

Programazio Modularra eta Objektuei Bideratutako Orientazioa– 1. Laborategia

Aurrebaldintzak

Ikaslea *Eclipse* ingurunearekin lan egiteko trebezia baduela suposatzen da, jakinda aurreko lauhilabetekoan, Oinarrizko Programazioan erabili izan duela *Adako* programak garatzeko.

Hezkuntza helburuak

Laborategi honetan PMPBO erabiliko dugun Java-ikuspegia aurkeztuko da. *Eclipse*en Javaz programatzeko oinarrizko kontzeptuak ere landuko dira, hala nola proiektu, pakete eta klase sinpleen sorkuntza, geroago, hurrengo laborategietan horien konplexutasuna handituz.

Laborategi hau bukatzean, ikasleak hurrengo gaitasunak lortuko ditu:

- *Eclipse* ingurunearen funtzionamendua ulertuko du, eta Java ikuspegia erabiltzen trebea izango da.
- Lan lekua (*workspace*) sortzeko gai izango da, bertan bere proiektuak gordez.
- Proiektuak, paketeak eta klase sinpleak sortzeko gai izango da, non klaseak atributu eta metodo sinplez ornituta agertuko diren (eraikitzaile...).
- Java lengoaiaren syntaxian trebatuko da.

Motibazioa

Kurtso honetan zehar, kodea idazteko *Eclipseko* Java-ikuspegia erabiliko da. Beraz, bere funtzionamendua menperatzea komenigarria da.

Planteatzen diren atazen ebazpenean Objektuei Bideratutako paradigma erabiliko da. Beraz, hemen aurkeztuko diren oinarrizko kontzeptuak jakitea ezinbestekoa da.

Kontzeptua hauek, esan bezala, osatzen joango gara kurtsoa aurre joan ahala.

Beste aldetik, lan lekuen (*workspaces*) kudeaketa ondo menperatzea komenigarria da, hortxe lan eta gorde egingo direlako ikasle edo talde bakoitzak sortutako lana.




1. ataza: Eclipse ingurunea eta Java ikuspegia

Jarduera simple honetan, defektuzko lan lekua ez den beste lan leku desberdinean, *Eclipse* ingurunea zabaltzea eskatzen da, (adibidez, USB gordegailu bateko karpeta bateko lan lekua), eta hau egitea Java ikuspegiaren barruan.

Laguntza

Eclipse abiaraztean, aplikazioak egingo duen lehenengo gauza lan leku (*workspace*) bategatik galdetzea da. *Eclipse* abiarazten den lehengo biderra denean, lan lekuaren defektuzko bidea ...\\eclipse\\workspace izango da.

Hala ere, eta ikasle askok ordenagailu berdinak konpartitzen dutenez, **norberak bere lan leku propioa** sortzea garrantzitsua da, arazoak saihesteko. Komenigarria da **kanpo euskarri** bat erabiltzea (adibidez, *pendrive* bat) lan lekua gordetzeko, honela egindako guztia gero errez eraman dezakegu etxera. Bestela, *Eclipse* egindako proiektu, pakete eta klaseak esportatzea *eta inportatzea* ahalbideratzen du (File→Export→General→File System eta File→Import→General→File System, hurrenez hurren). Irakasgai honetan gomendatzen dizuegu laborategiko ordenagailuetan **ezer ez uztea**.

Eclipse abiaraztean ongietorri orri bat zabaltzen da, eta geroago orri hau atzitu nahiko bagenu menutik Help→Welcome egin beharko genuke. Behin ongietorri orria ixten dugula, ingurunean bertan sartuko gara. Esan dugunez, ikasgai honetan **Java ikuspegiarekin** lan egingo dugu. Ikuspegi hau kargatu nahiko bagenu, menuak erakusten duen Window→Open Perspective→Java, egingo genuke. Edo bestela zuzenean,  (goialdeko) botoarekin gero →Java aukeratuz. Geroago, 5. laborategian, oso baliagarria izango zaigun **Debug ikuspegia** ere erabiliko da.

