CHOUCAIR
Business Centric Testing

Workshop de Automatización de Pruebas

NIVELACIÓN EN JAVA

Noviembre de 2025

Formador

Yeison Arias Castrillón

14 años de experiencia en Pruebas

Trayectoria Profesional

- 2008-2009

  Analista de Pruebas

- 2009-2015

  Automatizador de Pruebas

- 2015-2017

  Coordinador de Pruebas de Software

- 2017-2022

  Arquitecto de Automatización

- 2022-Actualmente

  Especialista de Producto

Contacto

yarias@choucairtesting.com

yfarias@gmail.com

CHOUCAIR
Business Centric Testing

Pautas

Horarios

Horario de Inicio

4:30 p.m

Horario Finalización

7:30 p.m

Break

6:00 p.m (15 ~ 20 minutos)

?

Espacio de Preguntas y Evaluar como vamos

7:20 p.m

Reglas de Participación

- Preguntar en cualquier momento
- Mientras no estemos interviniendo, mantener el micrófono apagado
- Participación activa

CHOUCAIR
Business Centric Testing

Introducción a la programación

La programación hace referencia al efecto de programar, es decir, de organizar una secuencia de pasos ordenados a seguir para hacer cierta cosa.

Este término puede utilizarse en muchos contextos, por ejemplo:

- "Vamos a programar una salida para este fin de semana largo (puente)"
- "La programación de ese canal de televisión está bien lograda"

CHOUCAIR
Business Centric Testing

¿Qué se requiere para ser un buen Programador?

Curiosidad

Análisis de Problemas

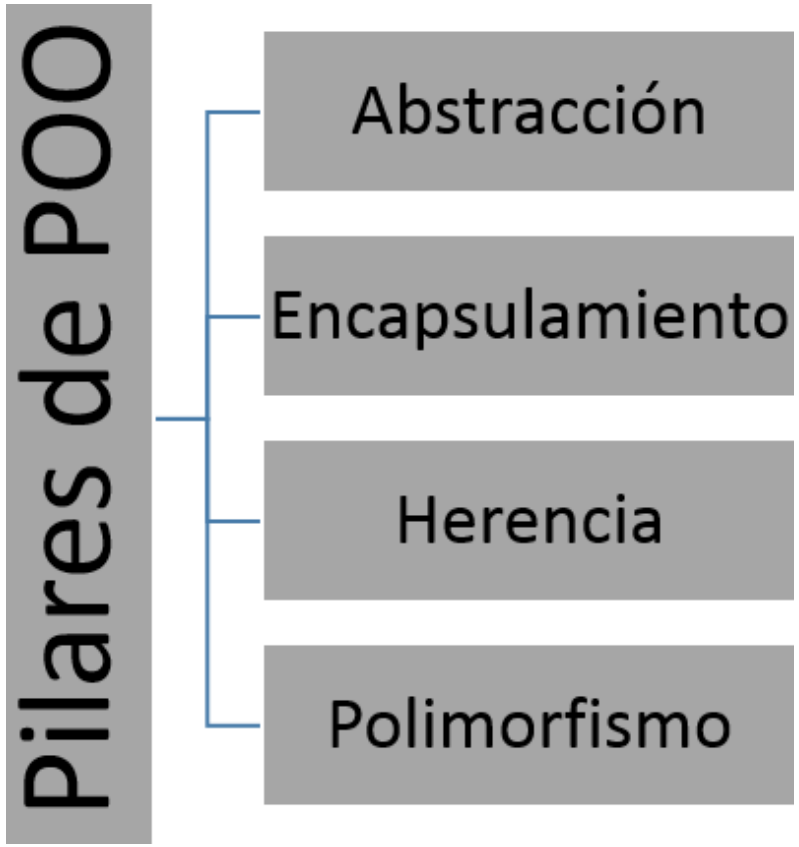
Aprendizaje

Interacción Social

Manejo del tiempo

Creatividad

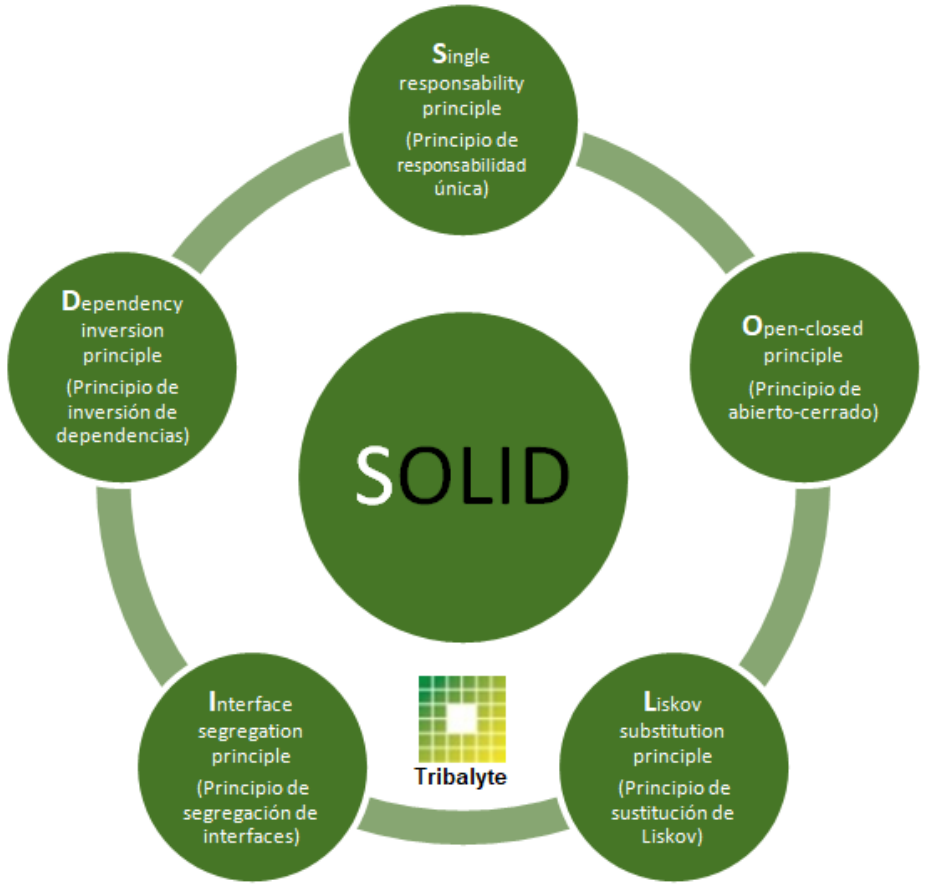
Pilares de la Programación Orientada a Objetos (POO)



