SOLID laborategia

Aritz Plazaola Cortabarria

kodea: https://github.com/Poxito/SOLID

OCP

1. Sheet klaseari Diamond klasea gehitzen badugu OCP printzipoa apurtuko litzateke, Sheet klasean aldaketak egitea derrigortuko bailiguke.

```
public class Diamond extends Figure{

	int x, y, aldea, altuera, zabalera;

	public Diamond(int aldea, int x, int y, int altuera, int
zabalera) {
		this.aldea = aldea;
		this.x = x;
		this.y = y;
		this.altuera = altuera;
		this.zabalera = zabalera;
	}

	public void draw() {
		System.out.println("Diamond " + x + ", " + y + ", " +
aldea);
	}
}


public class Sheet {

Vector<Square> squares = new Vector<Square>();
	Vector<Circle> circles = new Vector<Circle>();
	Vector<Diamond> diamonds= new Vector<Diamond>();

	public void addCircle(Circle c){
		circles.add(c);
	}

	public void addSquare(Square s){
		squares.add(s);
	}

	public void addDiamond(Diamond d){
		diamonds.add(f);
```

```
	}

	public void drawFigures(){
		Enumeration<Square> esquares = squares.elements();
		Square s;
		while (esquares.hasMoreElements()){
			s=esquares.nextElement();
			s.draw();
		}
		Enumeration<Circle> ecircles = circles.elements();
		Circle c;
		while (ecircles.hasMoreElements()){
			c = ecircles.nextElement();
			c.draw();
		}
		Enumeration<Diamond> ediamonds = diamonds.elements();
		Diamond d;
		while (ediamonds.hasMoreElements()){
			f = ediamonds.nextElement();
			f.draw();
		}

	}

}
```

Egin beharreko aldaketak: Sheet klasean Circle, Square eta Diamond motako objektuak zuzenean tratatu beharrean, Figure izeneko klase abtraktu bat sortu eta sortutako objektu horiek tratatu. Circle, Square eta Diamond klaseek Figure klasea zabalduko (extends) dute.

2. Honakoa da main metodoa

```
public static void main(String args[]) {
	Square square1 = new Square(2, 3, 4, 5);
	Circle circle1 = new Circle(3, 4, 5);
	Circle circle2 = new Circle(1, 2, 3);
	Diamond diamond1 = new Diamond(3, 2, 2, 4, 2);

	Sheet sheet = new Sheet();
	sheet.addFigure(circle1);
	sheet.addFigure(circle2);
```

```
sheet.addFigure(square1);
      sheet.addFigure(diamond1);

sheet.drawFigures();
}
```

Output:

Circle 3, 4, 5
Circle 1, 2, 3
Square 2, 3, 4, 5
Diamond 2, 2, 3

3. Bai. Klase bakoitzak inplemetazio desberdineko metodo berdinak zuten (*draw()*), errepikatutako metodo horiek Figure izeneko klase abstraktuan gordetzen badira, kodea enkapsulatzen (**Extracting**) ari gara. Eta kodea enkapsulatzea errefaktorizazio kontsideratzen da..
4. Figure klase abstraktuan sortuko litzateke metodoa eta gero beste klase haurrek metodoa berridatziko lukete.

```
public class Sheet {

      Vector<Figure> figures = new Vector<Figure>();

      public void addFigure(Figure f){
            figures.add(f);
      }

      public void drawFigures(){

            Enumeration<Figure> efigures = figures.elements();
            Figure f;
            while (efigures.hasMoreElements()){
                  f = efigures.nextElement();
                  f.draw();
            }

      }

      public void getAllAreas() {
            Enumeration<Figure> efigures = figures.elements();
            Figure f;
```

```
            while (efigures.hasMoreElements()){
                f = efigures.nextElement();
                System.out.println(f.getArea());
            }
        }
    }

    public static void main(String args[]) {
        Square square1 = new Square(2, 3, 4, 5);
        Circle circle1 = new Circle(3, 4, 5);
        Circle circle2 = new Circle(1, 2, 3);
        Diamond diamond1 = new Diamond(3, 2, 2, 4, 2);

        Sheet sheet = new Sheet();
        sheet.addFigure(circle1);
        sheet.addFigure(circle2);
        sheet.addFigure(square1);
        sheet.addFigure(diamond1);

        sheet.drawFigures();

        sheet.getAllAreas();
    }
```

Output:

```
Circle 3, 4, 5
Circle 1, 2, 3
Square 2, 3, 4, 5
Diamond 2, 2, 3
Circle area = 78.53982
Circle area = 28.274334
Square area = 20.0
Diamond area = 4.0
```