Eclipse ikuspegi bat, menuak eta elementuak ikusteko modu edo konfigurazio bat baino ez da. Java, ikuspegiak defektuz erakusten dituen eremuak hurrengoak dira: ezker aldean, *Package Explorer* (hemen Java proiektuen baliabideak kokatzen dira) eta *Hierarchy* (hemen erabiltzen diren moten hierarkia ikus daiteke); eskuinaldean, *Outline* (honek momentuan lantzen ari den fitxategiaren informazioa agertzen da, eta informazio hau modu desberdinetan ager daiteke fitxategiaren motaren arabera); eta bukatzeko, bekaldean, *Problems* (eremu honetan konpilazio akatsak agertuko dira), *Javadoc* (eremu honetan, Javak duen aukeratuta dagoen elementuaren gaineko informazioa agertuko da) eta *Declaration* (eremu honetan aukeratutako elementuaren jatorria edo deklarazioa agertuko da). *Eclipse*en beste hainbat ikuspegi existitzen dira, eta ikus daitezke menú Window→Show View eginez. Adibidez, *Search* (bilaketa baten emaitzak erakusten dituena), *Tasks* (amaitu gabe gelditu diren atazen zerrenda erakusten duena, hori lortzeko programatzen ari garen programan (metodoan) "//TODO atazaren_izena" gehitzea nahikoa da) edo *Debug* (lehen aipatu dugun bezala kodearen egikariketa jarraitzeko eta erroreak topatzeko baliagarria).

Edozein softwarearekin bezala, askotan egiten diren ekintzak errazteko *Eclipse*en **laburbideak** (*shortcuts*) erabiltzea ahalbideratzen du. Guzti hauen artean garrantzitsuenak **Ctrl-Shift-L** da, honek beste laburbideen zerrenda pantailaratzen duelako. Eta laburbide guztien artean dudarik gabe erabiliena **Ctrl-Space** da.



2. ataza: proiektuak, paketeak eta klaseak

Behin garapen ingurunea prestatuta dugula, orain gure lehen Java proiektua sortzeko prest gaude. Ariketa honetan, Pertsona deritzon **klasea**, dagozkion **atributuekin** (kasu honetan izena eta adina) sortuko duzue. Baita klasea memorian kargatzeko metodo eraikitzailea.

Laguntza

Lehenengoz, iturri kodea gordetzeko proiektu bat sortu behar da, eta Java programa bat sortzeko behar diren fitxategiak. Horretarako, menú File→New→Project→Java→Java Project egingo duzue. Printzipioz ez dago murriztapen finkorik proiektuei edozein izena jartzeko. Kasu honetan **PMOO** izena jarriko diogu gure programari, (ez aldatu defektuz esleituta dauden aukerak) eta, *Eclipsek* galdetuko balu, erantzun proiektua Java ikuspegiari lotu nahi zaiola. Behin proiektua sortu dugula, laborategi honi dagozkion fitxategiak biltzeko **pakete** bat behar da. Sortu berri dugun proiektuan (Package Explorer eremuan ikusten dena), sakatu eskumako botoia src karpeta gainean (*source karpeta*) eta aukeratu New→Package.

Javak (Adak ez bezala) maiuskulak eta minuskulak ez ditu modu berdinean tratatzen, beraz OSO GARRANTZITSUA DA, elementu desberdinak izendatzeko irizpideak jarraitzea. Irizpide honek paketeen izenak beti minuskulaz idatzi behar direla adierazten du. Atributuak, et metodoak (metodo eraikitzailea izan ezik), parametroak eta aldagaiak ere minuskulaz izendatuko dira. Hitzarmenez **lowerCamelCase** estiloa deritzona (hau da, minuskulaz hasita, baina hitz bat baino gehiago balego hitz bakoitza Maiuskulaz hasiko da; adibidez, **zenbakiBat**). Parametroan kasuan, beti hasiko dira 'p' letra minuskulaz. Klaseen izenak aldiz maiuskulaz hasiko dira, **UpperCamelCase** deritzon estiloa jarraituz (hau da, maiuskulaz hasita, eta hitz bat baino gehiago balego hitz bakoitza ere Maiuskulaz hasiko da; adibidez, **PertsonaHeldua**). Interfazeen izenak 'I' letra maiuskulaz hasiko dira. Eta bukatzeko, konstanteen izenak maiuskulaz (letra guztiak) definituko dira. Hurrengo taulak irizpide guztiak biltzen ditu.