Tomado de la página de tokioschool.com

CHOUCAIR
Business Centric Testing

Principios de la Programación Orientada a objetos



Tomado de la página de tribalyte

CHOUCAIR
Business Centric Testing

Cohesión y acoplamiento

¿Qué es el acoplamiento?

Es el grado en que los módulos de un programa dependen unos de otros.
Si para hacer cambios en un módulo del programa es necesario hacer cambios en otro módulo distinto, existe acoplamiento entre ambos módulos

¿Qué es cohesión?

Cohesión no implica dependencia. Tal como dice el significado de la palabra en castellano, es el efecto de reunirse o adherirse las cosas entre sí o la materia de la que está formada.
Algo está cohesionado si tiene sentido, y una dirección común.

Tomado de la página de disrupciontecnologica.com

¿¿¿Y qué es mejor???

EN LA PROGRAMACIÓN ORIENTADA A OBJETOS

Alta cohesión y bajo acoplamiento

Tomado de la página de disrupciontecnologica.com

Beneficios:

- **Legibilidad**
  El código es más fácil de leer y entender
- **Reusabilidad**
  Permite reutilizar componentes fácilmente
- **Mantenibilidad**
  Facilita el mantenimiento del código
- **Alta capacidad de prueba**
  Mejora la capacidad de realizar pruebas

CHOUCAIR
Business Centric Testing

Buenas prácticas en la programación

1. Definición de requisitos. Alcance del proyecto.
2. Dividir los desarrollos en fases o entregables.
3. Elección de un IDE que se adapte a tus necesidades.
4. Estandarizar las reglas del desarrollo.
5. DRY: Don't repeat yourself.
   Duplicar datos, lógica o función en tu código no solo hace que tu código sea largo, sino que también inviertas mucho tiempo en mantener.
6. No inventes.
   Si ya se disponen de módulos y librerías ya testeadas y optimizadas, no pierdas el tiempo en un desarrollo. Los IDE actuales disponen de librerías que te ayudarán en tus desarrollos.
7. Comenta tu código.
8. Divide y vencerás.
9. Testeo de código.
10. Optimización.
11. Seguridad.
    - El acceso a los ficheros (tanto de forma física, como permisos).
    - La posibilidad de modificar el código en la misma ejecución o la inyección sql.
    - Desbordamiento de buffer al hacer uso de un array sin tamaño controlado.
    - Formateo de los datos de entrada (en formularios).
    - Actualización del IDE y las funciones. Una función obsoleta puede originar un agujero de seguridad en nuestro código.
    - Uso de contraseñas en el código. Las contraseñas deben estar cifradas y en la base de datos.
12. Documentación.

CHOUCAIR
Business Centric Testing

Buenas prácticas en la programación

Estas 12 medidas nos garantizan una buena metodología a la hora de desarrollar, que nos va a permitir obtener beneficios claros, si seguimos los pasos indicados. Es por esto por lo que merece la pena invertir tiempo en la definición previa y desarrollo de un proyecto.

Beneficios principales

- Facilitar mantenimiento: Código organizado y bien documentado que permite actualizaciones y correcciones eficientes
- Facilitar la escalabilidad: Estructura flexible que permite el crecimiento del proyecto sin comprometer la calidad
- Facilitar el testeo, localización y resolución de errores: Código modular que simplifica la identificación y corrección de problemas

Referencia: Robert Cecil Martin, el «Tío Bob», es uno de los fundadores principales de los métodos ágiles, autor de «clean code» (para muchos de nosotros, uno de los mejores libros sobre buen y mal código), es uno de los más influyentes autores en programación orientada a objetos.

CHOUCAIR
Business Centric Testing

Patrones de diseño

DEFINICIÓN

Es una solución general reusable que puede ser aplicada a problemas que ocurren comúnmente en el desarrollo de software, es la descripción o plantilla de cómo resolver un problema que puede ser usada en diferentes situaciones.

