Software Ingeniaritza

2015-ko Maiatzak 22

1. DISEINUA (2 puntu)

Motorrak eraikitzen dituen *Motor-Race* enpresak bere bezeroek eskatutako motorren fabrikazioa kudeatzeko aplikazio bat eskatu digu. Horretarako honako informazioa eman digu:

a) **Eskaera Katalogo** bat, oraindik fabrikatu gabe dauden motorren informazioarekin.

b) **Eskaera** bakoitzak eskatzen den motor mota, eraikitzeko behar den denbora eta eraikitzeko beharrezkoak diren materialen zerrenda izango du, bakoitzetik beharrezkoak diren kantitateekin.

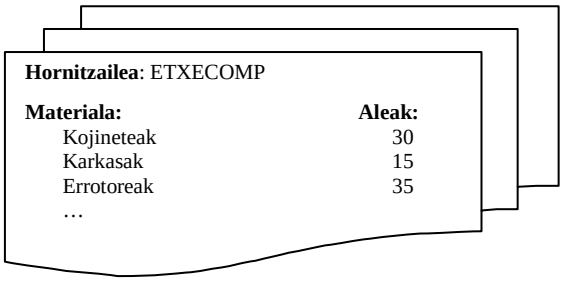
c) Eskaerak bete ahal izateko eskuragarri dauden materialak dituen **Biltegi** bat.

d) Biltegiko **Material** bakoitzak bere izakinak, beharrezko stock minimoa eta bere hornitzailearen identifikadorea izango ditu.

Eskatzen diguten aplikazioa astero egikarituko da hurrengo zazpi egunetan eraiki behar diren motorren eraikuntza planifikatzeko. Eskaera bat planifikatzen denean, astean gelditzen den denbora eta eraikitzeko beharrezkoak diren materialak eguneratu beharko dira, eskaera katalogotik ezabatuz. Biltegian beharrezko izakinen kantitate nahikorik ez balego, `MaterialGutxiegi` salbuespena emango da eta eskaera katalogoan mantenduko da.

1. Irudiak eskatutako aplikazioaren klase diagrama erakusten du. Planifikazio prozesua Eskaera Katalogoaren `planifikatu` metodoarekin hasten da.

Aplikazioaren lehenengo entregan, *Motor-Race*–k hasierako diseinua hedatu nahi duela esan digu, biltegian berritu behar diren materialen **Eskaera Orriak** automatikoki sortzen dituen funtzionalitatea gehituz. Orri horiek (ikusi ***2. Irudia***) hornitzaile bakoitzeko biltegian astero berritu behar diren materialak eta aleak dituzte. Eskatu beharreko kantitatea, material horren stock minimoa baina %20 izateko beharrezkoa dena izango da.

