COLEGIO DE BACHILLERES

[Guía de estudio]

QUINTO
SEMESTRE

Programación en Java

PLAN 2014
ACTUALIZADO

https://huelladigital.cbachilleres.edu.mx

PLAN 2014
ACTUALIZADO

CRÉDITOS

Autor:
Gabriel David Durruti Bautista

Actualización:
Gabriel David Durruti Bautista

Coordinador:
Luis Gabriel Mondragón Torres
Coordinador de Proyectos de Formación Laboral en Informática

PRESENTACIÓN

Con la finalidad de acompañar el trabajo con el plan y programas de estudio vigentes, además de brindar un recurso didáctico que apoye al cuerpo docente y al estudiantado en el desarrollo de los aprendizajes esperados; el Colegio de Bachilleres desarrolló, a través de la Dirección de Planeación Académica y en colaboración con el personal docente de los veinte planteles, las guías de estudio correspondientes a las tres áreas de formación: básica, específica y laboral.

Las guías pretenden ser un apoyo para que las y los estudiantes trabajen de manera autónoma con los contenidos esenciales de las asignaturas y con las actividades que les ayudarán al logro de los aprendizajes; el rol del cuerpo docente como mediador y agente activo en el aprendizaje del estudiantado no pierde fuerza, por el contrario, se vuelve fundamental para el logro de las intenciones educativas de este material.

Las guías de estudio también son un insumo para que las y los docentes lo aprovechen como material de referencia, de apoyo para el desarrollo de sus sesiones; o bien como un recurso para la evaluación; de manera que, serán ellos, quienes a partir de su experiencia definirán el mejor uso posible y lo adaptarán a las necesidades de sus grupos.

El Colegio de Bachilleres reconoce el trabajo realizado por el personal participante en la elaboración y revisión de la presente guía y agradece su compromiso, entrega y dedicación, los cuales se reflejan en el servicio educativo pertinente y de calidad que se brinda a más de 90,000 estudiantes.

INTRODUCCIÓN

En la actualidad existe una gran variedad de dispositivos electrónicos, tales como lectores de Blu-ray y DVD, pantallas inteligentes de alta definición, teléfonos, computadoras, incluso refrigeradores de nueva generación, etc. que se van ajustando a las necesidades de los seres humanos; la situación va cambiado y se van acrecentando las necesidades a tal ritmo, que se necesitan personas que sepan cómo dirigir, controlar y automatizar el actuar de las maquinas, por lo que la oferta laboral es amplia.

Ahí es en donde entra el Auxiliar de Programador del Colegio de Bachilleres, el cual debe cumplir con las competencias profesionales requeridas, debido a esto, en esta asignatura se abordan los temas apegados a los requerimientos de las empresas.

En este material aprenderás a definir los conceptos básicos de la Programación en JAVA, estructurar un programa de acuerdo con todos sus elementos, construir clases y darle seguridad a tu aplicación, para brindar confianza a los clientes potenciales y/o contratantes.

Es importante precisar que para esta asignatura requieres utilizar una computadora o un teléfono inteligente o tableta (con sistema operativo Android 4.0 o superior, o iPhone versión iOS 7 o superior).

ÍNDICE

CORTE

1

Conocimiento de los fundamentos de programación en Java

Aprendizajes esperados:

Contenidos específicos	Aprendizajes esperados
1. Principios de programación en Java 1.1 Entorno de Java 1.2 Sintaxis y librerías propias del lenguaje Java 1.3 Programas sencillos.	1. Elabora programas básicos en consola a partir de la programación orientada a objetos.
2. Metodología de programación orientada a Objetos (POO) 2.1 Conceptos de la programación orientada a objetos: encapsulación, herencia, polimorfismo, propiedades, métodos y eventos.	2. Aplica la lógica computacional para desarrollar modelos y programas orientados a objetos.

PROPÓSITO

Utilizar los fundamentos de programación aplicados en un lenguaje de programación orientado a objetos, a fin de proponer una solución informática ante situaciones de su vida cotidiana.

CONOCIMIENTOS PREVIOS

Para que puedas lograr los aprendizajes esperados con el Corte 1, es necesario que refuerces los siguientes conocimientos:

Modelado de Sistemas y Principios de Programación

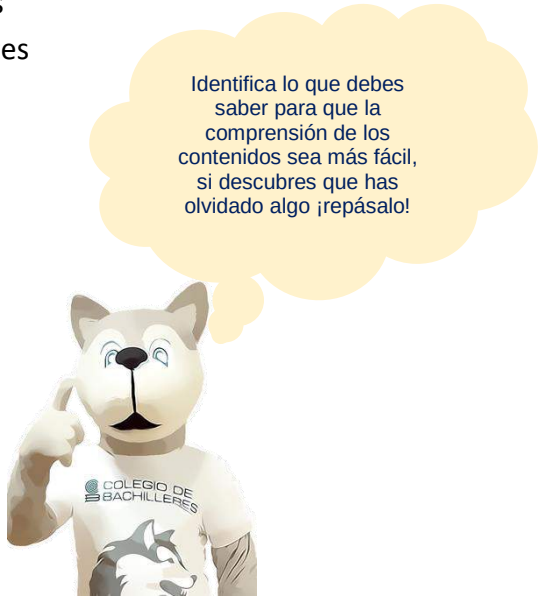
- ✓ Conocimiento en UML
- ✓ construcción de un diagrama de flujo
- ✓ Diagrama de Clases
- ✓ Documentación
- ✓ Librerías

Lenguaje y comunicación

- ✓ Sintaxis
- ✓ Verbos en el lenguaje inglés
- ✓ Sintaxis de la comunicación en idioma inglés

Matemáticas

- ✓ Variables
- ✓ Constantes



Instrucciones: Relaciona los siguientes conceptos (que se encuentran en la columna izquierda) con sus respectivas definiciones (que se encuentran en la columna derecha) según corresponda de forma correcta.

CONCEPTO	DEFINICIÓN
1. UML	___ Es aquella ubicación en la memoria qué va almacenando datos, También son conocidos como apuntadores.
2. VARIABLE	___ Son las reglas de cómo construir comandos e instrucciones a una computadora, Al mismo tiempo es una forma de comunicarse con la máquina.
3. TIPO DE VARIABLE	___ Es el tipo de dato que va a contener una variable, pudiendo ser textual, numérica calculable de forma aritmética, o pudiendo hacer de verdaderos y falsos.
4. LENGUAJE DE ALTO NIVEL	___ Es aquel lenguaje en el cual una persona escribe sus instrucciones (de preferencia en inglés), el cual será transformado en binario a través de un compilador.
5. CONSTANTE	___ es la representación gráfica de los contenidos de un sistema, el cual todo programador debe saber para poder interpretarlo y pasarlo a un código de programación.
6. SINTAXIS	___ Es todo aquel valor que no cambia, y que impide que el usuario lo modifique.
7. DIAGRAMA	___ Es aquel lenguaje universal, representado por diagramas y qué es utilizado en cualquier parte del mundo, Funciona para diseñar sistemas

1. Principios de programación en java

1.1 Entorno de java

El Entorno de Desarrollo Integrado (IDE) define las herramientas y vistas que se pueden utilizar al momento de programar en Java, los entornos más usados para programar en Java son NetBeans y Eclipse.

1.2 Sintaxis y librerías propias del lenguaje de java

Para hablar de sintaxis y librerías, primeramente, debemos definir los tipos de datos simples que se manejan en Java, los cuales se presentan a continuación:

DATOS SIMPLES EN JAVA

VARIABLES

Definición de variable

En matemáticas la variable es un símbolo que toma diferentes valores numéricos, dicho en forma simple, son valores cambiantes. Por ejemplo: X, Y, Z, Resultado.

En programación en JAVA, la variable es conocida como una localidad en memoria que contiene datos de diferentes tipos.

Forma de construir en código Variables

La estructura para crear una variable es:

[tipo_de_variable] [nombre_variable] = [valor] ;

Ejemplo:

double resultadoDeSuma = 0 d ;

Reglas de construcción

- No se debe crear 2 o más variables con el mismo nombre.
- Las variables solo permiten almacenar datos del mismo tipo (es decir los tipos numéricos solo aceptan numéricos, los tipos textuales solo reciben textuales y así sucesivamente).

Tipos de variables

Numérico

Almacena datos que son calculables de forma aritmética, permiten arrojar resultados (se pueden sumar, restar multiplicar, dividir, etc.).

Valores de verdad

Almacena datos exclusivamente para comparación lógica (solamente recibe “true” y “false”, o en nuestro idioma “verdadero” y “falso”).

Textual

Almacena datos que llevan “ ” o ‘ ‘ con caracteres textuales, sirven para lectura de las personas, ejemplo ‘a’ “Este es un ejemplo”.

Tipos de datos Primitivos o primitives

Son aquellos tipos de variable que solo podrán almacenar solamente un tipo de valor del mismo contexto, dicho de otra manera, si el programador indica que la variable recibe un valor matemático, dicha variable no almacenará de otro tipo de valor, para identificar a los tipos de datos primitivos, la inicial del tipado es con minúscula y cuentan con un color especial.

primitivos		Tipado	Ejemplo
Numérico	Entero	**byte**	1
		short	100
		int	1000
		long	2000000l
	Decimal	**float**	3.14f
		double	3.141595d
Valores de Verdad		**boolean**	True
Textual		**char**	'a'

Tipos de datos Engrapados o Wrappers

Son aquellos tipos de variable que solo podrán almacenar solamente un tipo de valor del mismo contexto, dicho de otra manera, si el programador indica que la variable recibe un valor matemático, dicha variable no almacenará de otro tipo de valor, para identificar a los tipos de datos engrapados, la inicial del tipado es con Mayúscula y no tienen color especial.

Engrapados		**Tipado**	Ejemplo
Numérico	Entero	**Byte**	1
		Short	100
		Integer	1000
		Long	20000l
	Decimal	**Float**	3.14f
		Double	3.141595d
Valores de Verdad		**Boolean**	true
Textual		**Character**	'a'

Ejemplo en NetBeans

```
Integer variableX = 1;
```

CONSTANTES

Definición de Constante

En matemáticas, la constante es un símbolo contiene un valor matemático que nunca va a cambiar. Algunos ejemplos: Pi π, Euler e, IVA, ISR.

En programación en JAVA, la constante, es un valor explicito, se simboliza con un prefijo **FINAL**, la cual restringe al programador para modificar el valor.

La estructura para crear una constante es:

[prefijo_final] [tipo_de_variable] [NOMBRE_CONSTANTE] = [valor] ;

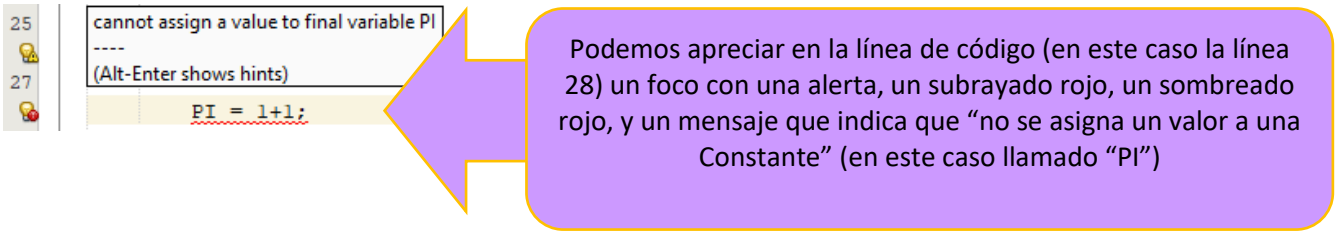
Ejemplo en NetBeans:

```
final double PI = 3.14159d;
```

Nótese que está el prefijo “final” que tiene un color distinguible, esta palabra reservada tiene como objetivo la restricción de modificar del valor, el nombre de la constante se caracteriza por estar en MAYUSCULAS.

Si se intenta modificar el valor de una constante, tendremos como consecuencia, un error de compilación.

Ejemplo de un error al intentar modificar el valor de una constante:



Actividad de aprendizaje 1

Crea un proyecto de Java, cuya clase “main”, contenga al menos una variable de cada una vista en esta sección y al menos una constante.

Parámetros

La clase “main” deberá contar con 16 variables (8 variables primitivas y 8 variables engrapados), cada una deberá contar con los diferentes tipos de valores que incluyan números enteros y decimales, textuales y valores de verdad según sea el caso.

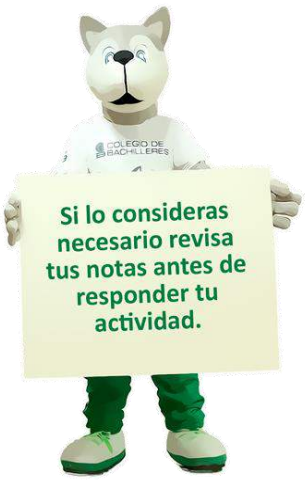
Ejemplo:

```
16      public static void main(String[] args) {
17          // TODO code application logic here
            byte varA;
            short varB;
            int varC;
            long varD;
            float varE;
            double varF;
            boolean varG;
            char varH;
26
            Byte varI;
            Short varJ;
            Integer varK;
            Long varL;
            Float varM;
            Double varO;
            Boolean varP;
            Character varQ;
35
            final double IVA = 0.16d;
            final int ISR = 15;
38      }
```

Puedes apoyarte en el siguiente video, ve realizándolo paso a paso.

YouTube: Programación en Java | Windows | S3-1 Variables y Constantes
http://bit.ly/CB-JAVA3-1

Guarda el proyecto con el nombre [NombreApellido_Act3-1_NombreActividad].



1.3 Programas sencillos

Para realizar programas sencillos, éstos deben de estar descritos en métodos, los cuales contienen toda la lógica de programación necesaria para su funcionamiento.

Métodos de conversión de datos

Definición de método

Es un bloque de instrucciones que puedes estar utilizando varias veces, esto depende de la necesidad del proyecto y de la secuencia del sistema.

En otras palabras, es:

- Una función reutilizable
- Evitas reescribir código

Tipos de método
Void (vacio)

No retorna ningún valor desde la función.

- Puedes hacer los procesos desde la clase
- Puedes traer los datos y procesarlos en el main

Return (retorno)

Si retorna valor desde la función, puedes ejecutar la instrucción desde la clase.

- No retorna ningún valor.
- Puede usar o no, los argumentos.
- *Se usa en impresiones, borrado de datos

Estructura de los Métodos

tipoDeFuncion nombreMetodo () { }

Ejemplo:

```
7
8   /**
9    *
10   * @author gddur
11   */
12  public class Usuarios {
13      int idUsuario;
14      String nombre;
15      String apellido;
16
17      void RegistrarUsuario(){
18          //código
19          //ejecución de valores
20      }
21
22      String LeerUsuario (){
23          //código
24          //ejecución de valores
25          return nombre; //regresa un valor
26      }
27
28  }
29
```

Ejemplo con método void

Ejemplo con método return

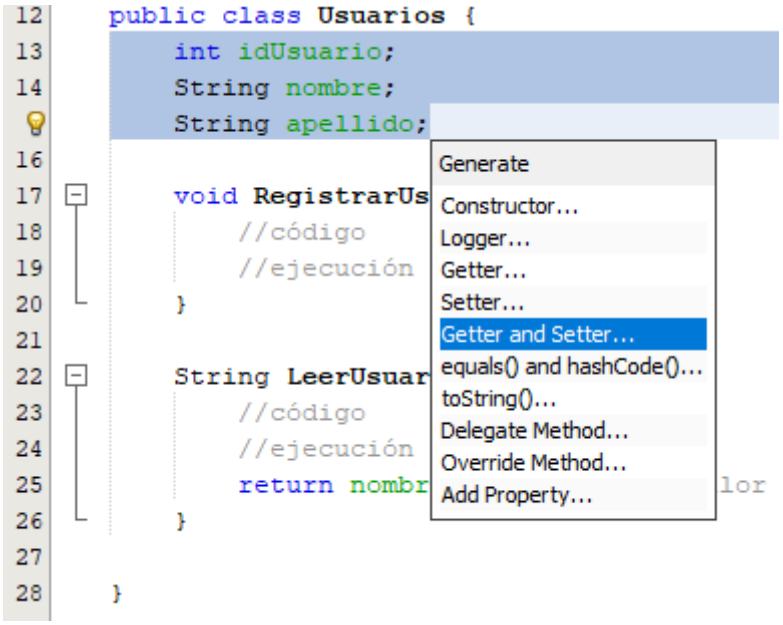
Como crear los métodos

Al igual que en la encapsulación, debemos generar el código de Setters y Getters, seleccionamos (dando clic y arrastrando el puntero) a cada una de las variables, damos clic con botón secundario de nuestro mouse, y seleccionamos “Insert Code…”

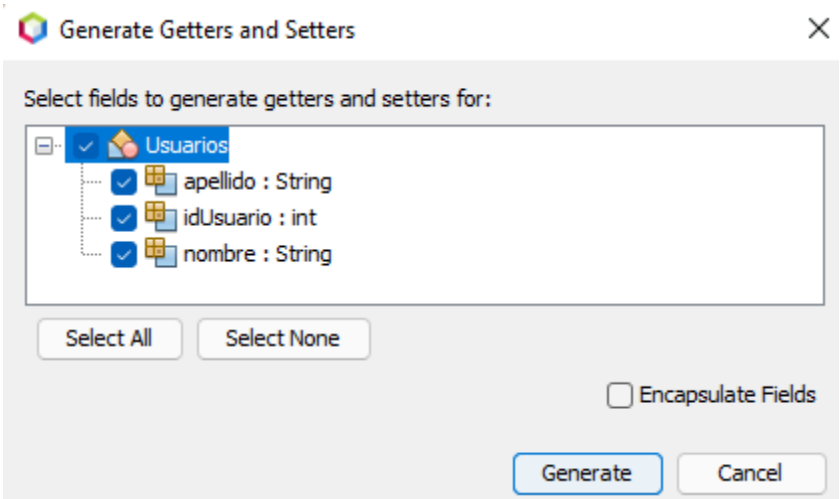
```
8   /**
9    *
10   * @author gddur
11   */
12  public class Usuarios {
13      int idUsuario;
14      String nombre;
15      String apellido;
16
17      void RegistrarUsuario(){
18          //código
19          //ejecución de valores
20      }
21
22      String LeerUsuario (){
23          //código
24          //ejecución de valores
25          return nombre; //regresa un v
26      }
27
28  }
```

Navigate
Show Javadoc Alt+F1
Find Usages Alt+F7
Call Hierarchy
Insert Code... Alt+Insert
Fix Imports Ctrl+Shift+I
Refactor
Format Alt+Shift+F
Run File Shift+F6
Debug File Ctrl+Shift+F5
Test File Ctrl+F6

Después Seleccionamos la opción “Getter y Setter”.



