<<<<<<<<<3-6 Sergio

**NOMBRE Y PREGUNTA A RESPONDER**

**1 respuesta por persona**

**Erick: 1**
**Daniel 15-19**
**Alicia 14 <- visón anónimo jajaja khe?**

1. Escribir para qué sirve la clase BigDecimalz

BigDecimal sirve para representar números con coma flotante, de manera precisa, se utiliza para realizar cálculos aritméticos financieros

/* para detallar en más info sobre…
http://java-white-box.blogspot.mx/2012/10/java-player-la-clase-bigdecimal.html
*/
http://www.javamexico.org/blogs/luxspes/por_que_usar_bigdecimal_y_no_double_para_calculos_aritmeticos_financieros

2. ¿Cuando una clase es una interfaz?
Cuando no tiene implementación (es completamente abstracta).
http://elvex.ugr.es/decsai/java/pdf/AC-interfaces.pdf

//información extra

- En Java, las interfaces se declaran con la palabra reservada **`interface`** de manera similar a como se declaran las clases abstractas.
- En la declaración de una interfaz, lo único que puede aparecer son declaraciones de métodos (su nombre y signatura, sin su implementación) y definiciones de constantes simbólicas.
- Una interfaz no encapsula datos, sólo define cuáles son los métodos que han de implementar los objetos de aquellas clases que implementen la interfaz.

/

3. ¿Cómo se declaran las clases abstractas en Java?

```
abstract class NombreClase{
        atributos;
        métodos;
}
```

4. ¿Cómo se declaran las interfaces en Java?

```
interface NombreInterfaz{
        métodos;
        public static final CONSTANTES;
}
```

//Las interfaces como tal no pueden tener atributos, pero de ser necesarios deben cumplir con lo anterior (static final)

5. ¿Cuando se utilizan clases abstractas?

En primer lugar, evitan que los usuarios de la clase puedan crear objetos de la misma, como dice la definición de clase abstracta.En segundo lugar, permiten crear interfaces que luego deben ser implementados por las clases que hereden de la clase abstracta.

6. ¿Cuando se utilizan interfaces?

Se usa cuando una clase padre, en este caso la interfaz,  no necesita instancia alguna de sí misma, sin embargo, todas las clases hijas les darán entido a sus métodos.

//De la 3 a la 6 utilice este sitio
https://es.slideshare.net/jcalmeida2/clases-abstractas-e-interfaces-en-java-15719490

7. ¿Qué sucede con la subclase que hereda de una clase abstracta y no redefine uno de los métodos de dicha superclase?
Si una clase hereda de una clase abstracta, deberá de sobreescribir todos los métodos abstractos, si no es así, la clase que hereda deberá ser pasada a abstracta.

8. ¿Qué sucede con la clase que implementa una interfaz y no redefine uno de los métodos de dicha interfaz?
si una clase implementa una interfaz y no implementa sus metodos, debe ser marcada como abstracta

9. ¿Una subclase es abstracta si hereda de una superclase abstracta y redefine todos los métodos de dicha superclase? ¿Porqué?

10. ¿Una clase es abstracta si implementa una interfaz y redefine todos los métodos de dicha interfaz? ¿Porqué?

11. Diferencia entre sobreescritura y sobrecarga de métodos
Alcance diferente, sobrecarga es en la mism clase mientras que sobreescritura es en clases derivadas y clases base

12. ¿Porqué se recomienda la notación @Override al redefinir un método abstracto?
El compilador te ayuda a checar si efectivamente hiciste un overrride o solo fue un error o escribiste algo mal
¿Existe error si no se agrega?
No, es algo opcional
13. Diferencia entre una constante "final" y una constante "static final"
Final-Puede asignarsele diferente valores durante la ejecucion al ser incializado
Static final- Todas la instancias comparten el mismo valor y no puede ser alterado despues de la primer inicializacion
14. ¿Se necesita instanciar un clase para utilizar un método static? ¿Por qué?
Las variables y los métodos marcados con el modificador *'static'*, es decir, variables y métodos estáticos pertenecen a la clase, no a una instancia de dicha clase en particular. De hecho, se pueden utilizar componentes estáticos sin tener una instancia de la clase. Pero en caso de que existan instancias de dicha clase, el componente estático de la misma será compartido por todas aquellas instancias existentes en un momento dado; solo existe una copia.

Para acceder a un componente estático de una clase solamente basta con utilizar el operador punto (.) después del nombre de la clase y posteriormente hacer referencia al método o variable estática.

http://monillo007.blogspot.mx/2008/05/variables-y-mtodos-estticos-en-java.html

15. ¿Qué es un error?
Se refiere a errores graves en la máquina virtual de Java, como por ejemplo fallos al enlazar con alguna librería. Normalmente en los programas Java no se tratarán este tipo de errores.