Elementua	Estiloa	Adibideak
Paketea	minuskulaz (puntu bananduta)	org.pmoo.packlaborategi1
Klasea	UpperCamelCase (izenak)	Pertsona, SalmentaZerrenda
Interfazea	UpperCamelCase (adjetiboak)	IOrdenagarria, ISerializagarria
Metodo eraikitzailea	UpperCamelCase (Klase-izena)	Pertsona, SalmentaZerrenda
Gainontzeko metodoak	lowerCamelCase (aditzak)	ordenatu, getAdina
Parametroa	lowerCamelCase (izenak)	pIzena, pIkasleenZerrenda
Atributua	lowerCamelCase (izenak)	luzeera koordX
Aldagaia	lowerCamelCase (izenak)	max, batazBestekoNota
Konstantea	MAIUSKULAZ(_ bananduta)	PI, MAX_ELEMK

Taula – Estilo eta irizpideak Javaz izenak jartzeko.


Hitzarmenez, packageak beti minuskulaz izendatzen dira. Irakasgai honetan irizpide hori jarraituko dugu, eta “pack” erroa ere gehituko diogu, beraz gure 1. laborategiarentzat izena honelako zerbait izan zitekeen **packlaborategi1**.

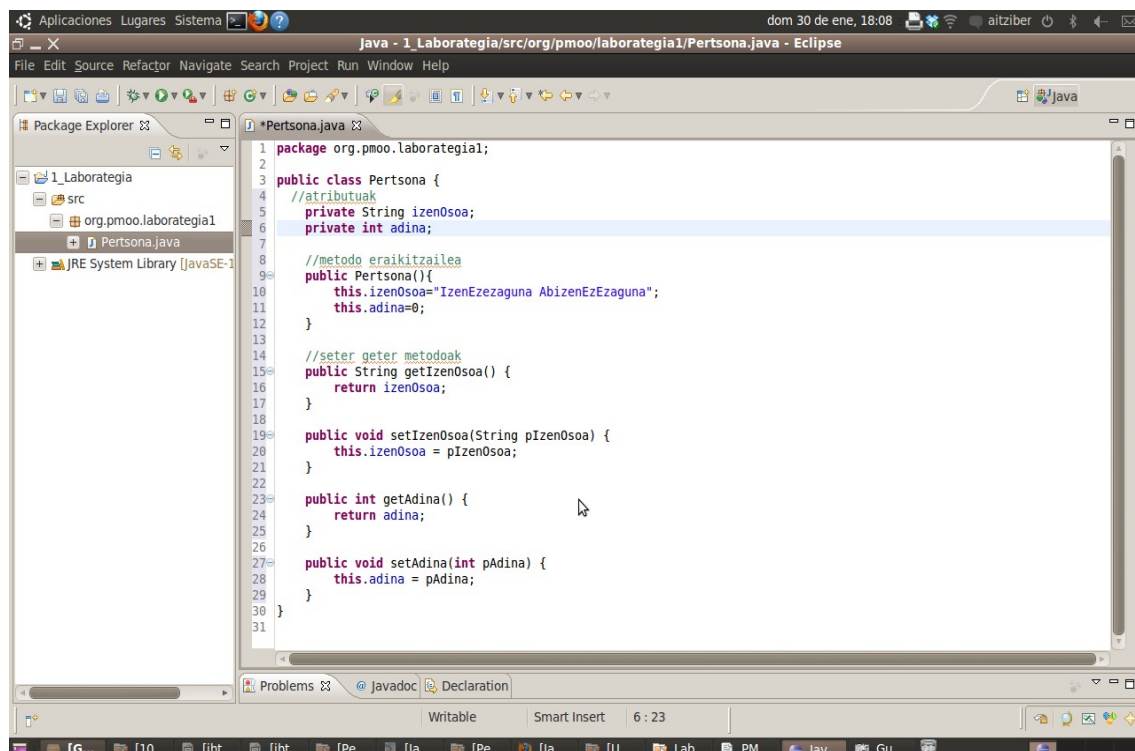
Behin iturri kodea gordetzeko package bat sortu dugula, gure lehenengo klasea sortzeko momentua da: **Pertsona** klasea. Horretarako, Package Explorer eremuan, packagearen izenaren gainean jarriko gara eta eskumako botoiarekin hautatu New→Class. *Eclipsek* klasearen izenaz galdetzean, adibidez **Pertsona** izena jarri genezake, gero *Finish* botoiari sakatuz.

