SOLID Printzipioak

Sarrera – Aldaketa

“Software sistemek bizi zikloan zehar aldaketak jasaten dituzte”

- Diseinu onei eta txarrei gertatzen zaie
- Diseinu onak egonkorragoak dira

Berdin da non, zer edo nola, beti **aldaketa** egongo da.

Diseinua ona izan arren, aplikazioa hazi egingo da edota **aldaketak** jasango ditu, bestela zaharkituta gelditukо da.

Aldaketa da konstante bakarra.

SOLID Printzipioak

- **S**ingle-Responsability Principle (SRP)
- **O**pen-Close Principle (OCP)
- **L**iskov Substitution Principle (LSP)
- **I**nterface Segregation Principle (ISP)
- **D**ependency Inversion Principle (DIP)

Jatorria:
http://mundogeek.net/archivos/2011/06/09/principios-solid-de-la-orientacion-a-objetos/

Single Responsability Principle (SRP)

Jatorria:
http://blogs.msdn.com/b/cdndevs/archive/2009/07/15/the-solid-principles-explained-with-motivational-posters.aspx

SRP

Moduluek eta klaseek SWeko **funtzionalitate bakarraren ardura** izan behar dute

SRP: Adibidea

```
public class CurrencyConverter {

    public BigDecimal convert(Currency from, Currency to, BigDecimal amount) {
        // gets connection to some online service and asks it to convert currency
        // parses the answer and returns results
    }

    public BigDecimal getInflationIndex(Currency currency, Date from, Date to) {
        // gets connection to some online service to get data about
        // currency inflation for specified period
    }
}
```

Zergatik kalkulatzen da inflazioa txanpon trukaketarekin?

Ez da intutiboa! Gainkargatuta dago!

Eta txanpon trukaketa zerbitzua aldatuko balitz? Edo inflazioa kalkulatzeko formatua?

Klasea bi kasuetan aldatu behar da!

SRP: Adibidea

```
public class CurrencyConverter {

    public BigDecimal convert(Currency from, Currency to, BigDecimal amount) {
        // gets connection to some online service and asks it to convert currency
        // parses the answer and returns results
    }

    public BigDecimal getInflationIndex(Currency currency, Date from, Date to) {
        // gets connection to some online service to get data about
        // currency inflation for specified period
    }
}
```

Zergatik kalkulatzen da inflazioa txanpon trukaketarekin?

Ez da intutiboa! Gainkargatuta dago!

Eta txanpon trukaketa zerbitzua aldatuko balitz? Edo inflazioa kalkulatzeko formatua?

Klasea bi kasuetan aldatu behar da!

SRP Adibidea

```
public class CurrencyConverter {

    public BigDecimal convert(Currency from, Currency to, BigDecimal amount) {
        // gets connection to some online service and asks it to convert currency
        // parses the answer and returns results
    }

    public BigDecimal getInflationIndex(Currency currency, Date from, Date to) {
        // gets connection to some online service to get data about
        // currency inflation for specified period
    }
}
```

Zergatik kalkulatzen da inflazioa txanpon trukaketarekin?

Eta txanpon trukaketa zerbitzua aldatuko balitz? Edo inflazioa kalkulatzeko formatua?

KONPONDU!!!

Ez da intutiboa! Gainkargatuta dago!

Klasea bi kasuetan aldatu behar da!

SRP: Adibidea

```
public class CurrencyConverter {
    public BigDecimal convert(Currency from, Currency to, BigDecimal amount) {
        // gets connection to some online service and asks it to convert currency
        // parses the answer and returns results
    }
}
```

Txanpon trukaketa zerbitzua aldatzen bada?
`CurrencyConverter` bakarrik aldatzen dugu!

Inflazioa kalkulatzeko formatua aldatzen bada?
`InflationIndexCounter` bakarrik aldatzen dugu!

```
public class InflationIndexCounter {
    public BigDecimal getInflationIndex(Currency currency, Date from, Date to) {
        // gets connection to some online service to get data about
        // currency inflation for specified period
    }
}
```

SRP: Adibidea

Bi ardura:
- Kaukotu
- Erabiltzailea DBtik lortu

```
public class UserAuthenticator {
    public boolean authenticate(String username, String password){
        User user = getUser(username);
        return user.getPassword().equals(password);
    }

    private User getUser(String username){
        st.executeQuery("select user.name, user.password from user where id=?");
        // something's here
        return user;
    }
}
```

SRP: Adibidea

```
public class UserAuthenticator {
    private UserDetailsService userDetailsService;
    public UserAuthenticator(UserDetailsService service) {
        userDetailsService = service;
    }
    public boolean authenticate(String username, String password){
        User user = userDetailsService.getUser(username);
        return user.getPassword().equals(password);
    }
}
```

DB-arekin zuzenean lanik ez!

