Patró Adapter

Enginyeria del Software

David Moreno Oliveras

Definició

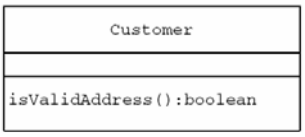
- Patró que converteix l'interfície d'una classe en una altra classe que és la que esperen els clients.
- Permet que les classes treballin juntes,permet una *compatibilitat* entre interfícies (adaptador).
- Es crea una nova classe amb accés a la classe interfície actual i amb la que es vol treballar.

Elements Bàsics

- Classe ***Client***
- Classe ***Target:*** classe abstracta que espera el client, amb la que treballa el client.
- Classe ***Adaptee***: Element que necessita ser adaptat (adaptee)
- Classe ***Adaptador***: Implementa la classe abstracta que espera el client mitjançant crides a l'objecte adaptat (Adaptador).

Disseny Inicial: Validar Adreces

- Un client vol validar adreces d'un lloc determinat (u.s)



Diversos objectes client poden crear un objecte "*Customer*" i invocar el mètode *isValidAddress* per validar les adreces clients.

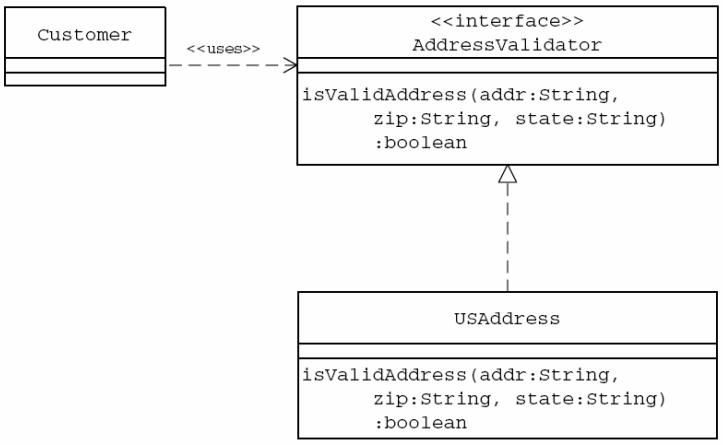
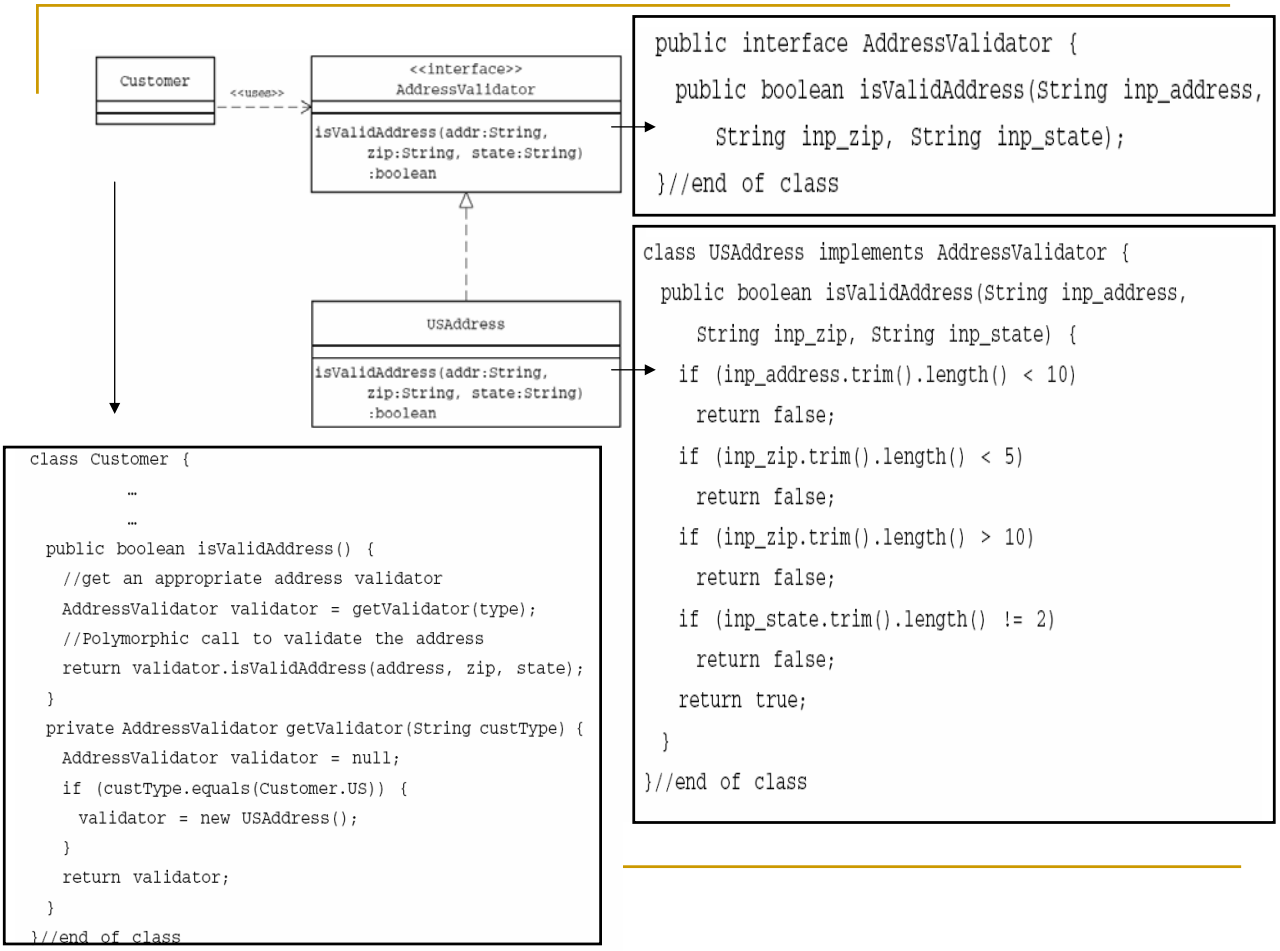
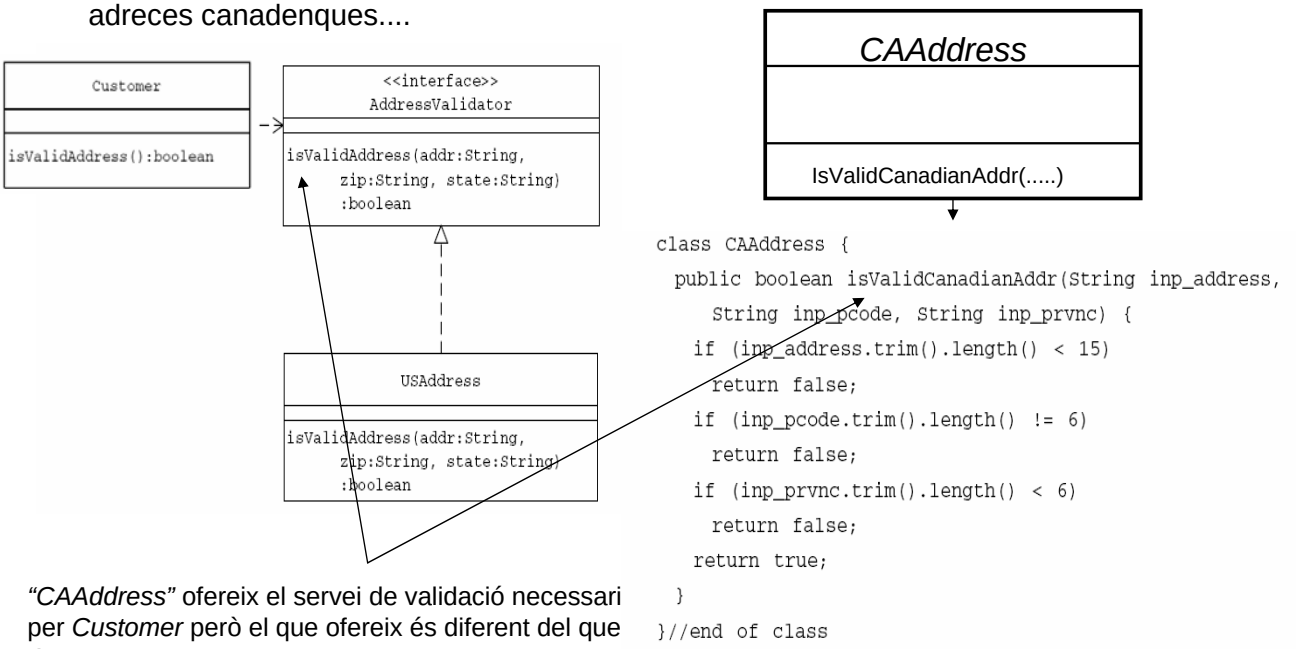