Behin **Pertsona** klasea sortuta dagoela, bere atributuak definitzen hasiko gara: **izenOsoa** (String motatakoa), eta **adina** (int motatakoa **_osokoa_**).

Orain arte egindakoa ikusita, erregistro eta klase bat gauza bera direla pentsatu genezake, baina ez da horrela. Erregistro batek ez du bere baitan funtzionalitaterik, aldiz klase batek bai. Klase bat errealitateko elementu bat errepresentatzen du, elementu honek dituen atributuak **eta baita elementuak duen portaera**, adibidez, pertsona batek gidatu dezake ala ez, pertsona batek beste pertsona bati odola eman diezaioke ala ez, hauek portaerak dira ez bakarrik atributuak. Lasai geroago ulertuko da hobeto adibidearekin. Ariketa honetan metodorik sinpleenak implementatzea eskatzen da:

- **Pertsona** klasearen metodo eraikitzailea: ezinbestekoa da metodo berezi honek eraikitzen duen objektuaren klaseak duen izen bera izatea; metodo honek memoria alokatuko du klase horretako objektu batentzat, aldi berean objektu horren atributuak hasieratuz. Orokorrean, metodo eraikitzaileari atributuak hasieratzeko balioak parametro bezala pasatzen zaizkio, baina momentuz implementatuko da atributuei zuzenean balio jakin batzuk esleitzuz, Honela, **Pertsona** berri bat sortzen dugun bakoitzean balio jakin hauek izango ditu defektuz (kasu honetan “ezezaguna” eta 0 izango dira balioak hurrenez hurren).

Hurrengo irudian, behin ariketa bukatu duzuela edizio eremuko egoera zein izan beharko litzatekeen erakusten da. Kodea idazten dugun bitartean *Eclipsek* nolabaiteko konpilazio, edo errore sintaktikoa topatuko balu gorriaz azpimarratu eta  ikonoarekin markatuko luke. Erroreak *Problems* eremuan ere kontsultatu daitezke, pantailaren behealdean.



This. partikula kodeak gauden klaseko objektuari erreferentzia egiten dio. Orokorrean ez da beharrezkoa, baina anbiguotasuna dagoen guztietan beharrezkoa bihurtzen da, beraz ohitura bezala egungo objektua erabili nahi dugun guztietan erabiliko dugu badaezpadan. Lehenago esan den bezala, pParametroIzena nomenklatura erailiko da metodoen parametro formalak izendatzeko, honela egitea erabaki da deskuiduak eta nahasketak saihesteko.



