

## NOMBRE Y PREGUNTA A RESPONDER

### 5 respuestas por persona

ERICK: 1-5

DAVID: 6-10

EMANUEL: 11-17,25

Hector 32-36

ALICIA: 58-64 (la mitad de la 62)

36-41

MACHI 42-54

55-76

:v

LA 21 ES INSTALAR SUBLIME TEXT 3

56 es hacer varios tutoriales

77-83 SON PERSONALES

La cincuenta y tantos es un manual de instalación de JDK xd

PARA LA PREGUNTA \* HAY QUE HACER EL CURSO:

<https://www.sololearn.com/Play/Java/#>

### 1. ¿Qué es un algoritmo y cuáles son sus características?

Un algoritmo es una secuencia de pasos lógicos y ordenados con las cuales le damos solución a un problema determinado.

Preciso: cada instrucción tiene que ser clara y determinada a una acción

Definido: debe obtenerse el mismo resultado siempre con las mismas condiciones de entrada

Finito: su diseño debe tener un número limitado de pasos

Ordenado: tienen una secuencia de pasos

Fuente: <https://prezi.com/sf8qawzyst9j/algoritmo-computacional-y-no-computacional/>

### 2. ¿Qué es un diagrama de flujo?

Un diagrama de flujo es una representación gráfica que puede describir la estructura lógica de un algoritmo (proceso)

<https://www.lucidchart.com/pages/es/qu%C3%A9-es-un-diagrama-de-flujo>






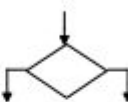
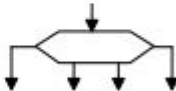

### 3. ¿Qué es un programa?



Un programa es un conjunto específico de instrucciones ordenadas para que una computadora las ejecute y realice una o varias tareas.

/\*Dos tipos básicos de programas informáticos son (1) un sistema operativo, que proporciona las instrucciones más fundamentales para las aplicaciones informáticas en sus operaciones, y (2) un programa de aplicación, que se ejecuta en el sistema operativo y realiza un trabajo específico \*/

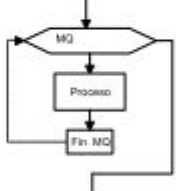
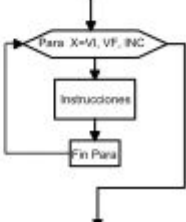
<http://searchsoftwarequality.techtarget.com/definition/program>

### 4. Realiza una tabla explicando la simbología de los diagramas de flujo.

	Símbolo	Función
Inicio/fin		Delimita el inicio y fin de un dfd.
Proceso		Cualquier tipo de operación que origine cambio de valor, asignación, operaciones aritméticas, etc.
Entrada		Cualquier tipo de de introducción de datos. Expresa lectura.
Salida		Para representar la impresión de un resultado. Expresa escritura.
Subrutina		Llamada a subrutina, función o procedimiento. Este es un modulo independiente del programa principal, que es llamado desde dicho programa, realiza una tarea determinada y regresa al terminar su tarea al programa principal.
Decisión		Indica operaciones lógicas o de comparación entre datos. En función del resultado obtenido, determina qué camino se debe seguir. Tiene dos salidas, respuesta <i>sí</i> o respuesta <i>no</i> . En algunas ocasiones la alternativa <i>no</i> se omite.
Selección múltiple		Para seleccionar una opción de varias predeterminadas. En su interior se almacena un selector que dependiendo del valor de una variable sigue por una de las ramas o caminos alternativos.
Flujo		Indican el flujo de datos en el dfd.

	Símbolo	Función
Conector		Sirve para enlazar dos partes cualesquiera de un dfd a través de un conector de salida y otro en la entrada. Se utiliza en la misma página del dfd.
Conector		Conexión entre dos puntos del dfd situados en páginas diferentes.

Se añaden los siguientes símbolos que, aunque nos son estándar, ayudarán a llevar a cabo una transición más sencilla entre los diagramas de flujo y los programas, dada la similitud que ofrecen con algunas instrucciones de programación.

	Símbolo	Función
Repetición "Mientras Que"		Sirve para repetir una operación o conjunto de operaciones <i>mientras</i> que la condición que se establezca sea verdadera.
Repetición "Para"		Sirve para repetir una operación o conjunto de operaciones un número determinado de veces. Se debe establecer un valor inicial, un valor final y un incremento.

<https://app.schoolology.com/course/1423463477/materials/gp/1423464111>  
es

## 5. ¿Qué es un lenguaje de programación?

es un modo práctico donde los seres humanos puedan dar instrucciones a un equipo para instruir a una computadora o dispositivo informático para realizar tareas específicas

[https://www.webopedia.com/TERM/P/programming\\_language.html](https://www.webopedia.com/TERM/P/programming_language.html)

## 6. ¿Qué es el lenguaje máquina?

El lenguaje de máquina o código máquina es el sistema de códigos directamente interpretable por un circuito microprogramable, como el microprocesador de una computadora o el microcontrolador de un autómata. Este lenguaje está compuesto por un conjunto de instrucciones que determinan acciones a ser tomadas por la máquina.

## **7. ¿Qué es el lenguaje ensamblador?**

Es un lenguaje de programación de bajo nivel. Consiste en un conjunto de mnemónicos que representan instrucciones básicas para los computadores, microprocesadores, microcontroladores y otros circuitos integrados programables. Implementa una representación simbólica de los códigos de máquina binarios y otras constantes necesarias para programar una arquitectura de procesador y constituye la representación más directa del código máquina específico para cada arquitectura legible por un programador.

## **8. ¿Qué es un lenguaje de alto nivel?**

se caracteriza por expresar los algoritmos de una manera adecuada a la capacidad cognitiva humana, en lugar de la capacidad que se la ejecuta de las máquinas.

## **9. ¿Qué es un paradigma de programación?**

Un paradigma de programación es una propuesta tecnológica adoptada por una comunidad de programadores y desarrolladores cuyo núcleo central es incuestionable en cuanto que únicamente trata de resolver uno o varios problemas claramente delimitados; la resolución de estos problemas debe suponer consecuentemente un avance significativo en al menos un parámetro que afecte a la ingeniería de software.

## **10. ¿Cuál es la clasificación de los lenguajes de alto nivel? Da ejemplos de cada clasificación**

Lenguajes de programación imperativos: entre ellos tenemos el Cobol, Pascal, C y Ada.

Lenguajes de programación declarativos: el Lisp y el Prolog.

Lenguajes de programación orientados a objetos: el Smalltalk y el C++.

Lenguajes de programación orientados al problema: son aquellos lenguajes específicos para gestión.

Lenguajes de programación naturales: son los nuevos lenguajes que pretender aproximar el diseño y la construcción de programas al lenguaje de las personas.

## 11. Explica:

### 1. Paradigma orientado a objetos.

El paradigma orientado a objetos (OO) define los programas en términos de comunidades de objetos. Los objetos con características comunes se agrupan en clases (un concepto similar al de tipo abstracto de dato (TAD)). Los objetos son entidades que combinan un estado (es decir, datos) y un comportamiento (esto es, procedimientos o métodos). Estos objetos se comunican entre ellos para realizar tareas.

### 2. Paradigma orientado a eventos.

La **programación dirigida por eventos** es un paradigma de programación en el que el flujo del programa está determinado por **eventos** o mensajes desde otros programas o hilos de ejecución.

Las aplicaciones desarrolladas con **programación dirigida por eventos** implementan un **bucle principal** o **main loop** donde se ejecutan las dos secciones principales de la aplicación: El selector de eventos y el manejador de eventos.

### 3. Paradigma funcional.

La programación funcional es un paradigma de programación declarativa basado en el uso de funciones matemáticas. La programación funcional tiene sus raíces en el cálculo lambda, un sistema formal desarrollado en los años 1930 para investigar la definición de función, la aplicación de las funciones y la recursión. El objetivo es conseguir lenguajes expresivos y *matemáticamente elegantes*, en los que no sea necesario bajar al nivel de la máquina para describir el proceso llevado a cabo por el programa, y evitar el concepto de *estado* del cómputo.

