# Software Ingeniaritza

2014-ko Maiatzak 29

## 1. PATROIAK (1 puntu)

Ondorengo kodean:

a) Erabili diren patroiak identifikatu.

b) Zein da patroi bakoitzaren helburua?

c) UML klase diagrama irudikatu, patroiak non dauden azalduz.

```
public interface TagOperator {
	void invokeEnd();
}
```

```
public class XMLParser extends DefaultHandler {
	private static XMLParser mXMLParser = new XMLParser();
	private String texto = null;
	private Character letra = null;
	private Definicion definicionActual = null;
	private TagOperatorS operatorFab;

	private XMLParser() {
		operatorFab = new TagOperatorS();
	}
	public static XMLParser getPDF2XMLParser() {
		return mXMLParser;
	}
	public void parseXmlFile(String pFile)
			throws XmlParsingException {
		SAXParserFactory saxParserFactory = SAXParserFactory.newInstance();
		try {
			SAXParser saxParser = saxParserFactory.newSAXParser();
			saxParser.parse(new FileInputStream(pFile), this);
		} catch (Exception e) {
			throw new XmlParsingException(e);
		}
	}
	public void characters(char[] pCh, int pStart, int pLength)
			throws SAXException {
			texto = new String(pCh, pStart, pLength).trim();
	}
	public void endDocument() throws SAXException {
		System.out.println("Finished processing the file");
	}
	public void endElement(String pUri, String pLocalName, String pName)
			throws SAXException {
		TagOperator op = operatorFab.getTagOperator(pName);
		if (op == null) {
			return;
		}
		op.invokeEnd();
	}
	public void startDocument() throws SAXException {
		System.out.println("Init parsing");
	}
	private class TagOperatorS {
		private Map<String, TagOperator> operators = null;
```

```
		public TagOperatorS() {
			operators = new HashMap<String, TagOperator>();
			operators.put("Definicion", new DefinicionTagOperator());
			operators.put("Letra", new LetraTagOperator());
			operators.put("Enunciado", new EnunciadoTagOperator());
			operators.put("Respuesta", new RespuestaTagOperator());
		}
		public TagOperator getTagOperator(String pName) {
			return operators.get(pName);
		}
	}
	// Tag operators
	private class DefinicionTagOperator implements TagOperator {
		public void invokeEnd() {
			// Kodea
			}
		}
	}
	private class LetraTagOperator implements TagOperator {
		public void invokeEnd() {
			// Kodea
		}
	}
	private class EnunciadoTagOperator  implements TagOperator {
		public void invokeEnd() {
			// Kodea
		}
	}
	private class RespuestaTagOperator implements TagOperator {
		public void invokeEnd() {
			// Kodea
		}
	}
}
```

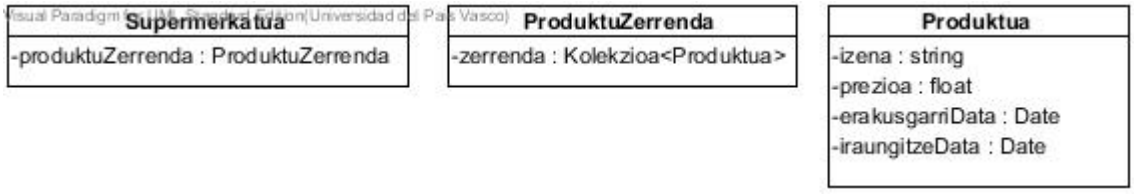
## 2. PATROIAK (1 puntu)

Sakeleko telefono baten bateria kontrolatuko duen aplikazio bat garatu nahi dugu. Aplikazio honek **Bateria** klasea izango du, bateria fisikoaren egoera kontrolatuko duena, bere karga maximoa eta momentuko karga dituelarik. Honetaz gain, aplikazioak **BateriaWidget** klase bat izango du, pantailan uneko karga erakutsiko duena, bateria %1 igo edo jaisten denean eguneratu behar delarik. Bukatzeko, **EnergiaKudeatzaileak** energia aurrezteko modua jarriko du martxan karga %30tik jaisten denean (pantailaren distira jaitsiko du, egunerapenak desgaituko ditu, etab. bateriaren gastua txikiagotzeko). Ondorengoa eskatzen da:

a) Zein patroi erabiliko zenituzke aplikazio hau inplementatzeko? Zergatik? Bere funtzionamendua azaldu zure aukera justifikatuz.
b) Aplikazio honen klase diagrama irudikatu, metodo garrantzitsuenak adieraziz.
d) Zure hitzekin eta laburki azaldu klaseen funtzionamendua eta noiz deitzen zaion metodo bakoitzari.