Después seleccionamos en la casilla de verificación, cada una de las variables, y damos clic en “Generate”.



Así quedará el código de la clase Usuarios.

Variables con Setters y Getters (en este caso son métodos void y return)	Métodos void y return

```java
public class Usuarios {
    int idUsuario;
    String nombre;
    String apellido;

    public int getIdUsuario() {
        return idUsuario;
    }

    public void setIdUsuario(int idUsuario) {
        this.idUsuario = idUsuario;
    }

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public String getApellido() {
        return apellido;
    }

    public void setApellido(String apellido) {
        this.apellido = apellido;
    }
```

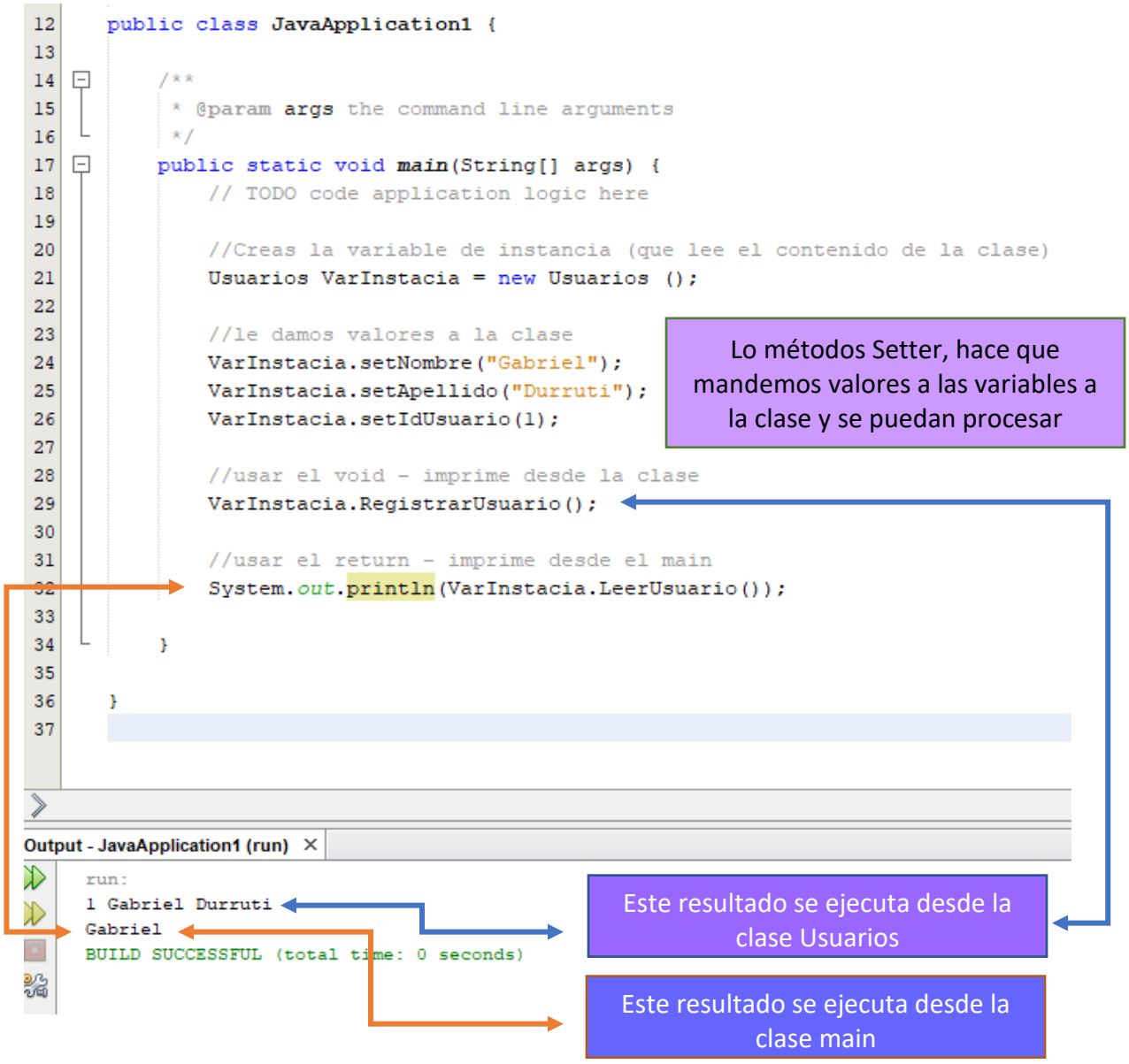
```java

    void RegistrarUsuario(){
        //código
        //ejecución de valores
    }

    String LeerUsuario (){
        //código
        //ejecución de valores
        return nombre; //regresa un valor
    }

}
```

Hacemos que imprima el nombre del usuario desde el método.

```java
    void RegistrarUsuario(){
        //código
        System.out.println(idUsuario + " " + nombre + " " + apellido );
        //ejecución de valores
    }
```

Usar el método Setter.

Desde la clase main, hacemos el siguiente código.



Actividad de aprendizaje 2

Crea un proyecto de Java, cuya clase “main”, contenga una ejecución de métodos desde una clase.

Parámetros

Las clases deben tener al menos una función void y una con return, por lo tanto, la variable de instancia deberá ejecutar cada una de ellas.

Ejemplo:

Puedes apoyarte en el siguiente video, ve realizándolo paso a paso.

YouTube: Programación en Java | Windows | S3-2 Métodos
http://bit.ly/CB-JAVA3-2

Guarda el proyecto con el nombre [NombreApellido_Act3-1_NombreActividad].

Nota. El docente o el asesor te indicará en qué lugar de almacenamiento en la nube subirás tus proyectos elaborados, de la misma forma guárdalos en tu computadora, para lo cual crea una carpeta llamada “programación en Java”.

Practicando con los códigos W3 School para programación en Java (requiere conexión a internet de forma constante). W3 Schools. (2020).
Java Tutorial de W3 Sitio web: https://www.w3schools.com/java

2. Metodología de Programación Orientada a Objetos (POO)

Introducción

Las empresas solicitan que los programadores de hoy en día sepan construir un código limpio y seguro, por lo que es necesario no perderse en el mar de código, para mejor comprensión y entendimiento de los procesos que una plataforma o un programa simple pueda efectuar.

2.1 Conceptos de la programación orientada a objetos (POO)

Encapsulación

Consiste en ocultar los atributos y estados de una clase y objeto, para beneficiar la seguridad de un sistema (evitar robo o espionaje).

Se necesitan varios componentes:

- ✓ Clase con atributos de acceso privado
- ✓ Operaciones de acceso publico
- ✓ Variable de instancia que invoque las Operaciones de acceso público (ubicada en una clase Main)

Ventajas de la programación orientada a objetos

- ✓ El código es limpio
- ✓ El código es más concreto
- ✓ Es más fácil de acceder
- ✓ Permite incrementar su construcción modulada
- ✓ Permite desarrollar y comprender mejor cada proceso
- ✓ Reutilizar código

Forma de construir encapsulado de una clase

Lo primero que tienes que hacer, es identificar un objeto para convertirlo en una clase.
Forma de generar un objeto.

Imagen 1. Ejemplifica cómo las clases tienen contenidos repetidos

Hay que representar textualmente A un objeto y hacernos las siguientes preguntas.

Elementos de un objeto

Entidad: ¿Quiénes? Generalizar en plural a un ente en un campo semántico.
Atributos: ¿Qué lo define? Componentes que definen a un ente.
Operaciones: Describir el "¿Qué hace?" y "¿Como lo hace?"

Ejemplo:

Entidad: ¿Quiénes? Gatos (se escribe en Plural).
Atributos: ¿Qué lo define? color, tamaño, peso, raza, genero (se escribe en singular).
Operaciones: "¿Qué hace?" y "¿Como lo hace?" Dormir, Comer, Jugar (se escribe en infinitivo, es decir con terminaciones en verbo).
Relevancia: Para resolver un problema, hay que obtener las características importantes.

Convertir un objeto en una clase

Una vez ya definido un objeto, la forma de pasarlo a UML y a su vez a un código de programación es de la siguiente forma.

Paso 1, se representa en diagrama de clases en UML de la siguiente manera:

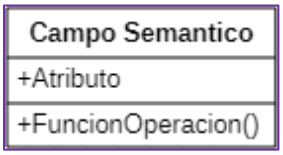


Imagen 2. Forma de construir un objeto en diagrama de clases.

Como en el ejemplo que se muestra a continuación:

Imagen 3. Ejemplo de construir el objeto "Gatos" en diagrama de clases.

Convertir un diagrama de clase (UML) en código de programación en JAVA

Para construir un diagrama de clase en UML en código de Java, generar una clase nueva de Java de acuerdo con el nombre de la clase, en este caso es “Gatos.java”.

```
public class Gatos {

}
```

Imagen 4. Escribir el código en Java.

El segundo paso que tenemos que hacer es generar los atributos con un acceso de tipo privado (private), después el tipo de variable y luego el nombre de nuestra variable que en este caso es el atributo.

```
public class Gatos {

    private String color;
    private String peso;
    private String raza;
    private String genero;

}
```

Imagen 5. se agrega los atributos del objeto en Java.

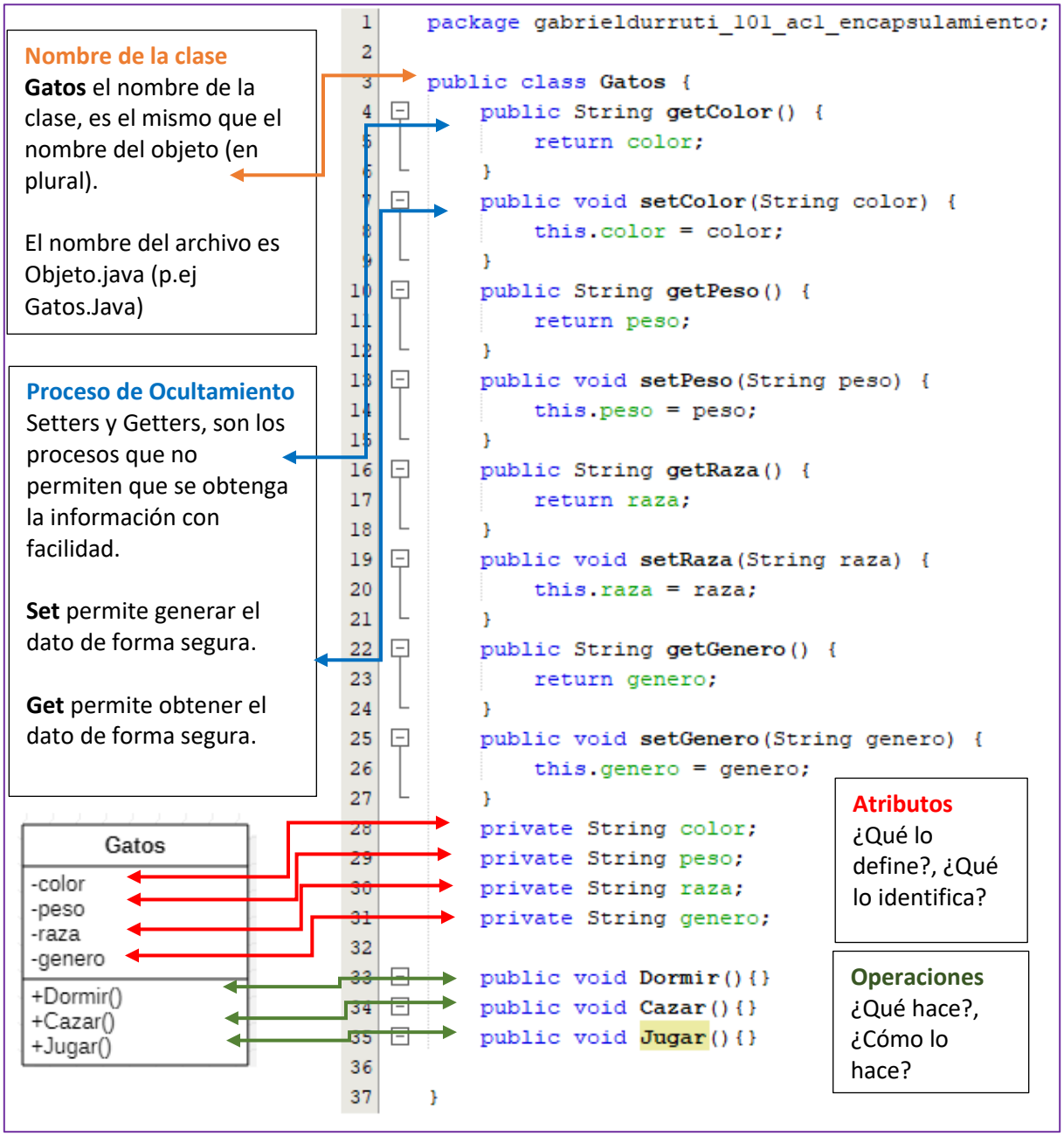
Después construimos las actividades u operaciones que realizará el objeto en Java.

```
public class Gatos {
    private String color;
    private String peso;
    private String raza;
    private String genero;

    public void Dormir(){}
    public void Cazar(){}
    public void Jugar(){}
}
```

Imagen 6. se agregan las operaciones de un objeto en Java.

Resultado final

Se agregan los procesos de ocultación y transmisión de datos, que son los “setters” y los “getters”.




```
1   package gabrieldurruti_101_acl_encapsulamiento;
2
3   public class Gatos {
4       public String getColor() {
5           return color;
6       }
7       public void setColor(String color) {
8           this.color = color;
9       }
10      public String getPeso() {
11          return peso;
12      }
13      public void setPeso(String peso) {
14          this.peso = peso;
15      }
16      public String getRaza() {
17          return raza;
18      }
19      public void setRaza(String raza) {
20          this.raza = raza;
21      }
22      public String getGenero() {
23          return genero;
24      }
25      public void setGenero(String genero) {
26          this.genero = genero;
27      }
28      private String color;
29      private String peso;
30      private String raza;
31      private String genero;
32
33      public void Dormir(){}
34      public void Cazar(){}
35      public void Jugar(){}
36
37  }
```

Imagen 7. Ejemplo de construir el diagrama de clase “Gatos” en código de java.

Clase de instancia

Para poder utilizar las clases a través de una variable lo que tenemos que hacer es generar una variable llamada instancia (la que es una propiedad que manda a referenciar una clase y sus contenidos públicos).