## 12. Menciona 5 lenguajes que usen el paradigma orientados a objetos

- JAVA
- javascript
- Python
- Perl
- C#

## 13. Menciona 3 lenguajes que soporten el paradigma funcional

- Haskell
- Miranda
- Scala

#### **14. ¿Qué es un compilador?**

El compilador es un programa que recibe como datos de entrada el código fuente de un programa escrito por un programador, y genera como salida un conjunto de instrucciones escritas en el lenguaje binario de la computadora donde se van a ejecutar.

#### **15. ¿Qué es un intérprete?**

Es un programa informático capaz de analizar y ejecutar otros programas. Los intérpretes se diferencian de los compiladores o de los ensambladores en que mientras estos traducen un programa desde su descripción en un lenguaje de programación al código de máquina del sistema, los intérpretes sólo realizan la traducción a medida que sea necesaria, típicamente, instrucción por instrucción, y normalmente no guardan el resultado de dicha traducción.

#### **16. ¿Qué es el código de bytes?**

Los Bytecodes o códigos de bytes son un conjunto de instrucciones muy parecidas al código máquina, pero que no son específicas para algún procesador.

<http://profejavaoramas.blogspot.mx/2010/04/codigos-de-bytes.html>

#### **17. ¿Qué es un IDE y menciona 3 que podemos usar para Java?**

Un entorno de desarrollo integrado o entorno de desarrollo interactivo, en inglés Integrated Development Environment (IDE), es una aplicación informática que proporciona servicios integrales para facilitarle al desarrollador o programador el desarrollo de software.

- NetBeans
- Eclipse
- IntelliJ IDEA

#### **18. ¿Qué es un Framework?**

**Un framework, entorno de trabajo[1] o marco de trabajo[2] es un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular que sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar.**

**un esquema (un esqueleto, un patrón) para el desarrollo y/o la implementación de una aplicación.**

## 19. Menciona 3 Frameworks de Java

//Dejo más para que escojan. Abajo está el enlace por si alguien quiere leer o escribir las descripciones

Spring MVC, Struts 2, Hibernate, Vaadin, Goggle Web Toolkit, Grails

<https://openwebinars.net/blog/los-7-mejores-frameworks-de-java-de-2017/>

## 20. ¿Qué es Sublime Text?

Sublime Text es un editor de código multiplataforma, ligero y sencillo. Su sistema de resaltado de sintaxis soporta varios lenguajes (C, C++, C#, CSS, D, Erlang, HTML, Groovy, Haskell, HTML, Java, JavaScript, LaTeX, Lisp, Lua, Markdown, Matlab, OCaml, Perl, PHP, Python, R, Ruby, SQL, TCL, Textile and XML). Soporta macros, *Snippets* y auto completar, entre otras funcionalidades. Cuenta con 22 combinaciones de color; se pueden conseguir más. Cuenta con *Minimap*, un panel que permite moverse por el código de forma rápida.

<https://www.genbeta.com/herramientas/sublime-text-un-sofisticado-editor-de-codigo-multiplataforma>

## 21. Si no lo tienes, instala Sublime Text 3 en tu computadora.

Yo no lo descargo porque ya lo tengo xd

<https://www.sublimetext.com/3>

## 22. ¿Qué es Java?

Java es un lenguaje de programación de propósito general orientado a objetos desarrollado por Sun Microsystems. También se puede decir que Java es una tecnología que no sólo se reduce al lenguaje sino que además provee de una máquina virtual Java que permite ejecutar código compilado Java, sea cual sea tanto hardware como software (sistema operativo) que haya debajo.

<http://personales.upv.es/rmartin/cursoJava/Java/Introduccion/PrincipalesCaracteristicas.htm>

## 23. Escribe la historia de Java (mínimo media cuartilla).

Java actualmente puede ser considerado uno de los lenguajes más extendido y usado del mundo, pero esto no siempre fue así. ¿Cual es la historia de Java?

Para hablar de la historia de java, primero debemos remontarnos a los años 80, donde C podía considerarse el lenguaje por antonomasia. Era un lenguaje versátil, que podía actuar a bajo nivel y resolvía problemas muy complejos. Era la cima de la programación estructurada, para resolver estos complejos algoritmos, se generaban grandes procedimientos con un código muy complicado de mantener a largo plazo. Por ello empezó a surgir como alternativa la programación orientada a objetos, y con ella nació C++. Entonces, ¿por qué surgió Java?

## Inicios de JAVA

Java nace en 1991 con el nombre "OAK", posteriormente cambiado por Green por problemas legales, y finalmente con la denominación actual JAVA.

El objetivo de java era crear un lenguaje de programación parecido a C++ en estructura y sintaxis, fuertemente orientado a objetos, pero con una máquina virtual propia. Esto se hizo bajo el principio, de poder ser usado bajo cualquier arquitectura "Write Once, Run Anywhere (escríbelo una vez, ejecútalo en cualquier sitio)".

En 1992 se presenta el proyecto verde, con los prototipos a bajo nivel. Entre 1993 y 1994 se trabaja para poder presentar un prototipo funcional (hotJava) donde se ve todo el potencial que JAVA puede ofrecer.

## Establecimiento

En 1995 finalmente, es presentada la versión alpha de java, y un año despues en 1996 es lanzado el primer JDK (JDK 1.0). El desarrollo de java a partir de entonces es imparable, se van presentando nuevos paquetes y librerías hasta la actualidad.

## Java hoy en día

A día de hoy, podemos decir, que Java es uno de los lenguajes más importantes del mundo. Con una comunidad extendida en todos los componentes y más de 9 millones de desarrolladores, existen millones de dispositivos que lo usan. Además, tras el surgimiento de android, java se establecido como el lenguaje de programación para móviles más extendido del planeta.

## Curiosidades de JAVA

El nombre de JAVA tiene una gran polémica con respecto a su origen, se discuten varias teorías:

- Una de las teorías más difundidas viene del café, se dice que una cafetería cercana donde los desarrolladores tomaban café se llamaba de la misma forma, de ahí vendría también su logo con una taza de café humeante.
- Otra versión dice que viene del acrónimo: Just Another Vague Acronym ("sólo otro acrónimo ambiguo más").
- Aparte de estas teorías, la versión más plausible es que se eligió su denominación al azar de una lista de posibles nombres.

<http://www.tuprogramacion.com/programacion/historia-de-java/>

## 24. Menciona 5 características del lenguaje Java (Da una descripción de ellas).

//Dejo varias para que escojan las que prefieran y para tener variedad cuando nos pregunte

- **Lenguaje totalmente orientado a Objetos.** Todos los conceptos en los que se apoya esta técnica, encapsulación, herencia, polimorfismo, etc., están presentes en Java.
- **Disponibilidad de un amplio conjunto de bibliotecas.** Como ya se mencionó anteriormente, Java es algo más que un lenguaje. La programación de aplicaciones con Java se basa no solo en el empleo del juego de



instrucciones que componen el lenguaje, sino, fundamentalmente, en la posibilidad de utilizar el amplísimo conjunto de clases que Sun pone a disposición del programador y con las cuales es posible realizar prácticamente cualquier tipo de aplicación.