## 3. GENERIZITATEA (1 puntu)

**produktuakEguneratu()** metodoa inplementatzea eskatzen da. Metodo honek iraungita ez dauden produktuak itzuliko ditu, 14 egun baino gehiago ikusgarri egon diren produktuei %10 deskontua aplikatu eta iraungitze dataren arabera ordenatu ondoren.

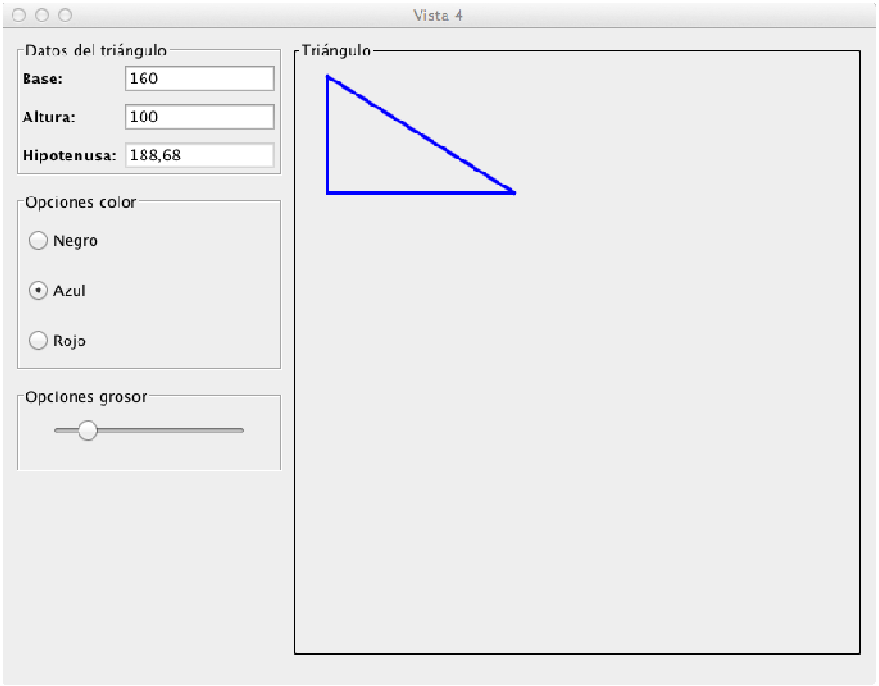



## 4. INTERFAZE GRAFIKOAK (1 puntu)

Ondoren agertzen den leihoa ikusita:

a) Leihoa inplementatzeko erabili diren osagai eta edukiontziak identifikatu, azken hauentzat zein *layout* erabiliko zenukeen azalduz.

b) Leihoak funtzionalitate egokia izan dezan inplementatu beharko liratekeen gertaera kudeatzaileak identifikatu.




| Interfaze Kudeatzailea | Adapterra | Metodo kudeatzailea |
|---|---|---|
| ActionListener | | void actionPerformed (ActionEvent e) |
| ChangeListener | | void stateChanged (ChangeEvent e) |
| ItemListener | | void itemStateChanged (ItemEvent evt) |
| ListSelectionListener | | void valueChanged (ListSelectionEvent evt) |
| MouseListener | MouseInputAdapter<br><br>MouseAdapter | void mouseClicked(MouseEvent evt)<br>void mouseEntered(MouseEvent evt)<br>void mousePressed(MouseEvent evt)<br>void mouseRelised(MouseEvent evt) |
| WindowListener | WindowAdapter | void windowClosed(WindowEvent evt)<br>void windowClosing(WindowEvent evt)<br>void windowActivated(WindowEvent evt)<br>void windowOpened(WindowEvent evt) |