Características

Soluciones probadas

Validadas por la comunidad

Expresivas

Comunican intención claramente

Fáciles de mantener

Simplifican el código

Muchos developers están familiarizados con los Patrones de diseño, así que podemos decir que es un tipo de estándar de desarrollo.

CHOUCAIR
Business Centric Testing

Patrones de diseño - Perspectivas

Estás totalmente perdido si piensas que...

- Debes usarlos solo porque la gente dice que es lo de hoy
- Son siempre la mejor opción sin importar la situación
- Si no aplicas Patrones de Diseño eres un mal developer

Estás en lo correcto si piensas que...

- Deberías usarlos cuando la situación lo requiere y la implementación mantiene tu código más simple
- Los Patrones de diseño son una solución general para problemas conocidos, pero se adaptan de acuerdo a contextos específicos
- Conoces de patrones de diseño y eres capaz de identificar cuando usarlos
- Los Patrones de Diseño ayudan a hacer el código más expresivo y menos complejo

Tomado de nearsoftjobs.com

Clasificación de los patrones de diseño

Patrones Creacionales

Proveen las técnicas para crear objetos. Algunos ejemplos son:

- Singleton (Instancia Única)
- Abstract Factory (Fábrica Abstracta)
- Builder Pattern (Constructor Virtual)

Patrones Estructurales

Describen la composición de los objetos y su organización. Ejemplos:

- Adapter Pattern (Adaptador)
- Decorator Pattern (Decorador)

Patrones de Comportamiento

Se enfocan en mejorar la comunicación entre los objetos en un sistema.

- Iterator Pattern (Iterador)
- Observer Pattern (Observador)
- Strategy Pattern (Estrategia)

Más detalle sobre patrones de diseño en JAVA: https://refactoring.guru/es/design-patterns/java

CHOUCAIR
Business Centric Testing

Modelos de desarrollo

TDD (Test Driven Development)

Esto significa escribir una prueba que falla porque la funcionalidad especificada no existe, entonces prosigo con escribir el código más simple que puede hacer pasar la prueba, a continuación, se realiza la refactorización para eliminar la duplicación y el ciclo se repite hasta que la funcionalidad esta lista.

BDD (Behaviour Driven Development)

Esto significa crear una especificación/requerimiento ejecutable que falla porque la característica no existe, entonces prosigo con escribir el código más simple que puede hacer pasar la especificación, a continuación, se realiza la refactorización para eliminar la duplicación y el ciclo se repite hasta que la especificación esta lista.

ATDD (Acceptance Test Driven Development)

Esto significa que todo el equipo discute en colaboración criterios de aceptación, con ejemplos, y luego los divide en un conjunto de pruebas de aceptación en concreto antes de que comience el desarrollo. ATDD es mas cercano a un proceso mas que una actividad.

