# Concurso:

# ¿Cuánto sabes de JAVA?

# Motivación:

- Para cambiar el ritmo de las jornadas y que no todas las actividades sean charlas
- Recordar conocimientos y aprender algo nuevo.
- Las preguntas pueden ayudarnos en futuros test psicotécnicos en una entrevista de trabajo.

# Reglas:

- Deben ponerse por grupos, hasta que haya 6-7 grupos en total.
- El test consistirá en 30 preguntas de JAVA. Habrá 10 preguntas más en caso de empate.
- Las preguntas irán aumentando poco a poco su complejidad.
- Los ganadores recibirán un pequeño premio

# Pregunta 1: JAVA es un lenguaje de programación orientado a

a) Orientado a eventos.

b) Orientado a objetos.

c) No orientado.

d) Orientado a aspectos.

# Pregunta 2: Cuál fue uno de los objetivos principales cuando fue diseñado

a) Que fuera sencillo para aprender.

b) Que tuviera más utilidades de programación a bajo nivel que C o C++.

c) Que permitiera ejecutarse en diferentes plataformas sin necesidad de recompilar.

d) Que fuera orientado a la programación de servicios web.

# Pregunta 3: Qué significan las iniciales JDK

a) Java Development Knowledge

b) Java Developer Knowledge

c) Java Developer Kit

d) Java Development Kit

# Pregunta 4: En qué se diferencian JRE de JDK

a) La JDK contiene un subconjunto de utilidades de la JRE.

b) La JRE contiene un subconjunto de utilidades de la JDK.

c) La JDK y la JRE son cosas completamente diferentes.

d) Tanto la JRE como la JDK contienen lo mismo pero se utilizan para diferentes plataformas.

# Pregunta 5: De qué tipo son los siguientes objetos

```
34

true

“Estamos en la pregunta…”

20.1
```

a) int, String, char, String, double

b) int, boolean, List, double

c) int, boolean, String, double

d) int, boolean, char[], double

# Pregunta 6: ¿Qué es una clase en JAVA?

a) Es un concepto similar al del array

b) Es un tipo particular de variable

c) Es un modelo o plantilla a partir de la cual creamos objetos

d) Es una categoría de datos ordenada secuencialmente

# Pregunta 7: Qué es una interface en JAVA

a) Define un contrato entre la clase que la implementa y el mundo exterior

b) Es un tipo JAVA que no se implementará en un objeto

c) Contiene unos métodos opcionales para las clases que la implementa

d) La primera y la tercera son ciertas

# Pregunta 8: ¿Qué elementos definen a un objeto en JAVA?

a) Su cardinalidad y su tipo

b) Sus atributos y métodos

c) La forma en que establece comunicación e intercambia mensajes

d) Su interfaz y sus elementos asociados

# Pregunta 9: ¿Qué código de los siguientes tiene que ver con herencia?

a) public class Componente extends Producto

b) public class Componente inherit Producto

c) public class Componente implements Producto

d) public class Componente belongs Producto

# Pregunta 10: ¿Qué es instanciar una clase?

a) Duplicar una clase

b) Eliminar una clase

c) Crear un objeto a partir de una clase

d) Conectar dos clases entre sí

# Pregunta 11: ¿Qué es JAVA Swing?

a) Una función utilizada para intercambiar valores

b) Es el sobrenombre de la versión 1.3 del JDK

c) Un framework de JAVA para Android

d) Una librería para construir interfaces gráficas

# Pregunta 12: ¿Qué es el bytecode en java?

a) El formato de intercambio de datos

b) El formato que obtenemos tras compilar un .java

c) Un tipo de variable

d) Un depurador de código

# Pregunta 13: ¿Qué significa sobrecargar (overload) un método?

a) Editarlo para modificar su comportamiento

b) Cambiarle el nombre dejándolo con la misma funcionalidad

c) Crear un método con el mismo nombre pero con diferentes argumentos

d) Añadirle funcionalidades a un método

# Pregunta 14: ¿Qué significa la palabra reservada “static”, y dónde se utiliza?

a) Las variables static son compartidas por la clase, no por una instancia concreta

b) La palabra reservada static se pueden usar para variables y métodos

c) Los métodos static no se pueden sobrecargar (overload)

