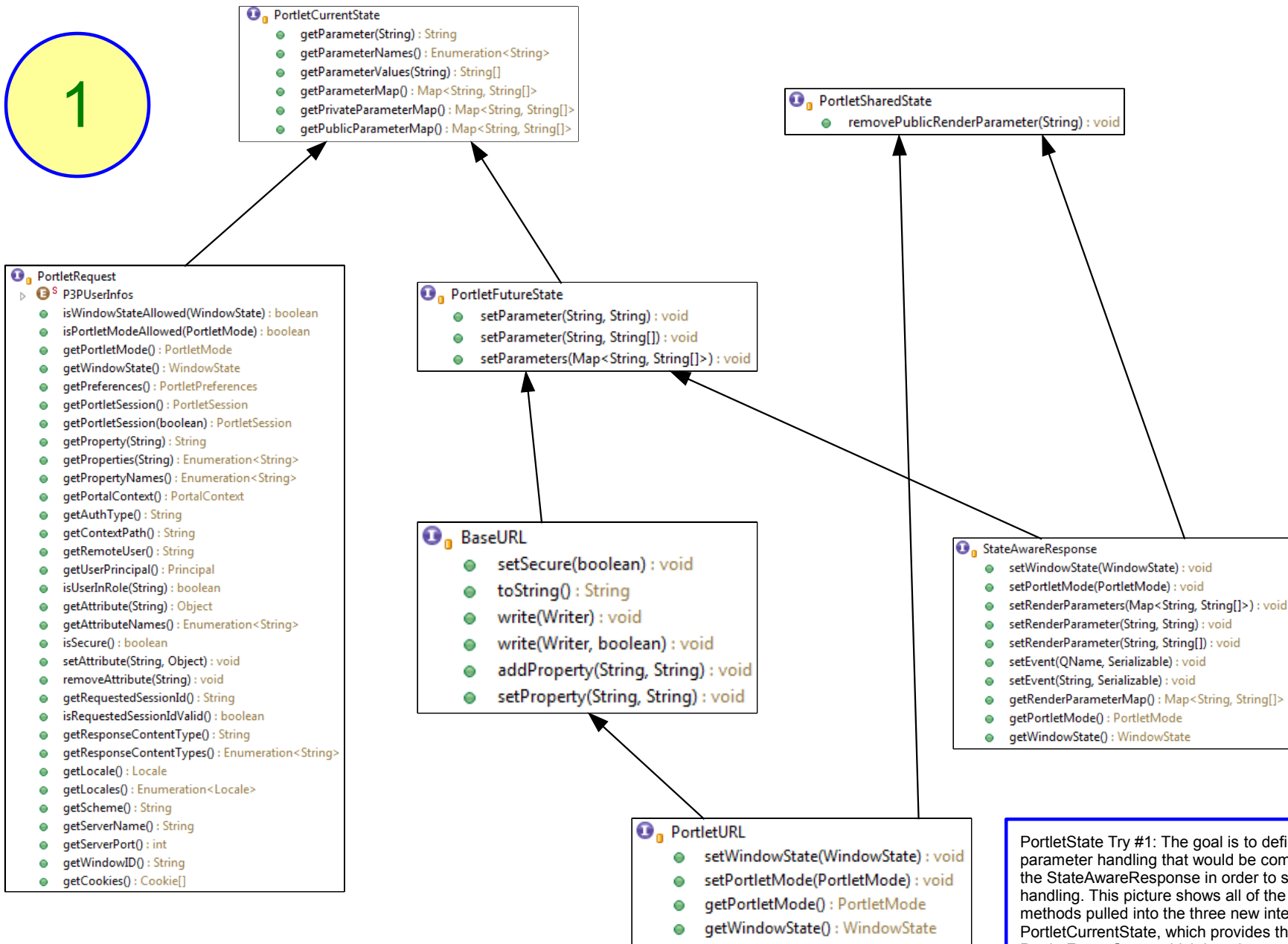