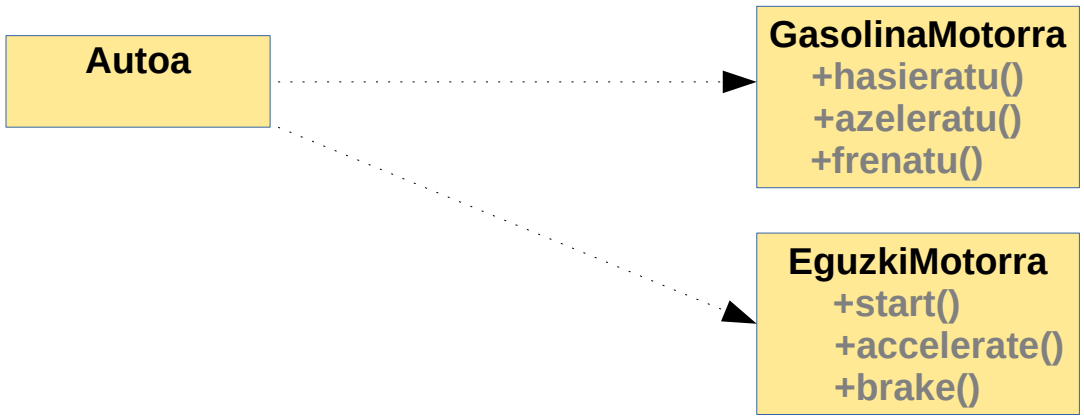
Kautoketa LDAP-arekin,
Klasea ez da aldatzen!

Open-Close Principle (OCP)

OCP

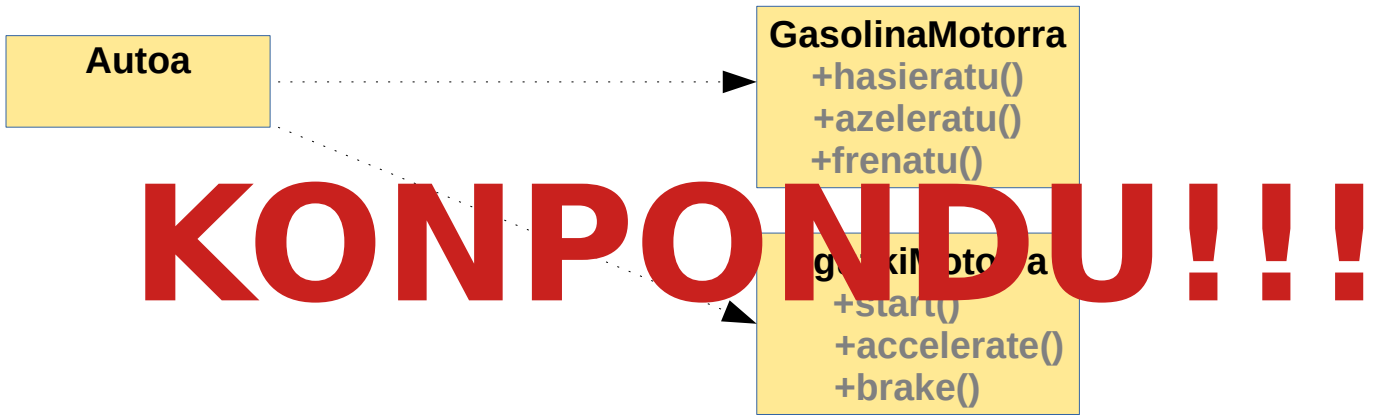
SW entitateak (funtzionalitateen) **hedapenerako irekiak**, baina (kodearen) **aldaketarako itxiak** izan behar dute