- **Lenguaje simple.** Java posee una curva de aprendizaje muy rápida. Resulta relativamente sencillo escribir applets interesantes desde el principio. Todos aquellos familiarizados con C++ encontrarán que Java es más sencillo, ya que se han eliminado ciertas características, como los punteros. Debido a su semejanza con C y C++, y dado que la mayoría de la gente los conoce aunque sea de forma elemental, resulta muy fácil aprender Java. Los programadores experimentados en C++ pueden migrar muy rápidamente a Java y ser productivos en poco tiempo.
- **Distribuido.** Java proporciona una colección de clases para su uso en aplicaciones de red, que permiten abrir sockets y establecer y aceptar conexiones con servidores o clientes remotos, facilitando así la creación de aplicaciones distribuidas.
- **Interpretado y compilado a la vez.** Java es compilado, en la medida en que su código fuente se transforma en una especie de código máquina, los bytecodes, semejantes a las instrucciones de ensamblador. Por otra parte, es interpretado, ya que los bytecodes se pueden ejecutar directamente sobre cualquier máquina a la cual se hayan portado el intérprete y el sistema de ejecución en tiempo real (run-time).
- **Robusto.** Java fue diseñado para crear software altamente fiable. Para ello proporciona numerosas comprobaciones en compilación y en tiempo de ejecución. Sus características de memoria liberan a los programadores de una familia entera de errores (la aritmética de punteros), ya que se ha prescindido por completo de los punteros, y la recolección de basura elimina la necesidad de liberación explícita de memoria.
- **Seguro (?).** Dada la naturaleza distribuida de Java, donde las applets se bajan desde cualquier punto de la Red, la seguridad se impuso como una necesidad de vital importancia. A nadie le gustaría ejecutar en su ordenador programas con acceso total a su sistema, procedentes de fuentes desconocidas. Así que se implementaron barreras de seguridad en el lenguaje y en el sistema de ejecución en tiempo real.
- **Indiferente a la arquitectura.** Java está diseñado para soportar aplicaciones que serán ejecutadas en los más variados entornos de red, desde Unix a Windows Nt, pasando por Mac y estaciones de trabajo, sobre arquitecturas distintas y con sistemas operativos diversos. Para acomodar requisitos de ejecución tan diversos o variopintos, el compilador de Java genera bytecodes: un formato intermedio indiferente a la arquitectura diseñado para transportar el código eficientemente a múltiples plataformas hardware y software. El resto de problemas los soluciona el intérprete de Java.

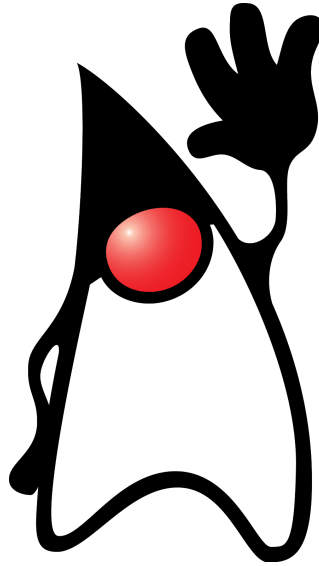
- **Portable.** La indiferencia a la arquitectura representa sólo una parte de su portabilidad. Además, Java especifica los tamaños de sus tipos de datos básicos y el comportamiento de sus operadores aritméticos, de manera que los programas son iguales en todas las plataformas. Estas dos últimas características se conocen como la Máquina Virtual Java (JVM).
- **Alto rendimiento.**
- **Multihebra.** Hoy en día ya se ven como terriblemente limitadas las aplicaciones que sólo pueden ejecutar una acción a la vez. Java soporta sincronización de múltiples hilos de ejecución (multithreading) a nivel de lenguaje, especialmente útiles en la creación de aplicaciones de red distribuidas. Así, mientras un hilo se encarga de la comunicación, otro puede interactuar con el usuario mientras otro presenta una animación en pantalla y otro realiza cálculos.
- **Dinámico.** El lenguaje Java y su sistema de ejecución en tiempo real son dinámicos en la fase de enlazado. Las clases sólo se enlazan a medida que son necesitadas. Se pueden enlazar nuevos módulos de código bajo demanda, procedente de fuentes muy variadas, incluso desde la Red.
- **Produce applets.** Java puede ser usado para crear dos tipos de programas: aplicaciones independientes y applets. Las aplicaciones independientes se comportan como cualquier otro programa escrito en cualquier lenguaje, como por ejemplo el navegador de Web HotJava, escrito íntegramente en Java. Por su parte, las applets son pequeños programas que aparecen embebidos en las páginas Web, como aparecen los gráficos o el texto, pero con la capacidad de ejecutar acciones muy complejas, como animar imágenes, establecer conexiones de red, presentar menús y cuadros de diálogo para luego emprender acciones, etc.

[https://es.wikibooks.org/wiki/Programaci%C3%B3n\\_en\\_Java/Caracter%C3%ADsticas\\_del\\_lenguaje](https://es.wikibooks.org/wiki/Programaci%C3%B3n_en_Java/Caracter%C3%ADsticas_del_lenguaje)

## 25. ¿Cómo se llama la mascota de Java?

Realiza un dibujo de ésta

Duke



## 26. ¿Qué es el API de Java?

Serie de clases y bibliotecas incluidas en Java Development Environment; estas clases están escritas en Java y son la base de un programa escrito en Java.

## 27. ¿Por qué aprender Java?

Por ser un lenguaje muy popular (lenguaje de programación más usado en el mundo) en todos los ámbitos: móviles, sitios web y buscadores, etc. Hay una gran documentación, es orientado a objetos (POO), es open source y cuenta con una comunidad de 10 millones de personas. Todas las apps de Android están escritas en Java y el 90% de las empresas de Fortune 500 usan servidores basados en Java para el back end.

## 28. ¿Qué son los Applets en Java?

Programas escritos en Java que cualquier navegador con Java incluido, puede correr.

## 29. ¿Qué es y sus características?

### 1. JDK

Java Development Kit: implementación de Java Platform (Standard, Enterprise, Micro Edition) para desarrolladores.

### 2. JVM

Java Virtual Machine: máquina virtual que permite a una computadora correr un programa en Java.

### 3. JRE

Java Runtime Environment: paquete de software que contiene lo necesario para que un programa en Java pueda correr.

### 30. ¿Qué es y sus características?

#### 1. Java SE

Standard Edition: API, máquina virtual, herramientas de desarrollo,

#### 2. Java EE

Enterprise Edition: está construida encima de SE, y provee de un API y runtime environment para desarrollar aplicaciones de gran escala.

#### 3. Java ME

Micro Edition: provee de un API y una máquina virtual “small-footprint” para desarrollar programas enfocados en móviles.

### 31. Menciona cuántas versiones hay de Java y qué cambió en cada una de ellas(hasta la actual)\*ahorita lo resumo\*./

- JDK 1.0 (23 de enero de 1996):

La primera versión del lenguaje contiene las clases principales , la máquina virtual y el API gráfico de AWT es un lenguaje que acababa de nacer.

- JDK 1.1 (19 de febrero de 1997) —

- una reestructuración intensiva del modelo de eventos AWT (Abstract Windowing Toolkit)
- clases internas (inner classes)
- JavaBeans
- JDBC (Java Database Connectivity), para la integración de bases de datos
- RMI (Remote Method Invocation)

- J2SE 1.2 (8 de diciembre de 1998) — Nombre clave *Playground*. Esta y las siguientes versiones fueron recogidas bajo la denominación Java 2

- la palabra reservada (keyword) strictfp
- reflexión en la programación
- la API gráfica ( Swing) fue integrada en las clases básicas
- la máquina virtual (JVM) de Sun fue equipada con un compilador JIT (Just in Time) por primera vez
- Java Plug-in
- Java IDL, una implementación de IDL (Lenguaje de Descripción de Interfaz) para la interoperabilidad con CORBA
- Colecciones (Collections)

- J2SE 1.3 (8 de mayo de 2000) — Nombre clave *Kestrel*.