16. ¿Qué es una excepción?
En sí, una excepción es un error en tiempo de ejecución. Se dividen en dos categorías: excepción marcada y no marcada.

Fuente: https://jarroba.com/excepciones-exception-en-java-con-ejemplos/

## 17. Diferencias entre excepción y error

Las excepciones representan errores que no son críticos, por ende, pueden ser tratados y 0p2, el programa sigue jalando. En cambio, los errores nel, esos te tumban tu programa como la pata del mameitor.

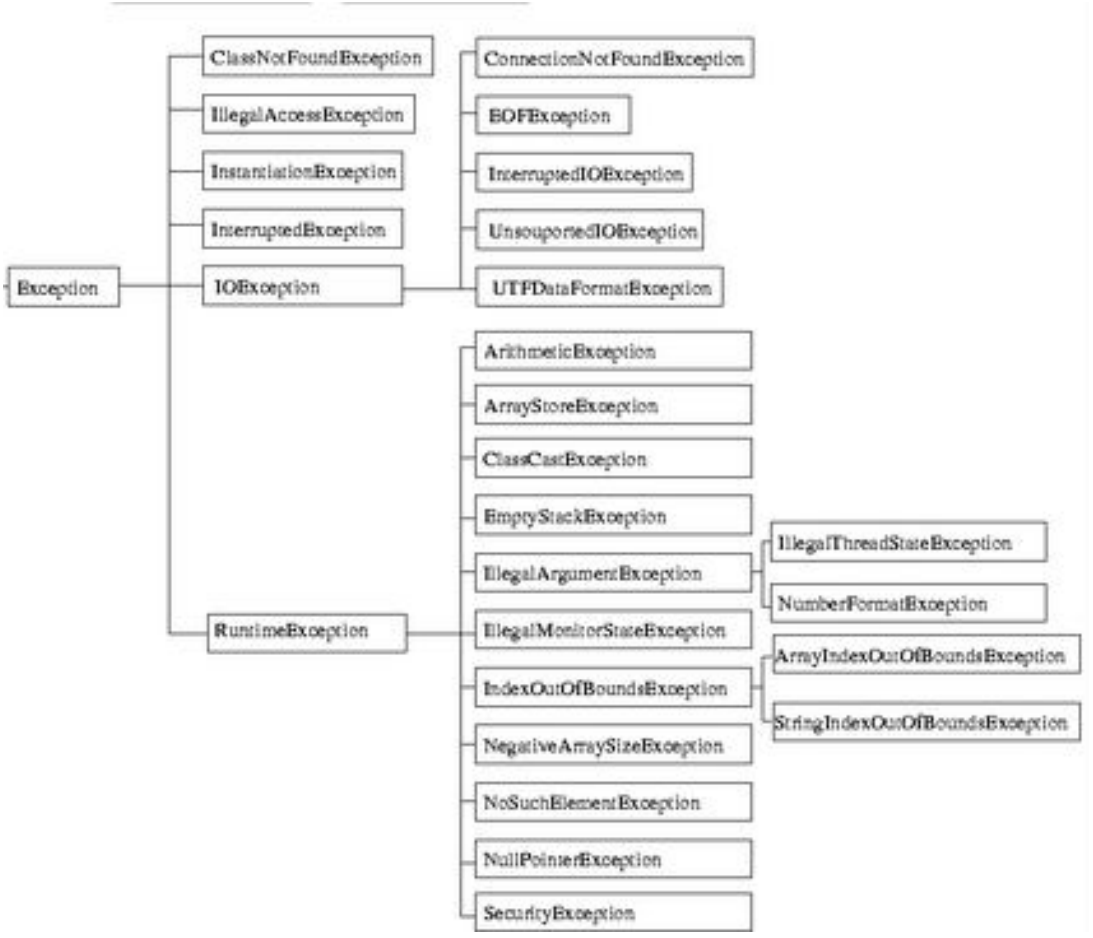
## 18. Describe los bloques: try, catch y finally

Los bloques try catch se ponen alrededor del código que pueda generar una excepción.
Try: todo el código dentro de esta sentencia será el código sobre el que se intentará capturar el error si se produce y una vez capturado hacer algo con él. Lo ideal es que no ocurra un error, pero en caso de que ocurra un bloque try nos permite estar preparados para capturarlo y tratarlo.
Catch: involucra el tratamiento (conjunto de instrucciones) que se le dará a lo que capture el try
Finally: define un conjunto de instrucciones necesarias tanto si se produce error o excepción como si no y que por tanto se ejecuta siempre.