d) Todas las anteriores

# Pregunta 15: ¿Qué hace la palabra reservada synchronized?

a) Comunica ese bloque de código con todos los hilos que se estén ejecutando

b) Sirve para hacer secuencial la ejecución de un método o trozo de código

c) Permite ejecutar ese método o trozo de código en paralelo con otras instancias

d) Usado en constructores, permite la instanciación de objetos segura

# Pregunta 16: ¿El borrado de tipos es un fenómeno de la JVM que…?

a) significa que en tiempo de ejecución no se tiene conocimiento de los tipos de objetos genéricos como List<Integer>

b) permite borrar en tiempo de ejecución determinados instancias de objetos para hacer código eficiente

c) las dos anteriores

d) permite que los métodos genéricos realicen las asignaciones a los tipos de los objetos instanciados

# Pregunta 17: ¿Cuál es la diferencia entre una interfaz y una clase abstracta?

a) En la interfaz no se pueden crear métodos abstractos y en una clase abstracta sí

b) Los métodos de la interfaz no pueden tener body, pero los métodos abstractos de una clase abstracta sí

c) Las clases abstractas pueden contener variables o métodos privados

d) Las interfaces no pueden ser instanciadas pero las clases abstractas sí

# Pregunta 18: ¿Cuál es el objetivo de un patrón de diseño en JAVA?

a) Proporcionar catálogos de elementos reusables en el diseño de sistemas software

b) Evitar la reiteración en la realización de soluciones frente al mismo problema

c) Imponer ciertas alternativas de diseño frente a otras

d) Eliminar la creatividad inherente al proceso de diseño.

# Pregunta 19: Según el GOF (Gang Of Four), ¿En qué categorías se dividen los patrones?

a) Patrones de frontend, de servicio y de backend

b) Patrones de interacción, de business delegate y de service locator

c) Patrones creacionales, estructurales y de comportamiento

d) Patrones de cliente, de presentación, de negocios, de integración y de recursos

# Pregunta 20: ¿Qué es una fachada (facade)?

a) Es un patrón de diseño, permite generar vistas de forma automática

b) Es un tipo de objeto JAVA que permite ordenar las clases por usabilidad

c) Es la vista base que une la cabecera, el menú y el footer para todas las vistas

d) Es un patrón de diseño que nos permite encapsular la comunicación entre dos objetos para facilitarla

# Pregunta 21: Sobre una interfaz

a) Puede contener métodos private, protected y public

b) El único modificador de método que puede contener es public

c) Si no se especifican, sus métodos son protected por defecto

d) Pueden contener métodos static a partir de Java 8

# Pregunta 22: Sobre una interfaz

a) Sus métodos no pueden tener body

b) Sus métodos pueden siempre pueden tener body

c) Sus métodos sólo pueden tener body si tienen el modificador static o default

d) Sus métodos sólo pueden tener body si se les aplica el modificador static

# Pregunta 23: Puede una interfaz tener el modificador abstract

a) Puede tenerlo pero no aporta nada porque es implícito

b) No puede tenerlo y da error de compilación

c) Puede tenerlo, pero sus métodos no podrán tener body

d) Sólo se puede poner el modificador abstract a partir de JAVA 8

# Pregunta 24: Se le puede poner abstract a un método de una interfaz

a) No, da un error de compilación

b) Sí, pero no aporta nada

c) Sólo se pueden utilizar si la interfaz tiene el modificador abstract también

d) Sí, y permite añadir un jerarquía a los métodos

# Pregunta 25: Un método abstracto…

a) Debe pertenecer a una interfaz o clase abstracta

b) Puede tener body

c) Es inmutable

d) No se puede sobrecargar

# Pregunta 26: En una clase abstracta…

a) Un método abstracto puede ser private

b) No se pueden crear métodos no abstractos

c) Los métodos no abstractos pueden ser private

d) Las variables de la clase solo pueden ser public o protected

# Pregunta 27: Si la clase Worker extiende a Person, podemos…

Código

```
Worker worker = new Worker();

Person person = new Person();
```

a) `person = worker;`

b) `worker = person;`

c) Ambas son correctas

d) Ninguna es correcta, necesitas hacer un casteo del estilo: `persona=(Trabajador)trabajador;`