Forma de construir una variable de instancia

La forma de crear una variable de instancia es de la siguiente manera:

NombreDeLaClase nombreDeLaVariable = nueva **NombreDeLaClase** ();

Por ejemplo, para llamar a la clase Gatos.java se crea de la siguiente manera:

Gatos varInstancia = new **Gatos**();

En este caso en lugar de "varInstancia", tú lo puedes poner el nombre que quieras, y esto se debe de hacer desde el método main qué es el que controla y permite ejecutar nuestro programa.

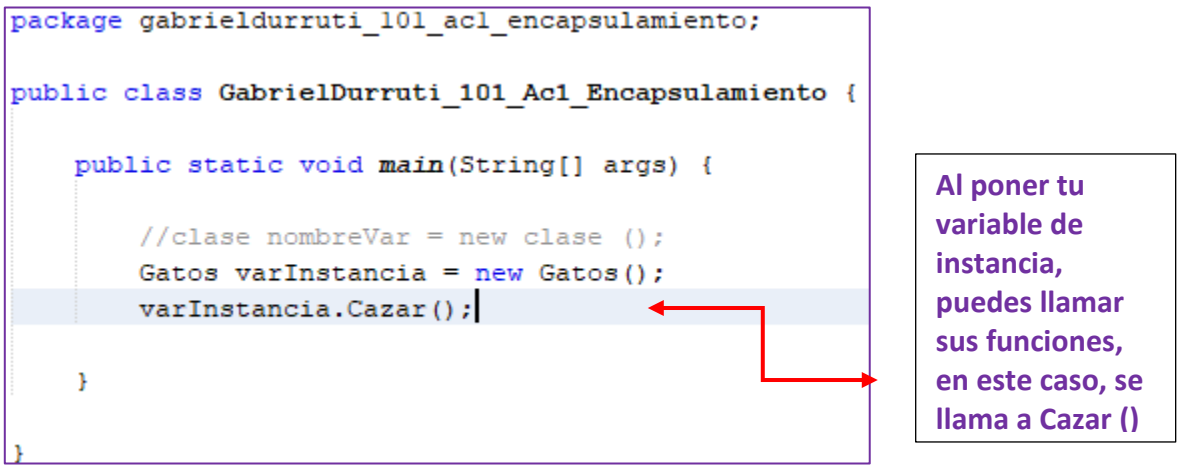


Imagen 8. Ejemplo de construcción de una variable de instancia dentro de una clase main y que invoca una operación.

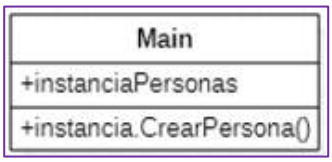
Actividad de aprendizaje 3

Realiza un proyecto de Java con una clase encapsulada (con los siguientes elementos), así como una clase main que contenga una instancia y que imprima una actividad, en lenguaje de programación en Java.

Parámetros:

La clase encapsulada deberá incluir las siguientes características:

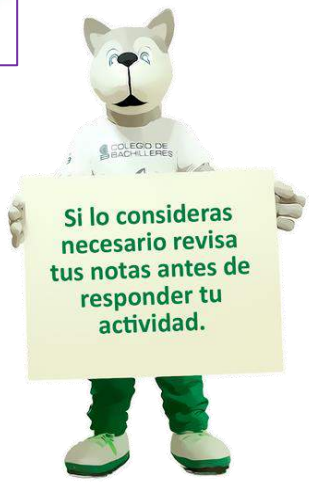
Y la clase Main deberá incluir



Puedes apoyarte en el siguiente video, ve realizándolo paso a paso:

YouTube: Programación en Java | Windows | S1 Encapsulación
http://bit.ly/CB-JAVA1

Guarda el proyecto con el nombre [NombreApellido_Act1_NombreActividad].



Herencia

En informática herencia es la capacidad de traer los datos de una clase a otra, Su principal ventaja es ahorrar código para no tener que estarlo repitiendo innecesariamente, y para tener mayor eficiencia en el consumo de procesamiento de una máquina.

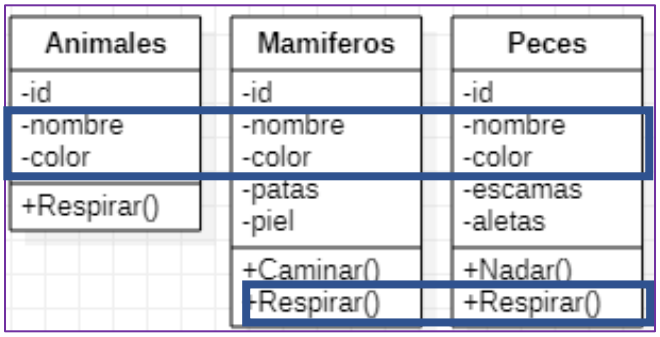


Imagen 9. Ejemplifica cómo las clases tienen contenidos repetidos.

Para evitar este tipo de repeticiones, y hacer un código más limpio y eficiente, debemos de crear una variable de instancia que invoque clases que son herederas o relacionadas entre sí.

Asimismo, tenemos que ocupar el uso de la palabra “extends”, en la clase para heredar las actividades y sus atributos.

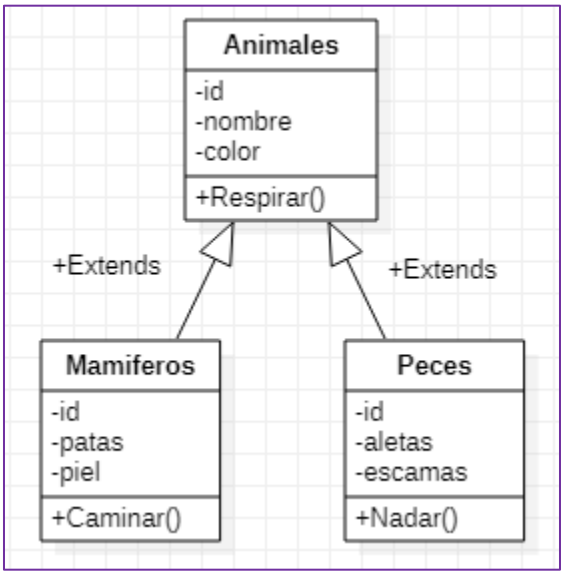


Imagen 10. Ejemplifica cómo las clases tienen contenidos diferentes
y aprovechan la clase “padre” o también llamada clase “base” o “super class”.

Forma de construir en código la herencia de clases

Vamos a tener que construir 3 diferentes tipos de clases, de las cuales una fungirá como la clase base o también llamada **clase padre**, y las clases que fungirán como las herederas también llamadas **hijas** o **sub clases**.

Elementos de la herencia en Java
Clase Padre o Super Class

La clase base (clase padre) sólo lleva su nombre.
Ejemplo: public class Animales { }

Clases hijo o Subclases

Las clases hijas (sub clases) Llevan el nombre del objeto seguido de la palabra “extends” seguido por el nombre de la clase padre.
Ejemplo: public class Peces **extends** Animales { }

El ejemplo (aplicando lo realizado en el tema “Encapsulamiento”) queda de la siguiente manera:

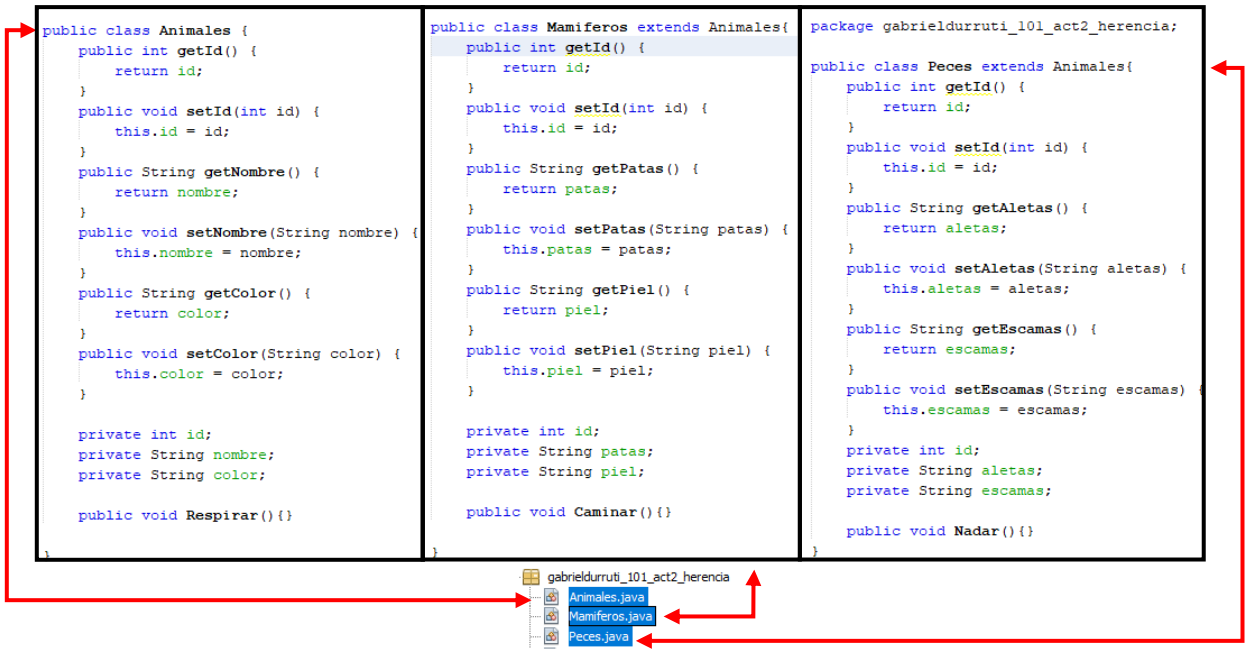


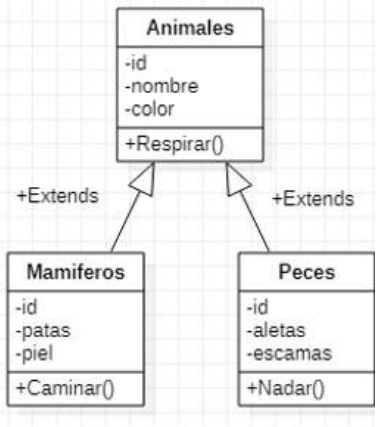
Imagen 11. Ejemplo de cómo queda la codificación de los elementos de la herencia.

Actividad de aprendizaje 4

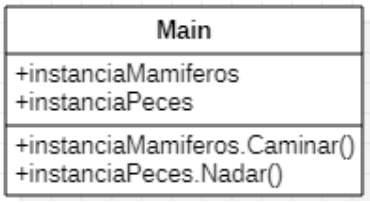
Realiza un proyecto de Java con una clase padre llamada “animales” y dos subclases llamadas “Mamíferos” y “Peces”. Cada una llevará 3 atributos y una operación, Así como una clase main que invoque las operaciones mediante la utilización de una variable de instancia.

Parámetros

Las clases deberán incluir las siguientes características:

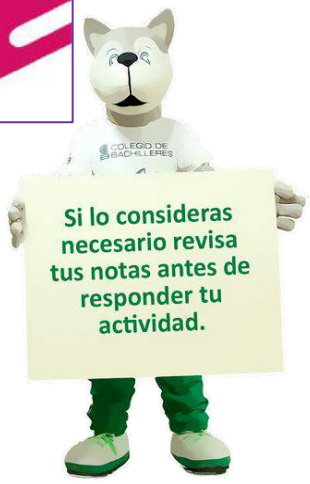


Y la clase Main deberá incluir



Puedes apoyarte en el siguiente video, ve realizándolo paso a paso:

YouTube: Programación en Java | Windows | S2 Herencia
http://bit.ly/CB-JAVA2



Guarda el proyecto con el nombre [NombreApellido_Act2_NombreActividad].

Polimorfismo

Definición de polimorfismo
Es el proceso de transformar un objeto de tal forma tenga múltiples formas.

Clasificación del polimorfismo
El polimorfismo se divide en varios tipos, pero solamente vamos a ocupar las siguientes dos divisiones.

Subtipo o Inclusivo
Es aquel objeto abstracto qué puede referenciar mediante una variable de instancia a una o varias cadenas de clases relacionadas.

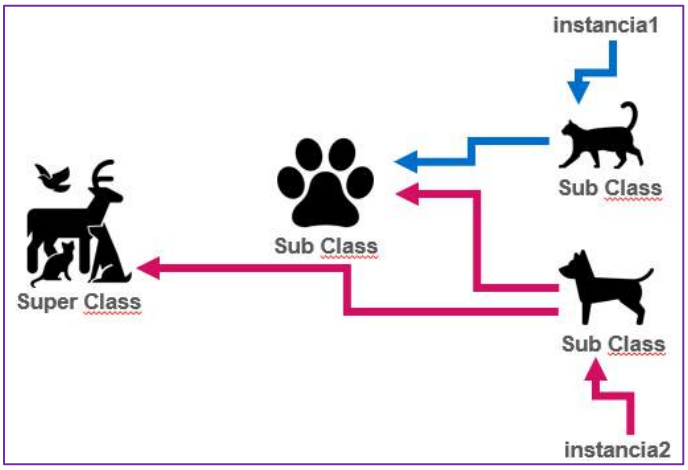


Imagen 12. Simbolismo de cómo se hace un subtipo por clases.

Ad Hoc

Es todo aquel objeto qué tiene múltiples funciones con diferentes argumentos, también se le conoce con el nombre de sobrecarga de funciones con el mismo nombre.

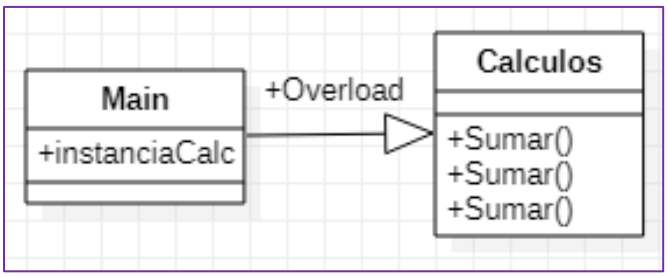


Imagen 13. Ejemplo de cómo queda la codificación de los elementos de la herencia.

Otros tipos

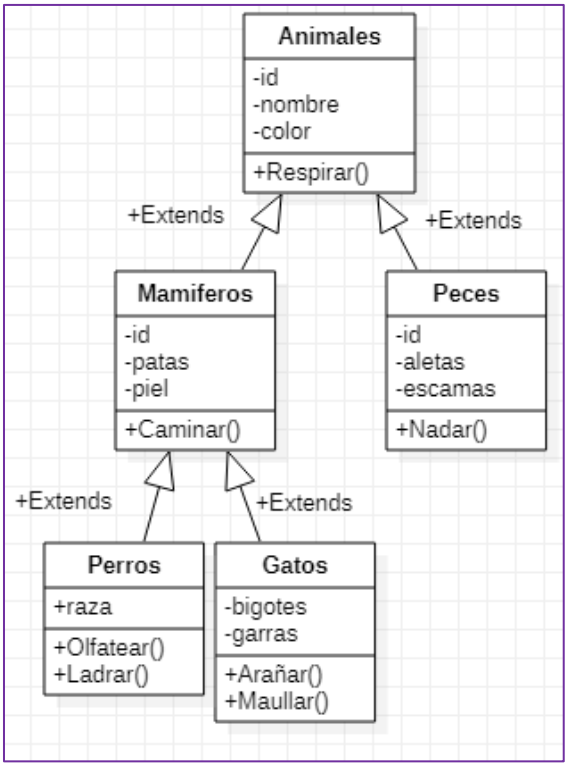
Existen otros diferentes tipos de polimorfismo cómo paramétrico, Fila, estática, Dinámica, Duck Typing, Polipismo (Polytypism o Programación genérica).

Actividad de aprendizaje 5

Modifica la actividad número 2, en este caso vas a incorporar dos subclases, llamadas "perros" y "gatos", y en la variable de instancia en el main, crearás dos variables de instancia adicionales las cuales deberán convertirse en otras clases.

Parámetros

Las clases adicionales deberán contener los siguientes atributos.



Y la clase Main deberá Incorporar dos nuevas variables de instancia, las cuales se conviertan en otras clases.

