according to the following:

<table>
<thead>
<tr>
<th>state</th>
<th>toString() returns:</th>
<th>edible() returns:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Embryo</td>
<td>“I’m an egg”</td>
<td>false</td>
</tr>
<tr>
<td>Larval</td>
<td>“I’m a caterpillar”</td>
<td>true</td>
</tr>
<tr>
<td>Pupa</td>
<td>“I’m a cocoon”</td>
<td>true</td>
</tr>
<tr>
<td>Adult</td>
<td>“I’m a butterfly”</td>
<td>false</td>
</tr>
</tbody>
</table>

A **ScrubJay** is another **Visitor**. When it visits, it will try to eat the butterfly with **public void eat(Butterfly b)** and, if b is edible the eat method will print “yumm!” otherwise “yuck!”.

1a. [5%] Finish the following driver code so that Butterfly b is visited by ScrubJay j followed immediately by Season s:

```java
Butterfly b = new Butterfly(); // first in the Embryo state
Season    s = new Season();    // a visitor to Butterfly b
ScrubJay  j = new ScrubJay();  // another visitor to b

for (int i = 0; i < 3; i++) {
    System.out.println("Butterfly: "+ b.toString());
    // now add visitations from j followed by s:
}
```

1b. [5%] What is the printout that results from the above code?
2. [30%] Finish implementing Butterfly and its states (which use the state pattern for edible and toString):

```java
public class Butterfly {
    public State currentState;
    public State embryo;
    public State larval;
    public State pupa;
    public State adult;

    public Butterfly() { setEmbryo(); }

    public void setEmbryo() { currentState = new Embryo(); }
    public void setLarval() { currentState = new Larval(); }
    public void setPupa() { currentState = new Pupa(); }
    public void setAdult() { currentState = new adult(); }

    public void acceptVisitor(Visitor v) {

        public boolean edible() {

        public boolean toString() {
```
3. [40%] ScrubJays do not just visit Butterflies; they also visit each other, and when they do they print out “squawk!”. Write Visitor and ScrubJay. Note that Visitor needs to deal with all possible visitees thus far defined.

abstract public class Visitor {

public class ScrubJay
4. [20%] Write the **Visitor** class **Season**. Make sure it is well-defined for all its possible visitees.