Tomado de la página de nearsoftjobs.com

CHOUCAIR
Business Centric Testing

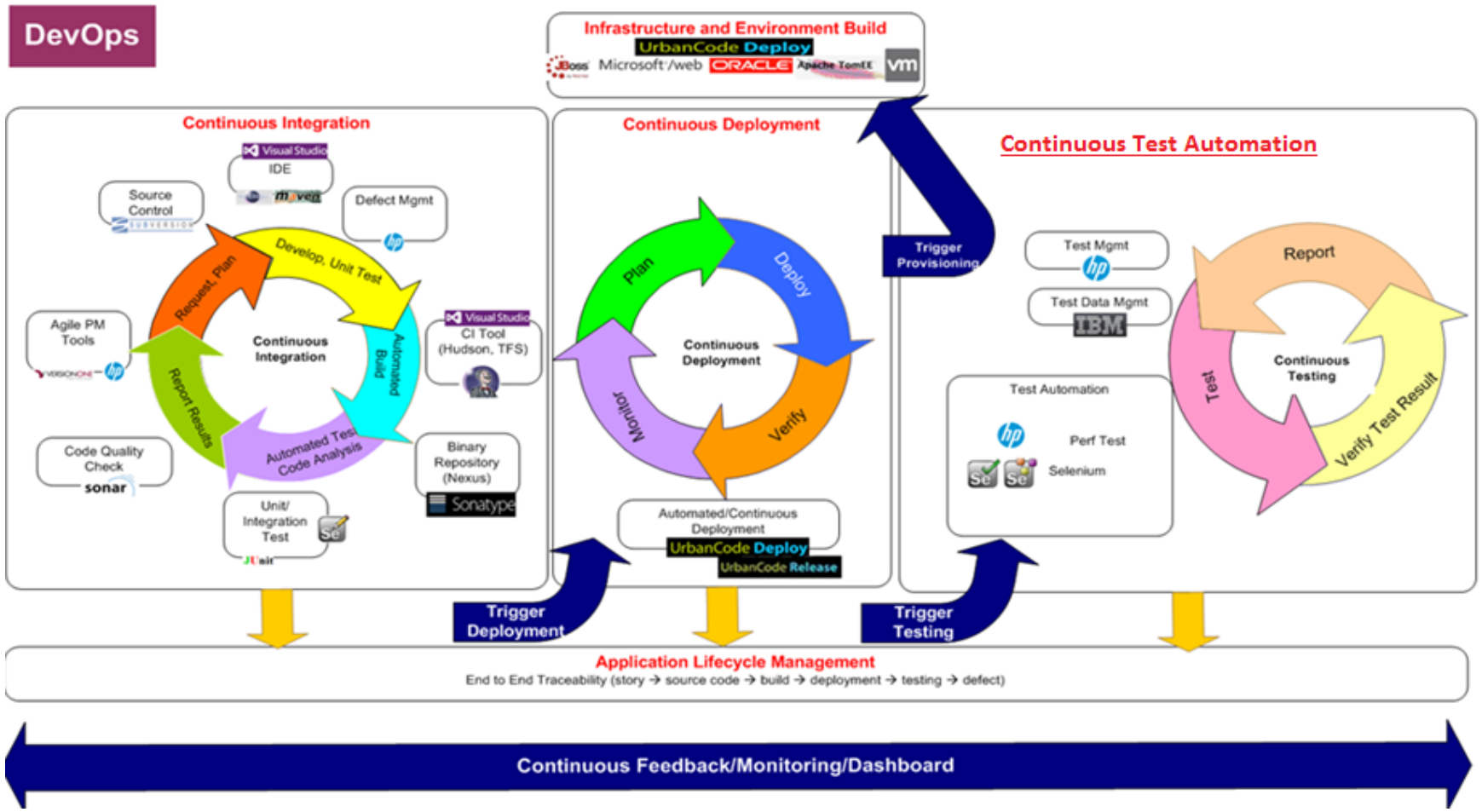
Modelos de desarrollo - Diferencias

Diferencia	ATDD	TDD	BDD
Colaboradores	Cliente	Desarrollador y Tester	ATDD + TDD
Objetivo	Participa los usuarios en la fase de diseño	Una práctica de desarrollo	Unit tester y negocio con un lenguaje común
Documento	Criterio de aceptación ejemplificativos	Documento de requisito	Documento escrito en lenguaje Gherkin
Automatización	No es necesario	Necesario	Necesario
Testing	Cada historia de usuario	Cada funcionalidad	Cada historia de usuario

Tomado de la página de vitaminac.github.io

CHOUCAIR
Business Centric Testing

DevOps



Tomado de secureservercdn.net

CHOUCAIR
Business Centric Testing

Gestores de proyectos

Sirven para construir proyectos y descargar las distintas dependencias que sean requeridas para el desarrollo.

Maven

Se basa en ficheros xml y viene con objetivos definidos para realizar ciertas tareas como la compilación del código y la generación del proyecto empaquetado, pero Maven también se encarga de obtener todas las dependencias del proyecto como de subirlo al repositorio final para que otros proyectos que dependan de él puedan usarlo.

Gradle

Esta herramienta se configura con ficheros DSL basados en Groovy, en vez de los xml que usaba Maven.