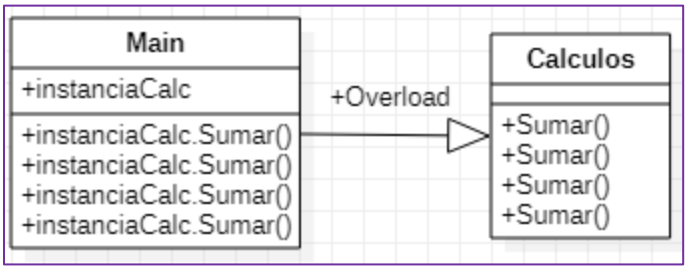
```
Animales varInstancia3 = new Perros(); //Polimorfismo Inclusivo (Instancia padre en hijo)
Mamiferos varInstancia4 = new Gatos (); //Polimorfismo Inclusivo
```

Actividad de aprendizaje 6

Crea un proyecto de Java, el cual contenga una clase que se llame “CalculosAritmeticos” y genera una sobrecarga de 4 diferentes tipos de argumentos, y una clase “Main”, la cual permite ejecutar los 4 diferentes tipos de función de sobrecarga.

Parámetros

La clase cálculos aritméticos deberá contar con los diferentes tipos de argumentos que incluyan números enteros y decimales.

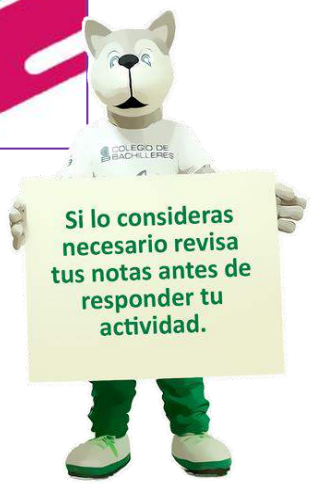


Y la clase Main deberá contar con una variable de instancia, la cual permita ejecutar la acción de sumar con los diferentes tipos de argumentos que incluyan números enteros y decimales.

Puedes apoyarte en el siguiente video, ve realizándolo paso a paso.

YouTube: Programación en Java | Windows | S3 Polimorfismo
http://bit.ly/CB-JAVA3

Guarda el proyecto con el nombre [NombreApellido_Act3_NombreActividad].



Instrucciones: ¿Cuál consideras que es tu nivel de dominio en el trabajo como programador en JAVA, en este primer corte?

Marca con una "X" la columna que corresponde.

	EXCELENTE	MUY BIEN	BUENO	DEBE MEJORAR
Crear un proyecto en Java				
Crear una clase en Java				
Crear un objeto con todo y sus atributos				
Crear el encapsulamiento de un objeto				
Crear el código del polimorfismo de una instancia				
Crear el código de la sobrecarga de una función				
Definir una clase				
Definir un atributo				
Definir la herencia de un objeto				
Crear el código de la herencia de un objeto				

FUENTES CONSULTADAS

Sitios web

Picodotdev. (2020). Los modificadores de acceso de clases, propiedades y métodos en Java. Recuperado el 18 de mayo de 2022 en https://picodotdev.github.io/blog-bitix/2020/01/los-modificadores-de-acceso-de-clases-propiedades-y-metodos-en-java/

Víctor. (2020). Operadores de Asignación y Aritméticos Java. Recuperado el 18 de mayo de 2022 en https://www.manualweb.net/java/operadores-asignacion-aritmeticos-java/

Natalia, Contreras R. (2009). Escala de estimacion (autoevaluacion). Recuperado el 18 de mayo de 2022 en http://natilabombera.blogspot.com/2009/07/escala-de-estimacion-autoevaluacion.html

W3 Schools. (2020). Java Tutorial. Recuperado el 18 de mayo de 2022 en https://www.w3schools.com/java/

Documento digital

Universidad informática. (2020). Guía de sintaxis del lenguaje Java 2. Recuperado el 18 de mayo de 2022 en https://www.mhe.es/universidad/informatica/8448132904/archivos/general_apendice3.pdf

Tabla de imágenes del corte 1

Figura	Origen	Referencia
1	Propia del autor	https://youtube.com/c/GabrielDurruti
2	STAR UML	http://staruml.io/
3	STAR UML	http://staruml.io/
4	Extracción de APACHE IDE NETBEANS 12.0	https://netbeans.org/
5	Extracción de APACHE IDE NETBEANS 12.0	https://netbeans.org/
6	Extracción de APACHE IDE NETBEANS 12.0	https://netbeans.org/
7	Extracción de APACHE IDE NETBEANS 12.0	https://netbeans.org/
8	Extracción de APACHE IDE NETBEANS 12.0	https://netbeans.org/
9	STAR UML	http://staruml.io/
10	STAR UML	http://staruml.io/
11	Extracción de APACHE IDE NETBEANS 12.0	https://netbeans.org/

CORTE

2

Elaboración de programas en Java

Aprendizajes esperados:

Contenidos específicos	Aprendizajes esperados
1. Estructura de control secuencial en Java 1.1 Normas de estructura y sintaxis. **2. Estructura de control condicional en Java** 2.1 Simple, doble y múltiple 2.2 Sintaxis **3. Estructuras de control repetitivas en Java** 3.1 Sintaxis de las estructuras: For, while, do while	1. Elabora programas en Java con variables, constantes y estructuras de datos, estructuras de control secuenciales, condicionales y repetitivas en los métodos de una clase.

Al finalizar el corte el estudiante será capaz de programar soluciones informáticas.

CONOCIMIENTOS PREVIOS

Para que puedas lograr los aprendizajes esperados con el Corte de evaluación 2, es necesario que refuerces los siguientes conocimientos:

Modelado de Sistemas y Principios de Programación

- ✓ Construcción de un diagrama de flujo
- ✓ Diagrama de flujo según la ANSI y OSI
- ✓ Librerías
- ✓ interacción Humano-Computadora

Lenguaje y comunicación

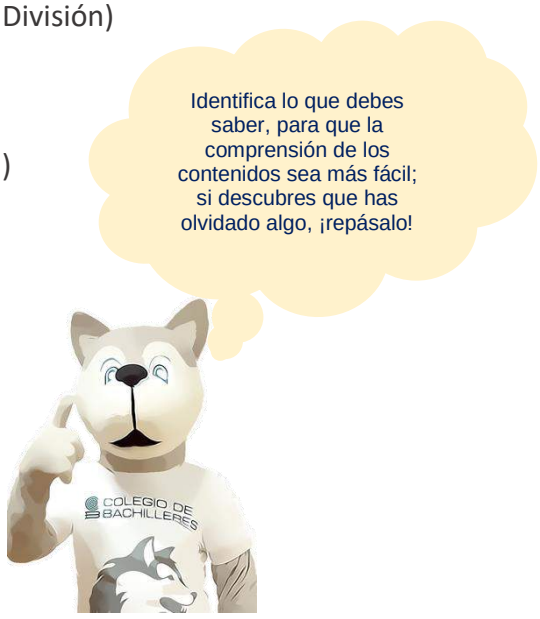
- ✓ Sintaxis
- ✓ Verbos en idioma inglés
- ✓ Sintaxis de la comunicación en idioma inglés

Matemáticas

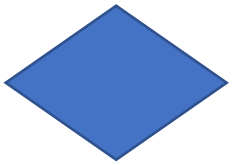
- ✓ Variables
- ✓ Constantes
- ✓ Aritmética (suma, resta, multiplicación y División)

Filosofía

- ✓ Tablas de verdad (Condicionales “Y”, “O”)
- ✓ Lógica proposicional
- ✓ Lógica procedimental



Instrucciones: Dependiendo del símbolo define el nombre del símbolo del diagrama de flujo y comenta cuál es su interpretación o función.

Símbolo ANSI / OSI	Nombre del símbolo	Función o interpretación
	P.ej. **"Proceso"**	P. ej. "**Procesa o Interpreta instrucciones que arrojan un resultado**"
		
		
		
		
		

Introducción

Como ya hemos visto antes, para comprender los procesos de programación en Java es preciso comprender cuál es su estructura y no solamente introducir código o incrustarlo directamente desde internet; al mismo tiempo y como ya lo habíamos visto en el tema de herencia y polimorfismo, la finalidad es evitar hacer código innecesariamente grande, para que el código no solamente sea más limpio, sino que sea más concreto, en este caso será más eficiente, de mayor comprensión.

1. Estructura de control secuencial en java

1.1 Normas de estructura y sintaxis

Es el conjunto de normas y forma de ordenamiento de la escritura en lenguaje de programación en Java.

Para generar la sintaxis en lenguaje de programación en Java es cómo se muestra en la siguiente imagen:

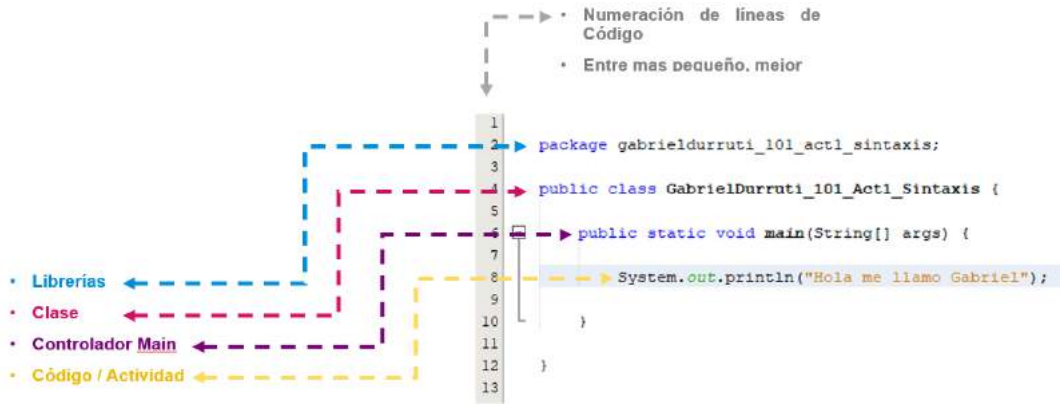


Imagen 12. Ejemplo de la estructura básica en Java.

Las librerías, es un conjunto de instrucciones previamente elaboradas y los cuales son referenciadas en el código.

la clase, en sí es el archivo contenedor de todo el código y que lleva un nombre para una mejor ubicación.

La clase **main** es aquella función que controla el programa, es necesario para que nuestro programa se ejecute correctamente.

Código o actividad, es el conjunto de instrucciones encapsuladas en llaves "{ }", las cuales permiten que se ejecute un algoritmo o múltiples algoritmos.

Tokens

Definición de Token

Son todos aquellos elementos textuales que forman parte del léxico (vocabulario) de los lenguajes de programación, en este caso, Java.

Los cuales se dividen en 5 ramas que se muestran en la imagen siguiente:

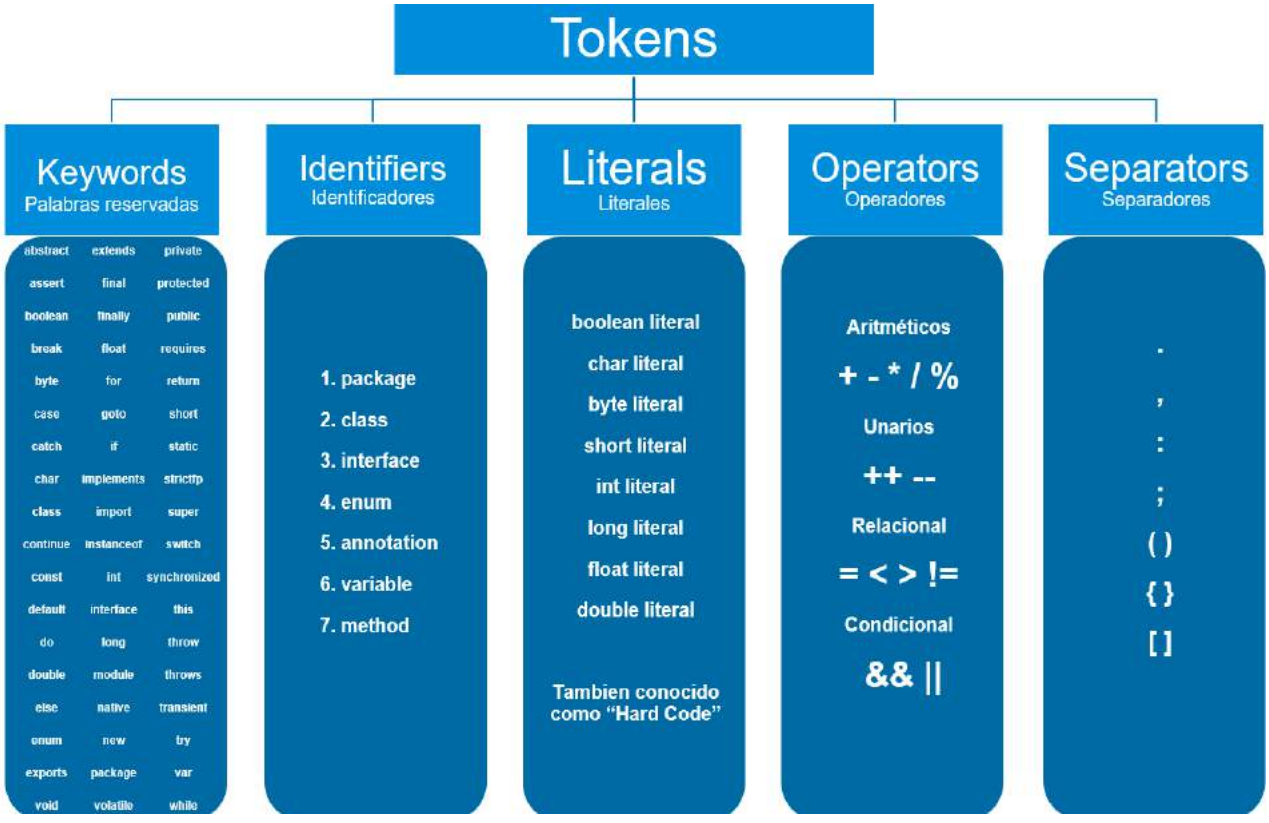


Imagen 13. clasificación de los Tokens.

Keywords – Palabras reservadas

Son aquellas palabras textuales que no pueden ser utilizadas como variables, y cuya función es la de contextualizar el elemento asociativo a la sintaxis.

P.ej. public static void NombreFuncion() { }
Las palabras en "**negritas**" demuestran el contexto de palabras reservadas.

Identifiers – Identificadores

Son aquellas palabras textuales, que demuestran el contexto del código.

P.ej. package gabrieldurruti_101_act3_polimorfismoadhoc;
Las palabras en "**negritas**" identifica la ubicación de la clase (en este caso un paquete).

Literals – Literales

También conocido como código duro, es incrustar datos directamente en el código, no han sido procesados, ni contenidos en alguna variable, ni de alguna fuente externa como una base de datos, o archivos.

P.ej. System.out.print ("El resultado es 3.1416");

Las palabras en "**negritas**" muestran cómo es la literal de un mensaje (puede ser numérico o textual que no fue calculado matemáticamente, ni extraído de otra fuente).

Operators – Operadores

Los operadores tienen la función de hacer cálculos aritméticos (suma, resta, multiplicación, división), hacer alguna comparación lógica cuyo resultado arroje un verdadero o un falso (igual que, menor que, mayor que o diferente que), Y en otros casos validaciones mediante condiciones que arrojan un resultado verdadero o falso (como la utilización de las aulas de verdad "y" así como "o").

P.ej. if (A > B) { };

El operador en "**negritas**" muestran cómo Se realizan las comparaciones mediante una función o bloque de código.

Separators – Separadores

El objetivo de los separadores es realizar algún tipo de Unión mediante la utilización de puntos para determinar un método

P. ej System.out.print

O segmentar algún tipo de operación en Java cómo se muestra a continuación:

P. ej case**:**
P. ej. If **(**A = B**) {**
    System.out.print (“mensaje”) **;**
}

Los separadores en “**negritas**” muestran cómo se segmentan los bloques de código.

ESTRUCTURA DE CONTROL

Antes que nada, te habrás encontrado con la evaluación de una situación como “saber si he aprobado una asignatura”, “¿te acuerdas de la contraseña de tu cuenta de internet?”, “si un equipo deportivo ha ganado un partido” o “si me alcanza para comprar un producto que me gusta”. Todos los conceptos están ligados a la lógica, razonamiento humano y a la programación.

Para generar una estructura de control debemos conocer las siguientes definiciones que se utilizan en la lógica.

Dato	Valor	Expresión o Condición
Cualquier cosa (puede ser imágenes, números, textos, etc.).	Es todo dato contenido en una variable o constante.	Es la comparación de 2 o más valores, el cual arroja un resultado en términos de valores de verdad (“verdadero” o “falso”).

Definición de Estructura de Control en programación

Es la evaluación matemática o lógica de uno o más valores que determinarán el rumbo de nuestra aplicación, programa o sistema.