# Pregunta 28: ¿Se puede crear una interfaz dentro de una clase?

a) No, las interfaces no se pueden crear dentro de una clase

b) Si, pero sólo se pueden crear private

c) Si, pero sólo si son publics

d) Si, en todos los casos (public, protected o private)

# Pregunta 29: ¿Se puede crear una clase dentro de otra clase?

a) No, las clases no se pueden crear dentro de otra clase

b) Si, pero sólo se pueden crear private

c) Si, pero sólo si son publics

d) Si, en todos los casos (public, protected, private…)

# Pregunta 30: ¿Se pueden crear una clase A y una interfaz B dentro de una clase C, y que A implemente a B?

a) No, se pueden crear A y B, pero A no puede implementar a B

b) No, porque no se puede crear la clase A

c) No, porque no se puede crear la clase B

d) Si, llegados a este punto me creo cualquier cosa

# Pregunta 31: De qué forma podemos declarar un Array de char en java

a) `char[] myCharArray = new char[3];`

b) `char[] myCharArray = {‘a’, ‘b’, ‘c’};`

c) `char[] myCharArray = new char[]{‘a’, ‘b’, ‘c’};`

d) De todas las formas anteriores

# Pregunta 32: Qué ocurre en el siguiente caso

**Código:**

```
char [] charArray = new char [10];

charArray[3] = 'a';

System.out.println(charArray);
```

a) Nos da la excepción "IndexOutOfBoundsException"

b) Funciona pero se muestran los espacios vacíos por consola

c) Nos da un fallo de compilación

d) Lo ignora y solo se muestra la 'a'

# Pregunta 33: Qué ocurre en el siguiente caso

```
public static void main(String[] args) {
i = 3;
Integer j = Integer.valueOf(2);
multiplica(i, j);
System.out.println(i+j);}

public static void multiplica(int i, Integer j) {
   i *= 10;j *= 10;}
```

a) Por consola se muestra el valor 5

b) Por consola se muestra el valor 50

c) Por consola se muestra el valor 32

d) Por consola se muestra el valor 23

# Pregunta 34: Qué ocurre en el siguiente caso

```
public static void
main(String[] args) {
int x = 5;
int y = 5;
y *= x++;
System.out.println(y);
}
```

a) Da fallo de compilación

b) Se muestra 25 por consola

c) Se muestra 30 por consola

d) Se muestra 5 por consola

# Pregunta 35: Qué ocurre en el siguiente caso

```
public static void
main(String[] args) {
    System.out.println(5&6);
}
```

a) Se muestra false por consola

b) Se muestra true por consola

c) Da error de compilación

d) Se muestra 4 por consola

# Pregunta 36: Qué ocurre en el siguiente caso

```
public static void
main(String[] args) {
System.out.println(5>>1);

}
```

a) Da error de compilación

b) La consola muestra 6

c) La consola muestra true

d) La consola muestra 2

# Pregunta 37: Qué ocurre en el siguiente caso

```
public static void
main(String[] args) {
Integer i8 =
Integer.valueOf(5);
Integer i1 = 5;
System.out.println(i1 ==
i8);
}
```

a) Da error de compilación

b) La consola muestra true

c) La consola muestra false

d) La consola muestra 5

# Pregunta 38: Qué ocurre en el siguiente caso

```
public static void
main(String[] args) {
Integer i8 = Integer.valueOf(5);
Integer i9 = Integer.valueOf("5");
System.out.println(i9 == i8);

}
```

a) Da error de compilación

b) La consola muestra true

c) La consola muestra false

d) La consola muestra 5

# Pregunta 39: Qué ocurre en el siguiente caso

```
public static void
main(String[] args) {
Integer i4 = 5787878;
Integer i5 = 5787878;
System.out.println(i4 == i5);
}
```

a) Da error de compilación

b) La consola muestra true

c) La consola muestra false

d) La consola muestra 5787878

# Pregunta 40: Qué ocurre en el siguiente caso

```
public static void
main(String[] args) {
Integer i2 = 5;
Integer i3 = new Integer(5);
System.out.println(i2 == i3);
}
```

a) Da error de compilación

b) La consola muestra true

c) La consola muestra false

d) La consola muestra 5