Tomado de la página de autentia.com

CHOUCAIR
Business Centric Testing

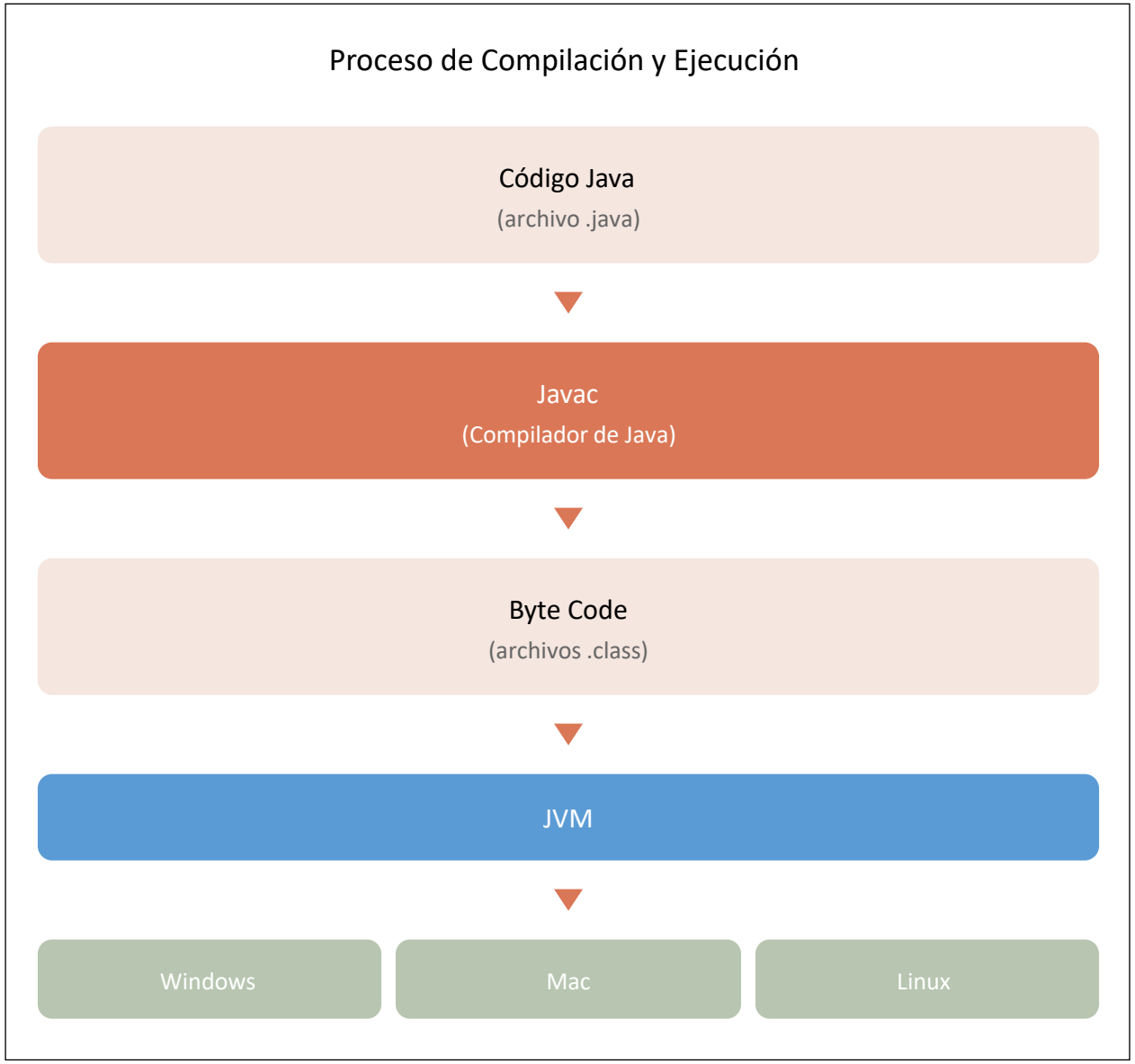
Entendamos qué es una máquina virtual

Concepto de Máquina Virtual

Entre el Byte Code (o el MSIL en el caso de .NET) y el sistema operativo se coloca un componente especial llamado Máquina Virtual que es el que realmente va a ejecutar el código.

Esta idea no tiene nada que ver con las máquinas virtuales a las que estamos acostumbrados hoy en día que ejecutan sistemas operativos completos, sino que es un concepto mucho más antiguo.

Tomado de la página de campusmvp.es



CHOUCAIR
Business Centric Testing

¿Qué es JDK y JRE?

JRE

Java Runtime Environment

Es la implementación de la Máquina virtual de Java que realmente ejecuta los programas de Java.

Normalmente el JRE está destinado a usuarios finales que no requieren el JDK, pues a diferencia de este no contiene los programas necesarios para crear aplicaciones en el lenguaje Java.

El JRE se puede instalar sin necesidad de instalar el JDK, pero al instalar el JDK, este siempre cuenta en su interior con el JRE.

