build an application that can determine payments for employees and invoices alike.

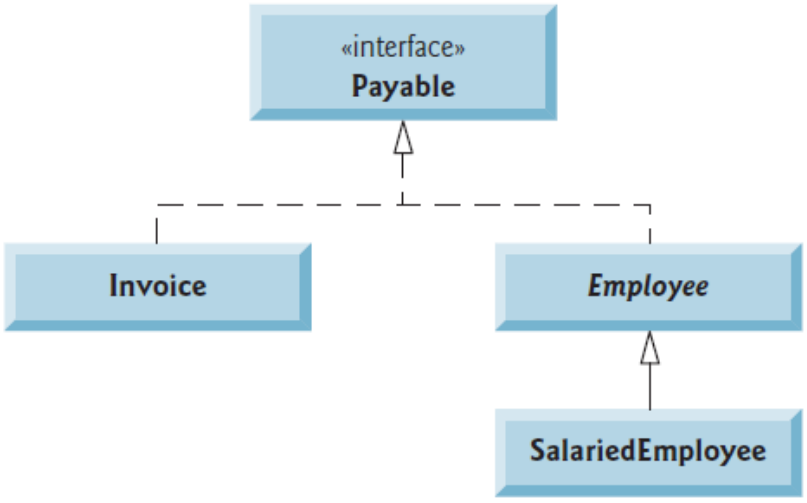
Create interface **Payable**, which contains method **getPaymentAmount** that returns a double amount that must be paid for an object of any class that implements the interface.

Method **getPaymentAmount** is a general-purpose version of method **earnings** of the **Employee** hierarchy—method **earnings** calculates a payment amount specifically for an **Employee**, while **getPaymentAmount** can be applied to a broad range of unrelated objects.

Implement class **Invoice**, which implements interface Payable.

Now, modify class Employee such that it also implements interface Payable.

Update Employee subclasses created previousley to "fit" into the Payable hierarchy by renaming SalariedEmployee method earnings as getPaymentAmount and so on.



Use this modified tester to have elements of Employee subclasse, and other elements purchased for the office.

public class PayableInterfaceTest

```
{  public static void main( String[] args ) {

// create four-element Payable array
Payable[] payableObjects = new Payable[ 4 ];
// populate array with objects that implement Payable
payableObjects[ 0 ] = new Invoice( "01234", "seat", 2, 375.00 );
payableObjects[ 1 ] = new Invoice( "56789", "tire", 4, 79.95 );
payableObjects[ 2 ] = new SalariedEmployee( "John", "Smith", "111-11-1111", 800.00 );
payableObjects[ 3 ] = new SalariedEmployee( "Lisa", "Barnes", "888-88-8888", 1200.00 );
System.out.println("Invoices and Employees processed polymorphically:\n" );
// generically process each element in array payableObjects
for ( Payable currentPayable : payableObjects )
{
  // output currentPayable and its appropriate payment amount
System.out.printf( "%s \n%s: $%,.2f\n\n",
currentPayable.toString(),"payment due", currentPayable.getPaymentAmount();
    }//endfor
  } // end main
} // end class PayableInterfaceTest
```

Draw the complete UML