- la inclusión de la máquina virtual de HotSpot JVM (la JVM de HotSpot fue lanzada inicialmente en abril de 1999, para la JVM de J2SE 1.2)
- RMI fue cambiado para que se basara en CORBA
- JavaSound

- se incluyó el Java Naming and Directory Interface (JNDI) en el paquete de bibliotecas principales (anteriormente disponible como una extensión)
  - Java Platform Debugger Architecture (JPDA)
- J2SE 1.4 (6 de febrero de 2002) — Nombre clave *Merlin*. Este fue el primer lanzamiento de la plataforma Java desarrollado bajo el Proceso de la Comunidad Java como JSR 59. Los cambios más notables fueron: comunicado de prensalista completa de cambios
  - Palabra reservada `assert` (Especificado en JSR 41.)
  - Expresiones regulares modeladas al estilo de las expresiones regulares Perl
  - Encadenación de excepciones Permite a una excepción encapsular la excepción de bajo nivel original.
  - non-blocking NIO (New Input/Output) (Especificado en JSR 51.)
  - Logging API (Specified in JSR 47.)
  - API I/O para la lectura y escritura de imágenes en formatos como JPEG o PNG
  - Parser XML integrado y procesador XSLT (JAXP) (Especificado en JSR 5 y JSR 63.)
  - Seguridad integrada y extensiones criptográficas (JCE, JSSE, JAAS)
  - Java Web Start incluido (El primer lanzamiento ocurrió en marzo de 2001 para J2SE 1.3) (Especificado en JSR 56.)
- J2SE 5.0 (30 de septiembre de 2004) — Nombre clave: *Tiger*. (Originalmente numerado 1.5, esta notación aún es usada internamente.[2]) Desarrollado bajo JSR 176, Tiger añadió un número significativo de nuevas características comunicado de prensa
  - Plantillas (genéricos) — provee conversión de tipos (type safety) en tiempo de compilación para colecciones y elimina la necesidad de la mayoría de conversión de tipos (type casting). (Especificado por JSR 14.)
  - Metadatos — también llamados anotaciones, permite a estructuras del lenguaje como las clases o los métodos, ser etiquetados con datos adicionales, que puedan ser procesados posteriormente por utilidades de proceso de metadatos. (Especificado por JSR 175.)
  - Autoboxing/unboxing — Conversiones automáticas entre tipos primitivos (Como los `int`) y clases de envoltura primitivas (Como `Integer`). (Especificado por JSR 201.)
  - Enumeraciones — la palabra reservada `enum` crea una `typesafe`, lista ordenada de valores (como `Dia.LUNES`, `Dia.MARTES`, etc.). Anteriormente, esto solo podía ser llevado a cabo por constantes enteras o clases construidas manualmente (`enum pattern`). (Especificado por JSR 201.)
  - Varargs (número de argumentos variable) — El último parámetro de un método puede ser declarado con el nombre del tipo seguido por tres puntos (e.g. `void drawtext(String... lines)`). En la llamada al método, puede usarse cualquier número de parámetros de ese tipo, que serán almacenados en un array para pasarlos al método.
  - Bucle `for` mejorado — La sintaxis para el bucle `for` se ha extendido con una sintaxis especial para iterar sobre cada miembro de un array o sobre

cualquier clase que implemente Iterable, como la clase estándar Collection, de la siguiente forma:

```
void displayWidgets (Iterable<Widget> widgets) {  
    for (Widget w : widgets) {  
        w.display();  
    }  
}
```

Este ejemplo itera sobre el objeto Iterable widgets, asignando, en orden, cada uno de los elementos a la variable w, y llamando al método display() de cada uno de ellos. (Especificado por JSR 201.)

- Java SE 6 (11 de diciembre de 2006) — Nombre clave *Mustang*. Estuvo en desarrollo bajo la JSR 270. En esta versión, Sun cambió el nombre "J2SE" por Java SE y eliminó el ".0" del número de versión.[3] Está disponible en <http://java.sun.com/javase/6/>. Los cambios más importantes introducidos en esta versión son:
  - Incluye un nuevo marco de trabajo y APIs que hacen posible la combinación de Java con lenguajes dinámicos como PHP, Python, Ruby y JavaScript.
  - Incluye el motor Rhino, de Mozilla, una implementación de Javascript en Java.
  - Incluye un cliente completo de Servicios Web y soporta las últimas especificaciones para Servicios Web, como JAX-WS 2.0, JAXB 2.0, STAX y JAXP.
  - Mejoras en la interfaz gráfica y en el rendimiento.
- Java SE 7 — Nombre clave *Dolphin*. En el año 2006 aún se encontraba en las primeras etapas de planificación. Su lanzamiento fue en julio de 2011.
  - Soporte para XML dentro del propio lenguaje.
  - Un nuevo concepto de superpaquete.
  - Soporte para closures.
  - Introducción de anotaciones estándar para detectar fallos en el software.
- No oficiales:
  - NIO2.
  - Java Module System.
  - Java Kernel.
  - Nueva API para el manejo de Días y Fechas, la cual reemplazará las antiguas clases Date y Calendar.
  - Posibilidad de operar con clases BigDecimal usando operandos.
- Java SE 8 — lanzada en marzo de 2014. Cabe destacar:
  - Incorpora de forma completa la librería JavaFX.
  - Diferentes mejoras en seguridad.
  - Diferentes mejoras en concurrencia.
  - Añade funcionalidad para programación funcional mediante expresiones Lambda.
  - Mejora la integración de JavaScript.
  - Nuevas API para manejo de fechas y tiempo (date - time).

- Java SE 9 — prevista para 23 de marzo del 2017:

### 32. ¿Cuáles son los 5 principios bajo los que se creó el lenguaje?

//y “write once run anywhere”? :v

empieza a contar desde 0?(supongo es la 2)

1. Debería usar el paradigma de la programación orientada a objetos.
2. Debería permitir la ejecución de un mismo programa en múltiples sistemas operativos.
3. Debería incluir por defecto soporte para trabajo en red.
4. Debería diseñarse para ejecutar código en sistemas remotos de forma segura.
5. Debería ser fácil de usar y tomar lo mejor de otros lenguajes orientados a objetos, como C++.

[https://es.wikipedia.org/wiki/Java\\_\(lenguaje\\_de\\_programaci%C3%B3n\)#Filosof%C3%ADa](https://es.wikipedia.org/wiki/Java_(lenguaje_de_programaci%C3%B3n)#Filosof%C3%ADa)

### 33. ¿Cuándo usar programación orientada a objetos y cuándo usar programación estructurada? Dar 3 ejemplos.

Estructurada:

cuando busquemos

- mayor entendimiento (y posible modificacion) de nuestro programa.
- Estructuras más notoriamente relacionadas

Orientada a objetos:

cuando busquemos

- Encapsulación.
- Facil creación de nuevos tipos de objetos a partir de los existentes
- facilita la creación de programas.

Ejemplos:

Deseo realizar una acción concreta, rápida de entender y mejorar. (utilizar estructurada)

Realizaré un proyecto mediano o grande y moderno. (utilizar poo).

Deseo crear un programa rápidamente, y que tenga una curva de aprendizaje menor. (utilizar estructurada)

### 34. Menciona 5 mitos y 5 realidades de Java

#### Mito: esta bien culero programar en java

realidad: la neta si :v, java ya está muerto(nomas que no le han avisado), ya dejenlo.

Mito:

La seguridad y la independencia de la máquina “son gratis”

Write once, run anywhere”

Java es un lenguaje de programación para la web

Java es un lenguaje de programación de propósito general

java esta muerto || java acabará con x tecnología.

Realidad:

Los compiladores JIT compilan el programa al cargarlo

Aplicaciones un 20% más lentas que

en C++

Se puede conseguir, aunque se debe comprobar.

Java es un lenguaje de programación de propósito general

Siempre tendrá sus ventajas y desventajas.