3. ataza: Javako metodoen programazioa

Hemen bukatzen da tutorial hau, beraz ikasleak bere kabuz lan egitea espero da. Orain, **Pertsona** klasean funtzionalitate gehiago sartzea eskatzen da. Bereziki:

- 4 atributu berri: idPertsona (osokoa), sexua (karaktere, 'E' edo 'G'), nazionalitatea (String) eta odolMota (String).
- *idBerdinaDu()* metodoa inplementatu, boolear bat bueltatuz. Boolear honen balioa true izanik baldin eta metodo honi pasatzen diogun beste pertsona baten idPertsona eta gure pertsonaren (this) idPertsona berdinak balira, bestela false bueltatuko du.
- *gidatuDezake()* metodoa inplementatu, boolear bat bueltatuz. Metodo honek pertsonak gidatzeko behar den adina daukanean *true* bueltatuko du, bestela *false*. Beti suposatuz herrialde desberdinetan gidatzeko adinak desberdinak direla, adibidez Etioipian 14 urterekin izango da, Australia eta Estatu Batuetan 16 urterekin, 17 urterekin Erresuma Batuan, eta 18 urterekin munduko gainontzeko herrialdeetan.
- *odolaEmandiezaioke()* metodoa inplementatu, boolear bat bueltatuz. Metodo honek, pertsona bat parametro bezala hartuta eta gure pertsona izanda (this) beherago agertzen den taularen arabera erabakiko du ea gure pertsonak, beste pertsona horri eman diezaioke odola. Posible balitz *true* bueltatuko du, bestela *false*.
- *getIzenarenIniziala()* metodoa inplementatu gure pertsonaren izenaren iniziala bueltatzeko (oharra: erabili dezakezue String klaseak inplementatuta duen charAt() metodoa). Egin gauza bera abizenarekin *getAbizenarenIniziala()*.

Laguntza:

		Emailea							
		O-	O+	B-	A-	B+	A+	AB-	AB+
aileaHartz	AB+	X	X	X	X	X	X	X	X
	AB-	X		X	X			X	
	A+	X	X		X		X		
	A-	X			X				
	B+	X	X	X		X			
	B-	X		X					
	O+	X	X						
	O-	X							

[Wikipediatik lortutako taula]

Java lengoaiaren elementuak

Hitz bereziak (reserved words): **abstract, boolean, break, byte, case, catch, char, class, continue, default, do, double, else, extends, final, finally, float, for, if, implements, import, instanceof, int, interface, long, native, new, null, package, private, protected, public, return, short, static, super, switch, synchronized, this, throw, throws, transient, try, void, volatile, while.**

Javaz 3 motatako iruzkinak erabil daitezke.

Iruzkin mota	Javako espresioa	Adibideak
Lerro bat edo gehiagokoak	<code>/* */</code>	<code>/* Hau hainbat lerroko iruzkin bat da*/</code>
Lerro bakarrekoa	<code>//</code>	<code>x=0; //x-ren hasieraketa</code>
Javadoc komandoetakoa (automatikoki dokumentatzeko atzetik agertzen den elementua	<code>/**</code> <code>*</code> <code>*</code> <code>*/</code>	<code>/**</code> <code>*Aurrebaldintza: Erabiltzailearen *adina.</code> <code>Balioa 0 baino handiagoa *izan behar da.</code> <code>*/</code> <code>public int erabiltzailearenAdina;</code>

Taula – Javazko iruzkinak.

Hurrengo taulak, balio konstanteak edo literalak adierazteko momentuan erabili behar diren erregelak zerrendatzen ditu.

Literala	Javazko espresioa	Adibideak
Integer (int)	Zenbaki osokoa	-1234
Integer(long)	Zenbaki osoko luzea	1234567L
Real (float)	Zenbaki erreala	12.34f
Real (doble)	Doitasun bikoitzeko zenbaki erreala	12.34 .123
Character (char)	'Karakterea'	'a'
String	"Karaktere-katea"	"Kaixo mundua" "" (kate hutsa)
Boolean	true false	true
Tabulazioa	\t	\t
Lerro-jauzia	\n	\n
Komila bakuna	\'	\'
Komila bikoitza	\"	\"
Kontrabarra	\\	\\

Taula – Javazko espresio literalak.