Figure 20.2 Customer/USAddress Validator: Class Association



Problema: Volem poder validar més adreces....

- Volem poder aplicar adreces pels canadencs
- Suposem que existeix una classe CAAddress amb la funcionalitat requerida per validar adreces canadenques....



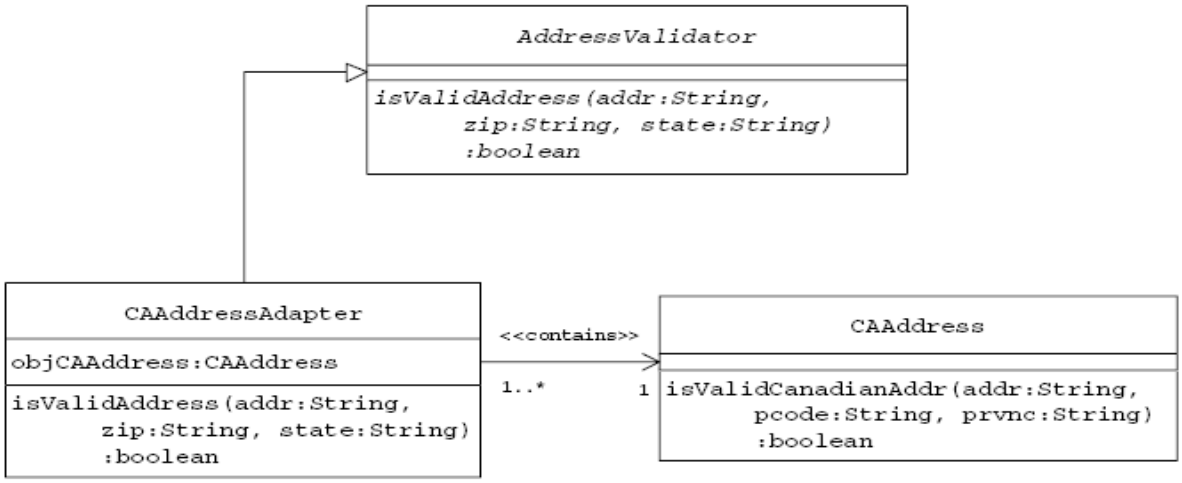
"CAAddress" ofereix el servei de validació necessari per *Customer* però el que ofereix és diferent del que *Customer* s'espera !!!

Solucions :

- A) Canviar l'interfície de la classe CAAddress ⇨ Poden haver altres aplicacions que la utilitzin d'aquesta manera...

  MALA OPCIÓ

- B) Patró Adapter ⇨ • Afegim nova classe interfície : "CAAddressAdapter" (Adaptador)



- Com que l'Adaptador CAAddressAdapter implementa l'interfície AddressValidator , els objectes client poden accedir als objectes de l'adaptador CAAddressAdapter sense problemes

Listing 20.7 Customer Class Using the CAAddressAdapter Class

```
class Customer {
          …
          …
  public boolean isValidAddress() {
    //get an appropriate address validator
    AddressValidator validator = getValidator(type);
    //Polymorphic call to validate the address
    return validator.isValidAddress(address, zip, state);
  }
  private AddressValidator getValidator(String custType) {
    AddressValidator validator = null;
    if (custType.equals(Customer.US)) {
      validator = new USAddress();
    }
    if (type.equals(Customer.CANADA)) {
      validator = new CAAddressAdapter();
    }
    return validator;
  }
}//end of class
```