### 35. Mencionar algunas críticas que ha recibido el lenguaje Java.

- La concisión del código generado,.
- La ausencia de determinadas funcionalidades existentes en otros lenguajes. Por ejemplo, en las versiones de Java previas a Java SE 5.0 se criticaba la falta de tipos genéricos en comparación con los *templates* de C++.
- **Error del Billón de Dólares:** Java es propenso al famoso **null pointer exception**. Esta excepción es responsable de gran parte de los problemas que surgen con el código en producción. Otros lenguajes tienen características que los hacen menos propensos o incluso inmunes a este tipo de error.
- **Verbosidad** es un lenguaje que requiere escribir demasiado para lograr tareas que deberían ser más sencillas.



- En un sentido estricto, Java no es un lenguaje absolutamente orientado a objetos, a diferencia de, por ejemplo, Ruby o Smalltalk. Por motivos de eficiencia, Java ha relajado en cierta medida el paradigma de orientación a objetos, y así por ejemplo, no todos los valores son objetos.
- El código Java puede ser a veces redundante en comparación con otros lenguajes. Esto es en parte debido a las frecuentes declaraciones de tipos y conversiones de tipo manual (casting).

### 36. Definir concretamente los siguientes conceptos:

#### 1. Clase

plantilla para la creación de [objetos de datos](#), define atributos y métodos.(cual hipervínculo, perro)

#### 2. Mensaje

Cualquiera de las dos:

*solicitud a un objeto para invocar uno de sus métodos.*

cuando el emisor le pide al receptor que aplique un método a sí mismo. Contiene el nombre del método y los argumentos del mismo.

#### 3. Objeto

La instanciación de una clase. Representa una identidad física

#### 4. Atributo

Propiedad de un objeto, determinan su estado.

#### 5. Método

operación que se puede realizar sobre un objeto.define el comportamiento de una clase.

#### 6. Instancia

articularización, realización específica u ocurrencia de una determinada clase

#### 7. Referencia

indicativo hacia un objeto.

### 37. ¿Qué es un Token?

Un **token** o también llamado **componente léxico** es una [cadena de caracteres](#) que tiene un significado coherente en cierto [lenguaje de programación](#). Ejemplos de tókenes podrían ser palabras clave (if, else, while, int, ...), [identificadores](#), números, signos, o un operador de varios caracteres, (por ejemplo, := "" :+"" ).

38. ¿Qué es un delimitador y cuales usa Java?

- 1) Carácter usado para separar las sub-cadenas, en la cadena a devolver.
- 2) Se llama delimitador a las palabras reservadas if, else, then, Select Case, switch, for, while, etc.

39. Menciona 20 palabras reservadas de Java y breve descripción.

//Elijan las que quieran//

1. **Abstract:** Declara clases o métodos abstractos.
2. **Double:** Tipo de Dato primitivo de punto flotante por defecto (32 bits).
3. **Int:** Tipo de Dato primitivo entero por defecto (32 bits).
4. **Strictfp:** Especifica bajo que standard se calcularán las operaciones con datos de punto flotante, para determinar el grado de precisión de los resultados.
5. **Boolean:** Tipo de Dato primitivo booleano (true o false).
6. **Else:** Evaluación de la condición lógicamente opuesta a un if o else if.
7. **Interface:** Declara interfaces.
8. **Super:** Hace referencia a la clase padre o al constructor de la clase padre del objeto actual.
9. **Break:** Rompe el flujo normal del bloque de código actual.
10. **Extends:** Indica que una clase o interfase hereda de otra clase o interfase.
11. **Long:** Tipo de Dato primitivo entero (64 bits).
12. **Switch:** Estructura de control condicional múltiple.
13. **Byte:** Tipo de Dato primitivo entero (8 bits).
14. **final:** Declara la clase, método o variable como "definitiva".
15. **native:** Indica que el método va a ser especificado en un lenguaje diferente a Java.
16. **synchronized :** Indica que el método, o bloque de código deberá prevenir que no sean cambiados los objetos a afectar dentro del bloque o método.
17. **case:** Verifica cada valor evaluado en un a sentencia switch.
18. **finally:** Determina el bloque de código que se ejecutará siempre luego de un try así sea que se capture o no una excepción.
19. **new:** Solicita al cargador de clases correspondiente, un objeto de esa clase.
20. **this:** Hace referencia el objeto actual o al constructor del objeto actual.
21. **catch:** Atrapa excepciones dentro de un bloque try
22. **float :** Tipo de Dato primitivo de punto flotante (64 bits).
23. **package:** Especifica el paquete al que pertenece esa clase o interfase.
24. **throw:** Lanza una excepción mediante código.
25. **char:** Tipo de Dato primitivo que almacena hasta un caracter UNICODE (16 bits).
26. **for:** Estructura de control cíclica.
27. **private:** Modificador de visibilidad de atributos y métodos limitándolos a la propia clase.
28. **throws:** Especifica la(s) excepciones(es) que podría lanzar el método.
29. **class:** Declara clases
30. **goto:** Palabra reservada que no es utilizada en el lenguaje de programación Java.
31. **protected:** Modificador de visibilidad de atributos y métodos limitándolos a la propia clase, paquete e hijo(s).

- 32. **transient**: Indica que el objeto no se debe serializar.
- 33. **const**: Palabra reservada que no es utilizada en el lenguaje de programación Java.
- 34. **if**: Estructura de control condicional.
- 35. **public**: Modificador de visibilidad de clases, interfaces, atributos y métodos haciéndolo visible al universo.
- 36. **try**: Declara un bloque de código que posiblemente lanzará una excepción.
- 37. **continue**: Rompe el flujo normal del bloque de código actual.
- 38. **implements**: Indica que una clase implementa a una (o varias) interfase(s).
- 39. **return**: Retorna (normalmente un valor) desde el método actual.
- 40. **void**: Indica que el método no retornará valor alguno.
- 41. **default**: Modificador de visibilidad de clases, interfases, atributos y métodos limitándolos a la clase y paquete.
- 42. **import**: Indica la(s) ruta(s) en la que se encuentran las clases y/o interfases usadas en el código
- 43. **short**: Tipo de Dato primitivo entero (16 bits).
- 44. **volatile**: Indica que a la referencia de la variable siempre se debería leer sin aplicar ningún tipo de optimizaciones ya que el dato almacenado tiene alta probabilidad de cambiar muy frecuentemente.
- 45. **do**: Estructura de control cíclica
- 46. **instanceof**: Operador que determina si un objeto es una instancia de una clase.
- 47. **static**: Indica que el método, variable o atributo pertenece a la clase y no a la instancia (objeto).
- 48. **while**: Estructura de control cíclica.

#### 40. ¿Qué es un identificador en Java?

Los identificadores son los nombres que el programador asigna a variables, constantes, clases, métodos, paquetes, etc. de un programa.

#### 41. Mencione 3 personajes importantes en la historia de Java y qué hizo cada uno

El lenguaje de programación Java fue originalmente desarrollado por James Gosling, de Sun Microsystems (la cual fue adquirida por la compañía Oracle), y publicado en 1995 como un componente fundamental de la plataforma Java de Sun Microsystems.

Entre junio y julio de 1994, tras una sesión maratónica de tres días entre John Ganga, James Gosling, Patrick Naughton, Wayne Rosing y Eric Schmidt, el equipo reorientó la plataforma hacia la Web.

#### 42. ¿Qué fue Sun Microsystems?

Sun Microsystems, Inc. era una compañía estadounidense que vendía computadoras, componentes de computadoras, software y servicios de tecnología de la información, y creó el lenguaje de programación Java, el sistema operativo Solaris, ZFS, el Sistema de archivos de red(NFS) y SPARC.

#### 43. ¿Qué es Oracle?

La compañía se especializa principalmente en desarrollar y comercializar software y tecnología de base de datos , sistemas de ingeniería en la nube y productos de software empresarial , particularmente sus propias marcas de sistemas de administración de bases de datos .

