

ADAPTER PATTERN


```
public interface ReservasSL {  
  
    String hotelInfo(String idHotel);  
  
    long creaReserva (String idHotel, String cliente, Date fecha, int dias);  
  
    String datosReserva (long codigo);  
  
}
```



```
public class MiSistemaReservas implements ReservasSL {  
  
    ...  
  
}
```


Problema: queremos integrarnos con un sistema internacional ... que usa su propia interface

nombres diferentes

```
public interface StandardReservasACME {
```

```
String getInfoHotel(String idHotel);
```

```
String createReservation (Date fecha, int dias, String idHotel, String cliente);
```

```
String getReservation (String codigo);
```

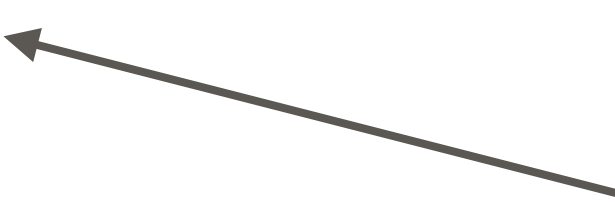
```
}
```

esto era un long

parámetros en otro orden

Solución 1

```
public class MiSistemaReservas implements ReservasSL, StandardReservasACME {  
  
    @Override  
    public String getInfoHotel(String idHotel) {  
        return hotelInfo(idHotel);  
    }  
  
    @Override  
    public String hotelInfo(String idHotel) {  
        return "info de hotel " + idHotel;  
    }  
  
    ...  
}
```



Adapta método existente a la nueva API

PROBLEMAS

- Rompe el principio de Single Responsibility
- ¿Y si salen nuevos estándares?
- ¿Y si proporcionan métodos incompatibles entre ellos?:
 - estándar1 : getInfo (String reserva)
 - estándar2: getInfo (String hotel)
- ... ¡Escala mal!

“

Convert the interface of a class into another interface clients expect. Adapter lets classes work together that couldn't otherwise because of incompatible interfaces. Aka Wrapper

— *GoF*

”


```
public class StandarReservasACMEAdapter implements StandardReservasACME {
```

```
    private ReservasSL miSistema;
```

```
    public StandarReservasACMEAdapter (ReservasSL miSistema) {  
        this.miSistema = miSistema;  
    }
```

la clase actua como
"envoltorio" del sistema
existente (ReservaSL)

```
@Override
```

```
public String getInfoHotel(String idHotel) {  
    return miSistema.hotelInfo(idHotel);  
}
```

← Adapta las llamadas a la API existente

```
@Override
```

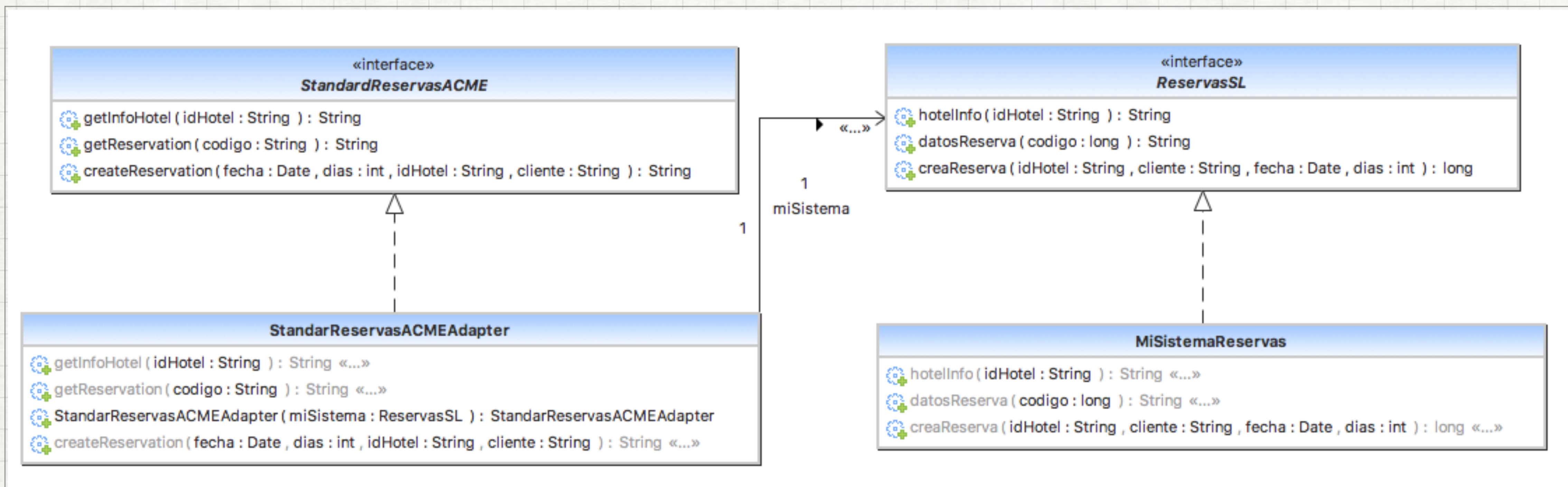
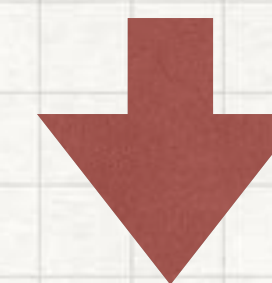
```
public String createReservation(Date fecha, int dias, String idHotel, String cliente) {  
    long codigo = miSistema.creaReserva(idHotel, cliente, fecha, dias);  
    return String.valueOf(codigo);  
}
```

```
@Override
```