SRP

1. Dedukzioa kalkulatzeaz arduratzen den klaseko metodoa aldatu beharko litzateke bertan *if-else* bat jarriz. BillDeduction klasea harduratuko da fakturaren araberako dedukzioa kalkulatzeaz.

```
public class BillDeduction {
	public float billDeduction;
	public int deductionPercentage;

	public BillDeduction() {

	}

	public Float deductionCalc(Float amount, int percentage) {
		if(amount > 50000) {
			return (amount * percentage + 5) / 100;
		}else {
			return (amount * percentage) / 100;
		}
	}
}
```

2. VAT-a kalkulatzeaz arduratzen den klaseak *VATpercentage* atributu bat gordeko luke momentuko portzentaia gordetzeko. %16tik %18ra aldatu nahi bada erakitzaileko parametroa aldatu beharko litzateke.

```
public class BillVAT {

	public float VATPercentage;

	public BillVAT(float percentage) {
		this.VATPercentage = percentage;
	}

	public Float vatCalc(float amount) {
		return (amount * VATPercentage);
	}
}
```

3. Fakturaren totala kalkulatzen duen klaseko (**BillTotal**) metoari (**totalCalc**) kodea parametro bezala pasako zaio, eta horrela metodoak *if-else* baten bidez kodea 0 duten fakturei ez zaie VAT-a aplikatuko.

```
public class BillTotal {
	public float billAmount;
```

```
	public BillTotal() {
	}

	public Float totalCalc(float amount, float VAT, float
deduction, String kodea) {
		if(kodea.equals("0")) {
			return (amount - deduction);
		}else {
			return (amount - deduction) + VAT;
		}
	}
}
```

4. Bill klasean atributu batzuk gehituko behar dira eta totalCalc() metodoa honela geratuko litzateke:

```
public BillDeduction deductor = new BillDeduction();
public BillVAT vatCalculator = new BillVAT((float) 0.16);
//%18 aldatu nahi bada erakitzaileko parametroa aldatu beharko
litzateke
public BillTotal totalCalculator = new BillTotal();


public void totalCalc() {
	// Dedukzioa kalkulatu
	billDeduction = deductor.deductionCalc(billAmount,
deductionPercentage);
	// VAT kalkulatzen dugu
	VAT = vatCalculator.vatCalc(billAmount);
	// Totala kalkulatzen dugu
	billTotal = totalCalculator.totalCalc(billAmount, VAT,
billDeduction, code);
}
```

LSK

1. Lehenik eta behin proiektu bat sortu dut. Ondoren bi fitxategi mota desberdin. Proiektura gehitu ostean metodoak exekutatu ditut.

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

	//1. Puntuan egin beharrekoa
	Project proiektua = new Project();

	ProjectFile pf1 = new ReadOnlyProjectFile("enuntziatua.pdf");
	ProjectFile pf2 = new ProjectFile("txostena.pdf");
	//Errefaktorizazioaren ostean proba hauek ez dute baliok.

	proiektua.addProject(pf1);
	proiektua.addProject(pf2);

	proiektua.loadAllFiles();
	proiektua.storeAllFiles();

}
```

Output:

```
file loaded from enuntziatua.pdf
file loaded from txostena.pdf
ERROR, can not be Saved
file saved to txostena.pdf
```

2. OCP betetzen du? Bai! Hasierako inplementazioarekin OCP printzipioa bermatzen da **Project** fitxategian ez baita aldaketarik egin behar.

3. LSK betetzen du? Ez! Klaseen arteko herentzia ez da ondo gauzatu. **ProjectFile** eta **ReadOnlyProjectFile** klaseek *storeFile*() metodoaren inplementazio desberdina dute, hau da, **ReadOnlyProjectFile** klase haurrak **ProjectFile** guraso klasearen portaera aldatzen du.

4. Arazoa *storeFile*() metodoan dago, beraz metodo hori interfaze batera mugituko dugu.

```
public interface IStoreFile {

	public void storeFile();

}
```

Ondoren, **IStoreFile** interfazeko metodoa inplementatzen duen **StoreableProjectFile** klasea sortuko dugu. **StoreableProjectFile ProjectFile** klasearen klase haur bat izango da.