Javaz, aldagaien deklarazioa **<atzipen mota> <aldagaiaren mota> <aldagaiaren izena>; eskema** jarraituz egiten da. Hurrengo taulak oinarritzko moten zerrenda erakusten du.

Mota	Javako espresioa	Adibideak
Atzipen mota	(Defektuzko atzipena paketetik) public (Atzipena edozein klasetik) private (bakarrik klase beratik) protected (bakarrik herentziaz sortutako ume klaseetatik)	long fakt; public long fakt; private long fakt; protected void setUp();
Integer	byte short int long <aldagaiaren izena>	int adina;
Real	float double <aldagaiaren izena>	Float batazbestekoNota;
Character	char <aldagaiaren izena>	char inicial;
String	String <aldagaiaren izena>	String Izenburua;
Boolean	boolean <aldagaiaren izena>	boolean topatua;
Deklarazio anizkoitza	<mota ac> <mota el> <izena1>, <izena2>,...;	private int i,j,k;

Taula – Javaz aldagaien deklarazioa.

Hurrengo taulak programazioko oinarritzko eragiketen sintaxia erakusten du.

Eragiketa	Operadore	Adibideak
Esleipena	=	x=0;
Eragiketa matematikoak	+ - * / %	i = i + 1; hondarra = num % 2;
Konparazioa	< <= > >= == !=	if(num>0) while(topatua!=false)
Eragiketa Logikoak	&& (eta) (edo) ! (ez)	if (x==0 && y==0) while(x<0 x>MAX) while (!(x>=0 && x<=MAX))
String-en konkatena (oinarrizko datu motak onartzen ditu)	+	String agurra = "Kaixo " + "mundua"; String str = "Balioa " + i + " da";

Taula – Javako oinarritzko eragiketak.

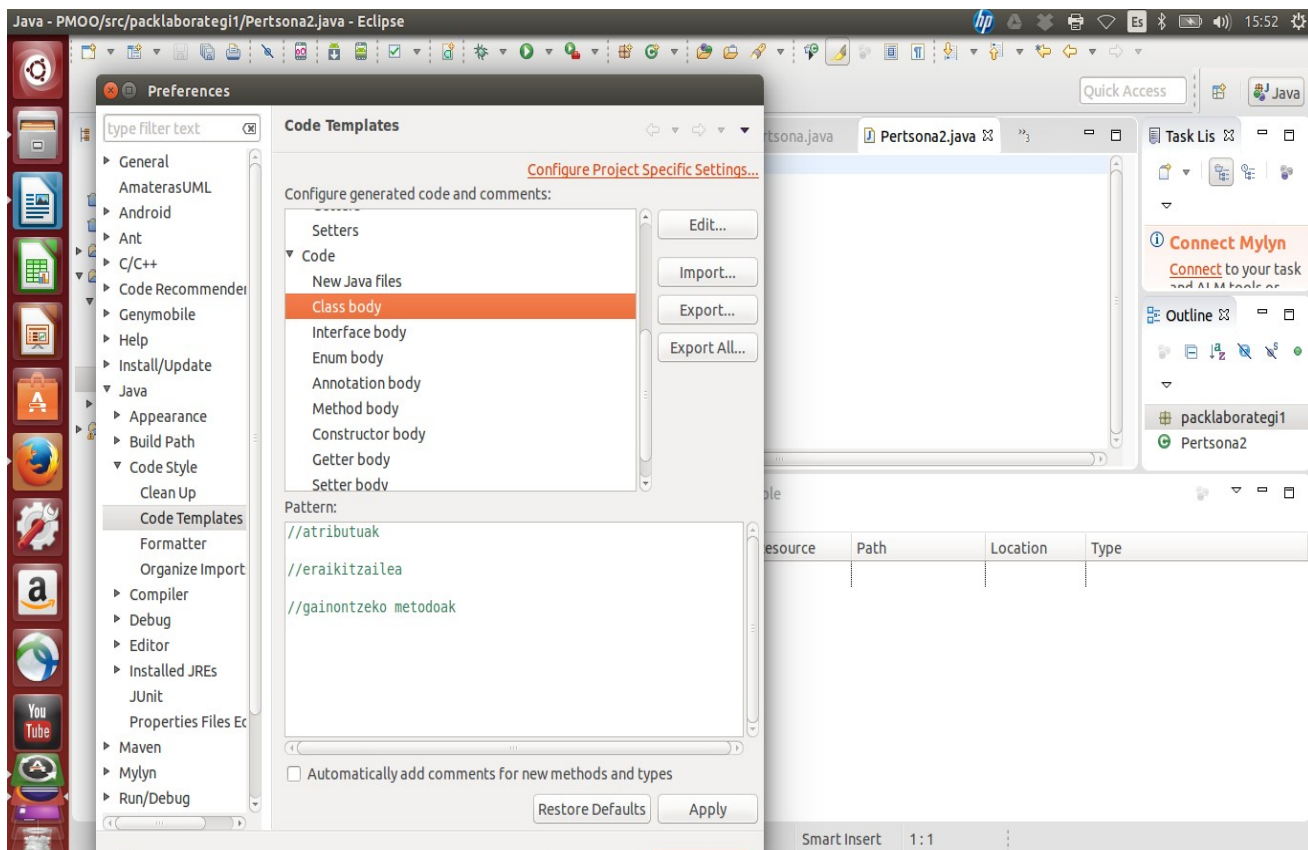
Bukatzeko, hurrengo taulan Javako oinarritzko aginduen sintaxia agertzen zaigu.