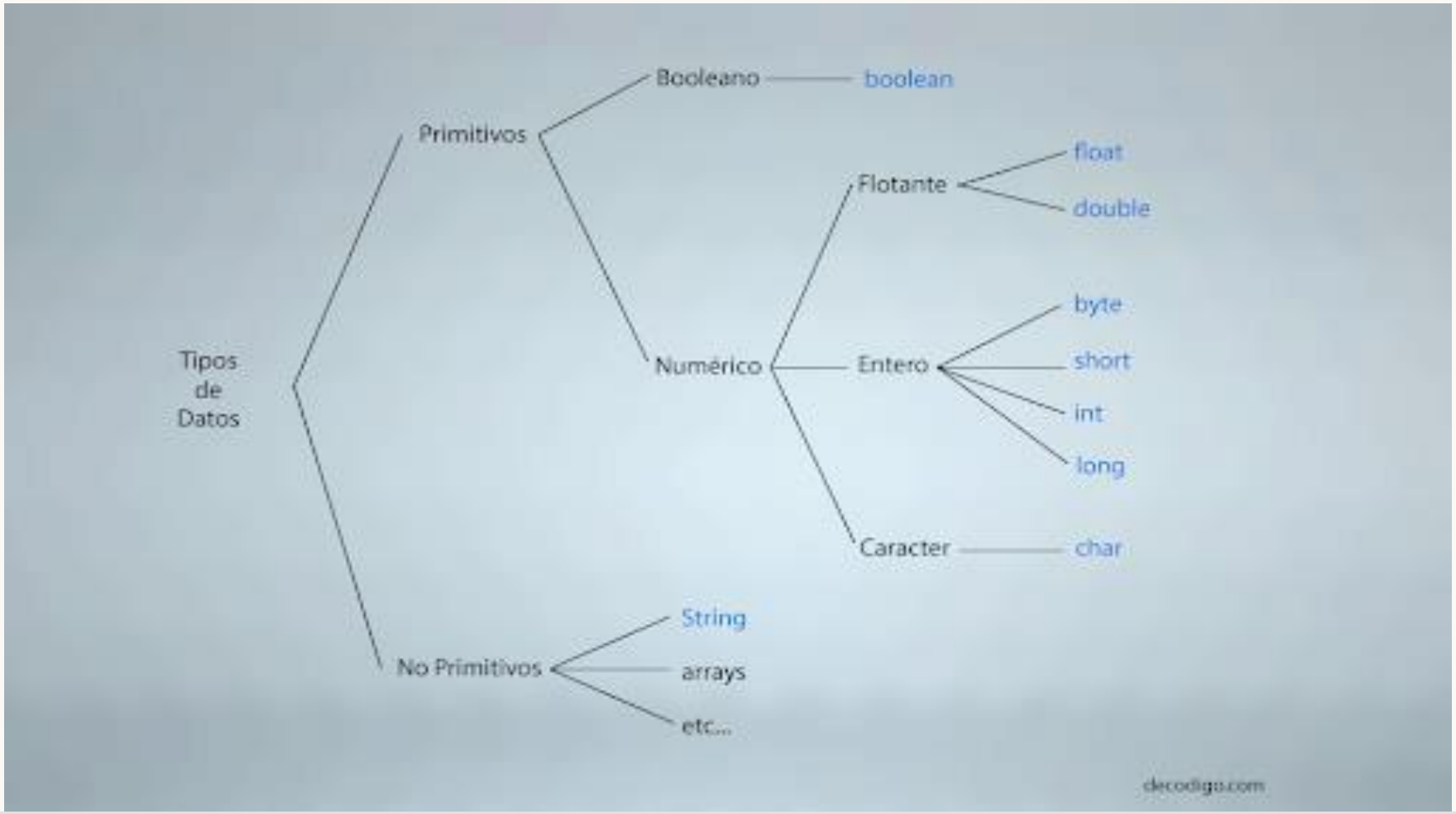
JDK

Java Development Kit

Es el compilador javac encargado de convertir nuestro código fuente (.java) en bytecode (.class).

El bytecode posteriormente será interpretado y ejecutado con la JVM para después ver los resultados de ejecución en los distintos sistemas operativos.

Tipos de datos en Java



CHOUCAIR
Business Centric Testing

Operadores en Java

Operador	Uso	Descripción
+	op1 + op2	Suma op1 y op2 (*)
-	op1 - op2	Resta op2 de op1
*	op1 * op2	Multiplica op1 y op2
/	op1 / op2	Divide op1 por op2
%	op1 % op2	Obtiene el resto de dividir op1 por op2
++	op ++	Incrementa op en 1; evalúa el valor antes de incrementar
++	++ op	Incrementa op en 1; evalúa el valor después de incrementar
--	op --	Decrementa op en 1; evalúa el valor antes de decrementar
--	-- op	Decrementa op en 1; evalúa el valor después de decrementar

Tomado de la página de manualweb.net

Operadores - Ejercicio

EJERCICIO PRÁCTICO

Cálculo del Volumen de una Caja

Elaborar un programa que permita determinar el volumen de una caja de dimensiones A, B y C.

Fórmula:

$$Volumen = A \times B \times C$$

Donde A, B y C representan las dimensiones (largo, ancho y alto) de la caja.

Tomado de la página de campusmvp.es

CHOUCAIR
Business Centric Testing

Manejo de control de flujos

if-then-else

```
if (expresion) {
  // Bloque then
} else {
  // Bloque else
}
```

switch

```
switch (expresion) {
  case valor1:
    bloque1;
    break;
  case valor2:
    bloque2;
    break;
  case valor3:
    bloque3;
    break;
  …
  default:
      bloque_por_defecto;
}
```

EJERCICIO PRÁCTICO

Cálculo del Volumen de una Caja

Elaborar un programa que permita determinar el volumen de una caja de dimensiones A, B y C, pero que además valide si las dimensiones son positivas. Si alguna dimensión es cero o negativa, el programa debe mostrar un mensaje indicando que los valores no son válidos. Si todas las dimensiones son correctas, calcular el volumen.

CHOUCAIR
Business Centric Testing

Manejo de control de flujos

Tabla de la verdad

P	Q	P && Q	P || Q	!P
True	True	True	True	False
True	False	False	True	False
False	True	False	True	True
False	False	False	False	True

Tomado de la página de campusmvp.es

CHOUCAIR
Business Centric Testing

Manejo de control de flujos

Ejercicio if

Ejercicio para los participantes…

Aplicando el control de flujo if, elabora un programa que muestre un mensaje de acuerdo a la edad de una persona:

- 0-10 años: niño
- 11-14 años: pre-adolescente
- 15-18 años: adolescente
- 19-25 años: joven
- 26-65 años: adulto
- >65 años: anciano

Tomado de campusmvp.es

CHOUCAIR
Business Centric Testing

Manejo de control de flujos - Ejercicio switch

EJERCICIO PROPUESTO

Control de flujo switch

Aplicando el control de flujo switch, realiza un programa que al ingresar un día en números del 1 al 7 muestre el día en texto.

Ejemplo:

- Entrada: 1→ Salida: "Lunes"
- Entrada: 2→ Salida: "Martes"
- Entrada: 7→ Salida: "Domingo"

Tomado de la página de campusmvp.es

CHOUCAIR
Business Centric Testing

¿Qué es equals?

Definición

Se encarga de comparar la cadena de texto contra un objeto. Devuelve true si las cadenas comparadas son iguales.

En caso contrario devolverá false.

```
String nombre = "Alejo";

if(nombre.equals("Alejo"))
{
    System.out.println("True");
}
```

CHOUCAIR
Business Centric Testing

Diferencia entre equals y "=="

En general, los operadores equals() y "==" en Java se usan para comparar objetos para verificar la igualdad, pero aquí hay algunas de las diferencias entre los dos:

1. La principal diferencia entre el método .equals() y el operador == es que uno aplica para objetos y variables y el otro sobre variables.

2. Se puede usar los operadores == para la comparación de referencias (comparación de direcciones) y el método .equals() para la comparación de contenido. En palabras simples, == comprueba si ambos objetos apuntan a la misma ubicación de memoria, mientras que equals() evalúa la comparación de valores en los objetos.