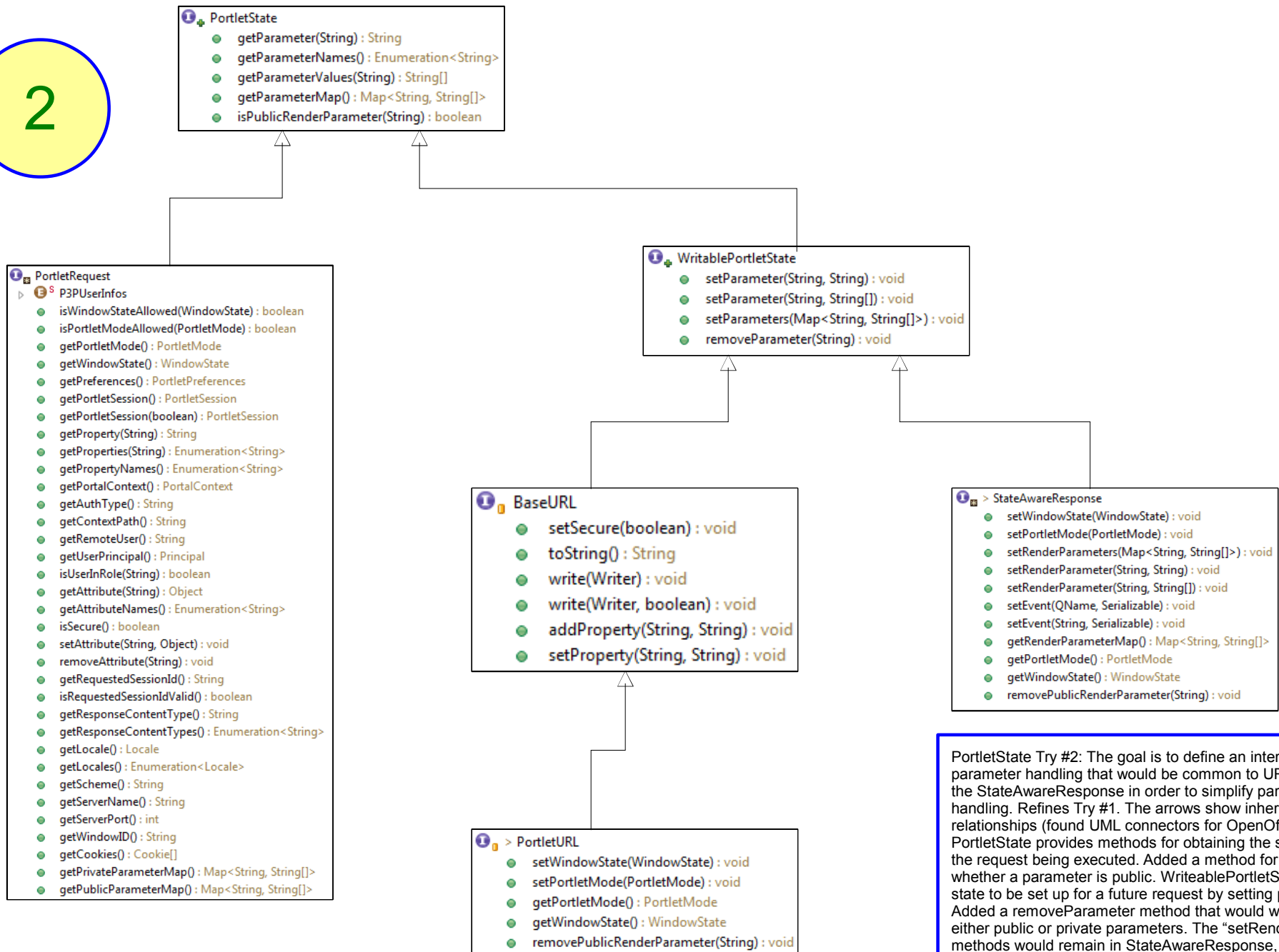