OCP: Adibidea



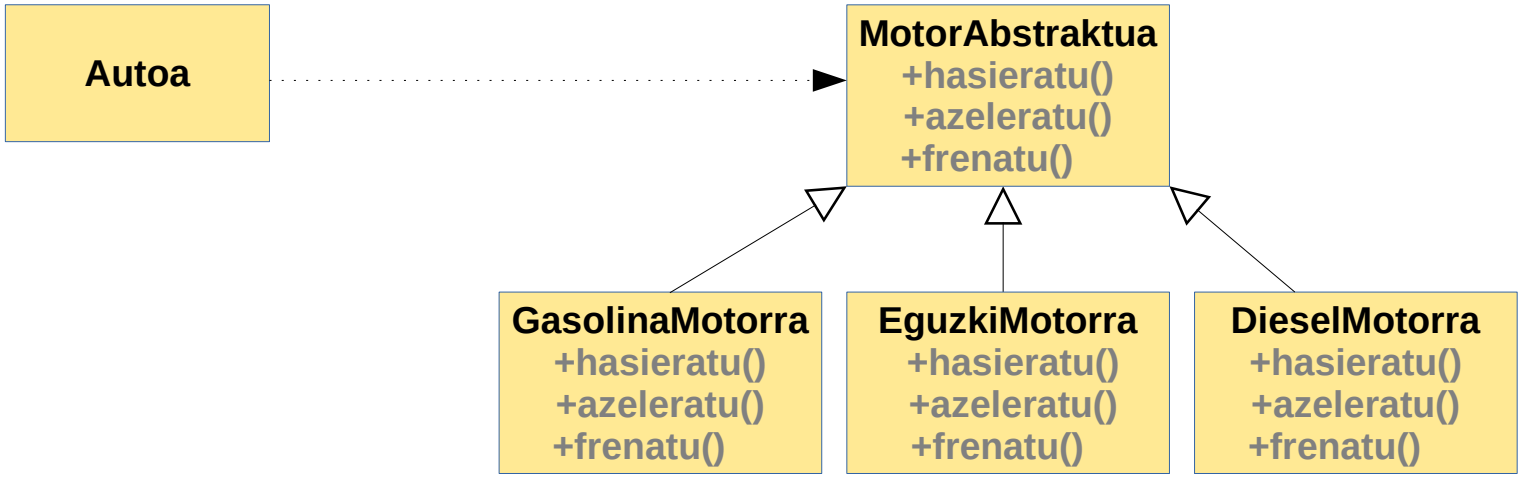
- Nola egin `Auto` batek `GasolinaMotorra` edo `EguzkiMotorra` erabiltzeko?
- `Autoa` klasea aldatu behar dugu!
  - ...gutxienez diseinu honetan

OCP: Adibidea



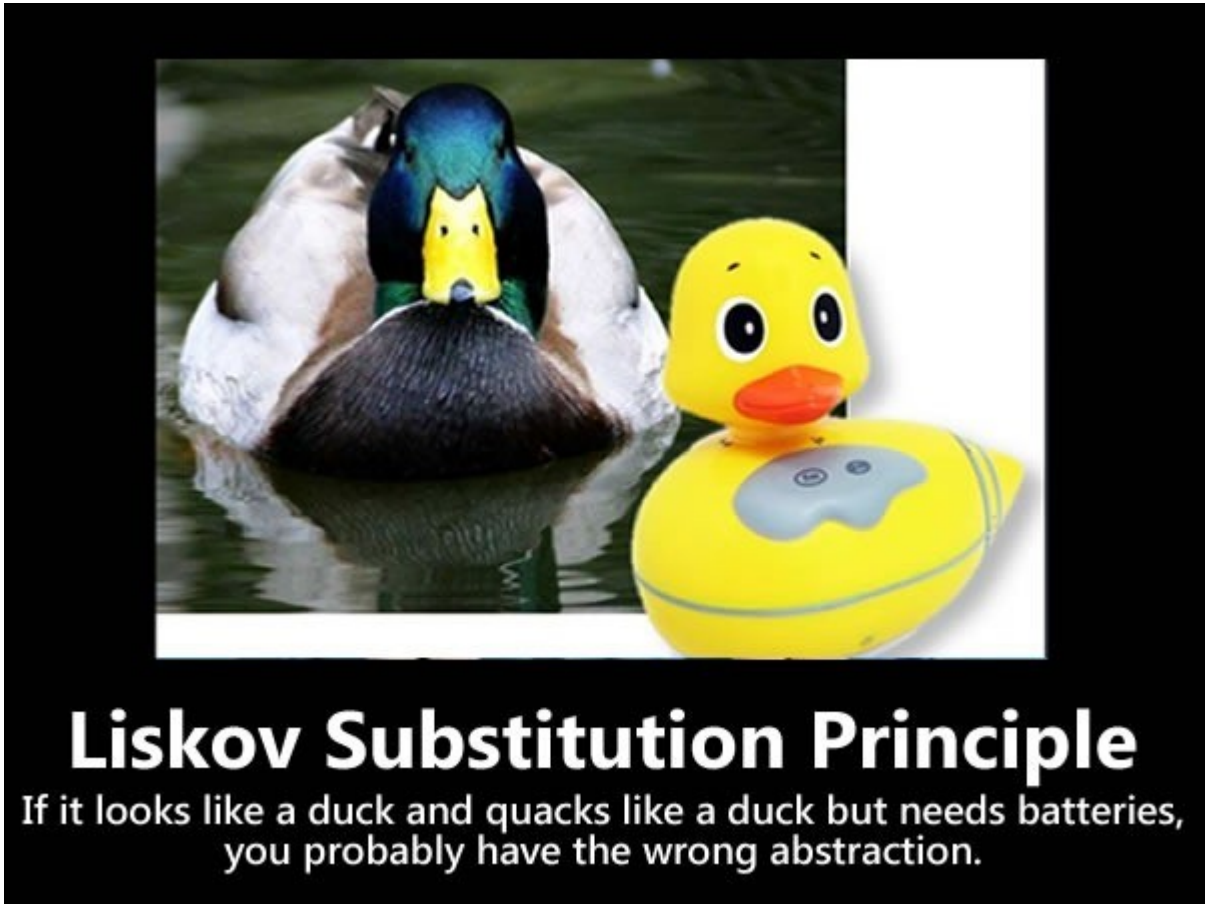
- Nola egin `Auto` batek `GasolinaMotorra` edo `EguzkiMotorra` erabiltzeko?
- `Autoa` klasea aldatu behar dugu!
  - ...gutxienez diseinu honetan