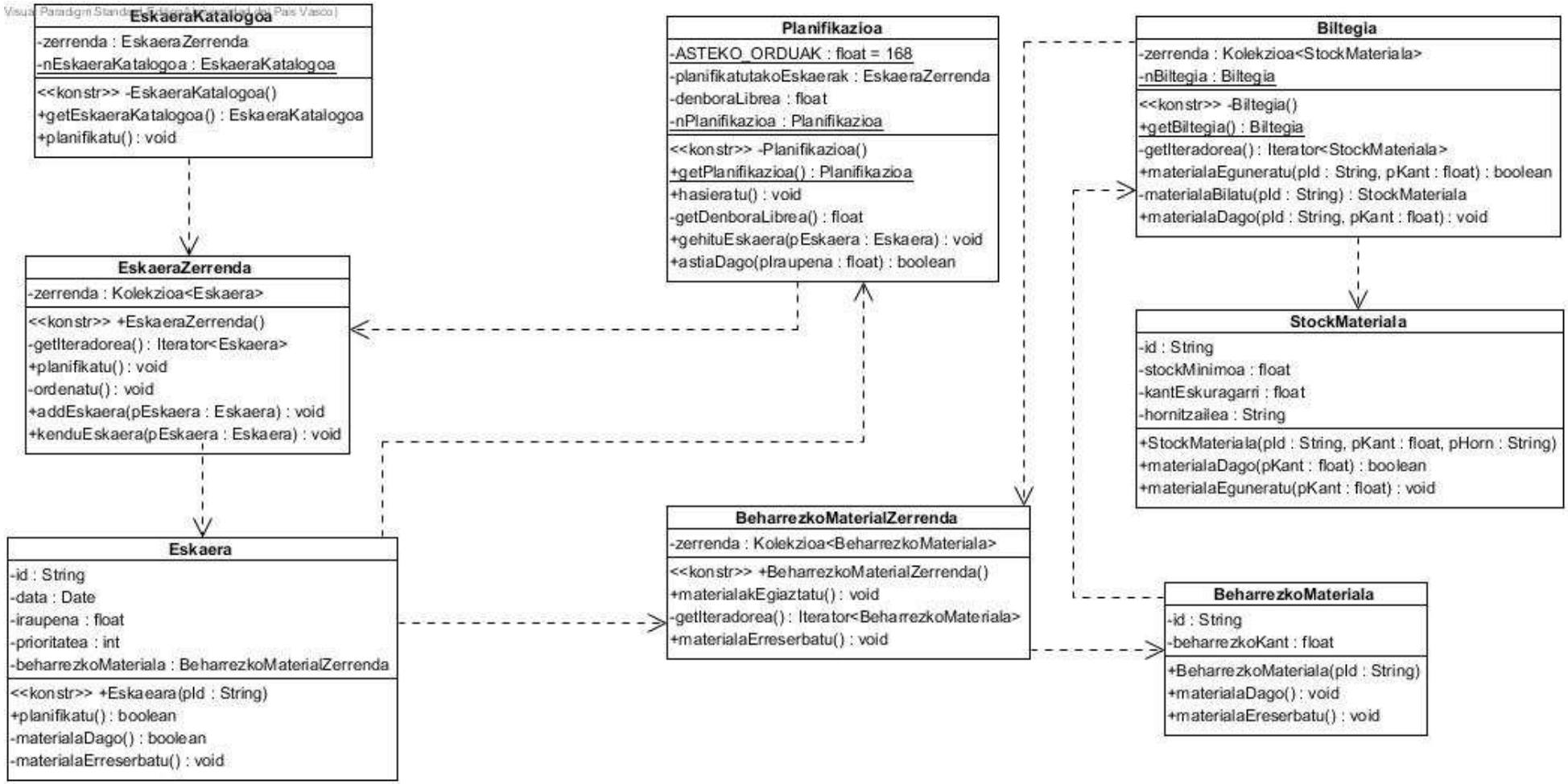


2. Irudia: *Eskaera orria*

Eskatzen da:

a) Klase diagrama hedatu edo aldatu funtzionalitate berria aurrera eramateko beharrezkoa den informazioarekin.

b) **`eskaeraOrriakSortu`** metodoaren sekuentzia diagrama egin.



1. Irudia: *Klase diagrama*

2. PATROIAK (2 puntu)

Urrutiko aginteak** (1 puntu)**:

Musika-kateak egiten dituen enpresa batean lanean gabiltza eta urrutiko aginteen softwarea garatzea eskatu digute. Aginteek ondorengo botoiak dituzte: *power, dvd, usb, irratia eta bolumena (+ eta - botoiak)*. Botoi hauetan *sakatu* egitean espero den portaera ondorengoa da:

- *power:* Katea itzalita badago piztu egiten du eta piztuta badago itzali egiten du.
- *dvd:* Katea piztuta badago *dvd* moduan jartzen du. Itzalita badago ez du ezer egiten.
- *usb:* Katea piztuta badago *usb* moduan jartzen du. Itzalita badago ez du ezer egiten.
- *irratia:* Katea piztuta badago *irratia* moduan jartzen du. Itzalita badago ez du ezer egiten.
- *bolumena (+ eta - botoiak)*: Katea piztuta badago aurreko edozein moduren bolumena kontrolatzen dute. Itzalita badago ez dute ezer egiten.

Eskatzen da:

a) ¿Zein patroi erabiliko zen(it)u(z)ke esandako softwarea garatzeko? Erantzuna arrazoitu.
b) UML klase diagrama zehaztu klase nagusiekin eta garrantzitsuak iruditzen zaizkizun metodoekin.
c) Aurreko klaseen inplementazioa idatzi.