1



PortletState Try #1: The goal is to define an interface for parameter handling that would be common to URLs and to the StateAwareResponse in order to simplify parameter handling. This picture shows all of the parameter handling methods pulled into the three new interfaces PortletCurrentState, which provides the getter methods, PortletFutureState, which has the setter methods, and PortletSharedState, which contains the single method that works only with public render parameters. The arrows show an inheritance relationship. PortletCurrentState allows you to get the parameters associated with the request currently being executed, while PortletFutureState allows you to set up state for a future request. The future request would be a render request if the state is set on the StateAwareResponse. If set on a URL, it would be the state that would come into play if the URL is activated.

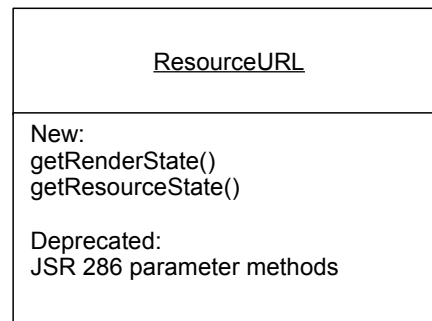
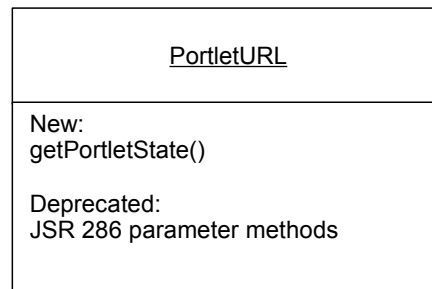
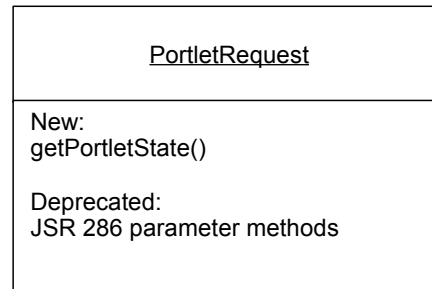
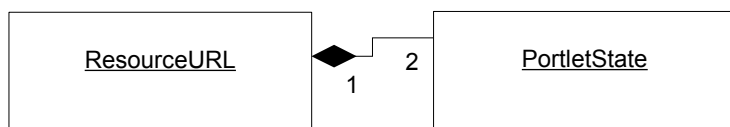
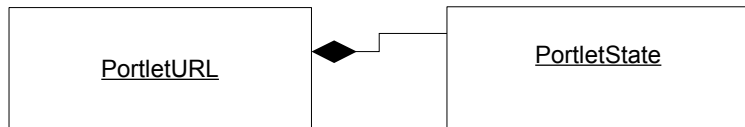
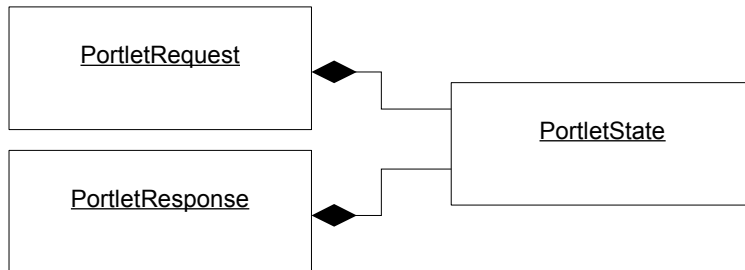
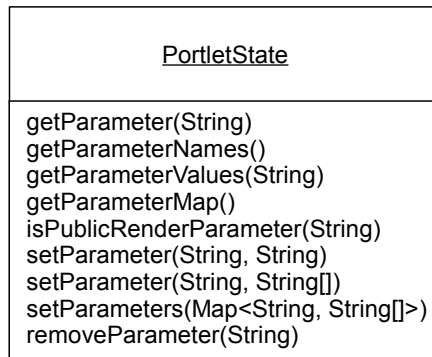
2



PortletState Try #2: The goal is to define an interface for parameter handling that would be common to URLs and to the StateAwareResponse in order to simplify parameter handling. Refines Try #1. The arrows show inheritance relationships (found UML connectors for OpenOffice, yay!) PortletState provides methods for obtaining the state valid for the request being executed. Added a method for determining whether a parameter is public. WritablePortletState allows state to be set up for a future request by setting parameters. Added a removeParameter method that would work with either public or private parameters. The "setRender..." methods would remain in StateAwareResponse, and would be deprecated. The removePublicRenderParameter method would remain in PortletURL and StateAwareResponse and would be deprecated.