#### 44. ¿Qué es un dato?

Un **dato** es una representación simbólica (numérica, alfabética, algorítmica, espacial, etc.) de un atributo o variable cuantitativa o cualitativa

#### 45. ¿Qué es un tipo de dato?

un **tipo de dato informático** o simplemente **tipo**, es un atributo de los datos que indica al ordenador (y/o al programador/programadora) sobre la clase de datos que se va a manejar.

#### 46. ¿Qué es una literal?

Una literal es un valor constante formado por una secuencia de caracteres.

[http://profesores.fi-b.unam.mx/carlos/java/java\\_basico2\\_4.html](http://profesores.fi-b.unam.mx/carlos/java/java_basico2_4.html)

#### 47. ¿Qué es una variable?

espacio en el sistema de almacenaje (memoria principal de un ordenador) y un nombre simbólico (un identificador) que está asociado a dicho espacio. Ese espacio contiene un valor.

#### 48. ¿Qué es una constante y cómo se declara en Java?

Una constante es una variable del sistema que mantiene un **valor inmutable a lo largo de toda la vida del programa**

```
static final nombreConstante = valor;
```

#### 49. Tipos de datos primitivos en Java y sus características.

- **byte**: El tipo de dato byte es un entero de 8 bits complemento a dos. Su valor mínimo es -128 y el máximo 127 (inclusive). El tipo de datos byte se puede utilizar para ahorrar memoria en grandes arrays, donde el ahorro de memoria realmente importa.
- **short**: El tipo de dato short es un entero de 16 bits complemento a dos. Su valor mínimo es -32,768 y el máximo 32,767. Se aplican las mismas directrices que con byte: puede utilizar short para ahorrar memoria en grandes arrays
- **int**: El tipo de dato int es un entero de 32 bits complemento a dos. Su valor mínimo es -2,147,483,648 y el máximo 2,147,483,647 (inclusive).  
Generalmente este tipo es la elección predeterminada para valores enteros a no ser que haya una razón (como las mencionadas anteriormente) para elegir otro. Este tipo de dato normalmente será lo suficiente grande para los

números que su program vaya a utilizar pero si necesita un rango más amplio, utilice long.

- **long**: El tipo de dato long es un entero de 64 bits complemento a dos. Su valor mínimo es -9,223,372,036,854,775,808 y el máximo 9,223,372,036,854,775,807 (inclusive). Utilice este tipo de dato cuando necesite un rango de valores más amplio que el proporcionado por int.
- **float**: El tipo de dato float es un dato en coma flotante IEEE 754 de 32 bits y precisión simple. Al igual que con byte y short, se recomienda usar un float (en vez de un double) si necesita ahorrar memoria en grandes array de números en coma flotante. Este tipo de dato nunca debería ser usado para valores precisos como, por ejemplo, divisas.
- **double**: El tipo de dato double es un dato en coma flotante IEEE 754 de 64 bits y precisión doble. Normalmente este tipo de dato es la elección predeterminada para valores decimales.
- **boolean**: El tipo de dato boolean solamente tiene dos valores posibles: true (verdadero) y false (falso). Utilice este tipo de datos como conmutadores para la evaluación de condiciones verdadero/falso. Este tipo de dato representa un bit de información
- **char**: El tipo de dato char es un solo carácter Unicode de 16 bits. Tiene un valor mínimo de '\u0000' (o «0») y un máximo de '\uffff' (o 65.535 inclusive).

50. Menciona todos los operadores en Java (aritméticos, booleanos, unarios, binarios, etc.).

Prior.	Operador	Tipo de operador	Operación
1	++	Aritmético	Incremento previo o posterior (unario)
	--	Aritmético	Incremento previo o posterior (unario)
	+, -	Aritmético	Suma unaria, Resta unaria
	~	Integral	Cambio de bits (unario)
	!	Booleano	Negación (unario)
2	(tipo)	Cualquiera	
3	*, /, %	Aritmético	Multiplicación, división, resto
4	+, -	Aritmético	Suma, resta
	+	Cadena	Concatenación de cadenas
5	<<	Integral	Desplazamiento de bits a izquierda
	>>	Integral	Desplazamiento de bits a derecha con inclusión de signo
	>>>	Integral	Desplazamiento de bits a derecha con inclusión de cero
6	<, <=	Aritmético	Menor que, Menor o igual que
	>, >=	Aritmético	Mayor que, Mayor o igual que
	instanceof	Objeto, tipo	Comparación de tipos
7	==	Primitivo	Igual (valores idénticos)
	!=	Primitivo	Desigual (valores diferentes)
	==	Objeto	Igual (referencia al mismo objeto)
	!=	Objeto	Desigual (referencia a distintos objetos)
8	&	Integral	Cambio de bits AND
	&	Booleano	Producto booleano
9	^	Integral	Cambio de bits XOR
	^	Booleano	Suma exclusiva booleana
10		Integral	Cambio de bits OR
		Booleano	Suma booleana
11	&&	Booleano	AND condicional
12		Booleano	OR condicional
13	? :	Booleano, cualquiera, cualquiera	Operador condicional (ternario)
14	=	Variable, cualquiera	Asignación
	*, /, %=		Asignación con operación
	+, -=		
	<<=, >>=		
	>>>=		
	&=, ^=,  =		

51. ¿Qué es un casting en Java?

Tomar un objeto de un tipo particular y convertirlos en otro de otro tipo

52. ¿Qué son los paquetes en Java?

Un **paquete java** es un grupo de tipos similares de clases, interfaces y subpaquetes. Se puede categorizar en dos formas, paquete integrado y paquete definido por el usuario.

53. Explicar la visibilidad de los miembros de una clase:

**public**

clase es visible para todas las clases en todas partes.

## protected

especifica que solo se puede acceder al miembro dentro de su propio paquete

## private

solo se puede acceder al miembro en su propia clase.

## 54. Escribe las convenciones de escritura de un código en Java.

- **Paquetes:** los nombres de paquetes se escriben en minúsculas:
- **Clases:** los nombres de clases deben ser escritos en letras minúsculas, todas las palabras seguidas y la primera letra de cada palabra en mayúscula, esta forma de nomenclatura también es llamada *lomo de camello* o *CamelCase* en inglés
- **Métodos:** Los métodos deben ser verbos escritos con la palabra inicial en minúscula y las siguientes palabras con la inicial en Mayúscula
- **Variables:** Deben escribirse en mayúsculas y minúsculas, con la inicial en minúscula, con la siguiente palabra con la inicial en mayúscula. Casi no se usan los signos de subrayado (\_) y evite el signo de dólar (\$).
- **Constantes:** Las constantes deben escribirse enteramente en mayúsculas y separando las palabras mediante underscore (subrayado).
- **Estructuras de Control:** cuando las sentencias forman parte de una estructura de control de flujo como un for, while, es necesario incluirlas en un paquete de sentencias con los corchetes ({}) aunque solo tengan una línea de código:
- **Espacios:** Los espacios son usados para entender mejor el código y distinguir el principio y el fin de las estructuras de control usese de acuerdo a la estructura de control las separaciones por sangría necesarias, por lo regular no son más de 4. Puede variar de acuerdo a las convenciones usadas.
- **Comentarios:** Use comentarios de una línea o varias, puede inclusive usar JAVADOC. No hay restricción de acuerdo a la forma en que deben estar escritos los comentarios.

## 55. Escribe diferencias entre C y Java (principalmente en manejo de memoria, arreglos y tipos básicos).

En C, el punto principal, para bien o para mal, está en el acceso a memoria. Este acceso se realiza mediante punteros que nos permiten gestionar DÓNDE y QUÉ se guarda en la memoria.

Java, al contrario, hace una administración automatizada de la memoria, por lo que el programador no tiene de qué preocuparse en cuanto a cómo gestionarla evitando así posibles errores que con C podríamos cometer al posicionar los punteros.