Filtroak** (1 puntu)**:

Utilitateak izeneko klase bat dugu. Bere *filtratu()* metodoak emandako irizpide bat betetzen ez duten elementuak array batetik kentzen ditu. Metodo honek parametro bezala jasotzen duen array-aren elementuek **Filtrable** interfazea inplementatzen dutela suposatzen du. Filtrable interfazeko *irizpideaBetetzenDu()* metodoak objektu batek ematen diogun irizpide bat betetzen duen edo ez bueltatzen digu. Utilitate klasea erabili nahi dugu **Erregistro** motako array batetik elementu baliogabeak kentzeko. Erregistro klaseak *baliagarriaDa()* metodo bat du, objektuaren baliagarritasuna bueltatzen diguna, baina ezin dugu klase horren kodea aldatu. Nola erabili dezakegu Utilitate klasea Erregistro array baten elementuak filtratzeko?

Eskatzen da:

a) ¿Zein patroi erabiliko zen(it)u(z)ke esandako softwarea garatzeko? Erantzuna arrazoitu.
b) UML klase diagrama zehaztu klase nagusiekin eta garrantzitsuak iruditzen zaizkizun metodoekin.
c) Aurreko klaseen inplementazioa idatzi.

3. HERENTZIA (1 puntu)

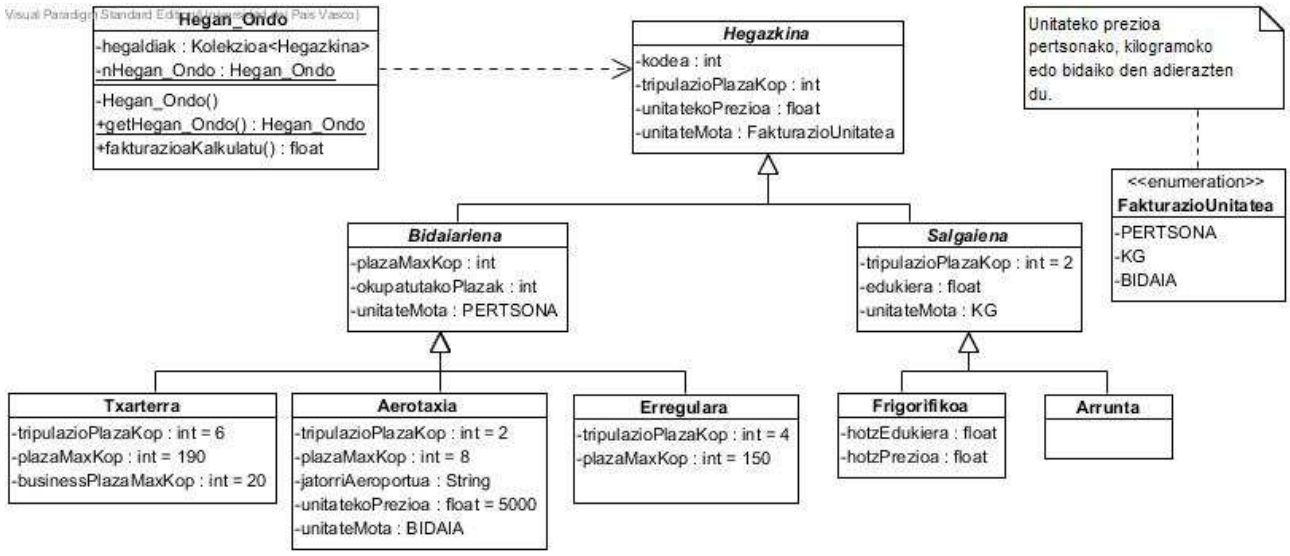
Bidaiari eta salgaien garraioan lan egiten duen *Hegan-Ondo* enpresak hegazkin flota bat du bere eskariak betetzeko. Hegazkin hauen ezaugarriak ***3. Irudiko*** hierarkian ikusi daitezke.

Hegan-Ondo akordio batera iritsi berri da posta eta telegrafo konpainiarekin aerotaxi eta salgai garraioko hegazkin arruntetan posta zakuak garraiatzeko. Akordio honek hegaldiko 300 eurotako prezio finko bat ezartzen du, posta konpainiak hegazkin hauetan libre dagoen espazioa aprobetxatu dezan.

Eskatzen da:

a) Hierarkia hedatu funtzionalitate berri hau kontutan hartzeko. Aldaketa honek ez du aurretik funtzionatzen ari ziren aplikazioetan eraginik izan behar.