2. Estructura de control condicional en java

Ejemplo del código en ejecución en NetBeans

Si el valor es de la Calificación es "7"	Si el valor es de la Calificación es "5"

```
public static void main(String[] args) {
    // TODO code application logic here
    double miCalificacionJava = 7;
    if (miCalificacionJava > 5)
    {
        System.out.println("Aprobó");
    }
    else
    {
        System.out.println("Reprobó");
    }
}

}
```

```
Output - JavaApplication1 (run)
run:
Aprob
BUILD SUCCESSFUL (total time: 0 seconds)
```

```
public class JavaApplication1 {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
        double miCalificacionJava = 5;
        if (miCalificacionJava > 5)
        {
            System.out.println("Aprobó");
        }
        else
        {
            System.out.println("Reprobó");
        }
    }
```

```
javaapplication1.JavaApplication1 > main > if (miCalificacionJava > 5)
Output - JavaApplication1 (run)
run:
Reprob
```

Actividad de aprendizaje 1

Crea un proyecto de Java, cuya clase "main", una expresión (comparación) de al menos 2 valores y arroje un resultado en su ejecución.

Parámetros

La clase "main" deberá contar con una expresión la cual compare 2 variables y arroje un resultado con "false" y con "true".

Puedes apoyarte en el siguiente video, ve realizándolo paso a paso.

YouTube: Programación en Java | Windows | S3-2 Estructura de Control
https://bit.ly/3MVyv70

Guarda el proyecto con el nombre [NombreApellido_Act3-2_NombreActividad].

2.1 Simple, doble y múltiple

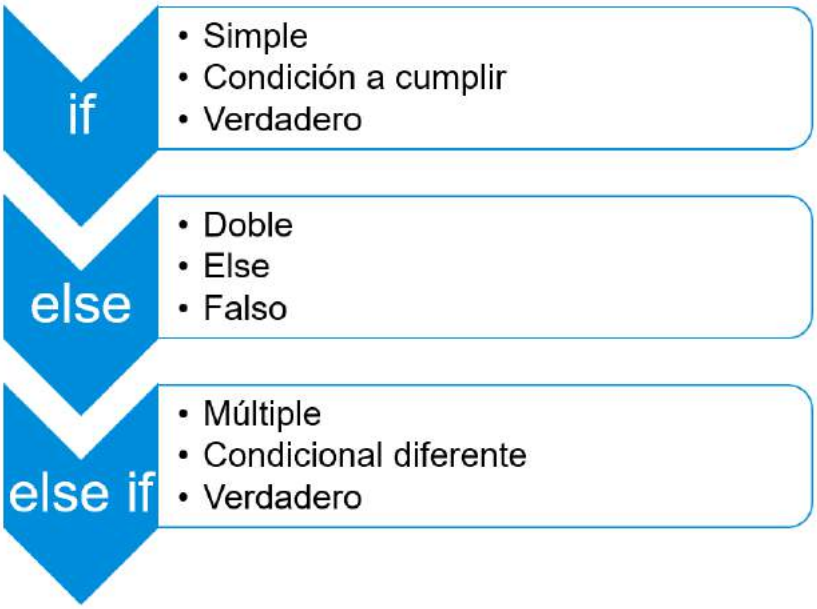


Imagen no. 14 estructura básica condicional.

2.2 Sintaxis

La estructura básica para generar una estructura de control condicional en lenguaje programación Java es el siguiente:

```
if (     ){
    //código
}
else if (     ){
    //código
}
else{
    //código
}
```

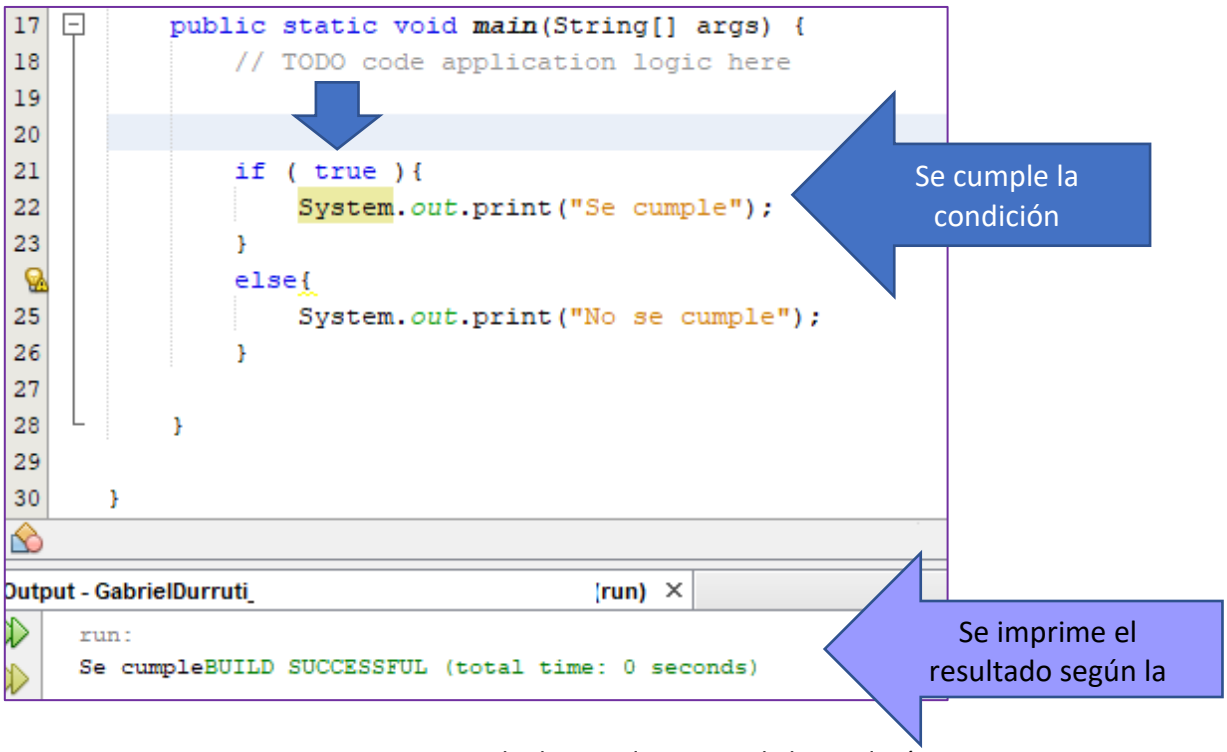


Imagen no. 15. Ejemplo de cuando se cumple la condición.

Ejemplo de cuando no se cumple una condición:

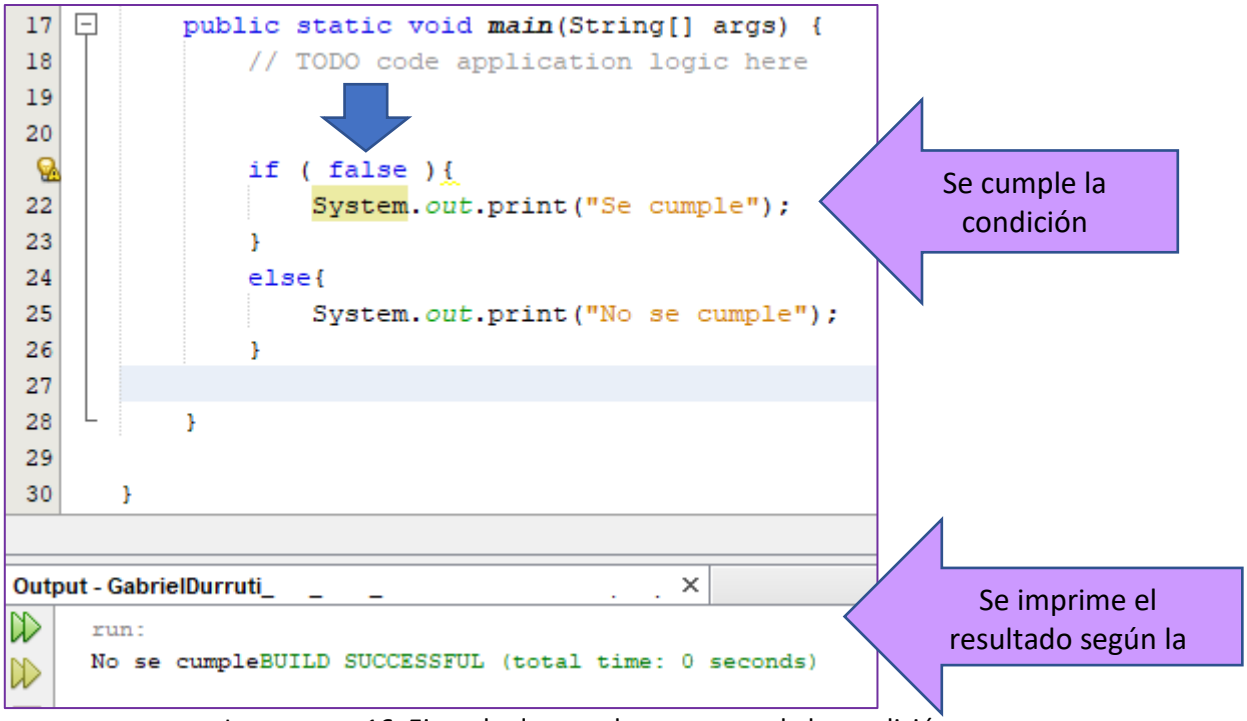


Imagen no. 16. Ejemplo de cuando no se cumple la condición.

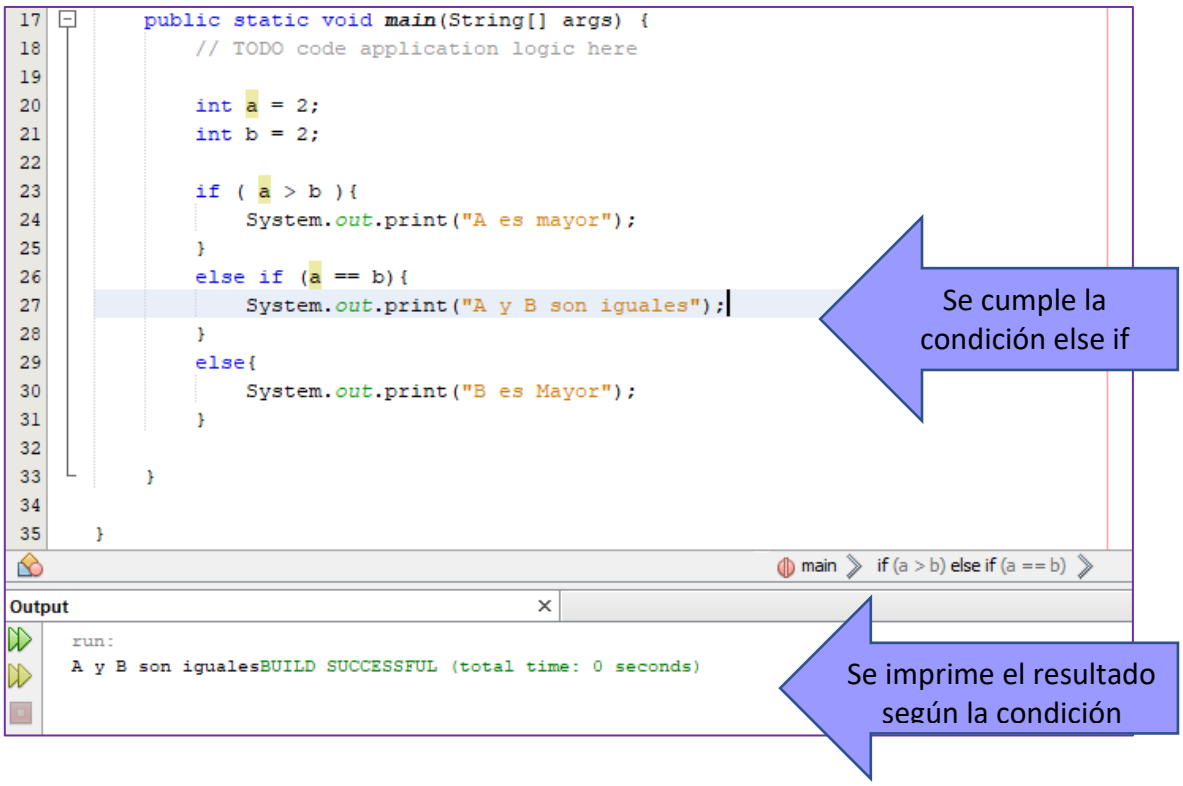
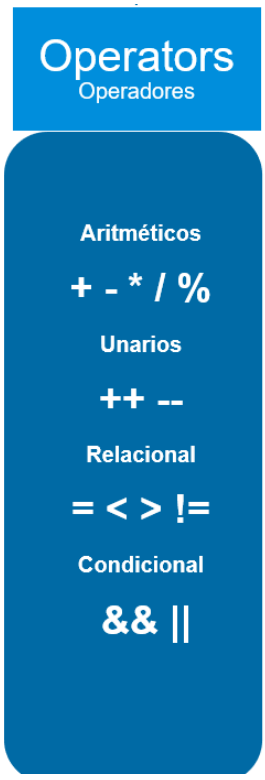


Imagen 17 Ejemplo de una tercera condición que se cumple.



Puedes utilizar los operadores Relacionales o Condicionales para hacer validaciones que necesites como, por ejemplo:

Preguntar si

Relacional	
Es igual que	==
Es menor que	<
Es mayor que	>
Es diferente que	!=

Condicional	
Si ambos valores coinciden (Y)	&& (shift + “6”)
Si uno u otro valor coinciden (O)	|| (La tecla a la izquierda del “1”)

Ejemplos:

```
if ( a < b) {   }
if ( a == b && c == d) {   }
```

3. Estructuras de control repetitivas en java

Una de las malas prácticas de la programación, es realizar conteos con literales (poner código sin procesar), si un cliente nos llegase a solicitar un conteo del 1 al 12, y lo hacemos sin procesar, podemos cometer el error de generar mucho código y mucha pérdida de tiempo como se puede ver en la imagen siguiente:

```
System.out.println(1);
System.out.println(2);
System.out.println(3);
System.out.println(4);
System.out.println(5);
System.out.println(6);
System.out.println(7);
System.out.println(8);
System.out.println(9);
System.out.println(10);
System.out.println(11);
System.out.println(12);
```

Como pudimos ver en la imagen, pusimos doce líneas de código, y perdimos bastante tiempo, existe una forma de hacerlo sin tener que recurrir a tantas líneas de instrucción.

Ciclos de iteración

Tienen como objetivo ejecutar las mismas instrucciones en un número de veces determinado, la condición esencial es que tiene que ser finito (básicamente para no saturar la memoria de nuestra computadora).

Elementos de los ciclos de iteración

Los elementos que componen los ciclos de iteración repetición, son los siguientes:

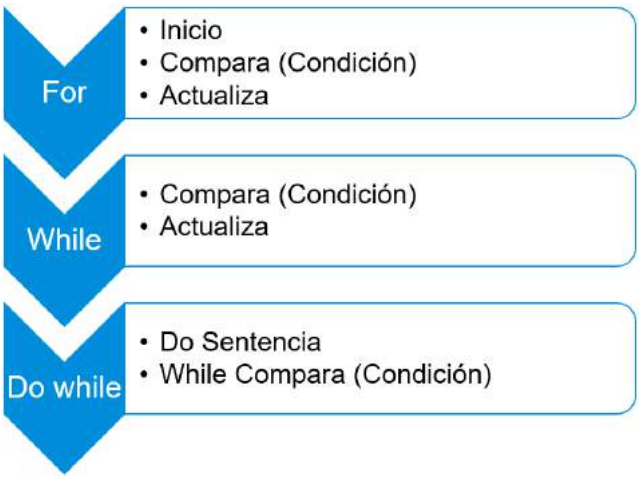


Imagen 18 Ejemplo de los contenidos de los ciclos iterativos.