Agindu mota	Javako espresioa	Adibideak
Agindu multzoa	<agindu multzoa> ::= {<agindua>;}+	x=0; y=0;
Baldintza	<baldintza_if> ::= if (<baldintza>) { <agindua>; { < agindu multzoa> } } <baldintza_if-else> ::= if (<baldintza>) { <agindua>; { < agindu multzoa> } } else { <agindua>; { <agindu multzoa> } } <baldintza_switch> ::= switch (<osokoen espresioa_karakteen espresioa>) { { case <balio> : <agindu multzoa> [break ;;] }+ [default : <agindu multzoa>] } {	if(x%2==0) { bikotiBai=true; } else { bikoitiBai=false; } switch(x%2) { case 0: bikoitiBai=true; break; case 1: bikoitiBai=false; break; default: //error break; }

Iterazioa	<pre> <for>::= for (<hasieraketa>; <baldintza>; <eguneraketa>) { <agindua>; { <agindu multzoa> } } < while>::= while (<baldintza>) { <agindua>; { <agindu multzoa> } } <do-while>::= do { <agindua>; { <agindu multzoa> } } while (<baldintza>); </pre>	<pre> /**suposatzen da fakt 1ra hasieratuta dagoela**// for(i=1;i<=x;i=i+1) { fakt=fakt*i; } while (i<=x) { fakt=fakt*i; i=i+1; } do { fakt=fakt*i; i=i+1; } while (i<=x); </pre>
-----------	---	--

Taula – Javako aginduak.

Gehigarria, nola sortu classeentzat txantilo bat



KONTUZ!! Switch-ean || ez da erabiltzen!!

```
switch ( <osokoen expresioa_karakteeren expresioa> ){  
    {case <balio> || <balio> : <agindu multzoa>  EZ!!! BEGIRATU BEHERAGO  
        [break;] }+  
    [default : <agindu multzoa>]  
}
```

Horrela jarri behar da: **case <balio> : **case** <balio> :**

```
switch ( <osokoen expresioa_karakteeren expresioa> ){  
    {case <balio> : case <balio> : <agindu multzoa>  
        [break;] }+  
    [default : <agindu multzoa>]  
}
```