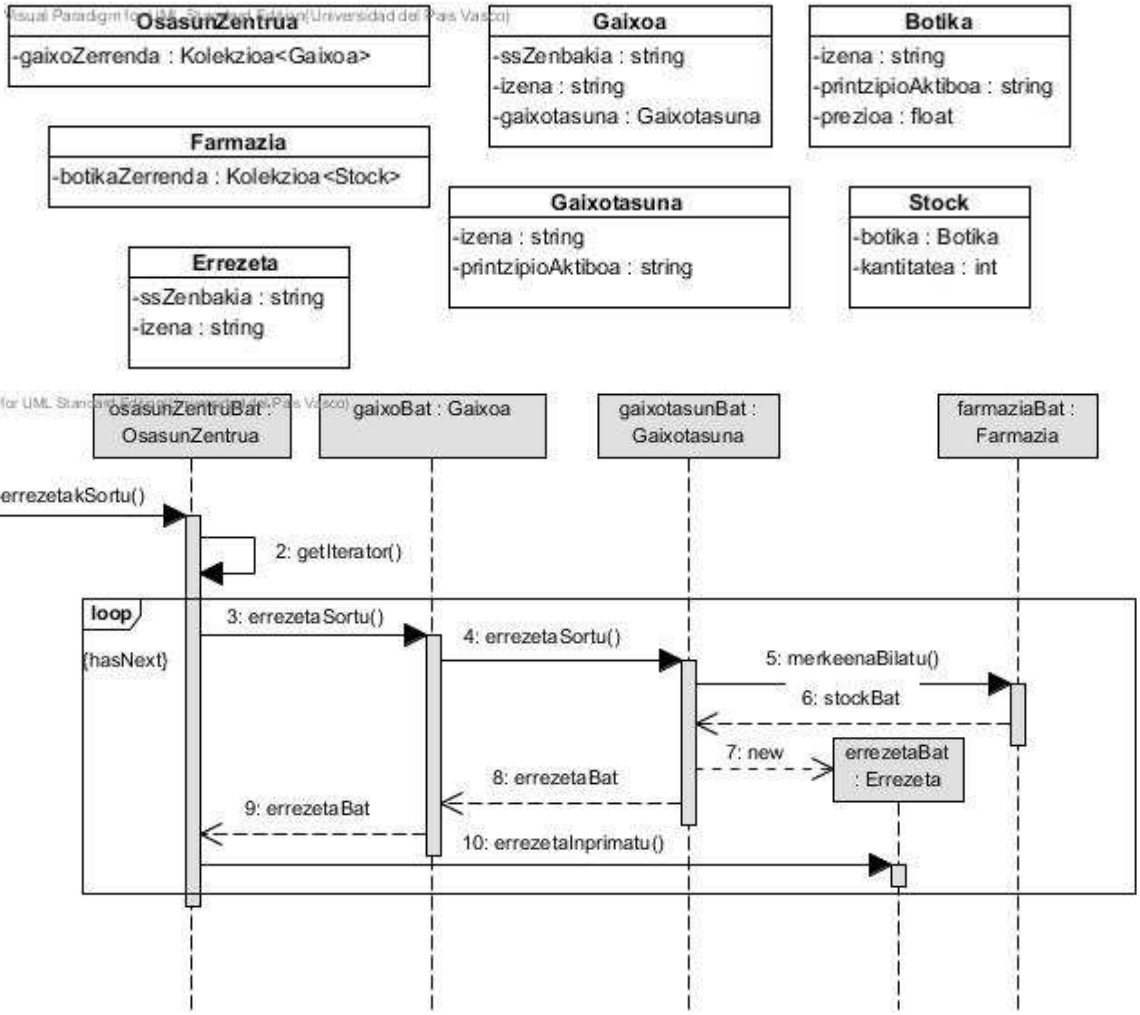
**OHARRA**: Triangeluaren lodiera *JSlider* osagaiak kontrolatzen du.

**5. DISEINUA (2 puntu)**

Bere gaixoei agindutako sendagaien koordainketa sistema kudeatzeko asmoz, **Osasun Zentru** batek gaixoen gaixotasunak tratatzeko botika merkeena aukeratzen duen aplikazio bat ezartzea erabaki du. Aplikazio honek egun batean zehar osasun zentruan egon diren gaixo guztien zerrenda bat kudeatzen du. **Gaixo** bakoitzari diagnostikatu zaion gaixotasuna gordetzen du, bere izena eta segurtasun zenbakiaz gain. Gaixotasun hori tratatzeko aplikazioak **Farmazian** printzipio aktibo egokia duen eta stock-ean dagoen botika merkeena bilatu beharko du. Ondoren, gaixo horrentzat botikaren errezeta sortuko du. Ondoren prozesu hau aurrera eramateko definitutako klaseak eta sekuentzi diagramaren zati bat aurkezten dira.

**Eskatzen da:**

a) Irudikatutako klaseen konstruktoreak eta prozesua egin ahal izateko metodoak zehaztu, bakoitzaren ikusgarritasuna eta sarrera eta irteera parametroak bere motekin adieraziz.
b) Sekuentzi diagrama osatu prozesua aurrera eramateko falta diren pausoekin (Nola aukeratzen da botika merkeena?).
**OHARRA**: Prozesua sinplifikatzeko, botika bakoitzak gaixotasun bakar bat tratatzeko balio duen printzipio aktibo bakarra duela suposatuko dugu.
d) Klaseen arteko dependentzia harremanak irudikatu.