Tomado de la página de campusmvp.es

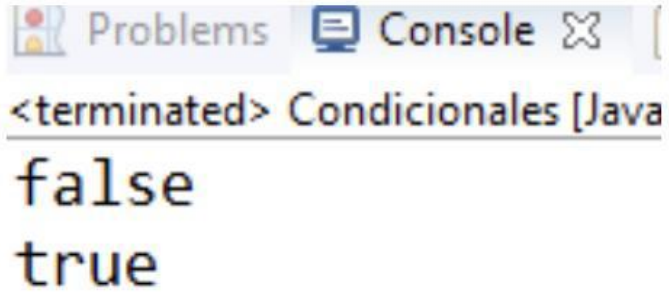
CHOUCAIR
Business Centric Testing

Diferencia entre equals y "=="

Ejemplo comparativo de código

```
String x = new String("Saludo");
String y = new String("Saludo");

System.out.println(x == y);
System.out.println(x.equals(y));
```



Bucles

ESTRUCTURA DE CONTROL

¿Qué son los Bucles?

Los bucles son una **estructura de control de total importancia** para el proceso de creación de un programa. Java y prácticamente todos los lenguajes más populares de la actualidad permiten hacer uso de estas estructuras.

Un ciclo en Java permite **repetir una o varias instrucciones cuantas veces sea necesario**, facilitando la ejecución de tareas repetitivas de manera eficiente.

Ejemplo Práctico

Si se desean escribir los números del **1 al 100**, no tendría sentido escribir cien líneas de código mostrando un número en cada una de estas. Para eso y para varias cosas más es útil un ciclo.

CHOUCAIR
Business Centric Testing

Bucles

TIPOS DE ESTRUCTURAS CÍCLICAS

Existen tres tipos principales de bucles en Java

Cada uno tiene una utilidad para casos específicos

1

Ciclos For

Estructura de control cíclica con número de iteraciones conocido

2

Ciclos While

Estructura iterativa controlada por una condición evaluada antes de ejecutar

3

Ciclos do While

Estructura que ejecuta al menos una vez antes de evaluar la condición

Bucles - Ciclo FOR

Definición

Son una estructura de control cíclica, que permite ejecutar una o varias líneas de código de forma iterativa (o repetitiva), pero teniendo control y conocimiento sobre las iteraciones.

Características del Ciclo For

- Es necesario tener un **valor inicial**
- Requiere una **condición** de control
- Debe tener un **valor final**
- Opcionalmente puede definir el **tamaño del paso** entre iteraciones

```
for(int i = 0; i < 10; i++)
{
    System.out.println("Número: "+i);
}
```

Tomado de campusmvp.es

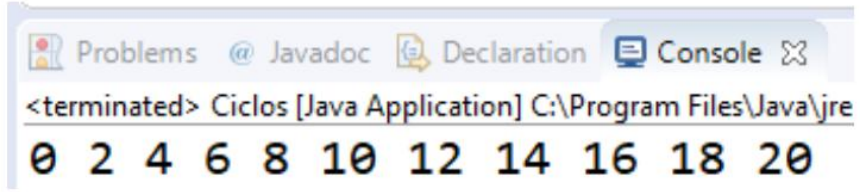
CHOUCAIR
Business Centric Testing

Ciclo FOR - Ejercicio guiado

Ejercicio

Desarrollar un programa que muestre los números pares entre 0 y 20 de forma ascendente.

```
for(int i = 0; i <= 20; i+=2)
{
    System.out.print(i+" ");
}
```



Tomado de la página de campusmvp.es

CHOUCAIR
Business Centric Testing

Ciclo FOR - Ejercicio propuesto

EJERCICIO PROPUESTO

Tabla de Multiplicar

Crear un programa que al proporcionar un valor muestre la tabla de multiplicar hasta el 10.

Ejemplo: Si el usuario ingresa el número 5, el programa debe mostrar:

```
5 x 1 = 5
5 x 2 = 10
...
5 x 10 = 50
```

Tomado de campusmvp.es

CHOUCAIR
Business Centric Testing

Bucles - Ciclo WHILE

Definición

Permite ejecutar una o varias líneas de código de forma iterativa (o repetitiva) al igual que el ciclo for, pero teniendo control y conocimiento sobre las iteraciones.

Características

- Control del ciclo por medio de una condicional en la declaración que determina si el ciclo continúa o se detiene
- Sintaxis más simple que el ciclo For
- Únicamente se tiene control del ciclo por medio de una condicional

Tomado de la página de campusmvp.es

```
boolean x = true;

while(x)
{
    System.out.println("Ciclo While");
    x = false;
}


    int i = 0;

    while(i < 10)
    {
        System.out.println(i);
        i++;
    }
```

CHOUCAIR
Business Centric Testing

Ciclo WHILE - Ejercicio guiado

Ejercicio

Desarrollar un programa que imprima los números impares entre 1 y 25

```
int i = 1;

while(i <= 25)
{
    if(i % 2 != 0)
    {
        System.out.print(i+" ");
    }
    i++;
}
```

Problems @ Javadoc Declaration Console

<terminated> Ciclos [Java Application] C:\Program Files\Java\jre1.8.0_171\bin\

```
1 3 5 7 9 11 13 15 17 19 21 23 25
```

Tomado de la página de campusmvp.es

Ciclo WHILE - Ejercicio propuesto

EJERCICIO PROPUESTO

Tabla de Multiplicar con WHILE

Crear un programa que al proporcionar un valor muestre la tabla de multiplicar hasta el 10 con el ciclo WHILE.