<http://www.ingenieroboss.com/programacion-c-vs-java/>

## 56. Realizar un Tutorial (impreso) con capturas de pantalla de:

¿Cómo Instalar Java JDK (última versión) en Linux?

¿Cómo hacer un hola mundo en Java?

```
class MyClass {  
    public static void main(String[] args) {
```



```
        System.out.println("Hola mundo");
    }
}
```

## ¿Cómo compilar y ejecutar desde terminal?

### 57. ¿Cómo se comenta en un código de Java?

```
-- esto es un comentario
/* esto es otro comentario */
```

### 58. ¿Por qué es importante comentar un código? (En cualquier lenguaje)

**Para** hacer el código fuente más fácil de entender con vistas a su mantenimiento o reutilización.

Se ha de tener en cuenta que los comentarios necesitan mantenimiento igual que el código y, por tanto, que un comentario preciso y conciso es más fácil de mantener que uno largo, repetitivo y complicado.

Los comentarios tienen una amplia gama de posibles usos: desde la mejora del código fuente con descripciones básicas hasta la [generación de documentación](#) externa. También se utilizan para la integración con sistemas de [control de versiones](#) y otros tipos de [herramientas de programación](#) externas.

### 59. ¿Qué significa que Java sea multiplataforma?

Es un término usado para referirse a los programas, [sistemas operativos](#), lenguajes de programación, u otra clase de software, que puedan funcionar en diversas plataformas. Por ejemplo, una aplicación multiplataforma podría ejecutarse en [Windows](#) en un procesador x86, en [GNU/Linux](#) en un procesador x86, y en Mac OS X en uno x86 (solo para equipos Apple) o en un PowerPC

--Otra definición

Esto significa que el [hardware](#) o [software](#) que es multiplataforma tiene la característica de funcionar de forma similar en distintas plataformas (distintos [sistemas operativos](#) por ejemplo).

### 60. ¿Qué significa que java sea portable?

Puede ser utilizada en cualquier ordenador que posea el sistema operativo para el que fue programada; esto significa que no es necesaria la instalación de bibliotecas adicionales en el sistema para su funcionamiento.

### 61. ¿Qué es un lenguaje de programación robusto? que crea software Altamente fiable

/\*Lo encontré aquí



[https://lenguajedeprogramacionjava.blogspot.mx/2011/06/1-definicion-y-principales.h  
tml](https://lenguajedeprogramacionjava.blogspot.mx/2011/06/1-definicion-y-principales.html)  
\*/

## 62. Escribe la definición de los siguientes conceptos

**Herencia:** La idea de la herencia es permitir la creación de nuevas clases basadas en clases existentes.

Cuando heredamos de una clase existente, reusamos (o heredamos) métodos y campos, y agregamos nuevos campos y métodos para cumplir con la situación nueva.

/\*Cada vez que encontremos la relación "es-un" entre dos clases, estamos ante la presencia de herencia.\*/

**Abstracción:** Abstracción es un término del mundo real que podemos aplicar tal cual lo entendemos en el mundo de la Programación Orientada a Objetos.

**Polimorfismo:** se refiere a la propiedad por la que es posible enviar mensajes sintácticamente iguales a **objetos** de **tipos** distintos. El único requisito que deben cumplir los objetos que se utilizan de manera polimórfica es saber responder al mensaje que se les envía.

### Upcast

convertir una instancia de una subclase a su superclase.

### Downcast

Convertir un objeto de una superclase a su subclase es llamado downcasting o conversión hacia abajo.

**Interfaces:** **Interfaz (Java)** Una **interfaz** en **Java** es una colección de métodos abstractos y propiedades constantes. En las **interfaces** se especifica qué se debe hacer pero no su implementación. Serán las clases que implementen estas **interfaces** las que describan la lógica del comportamiento de los métodos.

**Encapsulamiento:** Se refiere a la recolección de características que pueden pertenecer a una misma clase, y que están al mismo nivel de abstracción, las cuales serán ocultas para la protección del estado interno de un objeto. En Java el principio de ocultación es representado por los Modificadores de Acceso **public**, **private**, **protected**.

### Acoplamiento

Es el grado de interdependencia entre los módulos de un sistema.

### Cohesión

Medida que indica qué tan especializado es un módulo u objeto.

## 63. ¿Qué es la sobrecarga de métodos en Java?

La **sobrecarga de métodos** significa tener varios métodos con el mismo nombre pero con distintos **parámetros**.

## 64. ¿Qué es un constructor?

Los constructores son funciones o métodos que permiten realizar tareas de instanciación de objetos. Cuando un objeto es creado a partir de una clase, se llama al constructor que se encargará de inicializar los atributos del objeto, como así también cualquier llevar a cabo cualquier otra función necesaria. No es obligatorio el uso de un constructor para inicializar un objeto.

## 65. Menciona las diferencias entre los tipos de tipado: fuerte, débil, estático, dinámico.

El tipado débil es en donde no indicamos el tipo de variable al declararla.

Tipado fuerte: indicamos el tipo de dato al declarar la variable. Dicho tipo no puede ser cambiado nunca.

El *tipado estático* nos obliga a definir desde el principio el tipo de una variable, ejemplos de lenguajes con tipado estatico son C, C++, Pascal, Java, Objective-C, C#, entre otros, pero estos son los utilizamos en Qbit. Notese que C# esta incluido aunque exista la palabra clave **var**, la cual nos permite ahorrarnos un poquito de código, pero una vez que el compilador define el tipo este ya no puede cambiarse.

El *tipado dinamico* nos da la facilidad de no definir los tipos al declarar una variable, algunos ejemplos son PHP, JavaScript, Groovy, Python, Perl, entre otros, estos son los que mas usamos aquí.

## 66. ¿Qué tipo de tipado tiene Java? estático, fuertemente tipado

## 67. ¿Qué es el Garbage Collector y cómo lo puedo llamar en Java?

Es un proceso de la JVM que está revisando qué objetos pueden ser borrados y cuáles no.

El GC es un proceso de baja prioridad, por lo que no se pasa en todo momento liberando memoria, si no que pasa de vez en cuando, que podría ser en un tiempo muerto del procesador, aunque también nosotros podríamos *sugerirle* que pase, pero va a pasar cuando pueda y quiera, un par de ejemplos de cómo mandarlo llamar son estos:

```
System.gc();  
Runtime.getRuntime().gc();
```

## 68. Nombrar Aplicaciones "famosas" hechas en java (al menos 5).

**MINECRAFT** //niño rata

**GEOGEBRA**

**OPENFIRE**

**NOTEPAD++**

CHROME

69. ¿Qué es la certificación en Java?

70. Menciona 2 certificaciones oficiales en Java, descripción y requerimientos de cada una de ellas

71. ¿Qué son los diagramas UML?

Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos, funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y compuestos reciclados.