Listing 20.6 **CAAddressAdapter** as a Class Adapter

```
public class CAAddressAdapter extends CAAddress
  implements AddressValidator {
  public boolean isValidAddress(String inp_address,
     String inp_zip, String inp_state) {
    return isValidCanadianAddr(inp_address, inp_zip,
           inp_state);
  }
}//end of class
```

Quan un client que vol validar una adreça cadanenca, crida el mètode "**isValidAddress**"a l'instància de CAAddressAdapter, l'adaptador la tradueix a una crida al mètode "***isValidCanadian***"

Diagrama de classes amb Patro Adapter

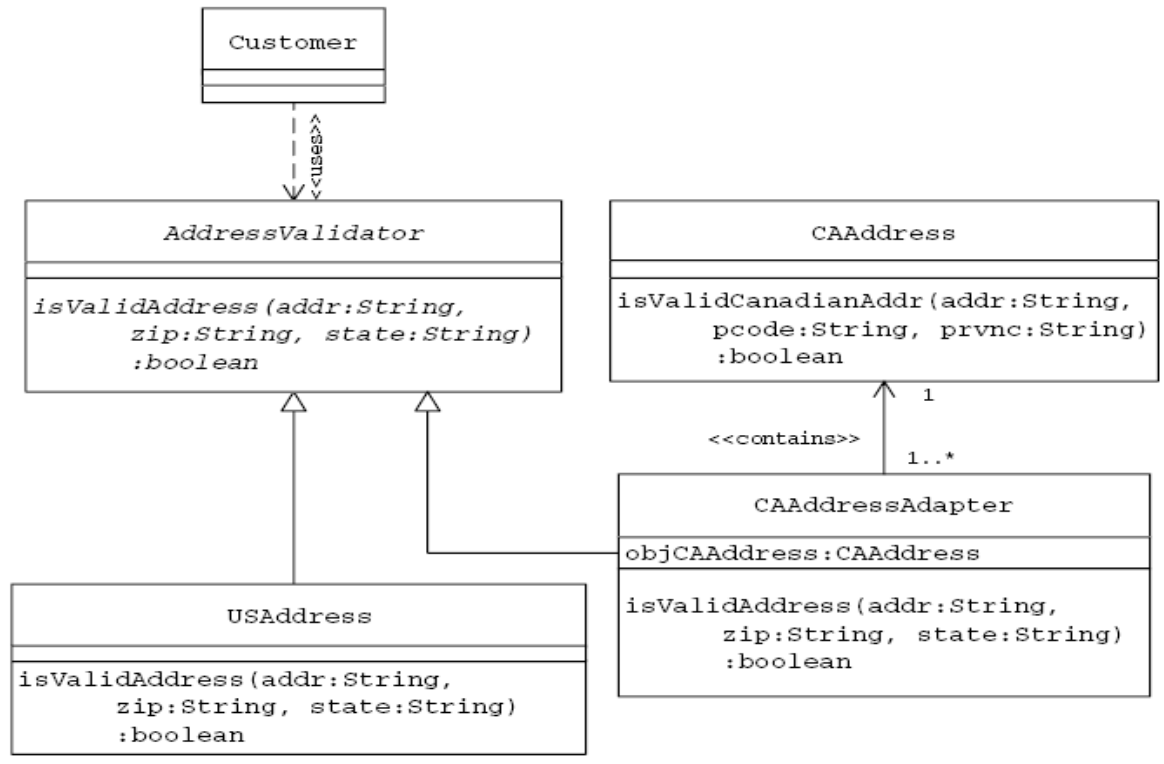


Figure 20.7 Address Validation Application: Using Object Adapter

```
class CAAddressAdapter extends AddressValidator {
  private CAAddress objCAAddress;
  public CAAddressAdapter(CAAddress address) {
    objCAAddress = address;
  }
  public boolean isValidAddress(String inp_address,
      String inp_zip, String inp_state) {
    return objCAAddress.isValidCanadianAddr(inp_address,
           inp_zip, inp_state);
  }
}//end of class
```