**Sort** - Class in net.sf.jga.algorithms
Adapts standard java Sort capabilities to the interfaces common to jga.
**Sort()** - Constructor for class net.sf.jga.algorithms.Sort
**sort(T[])** - Static method in class net.sf.jga.algorithms.Sort
Returns an iterable object over the sorted contents of the array.
**sort(T[], Comparator<? super T>)** - Static method in class net.sf.jga.algorithms.Sort
Returns an iterable object over the sorted contents of the array, using the given comparator to determine ordering.
**sort(Iterable<T>)** - Static method in class net.sf.jga.algorithms.Sort
Returns an iterable object over the sorted contents of the input.
**sort(Iterable<T>, Comparator<? super T>)** - Static method in class net.sf.jga.algorithms.Sort
Returns an iterable object over the sorted contents of the input, using the given comparator to determine ordering.
**sort(Iterator<T>)** - Static method in class net.sf.jga.algorithms.Sort
Returns an iterator object over the sorted contents of the input.
**sort(Iterator<T>, Comparator<? super T>)** - Static method in class net.sf.jga.algorithms.Sort
Returns an iterator object over the sorted contents of the input, using the given comparator to determine ordering.
**sort(Iterable<? extends T>, TCollection)** - Static method in class net.sf.jga.algorithms.Sort
Appends the sorted contents of the input to the output.
**sort(Iterable<T>, Comparator<? super T>, TCollection)** - Static method in class net.sf.jga.algorithms.Sort
Appends the sorted contents of the input to the output, using the given comparator to determine ordering.

---

**Filter** - Class in net.sf.jga.algorithms
Algorithms that return a subset of its input.
**Filter()** - Constructor for class net.sf.jga.algorithms.Filter
**filter(T[], UnaryFunctor<T, Boolean>)** - Static method in class net.sf.jga.algorithms.Filter
Returns the elements of the array that meet the given selection criteria.
**filter(Iterable<? extends T>, UnaryFunctor<T, Boolean>)** - Static method in class net.sf.jga.algorithms.Filter
Returns the contents of the input that meet the given selection criteria.
**filter(Iterator<? extends T>, UnaryFunctor<T, Boolean>)** - Static method in class net.sf.jga.algorithms.Filter
Returns the contents of the iterator that meet the given selection criteria.
**filter(Iterable<? extends T>, UnaryFunctor<T, Boolean>, TCollection)** - Static method in class net.sf.jga.algorithms.Filter
Appends the contents of the input that meet the given selection criteria to the output.
**filter(LT, UnaryFunctor<T, Boolean>)** - Static method in class net.sf.jga.algorithms.ListAlgorithms
Removes all elements from the list that do not meet the given selection criteria.
**Filter.FilterIterable**<T> - Class in net.sf.jga.algorithms
Produces Iterators that only return elements that meet a given condition.
**Filter.FilterIterable(Iterable<? extends T>, UnaryFunctor<T, Boolean>)** - Constructor for class net.sf.jga.algorithms.Filter.FilterIterable
Builds a FilterIterator that will return only qualifying elements of the given iterable.
**Filter.FilterIterator**<T> - Class in net.sf.jga.algorithms
Iterator that only returns elements that meet the given selection criteria.
**Filter.FilterIterator(Iterator<? extends T>, UnaryFunctor<T, Boolean>)** - Constructor for class net.sf.jga.algorithms.Filter.FilterIterator
Builds a FilterIterator that will return only qualifying elements of the given iterator.

---

**Transform** - Class in net.sf.jga.algorithms
Algorithms that process the elements of a collection, iteration, or iterable resource, and present the results.
**Transform()** - Constructor for class net.sf.jga.algorithms.Transform

**transform(T[], UnaryFunctor<T, R>)** - Static method in class net.sf.jga.algorithms.Transform
Passes each element in the array to the functor, iterating over the results
**transform(Iterable<? extends T>, UnaryFunctor<T, R>)** - Static method in class net.sf.jga.algorithms.Transform
Passes each element in the iterable resource to the functor, iterating over the results
**transform(Iterator<? extends T>, UnaryFunctor<T, R>)** - Static method in class net.sf.jga.algorithms.Transform
Passes each element in the iterator to the functor, iterating over the results
**transform(Iterable<? extends T>, UnaryFunctor<T, R>, RCollection)** - Static method in class net.sf.jga.algorithms.Transform
Appends the transformed contents of the input to the output.