72. ¿Qué tipos de diagramas UML existen? Describe cada uno

Perdón que no lo resuma

- Diagrama de clases Los diagramas de clase son, sin duda, el tipo de diagrama UML más utilizado. Es el bloque de construcción principal de cualquier solución orientada a objetos. Muestra las clases en un sistema, atributos y operaciones de cada clase y la relación entre cada clase. En la mayoría de las herramientas de modelado, una clase tiene tres partes, nombre en la parte superior, atributos en el centro y operaciones o métodos en la parte inferior. En sistemas grandes con muchas clases relacionadas, las clases se agrupan para crear diagramas de clases. Las Diferentes relaciones entre las clases se muestran por diferentes tipos de flechas.
- Diagrama de componentes Un diagrama de componentes muestra la relación estructural de los componentes de un sistema de software. Estos se utilizan principalmente cuando se trabaja con sistemas complejos que tienen muchos componentes. Los componentes se comunican entre sí mediante interfaces. Las interfaces se enlazan mediante conectores.
- Diagrama de despliegue Un diagrama de despliegue muestra el hardware de su sistema y el software de ese hardware. Los diagramas de implementación son útiles cuando la solución de software se despliega en varios equipos, cada uno con una configuración única.
- Diagrama de objetos Los diagramas de objetos, a veces denominados diagramas de instancia, son muy similares a los diagramas de clases. Al igual que los diagramas de clases, también muestran la relación entre los objetos, pero usan ejemplos del mundo real. Se utilizan para mostrar cómo se verá un sistema en un momento dado. Debido a que hay datos disponibles en los objetos, a menudo se utilizan para explicar relaciones complejas entre objetos.
- Diagrama de paquetes Como su nombre indica, un diagrama de paquetes muestra las dependencias entre diferentes paquetes de un sistema.
- Diagrama de perfiles El diagrama de perfil es un nuevo tipo de diagrama introducido en UML 2. Este es un tipo de diagrama que se utiliza muy raramente en cualquier especificación.
- Diagrama de estructura compuesta Los diagramas de estructura compuesta se utilizan para mostrar la estructura interna de una clase.

## De comportamiento

Muestran el comportamiento dinámico de los objetos en el sistema.

- Diagrama de actividades Los diagramas de actividad representan los flujos de trabajo de forma gráfica. Pueden utilizarse para describir el flujo de trabajo empresarial o el flujo de trabajo operativo de cualquier componente de un

sistema. A veces, los diagramas de actividad se utilizan como una alternativa a los diagramas de máquina del estado.

- [Diagrama de casos de uso](#) Como el tipo de diagrama de diagramas UML más conocido, los diagramas de casos de uso ofrecen una visión general de los actores involucrados en un sistema, las diferentes funciones que necesitan esos actores y cómo interactúan estas diferentes funciones. Es un gran punto de partida para cualquier discusión del proyecto, ya que se pueden identificar fácilmente los principales actores involucrados y los principales procesos del sistema.
- [Diagrama de máquina de estados](#) Los diagramas de máquina de estado son similares a los diagramas de actividad, aunque las anotaciones y el uso cambian un poco. En algún momento se conocen como diagramas de estados o diagramas de diagramas de estado también. Estos son muy útiles para describir el comportamiento de los objetos que actúan de manera diferente de acuerdo con el estado en que se encuentran en el momento.

## De interacción

- [Diagrama global de interacciones](#) Los diagramas generales o globales de interacción son muy similares a los diagramas de actividad. Mientras que los diagramas de actividad muestran una secuencia de procesos, los diagramas de interacción muestran una secuencia de diagramas de interacción. En términos simples, pueden llamarse una colección de diagramas de interacción y el orden en que suceden. Como se mencionó anteriormente, hay siete tipos de diagramas de interacción, por lo que cualquiera de ellos puede ser un nodo en un diagrama de vista general de interacción.
- [Diagrama de comunicación](#) El diagrama de comunicación se llamó diagrama de colaboración en UML 1. Es similar a los diagramas de secuencia, pero el foco está en los mensajes pasados entre objetos.
- [Diagrama de secuencia](#) Los diagramas de secuencia en UML muestran cómo los objetos interactúan entre sí y el orden en que se producen esas interacciones. Es importante tener en cuenta que muestran las interacciones para un escenario en particular. Los procesos se representan verticalmente y las interacciones se muestran como flechas.
- [Diagrama de tiempos](#) Los diagramas de sincronización son muy similares a los diagramas de secuencia. Representan el comportamiento de los objetos en un marco de tiempo dado. Si es solo un objeto, el diagrama es directo, pero si hay más de un objeto involucrado, también se pueden usar para mostrar interacciones de objetos durante ese período de tiempo.

### 73. ¿Cómo modelar las relaciones entre clases y objetos con UML? Incluir un ejemplo

### 74. ¿Qué tipos de errores existen en la programación

Perdon de nuevo por no simplificarlo :’v

**Errores de compilación:** Los errores de compilación, también conocidos como errores del compilador, son errores que impiden que su programa se ejecute. Cuando se presiona F5 para ejecutar un programa, Visual Basic compila el código en un lenguaje binario que entiende el equipo. Si el compilador de Visual Basic se encuentra con código que no entiende, emite un error de compilador. La mayoría de los errores del compilador se deben a errores cometidos al escribir el código. Por ejemplo, puede escribir mal una palabra clave, omitir alguna puntuación necesaria o intentar utilizar una instrucción End If sin antes utilizar una instrucción If.

**Errores en tiempo de ejecución:** Los errores en tiempo de ejecución son errores que aparecen mientras se ejecuta el programa. Estos errores aparecen normalmente cuando el programa intenta una operación que es imposible que se lleve a cabo. Un ejemplo de esto es la división por cero. Suponga que tiene la instrucción siguiente:  $Speed = Miles / Hours$  Si la variable Hours tiene un valor de 0, se produce un error en tiempo de ejecución en la operación de división. El programa se debe ejecutar para que se pueda detectar este error y si Hours contiene un valor válido, no se producirá el error. Cuando aparece un error en tiempo de ejecución, puede utilizar las herramientas de depuración de Visual Basic para determinar la causa.

**Errores lógicos:** Los errores lógicos son errores que impiden que el programa haga lo que estaba previsto. El código puede compilarse y ejecutarse sin errores, pero el resultado de una operación puede generar un resultado no esperado. Por ejemplo, puede tener una variable llamada FirstName y establecida inicialmente en una cadena vacía. Después en el programa, puede concatenar FirstName con otra variable denominada LastName para mostrar un nombre completo. Si olvida asignar un valor a FirstName, sólo se mostrará el apellido, no el nombre completo como pretendía. Los errores lógicos son los más difíciles de detectar y corregir, pero Visual Basic también dispone de herramientas de depuración que facilitan el trabajo.

### 75. ¿Qué es el archivo .class generado al compilar un programa en java?

El archivo .class es el resultado de compilar los archivos .java. El contenido de los archivos .class es netamente binario y no puede ser interpretado por un editor de texto (que parece ser el programa que usas para abrir el archivo).

Para ver el contenido del .class necesitas un decompilador. O, en su defecto, conocer las fuentes que originaron dicho archivo.

**76. ¿Qué es un archivo con extensión .jar?**

**JAVA ARCHIVES**

Recopila en un fichero, varios ficheros diferentes.

**77. Menciona lenguajes de programación en los que hayas programado y cuál es tu favorito.**

shellscript es mi fav, nunca lo cambiare

**78. ¿Qué esperas del curso de Java?**

**79. Empieza el curso de SoloLearn de Java. Termínalo antes del martes a la hora Prebe.**

Mandar certificado a los correos [rodrigo.vivas.proteco@gmail.com](mailto:rodrigo.vivas.proteco@gmail.com) y [luiszepedavarela@gmail.com](mailto:luiszepedavarela@gmail.com)

**80. Da un comentario sobre tu experiencia en los cursos del Programa de Tecnología en Cómputo (PROTECO)**

**81. Menciona cuál es tu concepto de “profesor” y de “alumno”. ¿Qué relación debe haber entre ellos?**

**82. Lee el artículo “Ir a clase, dar clase” del Ing. Juan Ocáriz Castelazo. Escribe un comentario (Un párrafo)**

**83. ¿Cuánto crees que vale tu tiempo? ¿Por qué?**

La tarea debe de estar COMPLETA y escrita a mano con LETRA LEGIBLE para tener

derecho a permanecer en la clase.

Estudien los conceptos, ya que se harán preguntas al inicio de la clase. No pretendo

que los aprendan todos para el primer día, sino que tengan al menos una idea de lo

que se va a ver para que la clase pueda ser más ágil. (Quizá algún día aleatorio pueda

haber algún examen).

Recuerden el compromiso que hicieron al momento de querer ingresar al programa,

así que den lo mejor de ustedes en estos cursos.