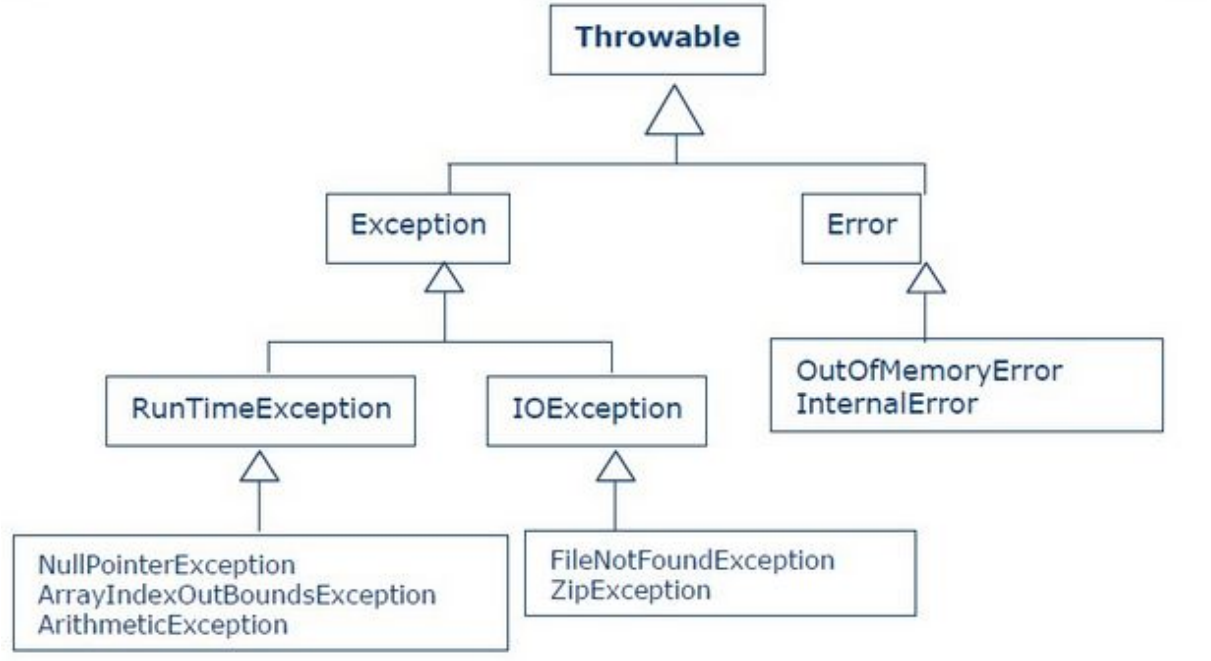
## 19. Explica las palabras throw y throws

La palabra throw es usada para explicitamente tirar una excepción de un método o cualquier parte del código
Throws es una palabra reservada que es usada en la declaración de un método para indicar que éste puede tirar una de las excepciones listadas

20. Mediante UML indica la jerarquía de clases para el manejo de excepciones y describe brevemente cada una de ellas




o esta



21. Leer los siguientes 2 artículos de opinión y e77scribir una conclusión por cada uno:
http://www.xataka.com/aplicaciones/20-anos-de-java-celebramos-su-tremenda-influencia-en-el-mundo-del-software-y-la-programacion
http://www.genbetadev.com/java-j2ee/por-que-empresas-que-empiezan-con-lenguajes-modernos-se-vuelven-a-java

22. Realiza un programa que simule un juego de pokemon.
Se tendrá una clase pokemon para poder instanciar 2 objetos. Cada objeto tendrá diferentes
valores en sus atributos.

Los atributos que debe tener son:

- Nombre
- Vida
- Nivel
- Ataque

Ataque tendrá dos atributos que son (Nombre y Daño)

El objetivo del programa es poner dos pokemon a luchar.

Cuando se inicie el programa se deberá pedir el nombre de los pokemon
Los demás atributos los pueden definir ustedes al momento de programar.

Una vez que se da los nombres, inicia la batalla.

Por medio de un sistema de turnos con un valor booleano y un ciclo de control se cambiará el turno del pokémon. En cada turno el usuario deberá elegir algún ataque que quiera realizar hacia el otro pokémon. La vida del pokemon atacado disminuirá dependiendo del valor del daño.
El pókémon perdedor será aquel que su vida llegue a 0.
El programa deber contener la mayoría de cosas que se han visto hasta ahora en el curso.