OCP: Adibidea



Ahal den einean, klase baten dependentzia **klase abstraktu** batekin izan behar du, ez konkretu batekin.

Liskov Sustitution Principle (LSP)



LSP

Klase batetik heredatutako azpiklase oro lehenengoa bezala erabili ahalko da, beren arteko desberdintasunak ezagutu barik ere.

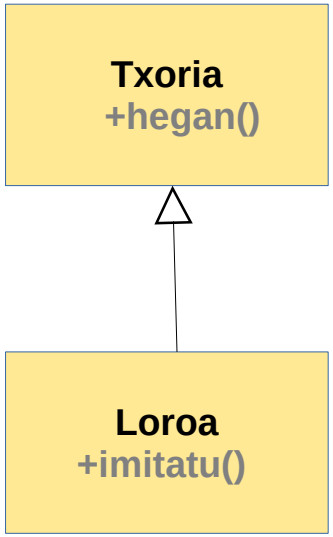
LSP: herentzia

Herentziak hurrengoa bermatuko du: **superklasearen edozein objekturen propietate** frogagarri **azpiklaseen edozein objekturentzat baliogarria** da.

B. Liskov, 1987

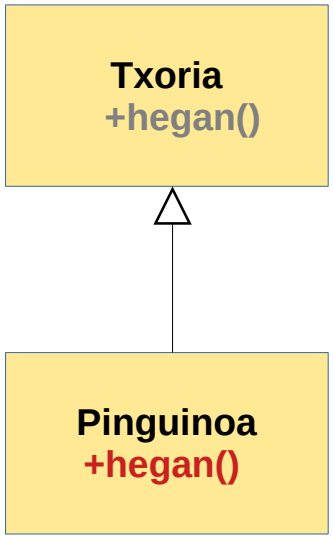
LSP: herentzia

```
abstract class Txoria {
  public abstract void hegan();
}

class Loroa extends Txoria {
  public void hegan(){...};
  public void imitatu(){...}
}

// …

Loroa nireMaskota = new Loroa();
nireMaskota.imitatu();    // Loroa izanda, imitatu() dezake
nireMaskota.hegan();      // Txoria izanda hegan() egin dezake
```



LSP: herentzia

```
class Pinguinoa extends Txoria {
   public void hegan() {
     new Exception(“ezin dut hegan egin!”);
   }
};
```