3.1 Sintaxis de las estructuras de control repetitivas: for, while, do while

FOR

La estructura básica para el ciclo iterativo “for” o “para” es el siguiente:

```
for ( inicio ; comparacion ; actualizacion )
{
    //Codigo
}
```

Ejemplo:

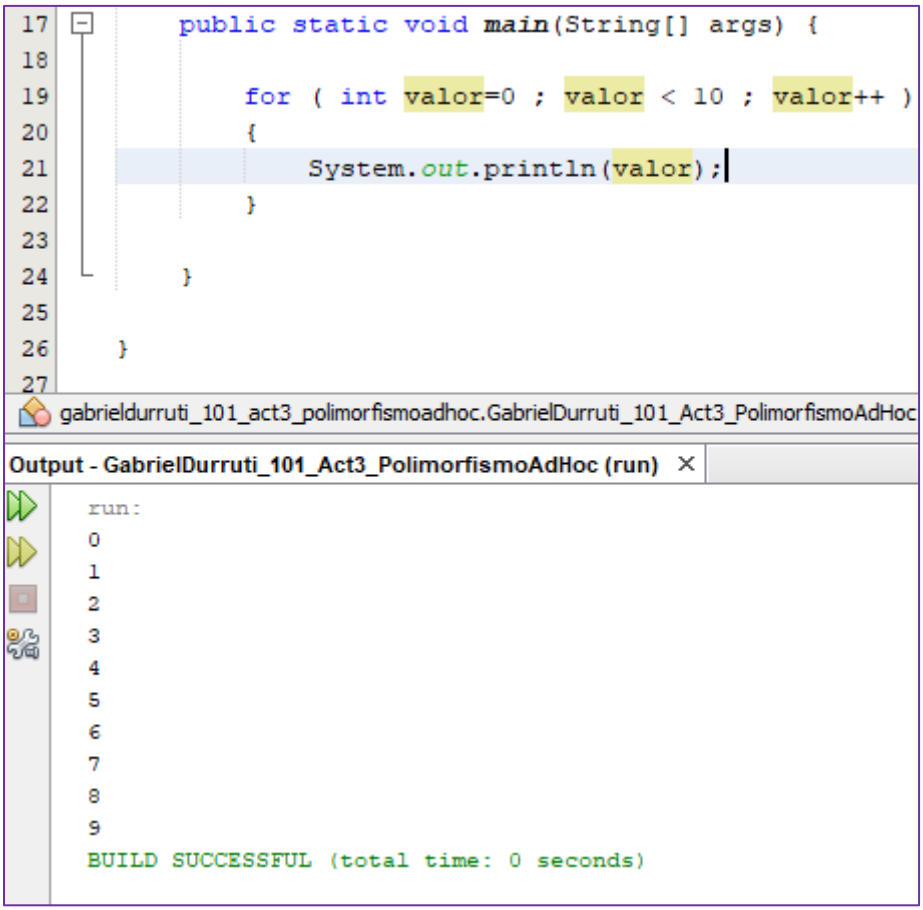


Imagen 19. Muestra de cómo se realiza el ciclo FOR

WHILE

La estructura básica para el ciclo iterativo “while” o “mientras” es el siguiente:

```
while ( comparacion )
{
    actualizacion;
    //Codigo;
}
```

Ejemplo:

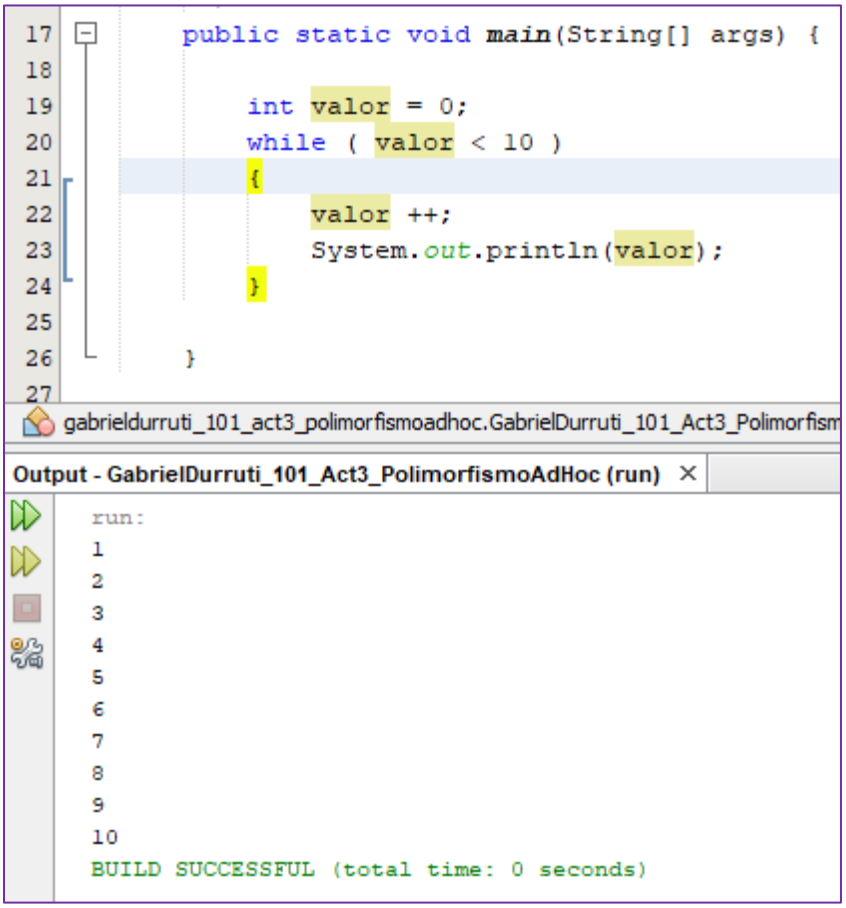


Imagen 20. Muestra de cómo se realiza el ciclo WHILE

DO WHILE

La estructura básica para el ciclo iterativo “do while” o “Haz Mientras” es el siguiente:

```
do {
   //código
}
while ( Comparacion hasta llegar a false );
System.out.println(//impresion de datos);
```

Ejemplo:

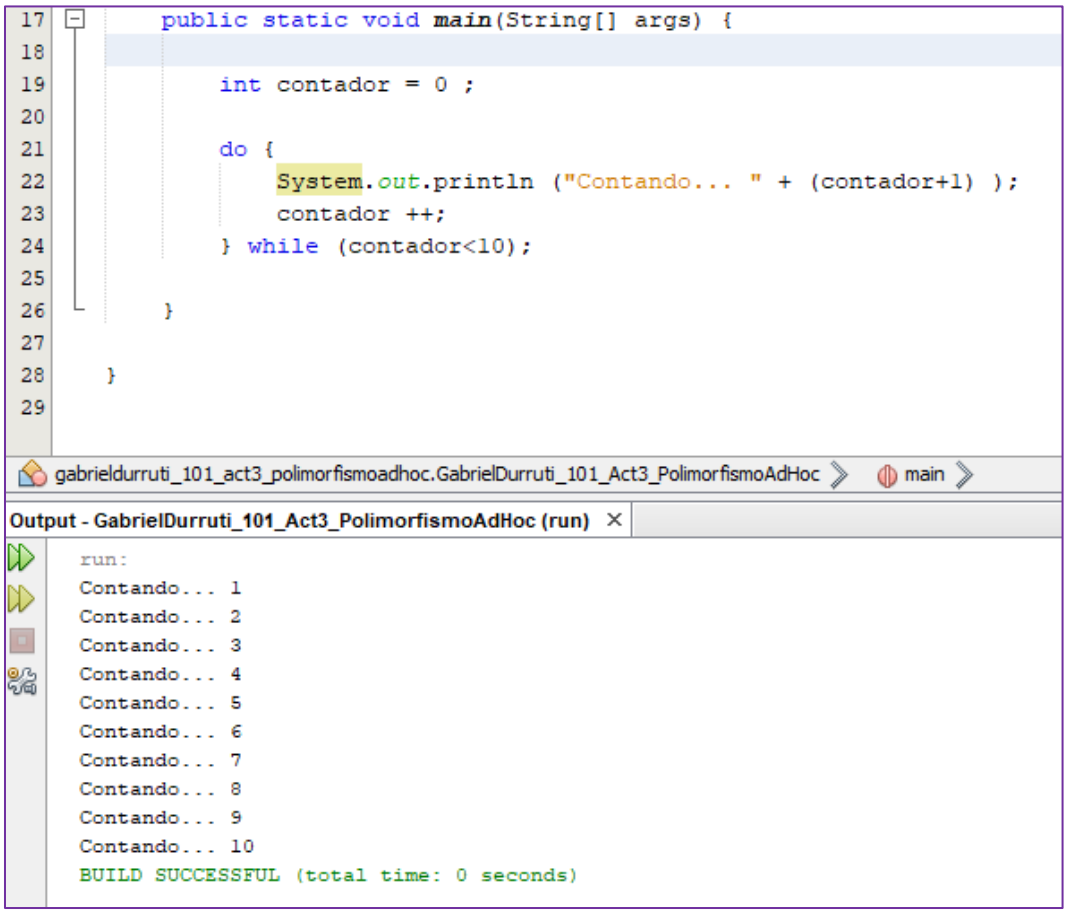


Imagen 21. Muestra de cómo se realiza el ciclo DO WHILE

Actividad de aprendizaje 2

Crea un proyecto de programación en Java, el cual permita, hacer conteos con las 3 estructuras secuenciales vistas en los ciclos iterativos.

Parámetros:

Tu proyecto debe contener 3 conteos con las estructuras:

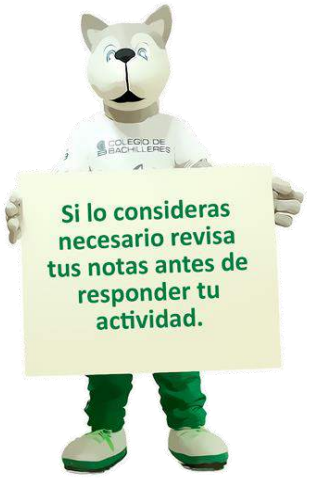
- ✓ FOR
- ✓ WHILE
- ✓ DO WHILE

Puedes apoyarte en el siguiente video, ve realizándolo paso a paso:

YouTube: Programación en Java | Windows | S6 Ciclos Iterativos
http://bit.ly/CB-JAVA6

Guarda el proyecto con el nombre [NombreApellido_Act6_NombreActividad].

Nota. El docente o el asesor te indicará en qué lugar de almacenamiento en la nube subirás tus proyectos elaborados, de la misma forma guárdalos en tu computadora, para lo cual crea una carpeta llamada "programación en Java"



Introducción a la Programación - A Elementos léxicos del Lenguaje de programación en C.
https://elvex.ugr.es/decsai/c/apuntes/tokens.pdf

Bucles en Java: for, while, do while Con Ejemplos.
https://javadesdecero.es/basico/bucles-for-while-do-while-ejemplos/

While y do while en Java (ciclos o bucles).
https://www.aprenderaprogramar.com/index.php?option=com_content&view=article&id=625:while-y-do-while-en-java-ciclos-o-bucles-ejemplo-break-salir-de-un-bucle-ejercicios-resueltos-cu00659b&catid=68&Itemid=188

Practicando con los códigos W3 School para programación en Java.
https://www.w3schools.com/java/

Instrucciones: ¿Cuál consideras que es tu nivel de dominio en el trabajo como programador en JAVA, en este segundo corte?

Marca con una "X" la columna que corresponde.

	EXCELENTE	MUY BIEN	BUENO	DEBE MEJORAR
Definir la estructura básica de Java				
Definir los elementos que componen a los tokens en Java				
Definir con claridad la estructura de la sintaxis de la programación en Java				
Saber cómo se conforma la estructura condicional en Java				
Saber cómo se conforma la estructura condicional doble en Java				
Construir una estructura de bucle iterativo For				
Construir una estructura de bucle iterativo While				
Construir una estructura de bucle iterativo do while				

Sitios web

Alex Walton. (2020). *Bucles en Java: for, while, do while con ejemplos*. Java desde cero. Recuperado el 18 de mayo de 2022 en https://javadesdecero.es/basico/bucles-for-while-do-while-ejemplos/

Alex Rodríguez. *While y do while en Java (ciclos o bucles). Ejemplo break: salir de un bucle. Ejercicios resueltos (CU00659B).* Aprende a programar. Recuperado el 18 de mayo de 2022 en https://www.aprenderaprogramar.com/index.php?option=com_content&view=article&id=625:while-y-do-while-en-java-ciclos-o-bucles-ejemplo-break-salir-de-un-bucle-ejercicios-resueltos-cu00659b&catid=68&Itemid=188

Documento digital

Fernando Berzal. *Introducción a la Programación - A Elementos léxicos del Lenguaje de programación en C.* Recuperado el 18 de mayo de 2022 en https://elvex.ugr.es/decsai/c/apuntes/tokens.pdf

Tabla de imágenes del corte 2

Figura	Origen	Referencia
12	Propia del autor	https://youtube.com/c/GabrielDurruti
13	Propia del autor	https://youtube.com/c/GabrielDurruti
14	Propia del autor	https://youtube.com/c/GabrielDurruti
15	Propia del autor	https://youtube.com/c/GabrielDurruti
16	Extracción de APACHE IDE NETBEANS 12.0	https://netbeans.org/
17	Extracción de APACHE IDE NETBEANS 12.0	https://netbeans.org/
18	Extracción de APACHE IDE NETBEANS 12.0	https://netbeans.org/
19	Propia del autor	https://youtube.com/c/GabrielDurruti
20	Extracción de APACHE IDE NETBEANS 12.0	https://netbeans.org/
21	Extracción de APACHE IDE NETBEANS 12.0	https://netbeans.org/
22	Extracción de APACHE IDE NETBEANS 12.0	https://netbeans.org/

CORTE

3

Desarrollo de soluciones informáticas en Java

Aprendizajes esperados:

Contenidos específicos	Aprendizajes esperados
1. Fase de planeación. 1.1 Plan de trabajo 1.2 Prototipos de prueba **2. Fase de diseño.** 2.1 Modelado de datos 2.2 Desarrollo de componentes a partir del plan de trabajo **3. Fase de desarrollo.** 3.1 Integración de los objetos creados que corresponde con la funcionalidad de la aplicación para PC 3.2 Pruebas de funcionalidad	1. Elabora la fase de planeación de un sistema informático para PC. 2. Elabora el diseño de los componentes de la solución en JAVA y los diagramas UML de las clases y objetos. 3. Desarrolla la aplicación funcional para PC en JAVA

PROPÓSITO

Al finalizar el corte el estudiante será capaz de realizar proyectos informáticos programados en Java.

CONOCIMIENTOS PREVIOS

Para que puedas lograr los aprendizajes esperados con el corte de evaluación 3, es necesario que refuerces los siguientes conocimientos:

Modelado de Sistemas y Principios de Programación

- ✓ Construcción de un diagrama de flujo
- ✓ Diagrama de clases.
- ✓ Diagrama de despliegue
- ✓ interacción Humano-Computadora

Crear y Administrar bases de datos

- ✓ Conceptualizar y definir y diferenciar los sistemas gestores de bases de datos
- ✓ Saber SQL
- ✓ Definir y codificar DDL
- ✓ Definir y codificar DML

Lenguaje y comunicación

- ✓ Lectura.
- ✓ Redacción.
- ✓ Comprensión.
- ✓ Verbos en idioma inglés.
- ✓ Sintaxis de la comunicación en idioma inglés.

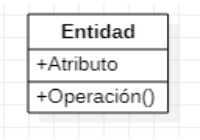
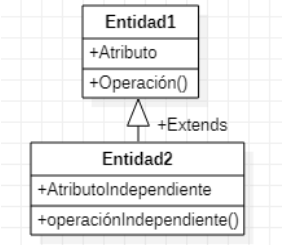
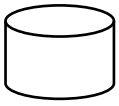
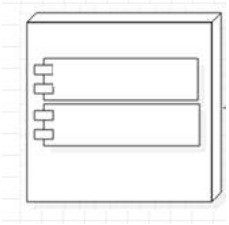
Matemáticas

- ✓ Interpretación del lenguaje algebraico.
- ✓ Variables
- ✓ Constantes
- ✓ Literales
- ✓ Aritmética (suma, resta, multiplicación y División).

Filosofía

- ✓ Tablas de verdad (Condicionales “Y”, “O”).
- ✓ Lógica proposicional.
- ✓ Lógica procedimental.

Instrucciones: Define el nombre del símbolo del diagrama de flujo y comenta cuál es su interpretación o función.

Símbolo UML	Nombre del símbolo	Función o interpretación
		
		
		
		
		
		

Introducción

En la actualidad, las grandes empresas que se dedican a desarrollar aplicaciones informáticas tienen un esquema de trabajo utilizado a nivel nacional e internacional, el cual busca la satisfacción y trata de cumplir con los requerimientos de los clientes (calidad). Este mismo modelo normalmente lo conocerás como "ingeniería del software", y es el que aprenderás a manejar.

1. Fase de planeación

La forma de esquema de desarrollo a partir de la planeación es muy importante, ya que tienes que considerar varios aspectos fundamentales a la hora de desarrollar un producto software (también llamado aplicación o programa informático).

Plan de trabajo

Lo que tienes que hacer es prever, situaciones y elementos importantes como los alcances económicos, los tiempos de desarrollo, Las personas con las que vas a trabajar (considerando sus perfiles, habilidades y competencias profesionales).

Para esto mismo tienes que aprender a desarrollar un mecanismo denominado "Diagrama de Gantt", el cual te permite ver todos estos elementos, y te permite ver los pros y los contras del desarrollo de tu aplicación o sistema informático.

