Sarrera

Laborategi honetan SOLID printzipioetan oinarritzen diren ariketa batzuk proposatzen dira.

Entregatu behar dena (TALDEKO LANA)

PDF dokumentu bat:
a) SOLID printzipio bakoitzeko atal batekin. Atal bakoitzean, printzipio bakoitzari eskatzen diren erantzunak arkeztuko dira.
b) Github esteka bat eclipse-ko proiektu batekin. Proiektuak SOLID printzipio bakoitzeko pakete bat izango du SOLID printzipioren izenarekin (adb: ocp). Bere barruan, eskatzen den inplementazioa gartuko da.

Ireki-Itxia Printzipioa (OCP).

Hurrengo klaseak emanda (klase batzuk bukatu gabe daude):

```
public class Figure {
        public abstract void draw(){};
}
public class Square extends Figure {float  lenght;}
public class Circle extends Figure {float diameter;}

public class Sheet{
        Vector< Square> squares=new Vector< Square>();
        Vector<Circle> circles= new Vector< Circle>();

        public void addCircle(Circle c){
                circles.add(c);
        }
        public void addSquare(Square s){
                squares.add(s);
        }
        public void drawFigures(){
                Enumeration<Square> esquares=squares.elements();
                Square s;
                while (esquares.hasMoreElements()){
                        s=esquares.nextElement();
                        s.draw();
                }
                Enumeration<Circles> ecircles=circles.elements();
                Circle c;
                while (circles.hasMoreElements()){
                        c=ecircles.nextElement();
                        c.draw();
                }
        }
}
```

Galderak:

1. Zer gertatuko litzateke Sheet klaseari "Diamond" gehituko bagenio? Sheet klaseak OCP printzipioa betetzen du?. Erantzuna justifikatu. Betetzen ez badu, behar diren aldaketak egin printzipioa betetzeko.
2. Aplikazio bat sortu behar diren klase guztiekin, eta programa nagusi batean, drawFigures() metodoari deitzen dion programa bat sortu, bi circle, square bat eta diamond objektu batekin.
3. Egin duzun aldaketa errefaktorizazio bat dela kontsideratzen duzu? Justifikatu erantzuna.
4. Figure bakoitzak azalera bat edukiko balu (hau da, getArea()), eta Sheet batean dauden irudi guztien azalera kalkulatu nahi bagenu, nola osatuko zenuke klaseak. Programa nagusia osatu, eskakizun hau frogatzeko.

Erresponsabilitate bakarreko printzipioa (SRP).

Hurrengo Bill klasea emanda:

```
public class Bill {
        public String code;
        public Date billDate;
        public float billAmount;
        public float VAT;
        public float billDeduction;
        public float billTotal;
        public int deductionPercentage;

        // Fakturaren totala kalkulatzen duen metodoa.
        public void totalCalc() {
                // Dedukzioa kalkulatu
                billDeduction = (billAmount * deductionPercentage) / 100;
                // VAT kalkulatzen dugu
                VAT = (float) (billAmount * 0.16);
                // Totala kalkulatzen dugu
                billTotal = (billAmount - billDeduction) + VAT;
        }
}
```

Esan genezake klase honen erresponsabilitatea faktura baten totala kalkulatzea dela, eta horixe da berez egiten duena. Baina ez da egia klaseak erresponsabilitate bakar bat daukadala: totalCalc() metodoaren inplementazioan arreta ipintzen badugu, fakturaren totala kalkulatzeaz gain, fakturaren dedukzio batzuk eta fakturaren VAT-a ere kalkulatzen duela ikusi dezakegu. Arazoa dago, etorkizunean VAT-a aldatu behar badugu edo dedukzioak beste modu desberdin batean kalkulatzen badira (Adibidez, faktura kopuruaren arabera dedukzio portzentai deberdinak aplikatzen dira) Bill klasea aldatu beharko dugula komentatutako arrazoiengatik; beraz, uneko diseinuarekin erresponsabilitateak akoplatuta daude, eta SRP printzipioa bortxatzen du.

Eskatzen da:

1. Aplikazioa errefaktoriazatu, erresponsabilitate bakoitza klase isolatu batean geratu dadin. Adierazi zein aldaketa egin beharko litzateke fakturaren dedukzioa (hau da, billDeduction) fakturaren arabera kalkulatuko balitz:

```
if (billAmount >50000)
        billDeduction = (billAmount * deductionPercentage +5) / 100;
else billDeduction = (billAmount * deductionPercentage) / 100;
```

2. Zein aldaketa egin beharko litzateke VAT-a %16-tik %18-ra aldatuko balitz?
3. Zein aldaketa egin beharko litzateke, 0 kodea duten fakturei VAT-a aplikatzen EZ bazaie.

Liskov printzipioa (LSK).