Question: Should PortletMode & WindowState be handled as "portlet state" in these interfaces? Or should we define an "ExtendedPortletState" containing the parameters, window state, and portlet mode?

# 3-1



PortletState Try #3: Implement concept of "Portlet State". Define a regular interface for state handling for the portlet itself and for URLs. The state is represented by the public and private parameters that determine processing during a portlet request. It corresponds to the JSR 286 idea of portlet parameters.

Each portlet request and each portlet response object has one portlet state object that they conceptually share. You use the getPortletState() method on the PortletRequest to obtain access to the PortletState object. Changes to the PortletState object obtained from the request are automatically reflected in the PortletResponse PortletState object and become the new portlet render state, so there is no need to explicitly set a new portlet state. The new state will be available during subsequent render requests.

Changing the portlet render state during processing of a render request or a resource request is not allowed, so an attempt to do so by using one of the setParameter... methods or by using the removeParameter() method will result in a javax.portlet.ReadOnlyException.

Each portlet URL has one portlet state object. Depending on how the URL is created (new "create" method TBD), the PortletState for the URL will initially be either empty, or will be populated with the PortletState valid for the current request. You use the getPortletState() method to obtain access to the PortletState object. Changes to the PortletState object are automatically stored with the URL, so there is no need to explicitly set a new portlet state. The new state will be available during the portlet request initiated through this URL..

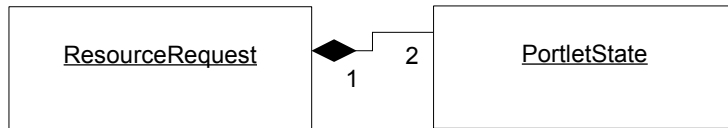
Each Resource URL has 1 or 2 portlet state objects.

The PortletState object representing the render state for the URL be populated with the render state valid for the current request. You use the getRenderState() method to obtain access to the read-only render state.

Changing the portlet render state on a ResourceURL is not allowed, so an attempt to do so will result in an InvalidStateException.

However, you can add additional state by using the getResourceState() method to obtain an empty PortletState object that allows additional resource parameters to be added.

# 3-2



<u>ResourceRequest</u>
New: getPortletState() getRenderState() getResourceState()
Deprecated: JSR 286 parameter methods

<u>MimeResponse</u>
New: createActionURL(int copyFlag) createRenderURL(int copyFlag) createResourceURL(int copyFlag)
Deprecated: None.

Each Resource Request has 1 or 2 portlet state objects.

The render state is the state that governed the render or resource request during which the resource URL leading to this resource request was generated. It is automatically copied to the resource URL when the URL is created and cannot be modified.

The resource state consists of additional state information that was appended to the URL after its creation. It is read-only and cannot be modified.

The PortletState obtained from the getPortletState() method is a read-only object representing the render state and resource state combined together. If a render state parameter and a resource state parameter have the same name, the resource parameter value(s) will appear in the value array before the render state value(s).

Note: I believe that it would be possible to add this new model while leaving the current JSR 286 model completely intact. We could specify that calling getPortletState() on a request activates the new model and deactivates the old model. That would avoid the problem of having to merge parameters set through the JSR 286 methods into the new PortletState model.

Although not strictly related to parameters, for completeness I wanted to show the proposal for new URL creation methods.

Three new create methods would be added, one for each URL type. The new methods would each contain a flag that would designate whether the currently active portlet state should be copied into the new URL. The flag would have the following values:

COPY\_NONE – no parameters would be copied; same behavior as the current createActionURL() and createRenderURL() methods.

COPY\_RENDER\_STATE – the currently active render state is copied to the URL.

COPY\_PORTLET\_STATE – the complete portlet state is copied to the new URL. During render request processing, this flag would have the same effect as the COPY\_RENDER\_STATE flag. However, during resource request processing, the portlet state represents the combined resource state and render state, so the result would be different