Diagrama de Gantt

Crear tareas con las fases de la "Ingeniería del Software" (el cual está basado en las fases del proceso administrativo aplicado a la programación).

Fase mecánica

- ✓ Planificación
- ✓ Organización

Fase dinámica

- ✓ Dirección
- ✓ Control

Descomposición

Cuando tengas un problema muy grande, a la hora de desarrollar un sistema informático, lo primero que tienes que hacer es segmentar el problema en partes pequeñas, a esto se le llama descomposición. Y eso te permite, resolver los problemas paso a paso, y en menos tiempo. Coloquialmente se le dice “Segmenta el problema en cachitos (partes pequeñas).

Definiciones importantes de la planeación

Hito

Es una parte importante del desarrollo.

Por ejemplo, lograr una conexión de base de datos al programa.
Por ejemplo, terminar todos los diagramas de clases para que sean codificados en Java.
Por ejemplo, lograr modificar la base de datos a petición del usuario.

Meta

Es cumplir con la totalidad de lo requerido.
Por ejemplo, lograr terminar todo el sistema y poderlo instalar.

Plan de trabajo Prototipos de prueba

Vamos a utilizar una aplicación qué encontrarás de forma gratuita en internet la cual se llama “Planner” https://wiki.gnome.org/Apps/Planner

Organización

Esta aplicación te permite ver toda la organización total de tu proyecto, en este caso vas a disponer de varios elementos los cuales vamos a numerar.

1. Estructura de la descomposición del trabajo (WBS) - Desglose de las actividades (Bloque de cada tarea y subtarea)
2. Tiempo planeado de la ejecución de la actividad
3. Calendarización de actividades
4. Dependencia entre tareas y subtareas (Lo que significa que, una vez terminada la tarea, se da inicio a la siguiente tarea)

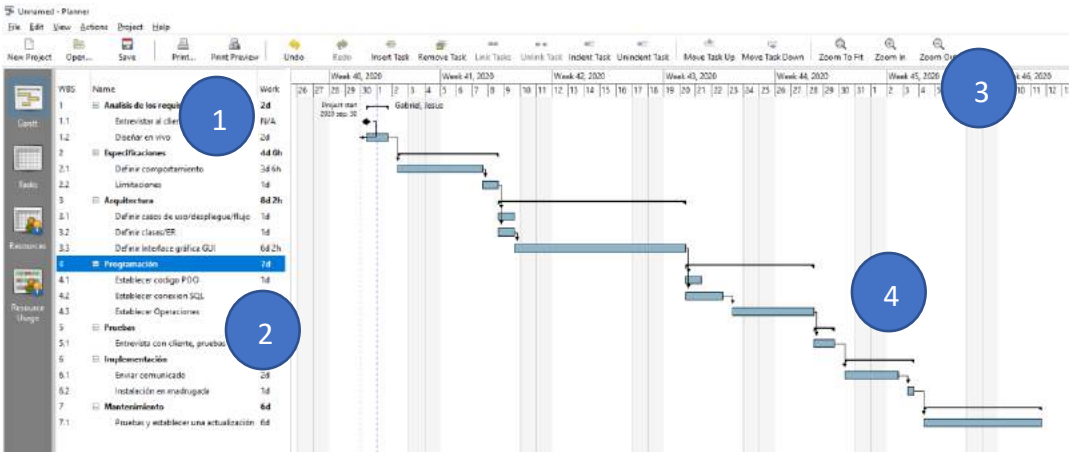


Imagen 22. Ejemplo de un diagrama de Gantt hecho en Planner.

Actividad de aprendizaje 1

Realiza un diagrama de Gantt, lo puedes elaborar a mano o con el programa "Planner" https://wiki.gnome.org/Apps/Planner

Parámetros:
El diagrama de Gantt deberá incluir los siguientes elementos:

- ✓ 3 tareas
- ✓ 3 subtareas por cada tarea
- ✓ 3 personas que funjan como recursos humanos
- ✓ una calendarización
- ✓ una temporización por cada tarea

Puedes apoyarte en el siguiente video, ve realizándolo paso a paso:

YouTube: Programación en Java | Windows | S7 Planificación de un proyecto software
http://bit.ly/CB-JAVA7

Guarda el proyecto con el nombre [NombreApellido_Act7_NombreActividad].

Prototipos de prueba

Un prototipo es un modelo funcional pero breve del sistema a desarrollar, que tiene como propósito mostrar al cliente la operatividad de la solución informática a desarrollar en Java.

2. Fase de diseño

Modelado de datos

Es el proceso que se realiza para levantar, analizar y modelar las estructuras de datos necesarias para el procesamiento de información de una entidad o negocio, lo que permitirá contar con estructuras de datos sólidas que eviten problemáticas de redundancia e inconsistencia, dando integridad a la base de datos a utilizar.

Desarrollo de componentes a partir del plan de trabajo

Toma de requerimientos

Superficialmente hablaremos, de que una de las tareas que debe realizar uno de los miembros del equipo, es la de entrevistar al cliente (puede ser ésta una empresa o un representante legal), el cual tiene que brindar los detalles técnicos y procedimentales para que nuestro sistema sea perfectamente funcional y resuelva todo el problema. esto se logra a partir de una entrevista con el cliente, en el cual preguntamos las necesidades de la empresa, de los operadores, de los usuarios y de los administrativos.

Análisis del problema

una vez contando con los datos elementales, de los pasos a seguir que realiza la empresa, el siguiente paso que tenemos que hacer es el de modelar (esta parte comúnmente la lleva a cabo el analista de sistemas)

Su tarea no es otra que llevar a cabo representaciones gráficas y técnicas de la operación del trabajo a realizar.

Principalmente son 3 y son las siguientes:

- ✓ Diagramas de causa y efecto – Control de calidad – NO UML
- ✓ Diagramas de caso de uso - UML
- ✓ Diagrama de flujo – UML

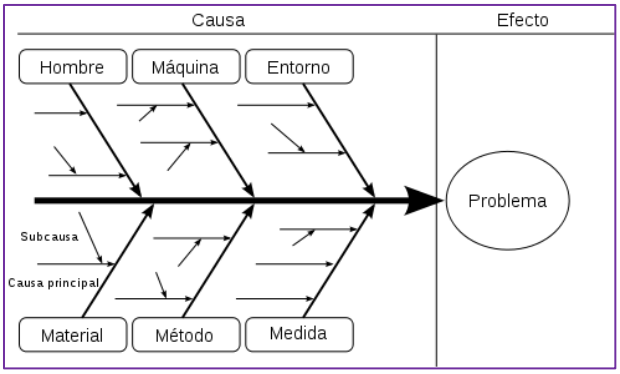


Imagen 23. Ejemplo de diagrama de Causa y efecto (también conocido como diagrama de Ishikawa), el cual no es UML y funciona para determinar los problemas, sus causas, sus soluciones y los efectos que propicia; el objetivo es mejorar el eliminar las causas de una falla o problema e incrementar el control de calidad.

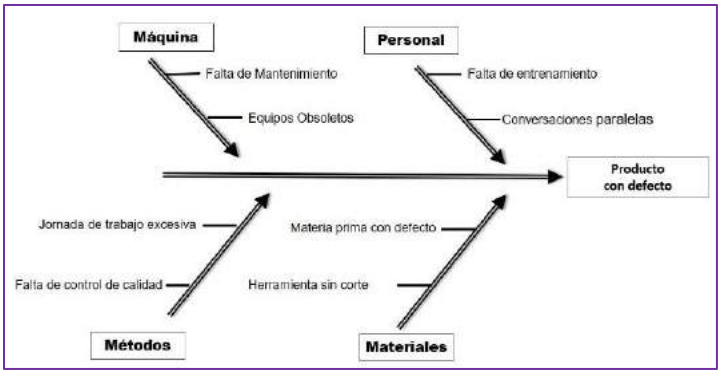


Imagen 24. Ejemplo de identificación de un problema, detallar causas e inferencia.

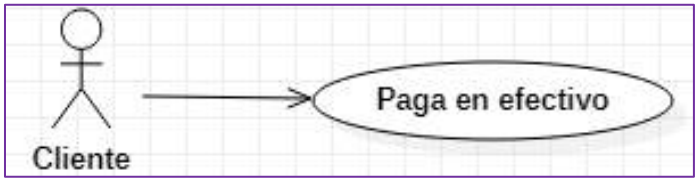


Imagen 25. Ejemplos del diagrama de casos de uso, en el cual se utiliza un actor, y un caso que infiere directamente en los procesos de trabajo

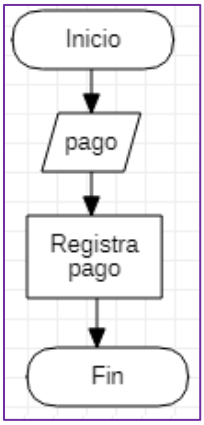


Imagen 26. Ejemplos del diagrama de flujo, en el cual se leen los datos y se procesa una actividad

Cómo podemos ver en la imagen, se pueden notar los factores y elementos que involucran el proceso de acción, para modular un problema o múltiples problemas. Esto nos ayudará a resolver el problema de forma más fácil.

Una vez completado el trabajo del **analista de sistemas**, que no es más que el de hacer los diagramas de abstracción, ahora tiene que definir otros diagramas, los cuales son una antesala al proceso de desarrollo de la programación orientada a objetos.

Estos diagramas son principalmente 3 y son los siguientes:

- ✓ diagrama de clases
- ✓ diagramas de entidad relación
- ✓ diagramas de despliegue

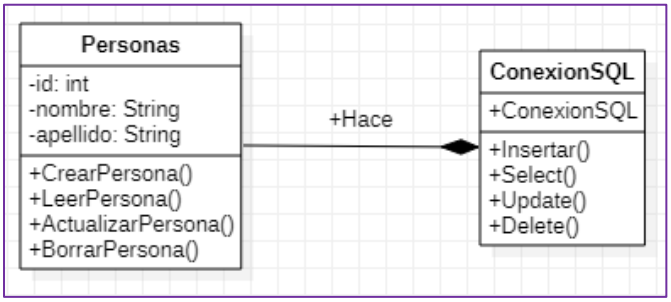


Imagen 27. Ejemplo de un diagrama de clase, el cual permite visualizar el nombre de la entidad, los atributos que la definen, y las operaciones que ésta realiza.

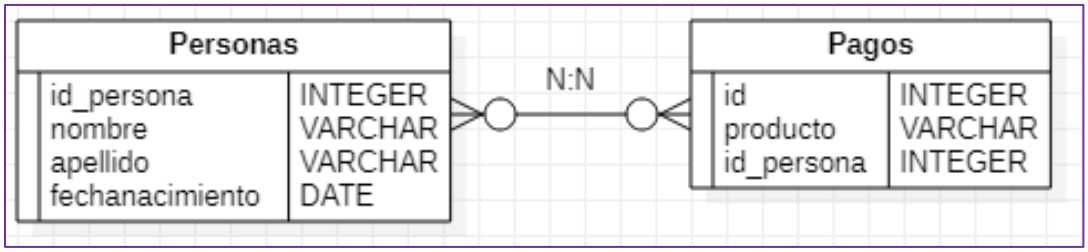


Imagen 28. Ejemplo de un diagrama entidad relación, en el cual las entidades van en plural, los atributos se nombran en singular y el tipo de dato que nos va a contener.

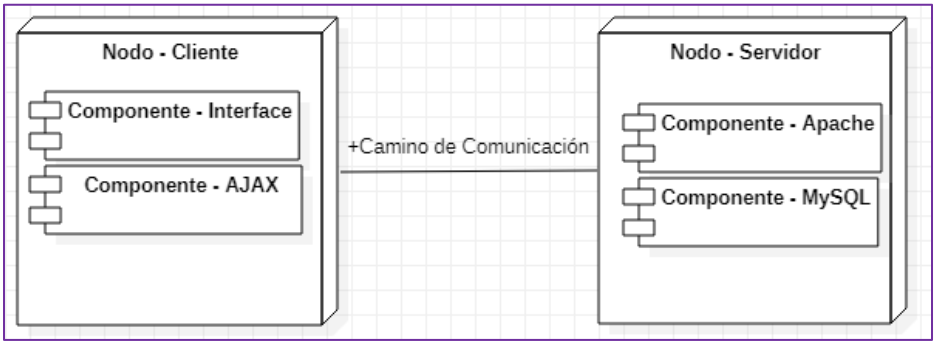


Imagen 29. Ejemplo de un diagrama de despliegue, el cual permite visualizar los nodos, los componentes y la ruta de comunicación de cada aplicación o servicio.

3. Fase de desarrollo

Una vez hecha la observación y el análisis de los requerimientos del cliente mediante la instrumentación diagramas, ahora viene la parte del desarrollo del sistema.

Integración de los objetos creados que corresponde con la funcionalidad de la aplicación para PC

Para unir todos los elementos gráficos con la parte del desarrollo se tiene que elaborar una interpretación abstracta de los algoritmos, el cual se realiza mediante el uso de diagramas de flujo o mediante el uso de pseudocódigo. Vamos a ver un ejemplo.

Modelado de la aplicación

Para lograr la conexión de una aplicación a una base de datos, no importando el lenguaje de programación, primero se crea el diagrama de flujo de la columna vertebral de la aplicación, para hacerlo es necesario que reconozcas los 3 procesos básicos de todo programa:

La base para conectar una base de datos a un programa (sin importar el lenguaje de programación) es el siguiente:

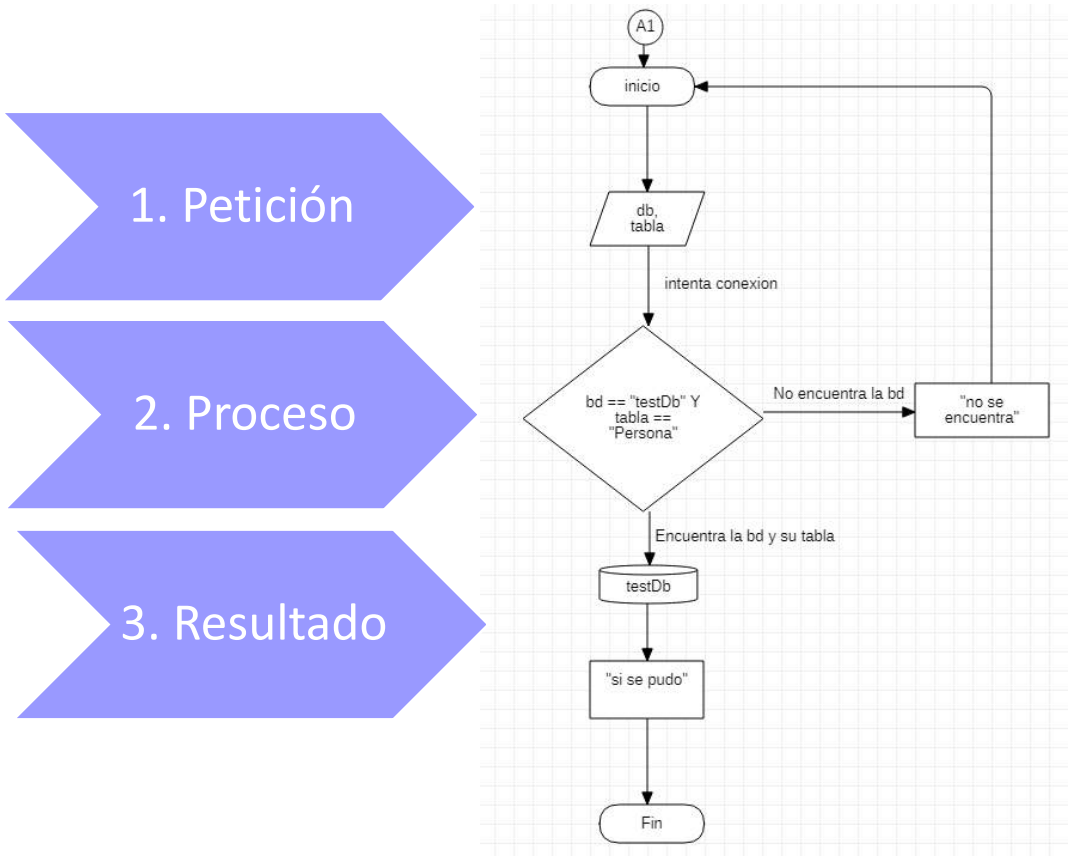


Imagen 30. Elaboración del proceso de conexión de la base de datos el programa.

El objetivo del anterior diagrama de flujo se debe a que constantemente estaremos utilizando esa misma secuencia de pasos para grabar, actualizar y escribir dentro de la base de datos.