```
public class StoreableProjectFile extends ProjectFile implements
IStoreFile{


	public StoreableProjectFile(String filePath) {
		super(filePath);
	}

	public void storeFile(){
		System.out.println("file saved to " + filePath);
	}
}
```

Fitxategiak gordetzerako orduan errore mezurik ez agertzeko **StoreableProjectFile** motako objectuak tratatuko ditugu.

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

	System.out.println("- - - - - - - - - - - - - - - - - - - -
-");
	//4. galderako errefaktorizazioaren probak
	Project project = new Project();

	ProjectFile pf3 = new
ReadOnlyProjectFile("kalifikazioa.pdf");
	ProjectFile pf4 = new StoreableProjectFile("azterketa.pdf");

	project.addProject(pf3);
```

```
        project.addProject(pf4);

        project.loadAllFiles();
        project.storeAllFiles();

}
```

DIP

1. Ez du DIP betetzen. *Student* klaseko **register** metodoa *Preconditions*, *Deduction* eta *SubjectQuotes* klaseen menpe dago. Metodoaren barruan sortzen diren instantziak begiratu behar dugu printzipioa betetzen ez duela jakiteko.
2. DIP printzipioa bete dezan **register** metodo barruan dauden instantziak (new Class()) ezabatu egin behar dira, eta horien ordez, instantziak parametro bezala eman behar zaizkio metodoari.

```
public void register(String subject, Preconditions p, Deduction d,
SubjectQuotes sq) {
        // Aurrebaldintzak konprobatzen dira
        boolean isPosible = p.isPosible(subject , subjectRecord);

        if (isPosible) {
                // Dedukzioa kalkulatu sex eta edadearen arabera
                int percentage = d.calcDeduction(sex, year);
                // Irakasgaiaren prezioa lortu
                int quote = sq.getPrice(subject);
                // HashMap batean gordetzen du eta ordaindu behar duen
balioa eguneratu
                subjectRecord.put(subject,null);
                toCharge = toCharge + (quote - percentage * quote /
100);
        }
}
```

```
public class Preconditions {

	public boolean isPosible(String subject, HashMap<String,
Integer> subjectRecord) {
		return true;
	}
}

public class Deduction {

	public int calcDeduction(String sex, String year) {
		return 1;
	}
}

public class SubjectQuotes {

	public int getPrice(String subject) {
		return 1;
	}
}
```

ISP

1. **EmailSender** klaseak email bat eta mezua behar ditu, eta horren ordez, pertsona bat eta mezua jasotzen ditu. **SMSSender** klaseak telefoono bat eta mezua behar ditu, eta horren ordez, pertsona bat eta mezua jasotzen ditu.

   ISP printzipioa bortxatzen da **EmailSender** eta **SMSSender** klaseek behar ez duten informazio (edo parametro) orokor bat jasotzen dutelako.

2. **EmailSender** eta **SMSSender** klaseek Person motako parametro baten ordez **IEmail** eta **ITelephone** interfazeetako parametro bat jasoko dute hurrenez hurren.

```
public class EmailSender {

	public EmailSender() {
```

```
	}

	public static void sendEmail(IEmail c, String message){
		// Mezu bat bidaltzen du Person klaseko korreo
helbidera.
		System.out.println("Emaila: " + c.getEmail() + ".
Mezua: "+ message);
	}

}


public class SMSSender {

	public SMSSender() {

	}

	public static void sendSMS(ITelephone c, String message){
		//SMS bat bidaltzen du Person klaseko telefono
zenbakira.
		System.out.println("Telefono zenbakia: " +
c.getTelephone() + ". Mezua: "+ message);
	}

}


public interface IEmail {

	public String getEmail();
	public void setEmail(String email);

}


public interface ITelephone {

	public String getTelephone ();
	public void setTelephone(String telephone);

}
```

Person klaseak bi interfazeak inplementatuko ditu.

```
public class Person implements IEmail, ITelephone{

	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; }

	@Override
	public void setEmail (String ea) { email = ea; }

	@Override
	public String getEmail () { return email; }

	@Override
	public void setTelephone(String t) { telephone = t; }

	@Override
	public String getTelephone() { return telephone; }
}
```

3. **GmailAccount** klaseak **IEmail** interfazeko metodoak inplementatuko ditu.

```
public class GmailAccount implements IEmail{

	String name, emailAddress;

	public GmailAccount() {

	}

	public GmailAccount(String name, String emailAddress) {
		this.name = name;
		this.emailAddress = emailAddress;
	}

	@Override
```

```
	public String getEmail() {
		return emailAddress;
	}

	@Override
	public void setEmail(String email) {
		emailAddress = email;

	}

}
```

EmailSender klaseko sendEmail metodoari, GmailAccount objektu batekin, deitzen dion programa honakoa da:

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

	GmailAccount ga = new GmailAccount("Gorka",
"gorka@gmail.com");
	EmailSender es = new EmailSender();

	es.sendEmail(ga, "Mezu hau Gmail kontu batetik bidali da.");
}
```