Requisitos:

- Solicitar al usuario un número
- Utilizar el ciclo WHILE para iterar del 1 al 10
- Mostrar el resultado de cada multiplicación

Tomado de campusmvp.es

CHOUCAIR
Business Centric Testing

Bucles - Ciclo DO WHILE

Definición

Esta estructura de iteración cumple el mismo objetivo de la estructura While con la variante que el ciclo do While ejecuta cuando menos una vez antes de evaluarse la condición del ciclo.

Característica Principal

Siempre se tendrá una iteración así el ciclo nunca haya entrado en ejecución, ya que la condición se evalúa al final del bloque.

Tomado de la página de campusmvp.es

```
do
{
    System.out.println("Do While");
}
while(false);
```


```
int i = 0;
do
{
    System.out.println(i);
    i++;
}
while(i > 5);
```

CHOUCAIR
Business Centric Testing

Ciclo DO WHILE - Ejercicio propuesto

EJERCICIO PROPUESTO

Tabla de Multiplicar con DO WHILE

Crear un programa que al proporcionar un valor muestre la tabla de multiplicar hasta el 10 con el ciclo DO WHILE.

Instrucciones:

- El programa debe solicitar un número al usuario
- Utilizar el ciclo DO WHILE para generar la tabla
- Mostrar los resultados desde 1 hasta 10

Tomado de la página de campusmvp.es

Paquetes en Java - package e import

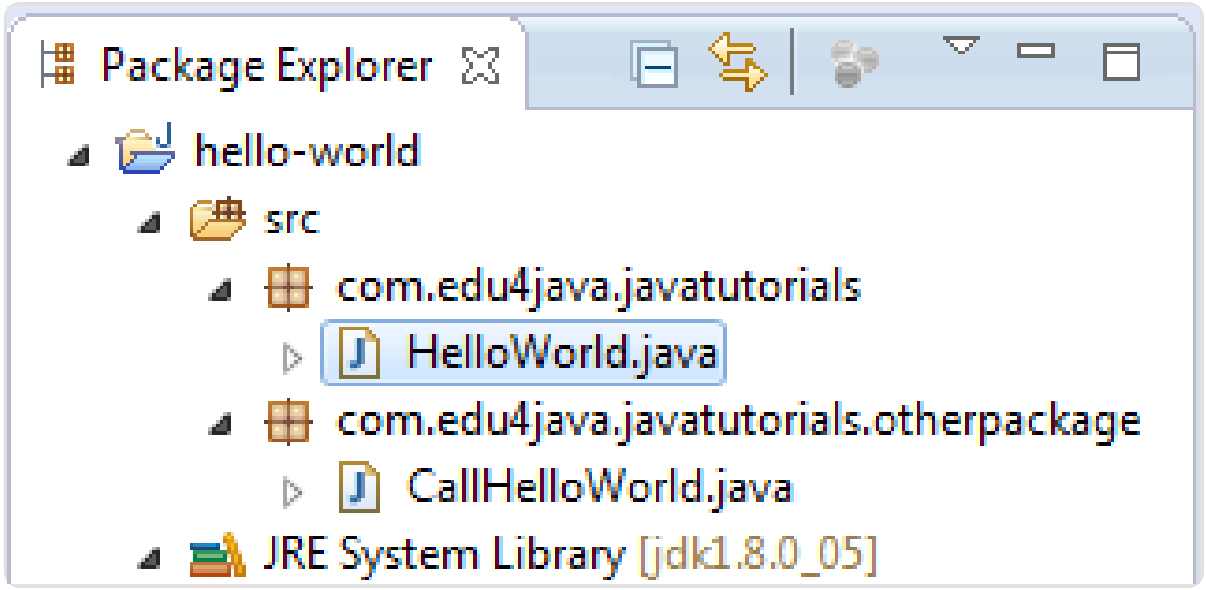
Convenciones de Java

Las convenciones del lenguaje Java recomiendan el uso de **paquetes (packages)** para la agrupación y organización de las clases, interfaces y anotaciones.

Espacios de Nombres

Estas divisiones se conocen como **espacios de nombres (namespaces)**, permitiendo una mejor estructura y evitando conflictos entre clases con el mismo nombre.

Beneficios: Organización modular, reutilización de código, control de acceso y mantenimiento simplificado.



CHOUCAIR
Business Centric Testing

Manejo de cadenas

Muchos Strings se crean a partir de cadenas literales. Cuando el compilador encuentra una serie de caracteres entre comillas (" y "), crea un objeto String cuyo valor es el propio texto. El esquema general es el siguiente: String nombre="cadena"; Cuando el compilador encuentra la siguiente cadena, crea un objeto String cuyo valor es Hola Mundo.

Longitud de cadenas:

```
String s = "abc";
System.out.println(s.length());
```

Devuelve: 3

Extracción de caracteres:

```
"abc".charAt(1)
```

Devuelve: 'b'

Métodos con subcadenas:

```
String s = "Víctor Cuervo";
s.substring(7);
```

Devuelve: "Cuervo"

Tomado de la página de campusmvp.es

CHOUCAIR®
Business Centric Testing

Muchas gracias