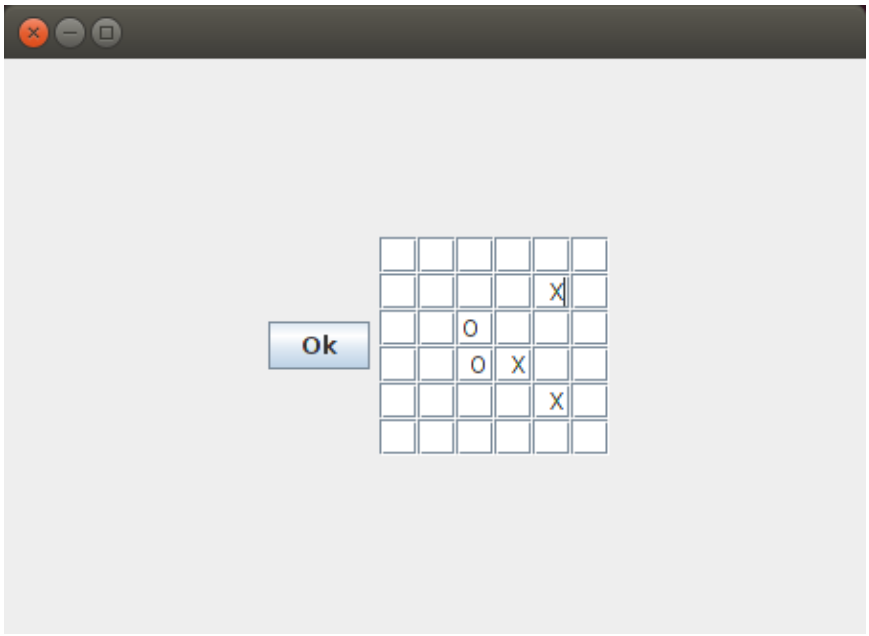
b) **Hegan_Ondo** klaseko *postaFakturaKalkulatu()* metodoa inplementatu. Metodo honek posta konpainiari fakturatuko litzaiokeen zenbatekoa kalkulatzen du. Inplementazioa errazteko beharrezko metodoak adierazi klase egokietan.



3. Irudia: *Klase diagrama*

4. INTERFAZE GRAFIKOAK (1 puntu)

6_lerroan jokoa garatu nahi da. Bere interfaze grafikoa ***4. Irudian*** ikusi daiteke.



4. Irudia: *Jokoaren leihoa*

Jokoaren garapen prozesuan hurrengo orrialdean agertzen den kodea idatzi da. Bertan **Bista** eta **Eredu** klaseak ikusi daitezke baina osatu gabe dago.

Eskatzen da:

Jokoaren inplementazioa bukatu, erabiltzaileak koadro batean balio bat sartzen duenean eredua egoki eguneratu dadin, uneoro bere informazioa leihoan agertzen denarekin koherentea izan dadin.

Koadro batean balio bat sartzean bistak ereduari dei bat egin behar dio:

```
public class Eredua { // SINGLETON BAT DA
        private char[] = new char[SIZE * SIZE];
        private static nEredua = new Eredua();

        private Eredua getEredua();
        public void aldatu(int pos, int balioa);
        // post: posizio jakin baten balioa aldatzen du
        // EZ DA INPLEMENTATU BEHAR!!!
}


public class Bista extends JFrame {
        private static final int SIZE = 6;
        private JPanel contentPane;
        // TODO Atributuak osatu
```

```
public Bista() {
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		setBounds(100, 100, 450, 300);
		contentPane = new JPanel();
		contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
		this.setContentPane(contentPane);
		contentPane.setLayout(new BorderLayout(0, 0));

		JPanel panel = new JPanel();
		contentPane.add(panel, BorderLayout.NORTH);

		JButton btnNewButton = new JButton("Ok");
		panel.add(btnNewButton);

		JPanel gridPanel = new JPanel(new GridLayout(SIZE, SIZE));
		for (int row = 0; row < SIZE; row++) {
			for (int col = 0; col < SIZE; col++) {
				JTextField newTextField = new JTextField("    ");
				gridPanel.add(newTextField);
	// TODO Tableroaren koadroak eraikitzen bukatu









			}
		}
		setLayout(new GridBagLayout());
		this.add(gridPanel);
	}

	private class Controller ... //Bukatu
	{	// TODO Kontrolatzailea inplementatu









	}
}
```