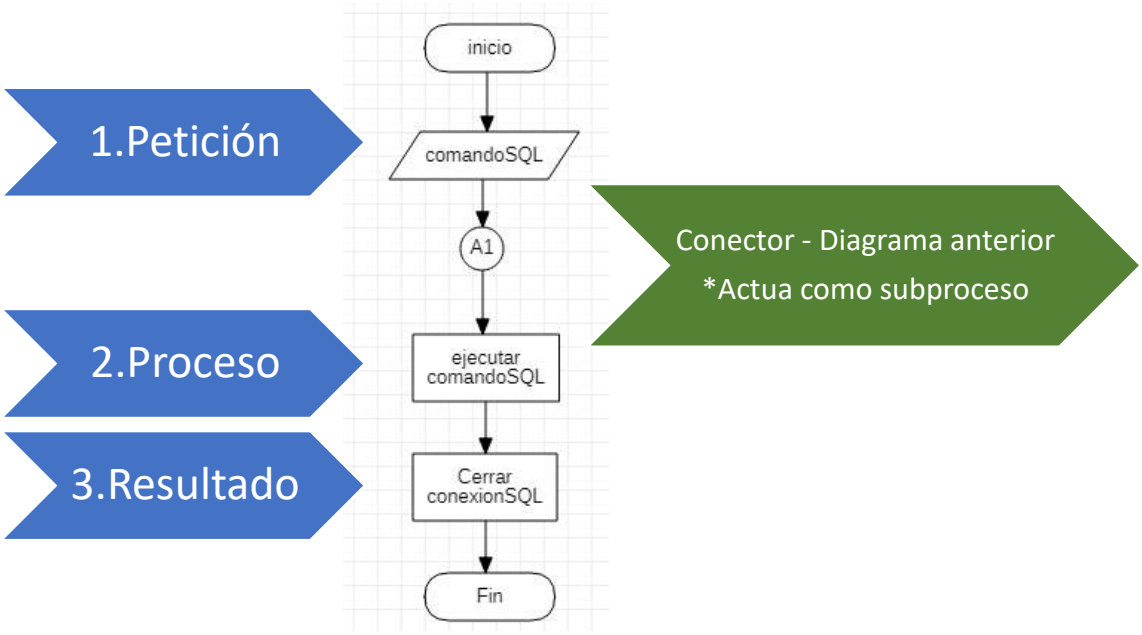


Imagen 31. Elaboración del proceso de conexión de la base de datos el programa.

Actividad de aprendizaje 2

Realiza 5 diagramas de UML (los puedes elaborar a mano o con el programa “STAR UML”).

Parámetros:

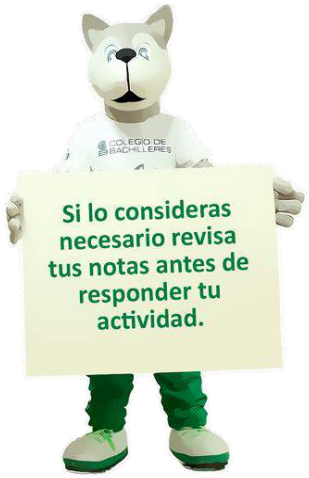
El diagrama de UML deberá incluir los siguientes elementos:

- ✓ 1 diagrama de clases que tenga 2 clases.
- ✓ 1 diagrama de casos de uso, que tenga un caso de uso.
- ✓ 1 diagrama de despliegue, que contenga dos nodos, un componente por cada nodo.
- ✓ 1 diagrama entidad relación.
- ✓ 1 diagrama de flujo el cual contenga una lectura de datos y un proceso.

Puedes apoyarte en el siguiente video, ve realizándolo paso a paso:

YouTube: Programación en Java | Windows | S8 Diagramas UML
http://bit.ly/CB-JAVA8

Guarda el proyecto con el nombre [NombreApellido_Act7_NombreActividad].



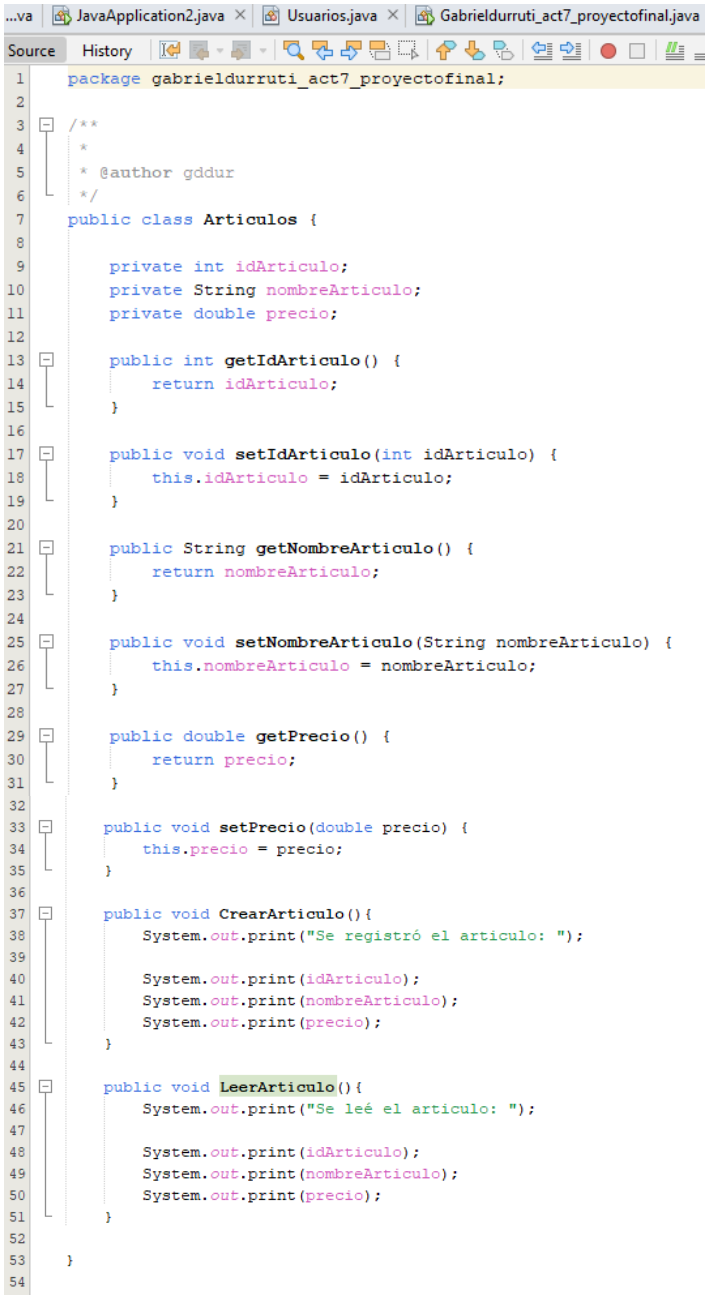
Desarrollo

Para concluir, deberás desarrollar un sistema DEMO (demostrativo muy básico) el cual utilice un par de encapsulados en las clases PERSONAS y ARTICULOS, utilizando los setters y los Getters.

Contenido de la clase “Personas”

```java
package gabrieldurruti_act7_proyectofinal;

/**
 *
 * @author gddur
 */
public class Usuarios {

    private int idUsuario;
    private String nombre;
    private String apellido;

    public int getIdUsuario() {
        return idUsuario;
    }

    public void setIdUsuario(int idUsuario) {
        this.idUsuario = idUsuario;
    }

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public String getApellido() {
        return apellido;
    }

    public void setApellido(String apellido) {
        this.apellido = apellido;
    }

    public void CrearUsuario(){
        System.out.println("Usuario registrado: " + idUsuario +"," + nombre+ "," +apellido);
    }
    public void LeerUsuario(){
        System.out.println("Usuario: " + idUsuario +"," + nombre+ "," +apellido);
    }
    public void ActualizarUsuario(){
        System.out.println("Usuario Actualizado");
    }
    public void BorrarUsuario(){
        System.out.println("Usuario Borrado");
    }

}
```

Contenido de la clase “Articulos”

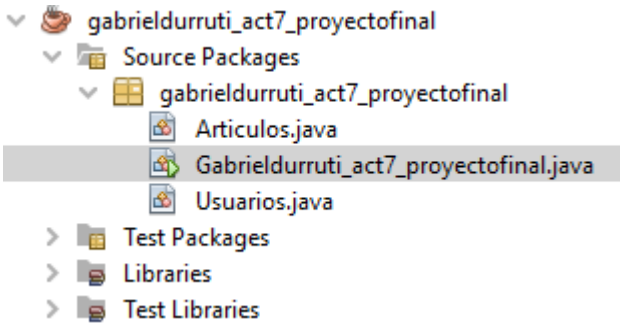



```java
package gabrieldurruti_act7_proyectofinal;

/**
 *
 * @author gddur
 */
public class Articulos {

    private int idArticulo;
    private String nombreArticulo;
    private double precio;

    public int getIdArticulo() {
        return idArticulo;
    }

    public void setIdArticulo(int idArticulo) {
        this.idArticulo = idArticulo;
    }

    public String getNombreArticulo() {
        return nombreArticulo;
    }

    public void setNombreArticulo(String nombreArticulo) {
        this.nombreArticulo = nombreArticulo;
    }

    public double getPrecio() {
        return precio;
    }

    public void setPrecio(double precio) {
        this.precio = precio;
    }

    public void CrearArticulo(){
        System.out.print("Se registró el articulo: ");

        System.out.print(idArticulo);
        System.out.print(nombreArticulo);
        System.out.print(precio);
    }

    public void LeerArticulo(){
        System.out.print("Se leé el articulo: ");

        System.out.print(idArticulo);
        System.out.print(nombreArticulo);
        System.out.print(precio);
    }

}

```

Contenido de la clase “MAIN”

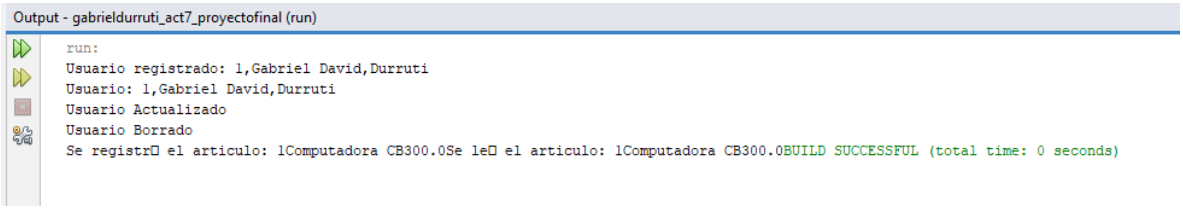
```
Source  History
1  package gabrieldurruti_act7_proyectofinal;
2
3  public class Gabrieldurruti_act7_proyectofinal {
4
5      /**
6       * @param args the command line arguments
7       */
8      public static void main(String[] args) {
9          // TODO code application logic here
10         // Se inicia la varInstancia de la clase Usuarios
11         // recordamos que trae los contenidos publicos de la clase
12         Usuarios varUsuarios = new Usuarios();
13         // Se inicia la varInstancia de la clase Articulos
14         // recordamos que trae los contenidos publicos de la clase
15         Articulos varArticulos = new Articulos();
16
17         //se indican los valores de las variables de Usuarios
18         varUsuarios.setIdUsuario(1);
19         varUsuarios.setNombre("Gabriel David");
20         varUsuarios.setApellido("Durruti");
21         //se leen las funciones con las variables
22         varUsuarios.CrearUsuario(); //recordemos que es un DEMO
23         varUsuarios.LeerUsuario();
24         varUsuarios.ActualizarUsuario();
25         varUsuarios.BorrarUsuario();
26
27         //se indican los valores de las variables de Articulos
28         varArticulos.setIdArticulo(1);
29         varArticulos.setNombreArticulo("Computadora CB");
30         varArticulos.setPrecio(300d);
31         //se leen las funciones con las variables
32         varArticulos.CrearArticulo(); //recordemos que es un DEMO
33         varArticulos.LeerArticulo();
34     }
35
36 }
```

El Contenido del proyecto debe quedar así



Resultados

El resultado en ejecución (con tu nombre) debe salir así:



Pruebas de funcionalidad

Todo sistema debe de ser probado y verificado para su próxima distribución y uso, por lo que es vital realizar pruebas de funcionalidad para detectar errores lógicos, de sintaxis y de entrada de datos.

Actividad de aprendizaje 3

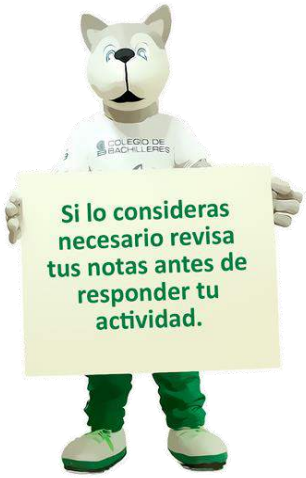
Instrucciones: Debes realizar un proyecto de programación en Java, el cual deberás integrar desde la planificación, los diagramas de UML, programación orientada a objetos, y una sobrecarga; la cual responda como una solución informática a la siguiente situación:

Problema

Un cliente solicita una aplicación de demostración el cual tenga impresión el nombre y apellido del usuario; El programa deberá permitir leer los datos que ponga el usuario (con un identificador, un nombre y apellido) y un artículo (que incluya un identificador, nombre del articulo y un precio), es tanta la urgencia del cliente que por el momento no solicita interfaces gráficas.

Puedes apoyarte viendo el siguiente vídeo

YouTube: Programación en Java | Windows | S9A Desarrollo
http://bit.ly/CB-JAVADFinal



STARUML.
http://staruml.io/

Planner Project management
https://wiki.gnome.org/Apps/Planner

Diagrama de Ishikawa
https://blogdelacalidad.com/diagrama-de-ishikawa/

Diagrama Causa-Efecto
https://www.progressalean.com/diagrama-causa-efecto-diagrama-ishikawa/

AUTOEVALUACIÓN

Instrucciones: ¿Cuál consideras que es tu nivel de dominio en el trabajo como programador en JAVA, en este tercer corte?

Marca con una “X” la columna que corresponde.

	EXCELENTE	**MUY BIEN**	**BUENO**	**DEBE MEJORAR**
Crear la estructura de una planeación en planner				
Asignar tiempos de desarrollo				
Definir con claridad los recursos humanos con los que se van a contar				
Asignar los recursos humanos a cada una de las tareas relevantes el plan de trabajo				
Hacer los diagramas de clases				
Pasar los diagramas a código de programación en Java				

Sitios web

W3 Schools. (2020). Java Tutorial de W3. Recuperado el 18 de mayo de 2022 en https://www.w3schools.com/java/
<
R&D company in the area of software development and engineering. STARUML. Recuperado el 18 de mayo de 2022 en http://staruml.io/

SvitozarCherepii. (2005 - 2020). Planner Project management. Recuperado el 18 de mayo de 2022 en https://wiki.gnome.org/Apps/Planner

Tabla de imágenes del corte 3

Figura	Origen	Referencia
22	Extracción de GNOME PLANNER	https://wiki.gnome.org/Apps/Planner
23	Imagen de Progressa Lean	https://bit.ly/2GGl3Xi
24	Imagen del Blog de calidad	https://bit.ly/3nswGCr
25	Extracción de STAR UML	http://staruml.io/
26	Extracción de STAR UML	http://staruml.io/
27	Extracción de STAR UML	http://staruml.io/
28	Extracción de STAR UML	http://staruml.io/
29	Extracción de STAR UML	http://staruml.io/
30	Extracción de STAR UML	http://staruml.io/
31	Extracción de STAR UML	http://staruml.io/

EVALUACIÓN FINAL

Instrucciones: Realiza un proyecto de programación en Java que integre la planificación, diagramas de UML, programación orientada a objetos y una sobrecarga de métodos; el cual responda como una solución informática a la siguiente situación:

Problema

El Colegio de Bachilleres te solicita un programa de demostración el cual tenga impresión el nombre y apellido del usuario; el programa deberá permitir leer los datos que ponga el usuario (con un identificador, un nombre y apellido), una clase artículo (que incluya un identificador, nombre del articulo y un precio), una clase de cálculo (que integre una suma, resta y multiplicación) es tanta la urgencia del cliente que por el momento no solicita interfaces gráficas.

La solución de este proyecto deberá ser entregada en una carpeta en la nube con tu nombre y grupo, con la siguiente información:

- Planificación del proyecto
- Diagramas UML
- Código fuente en Java en donde se incluya una sobrecarga de métodos.
- Pantallas de captura de la solución informática funcionando.

PLAN 2014

ACTUALIZADO

Somos Lobos Grises,
somos Bachilleres