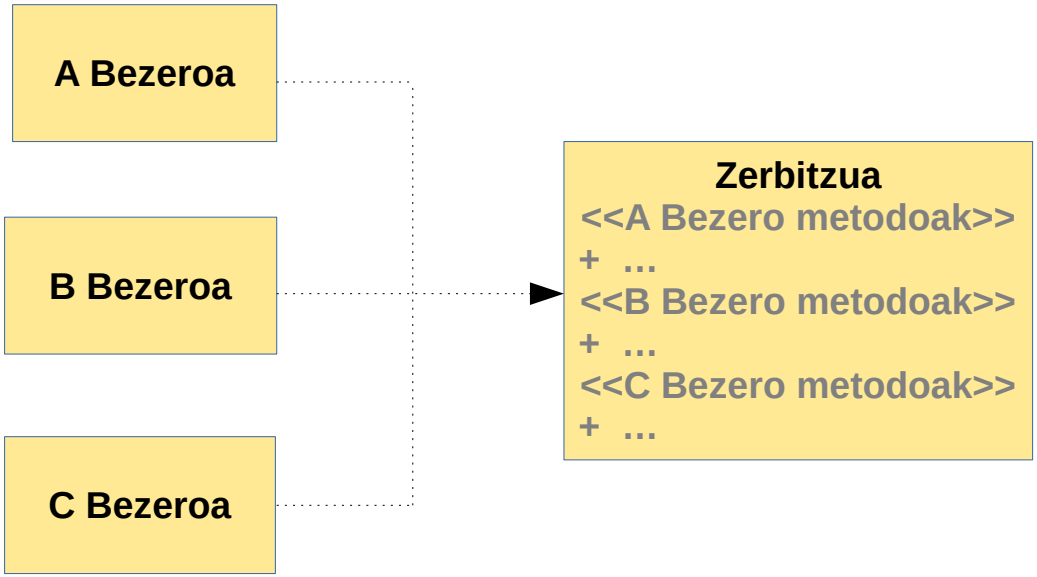
- Ez du *“Pinguinoek ezin dute hegan egin”* modelatzen
  - *“Pinguinoek hegan eginez gero, errorea!!”* modelatzen du
  - **Liskov printzipioa ez da betetzen**

Interface Segregation Principle (ISP)