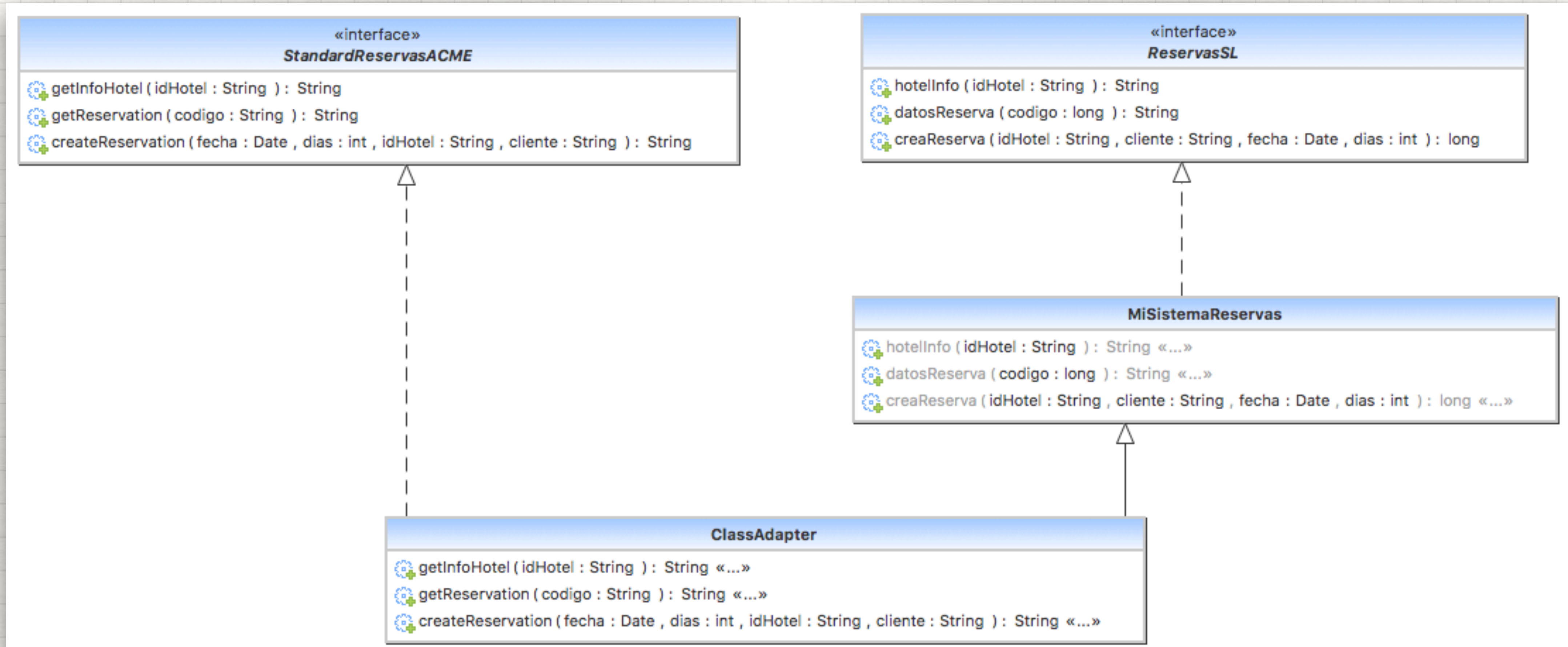
```
public String getReservation(String codigo) {  
    long codigoLong = Long.parseLong(codigo);  
    return miSistema.datosReserva(codigoLong);  
}
```

```
}
```

Nueva
API



Object adapter vs Class adapter



Class adapter

```
public class ClassAdapter extends MiSistemaReservas implements StandardReservasACME {  
  
    @Override  
    public String getInfoHotel(String idHotel) {  
        return hotelInfo(idHotel);  
    }  
  
    @Override  
    public String createReservation(Date fecha, int dias, String idHotel, String cliente) {  
        long codigo = creaReserva(idHotel, cliente, fecha, dias);  
        return String.valueOf(codigo);  
    }  
  
    @Override  
    public String getReservation(String codigo) {  
        long codigoLong = Long.parseLong(codigo);  
        return datosReserva(codigoLong);  
    }  
}
```

comparación con object adapter:
return miSistema.hotelInfo(idHotel);

ADAPTER PATTERN