ProjectFile objektuez osatutako *Project* klase bat daukagu. Klase honetako metodoen bidez proiektuaren fitxategiak gorde eta kargatu daitezke. Bere inplementazioa hurrengoa da:

```
public class Project {
	public Vector<ProjectFile> files=new Vector<ProjectFile>();

	public void addProject(ProjectFile p){
		files.add(p);
	}

	public void loadAllFiles(){
		for (ProjectFile f:files)
			f.loadFile();
	}

	public void storeAllFiles(){
		for (ProjectFile f:files)
			f.storeFile();
	}
}
```

ProjectFile klasearen inplementazioa hurrengoa da:

```
public class ProjectFile {
	public String filePath;

	public ProjectFile(String filePath){
		this.filePath=filePath;
	}
	public void loadFile(){
			System.out.println("file loaded from "+filePath);
	}
	public void storeFile(){
		System.out.println("file saved to "+filePath);
	}
}
```

Baina, aplikazioaren hurrengo bertsio batean, fitxategi batzuk ezin direla aldatu konturatu dira, hau da, irakurketako fitxategiak dira soilik. Horretarako, aurreko klasea espezializatzen duen klase berri bat sortu dute. Bere inplementazioa hurrengoa da:

```
public class ReadOnlyProjectFile extends ProjectFile{

	public ReadOnlyProjectFile(String filePath) {
		super(filePath);
		// TODO Auto-generated constructor stub
	}
	public void storeFile() {
			System.out.println("ERROR, can not be Saved");
	}
}
```

Galderak:

1. Programa batean, Project bat sortu 3 fitxategi mota desberdin batzuekin, eta jarraian bere metodoak exekutatu.
2. *Project* klaseak OCP printzipioa betetzen du?. Justifikatu erantzuna.
3. *Project* klaseak LSK (Liskov) printzipioa betetzen du?. Justifikatu erantzuna.
4. Errefaktorizatu aplikazioa OCP edo LSK betetzen ez badu.

Dependentzi inbertsioaren printzipioa (DIP).

Student klasea hurrengoa da:

```
public class Student {
        public String name;
        public String sex;
        public String year;
        // matrikulatuta dagoen espedientea
        public HashMap<String,Integer> subjectRecord;
        // ikasleak ordaindu behar duena
        public String toCharge;

        // Irakasgai batean matrikulatzen duen metodoa.
        public void register(String subject) {
                // Aurrebaldintzak konprobatzen dira
                Preconditions p=new Preconditions();
                boolean isPosible = p.isPosible(subject , subjectRecord);
                if (isPosible) {
                  // Dedukzioa kalkulatu sex eta edadearen arabera
                  Deduction d=new Deduction();
                  int percentage = d.calcDeduction(sex, year);
                // Irakasgaiaren prezioa lortu
                  SubjectQuotes sq=new SubjectQuotes();
                  int quote = sq.getPrice(subject);
                  // HashMap batean gordetzen du eta ordaindu behar duen balioa eguneratu
                  subjectRecord.put(subject,null);
                  toCharge = toCharge+(quote-percentaje*quote/100);
                }
        }
}
```

Galderak:

1. DIP printzipioa betetzen du. Justifikatu erantzuna.
2. Errefaktorizatu kodea betetzen ez badu.

Interfaze Bananduaren Printzipioa (ISP).

Hurrengo Person klasea ematen digute:

```
public class Person {
   String name, address, email, telephone;
   public void setName(String n) { name=n; }
   public String getName() { return name; }

   public void setAddress(String a) { address=a; }
   public String getAddress() { return address; }

   public void setEmail (String e) { email=ea; }
   public String getEmail () { return email; }

   public void setTelephone(String t) { telephone=t; }
   public String getTelephone() { return telephone; }

}
```

eta mail-ak eta SMS-ak bidaltzen dituzten bi klase osagarri::

```
public class EmailSender {
  public static void sendEmail(Person c, String message){
         // Mezu bat bidaltzen du Person klaseko korreo helbidera. }
}

public class SMSSender {
   public static void sendSMS(Person c, String message){
         //SMS bat bidaltzen du Person klaseko telefono zenbakira. }
}
```

Galderak:

1. Zer informazio behar dute EmailSender eta SMSSender klaseek egin behar duten funtzionalitatea betetzeko, eta zer informazio jasotzen dute? ISP printzipioa bortxatzen dutela uste duzu?.
2. Aurreko klaseak errefaktorizatu (EmailSender eta SMSSender), Person parametroa aldatuz interface batengatik (klase bakoitzak interfaze desberdina). Interface bakoitzak klase bakoitzak behar duen metodoez osatuta egongo da bere eginkizuna betetzeko, hau da, mail bat edo sms bat bidaltzeko. Person klasea ere aldatu.
3. GmailAccount klasea osatu. Klase honetako objektuek (aldaketa batzuekin) EmailSender klasera bidaltzeko ahalmena izan beharko lukete baina ez SMSSender klasera.

   ```
   public class GmailAccount {
      String name, emailAddress;
   }
   ```

   Programa bat sortu, EmailSender klaseko sendEmail metodoari deitzen diona GmailAccount objektu batekin.