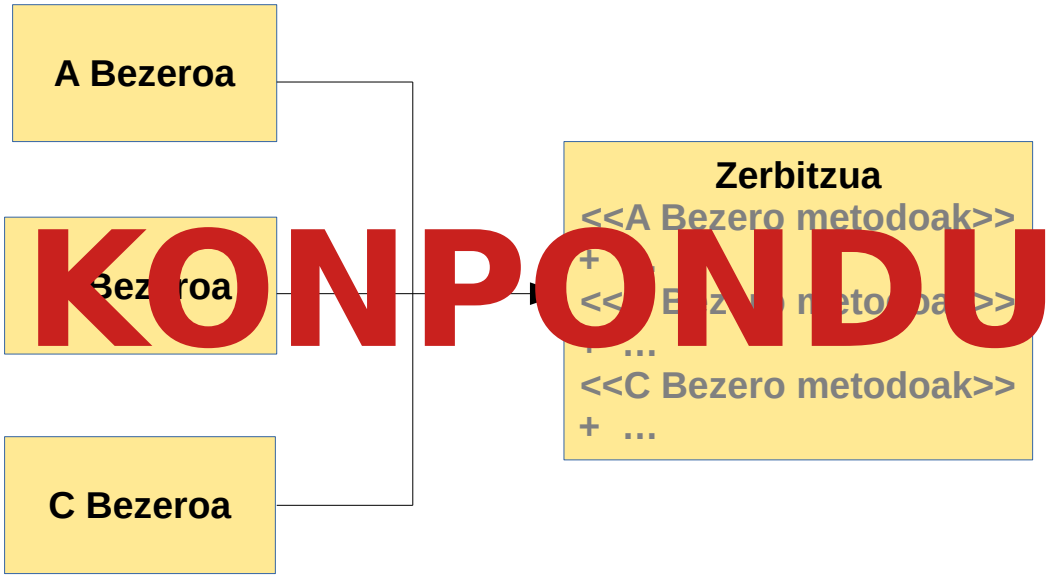
ISP

Bezeroek erabiltzen ez duten metodoen dependentziarik ez dute izan behar

ISP: Adibidea

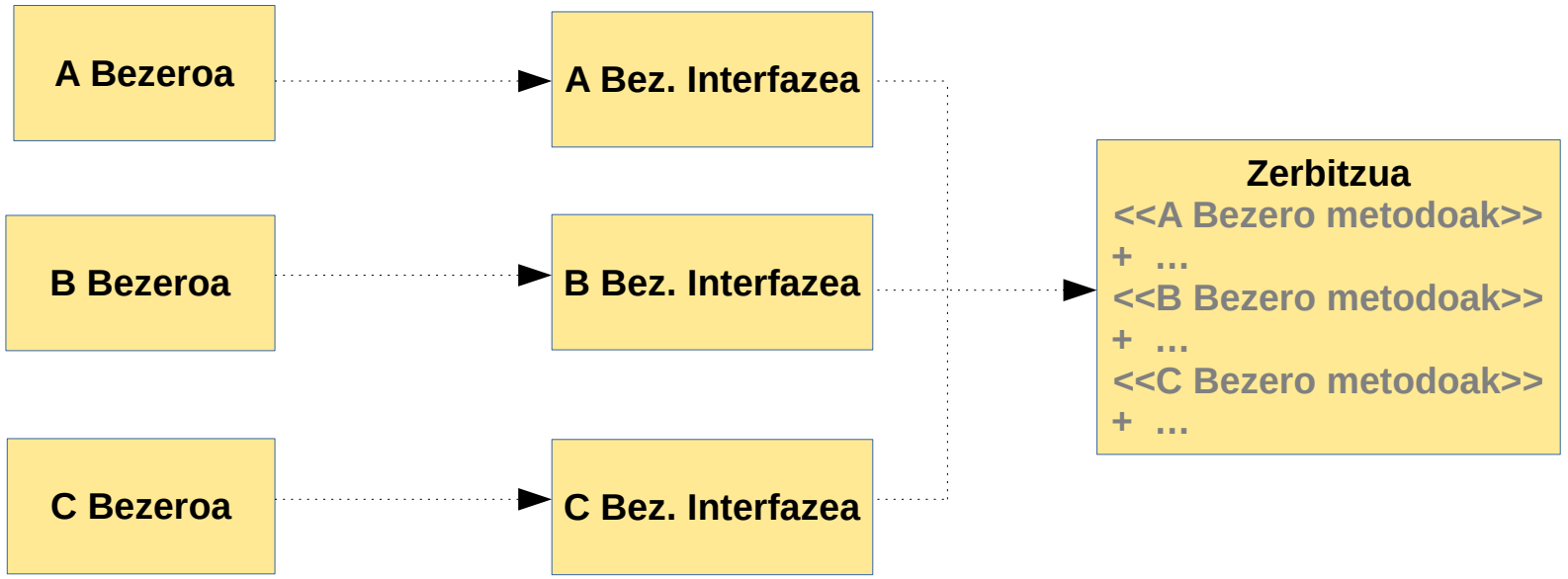


ISP: Adibidea

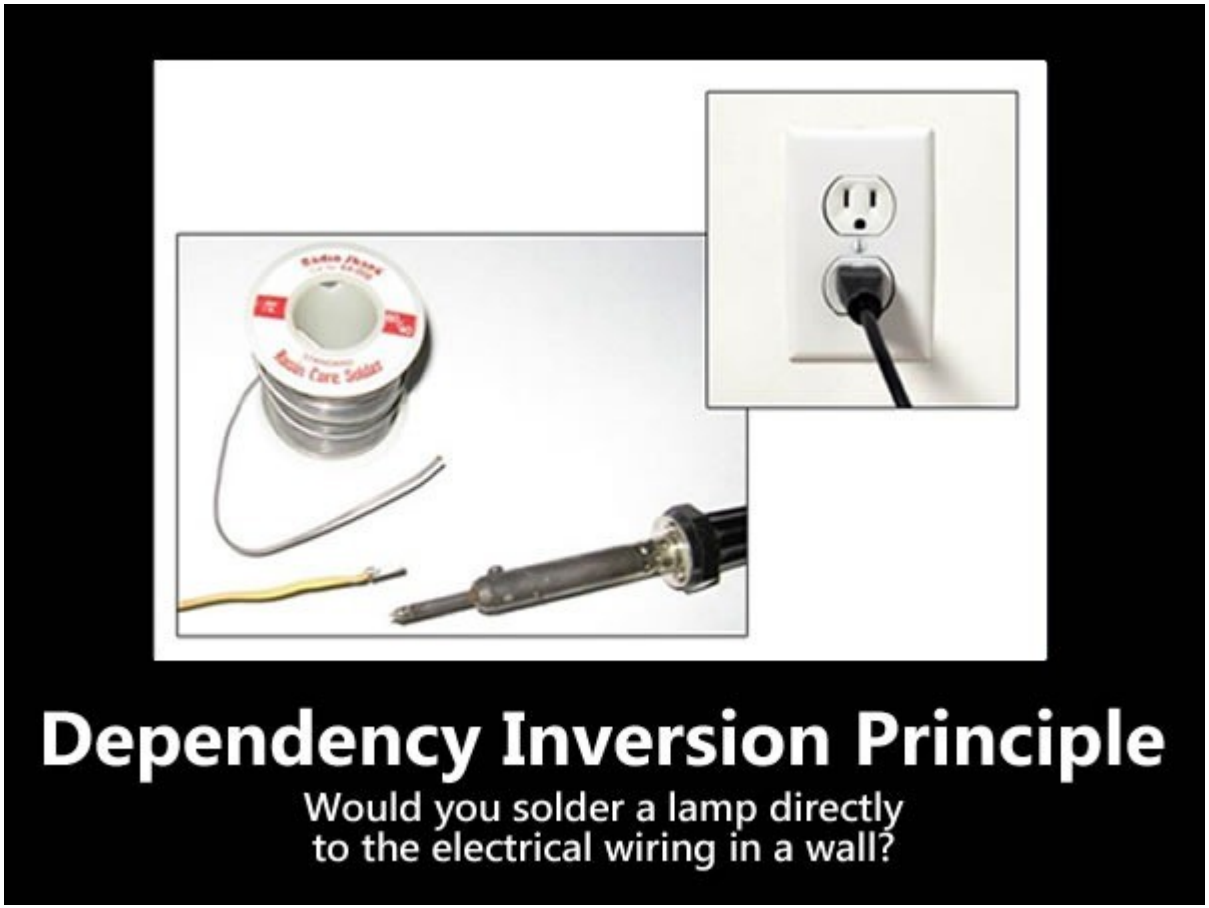


KONPONDU!!!

ISP: Adibidea



Dependency Inversion Principle (DIP)



DIP

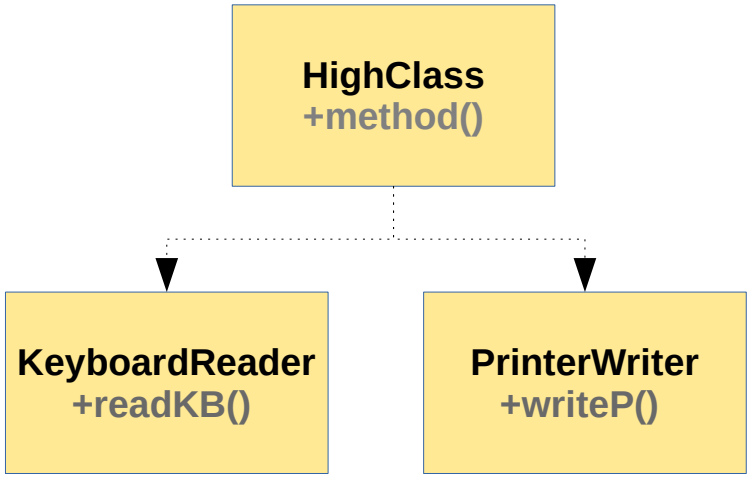
I. **Goi mailako moduluek** ezin dute behe mailako moduluekin menpekotasunik izan. **Menpekotasuna abstrakzioekin.**

