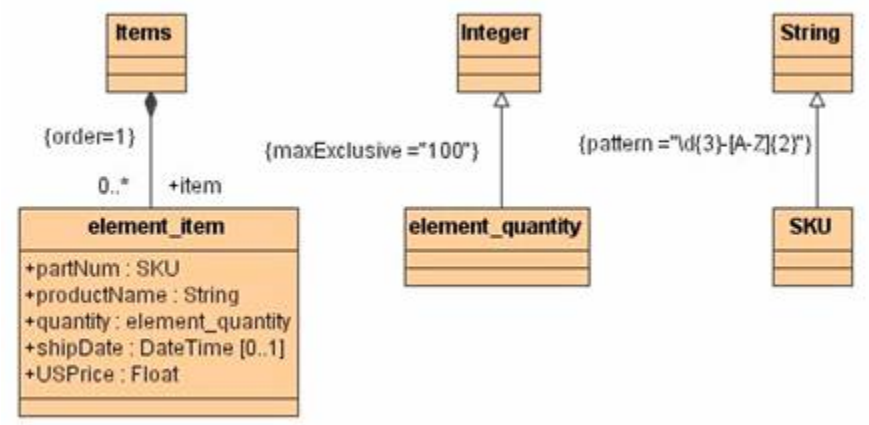
Import of Anonymous XSD Type Definitions

Schemas can be constructed by defining sets of named types such as PurchaseOrderType and then declaring elements such as purchaseOrder that reference the types using the type= construction. This style of schema construction is straightforward but it can be unwieldy, especially if you define many types that are referenced only once and contain very few constraints. In these cases, a type can be more succinctly defined as an anonymous type, which saves the overhead of having to be named and explicitly referenced.

The definition of the type Items in the PurchaseOrder schema contains two element declarations that use anonymous types (item and quantity). In general, you can identify anonymous types by the lack of a type= attribute in an element (or attribute) declaration, and by the presence of an un-named (simple or complex) type definition:

Figure: Anonymous type definitions

```
<xsd:complexType name="Items">
          <xsd:sequence>
      <xsd:element name="item" minOccurs="0" maxOccurs="unbounded">
        <xsd:complexType>
           <xsd:sequence>
             <xsd:element name="productName" type="xsd:string"/>
             <xsd:element name="quantity">
               <xsd:simpleType>
                  <xsd:restriction base="xsd:positiveInteger">
                    <xsd:maxExclusive value="100"/>
                  </xsd:restriction>
               </xsd:simpleType>
             </xsd:element>
             <xsd:element name="USPrice" type="xsd:decimal"/>
             <xsd:element ref="comment" minOccurs="0"/>
             <xsd:element name="shipDate" type="xsd:date" minOccurs="0"/>
           </xsd:sequence>
           <xsd:attribute name="partNum" type="typens:SKU" use="required"/>
        </xsd:complexType>
      </xsd:element>
   </xsd:sequence>
</xsd:complexType>
```



In the case of the item element, it has an anonymous complex type consisting of the elements productName, quantity, USPrice, and shipDate, and an attribute called partNum. In the case of the quantity element, it has an anonymous simple type derived from integer whose value ranges between 1 and 99.

The Importer now uses the following rule to create UML classes:

> **Mapping Rule:** Each xsd:element or xsd:element that contains an anonymous type is mapped to an UML class of name 'element_<element_name>'. note

The above rule explains the class diagram above9_import_-_export_mechanisms_9_3_4417. The complex type Items contains a sequence of item elements. This sequence becomes an orderer association to the type element_item. This type contains all elements of the anonymous type as if the type would have been declared non-anonymously having the name element_item. The second anonymous type found in Figure 439 is the simple type nested within the quantity element. Because this is a simple type we create an attribute quantity (not an association) of type element_quantity. The type element_quantity derives from Integer mapping the XSD restriction to a constraint on the generalization (maxExclusive = 100).