II. Abstrakzioek ezin dute xehetasunekin menpekotasunik izan. **Xehetasunek abstrakzioekin menpekotasuna**.

Martin, 1996

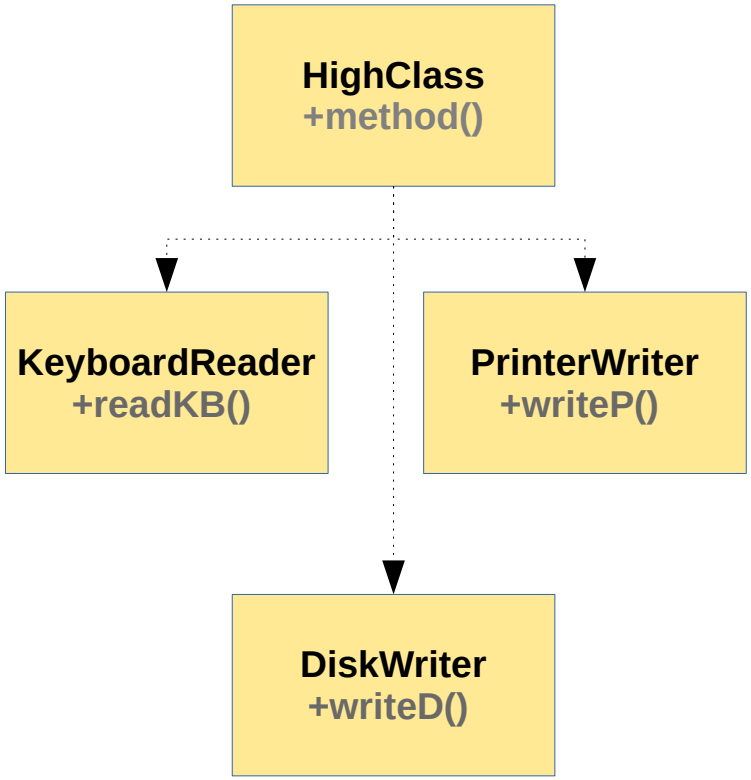
- OCPk helburua adierazten du. DIPek mekanismoa.
- Superklase batek ez ditu bere azpiklaseak ezagutu behar.
- Inplementazio xehetasunak dituzten moduluek ez dute beraien arteko menpekotasunik. Menpekotasuna abstrakzioen bidez definitzen dira.

DIP: Adibidea



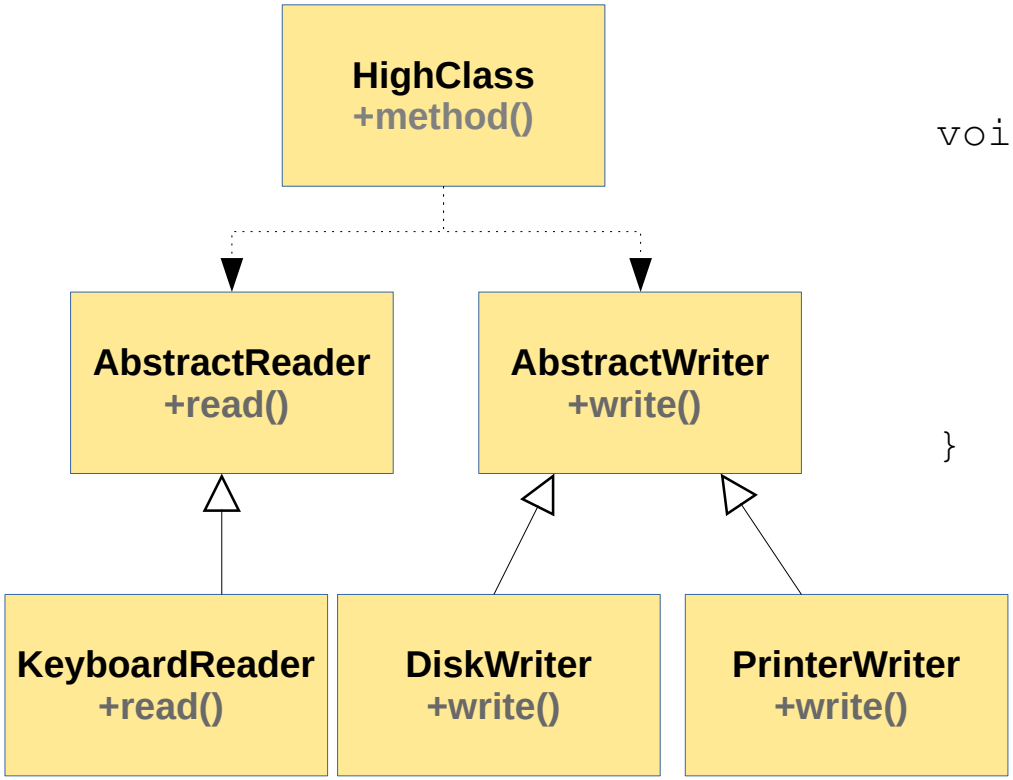

```
void method(){
    int c;
    KeyboardReader mKR;
    PrinterWriter  mPW;
    while((c=mKR.readKB())!= EOF)
       mPW.writeP(c);
}
```

DIP: Adibidea




```
void method(ODev pDev){
    int c;
    KeyboardReader mKR;
    PrinterWriter  mPW;
     DiskWriter     mDW;

    while((c=mKR.readKB())!= EOF){
      if(pDev==PRINTER) mPW.writeP(c);
       else             mDW.writeD(c);

    }

}
```

DIP: Adibidea




```
void method(Reader r, Writer w){
    int c;
    while((c=r.read())!= EOF){
      w.write(c);
    }

}
```

Laburbilduz

- SW sistemek **aldaketak** beren bizi zikloan
- SW diseinuak aldaketetara **moldatu** behar dira
- SOLID printzipioak hastapeneko pausuak, diseinu konplexuak egiteko teknika sofistikatuagoak