Trabajo práctico - Conceptos del lenguaje Java

[**Constructores**](#_f48bqatlobjb) **2**

[**Modificadores de visibilidad**](#_farr678xzhyo) **4**

[**Colecciones**](#_iqsnm9w6zmqw) **5**

[**Excepciones**](#_emh859shvdnp) **6**

[**Static**](#_fssvez452zch) **6**

[**Final**](#_zci8qcire89o) **7**

[**Clases abstractas**](#_nyg9rmremks6) **9**

[**Interfaces**](#_2eifgrseertk) **9**

[**Clases abstractas e interfaces**](#_13rwot29q0l6) **9**

[**Herencia**](#_qz0canj4om3d) **10**

[**Polimorfismo**](#_qz0canj4om3d) **10**

## 

## Constructores

### Ejercicios

1. ¿Qué salida produce este código? Justifique su respuesta.

**public** **class** PruebasConstructores {  
   
 **public** **static** void main(**String** [] args) {  
 **System**.out.println("Pruebas 1");  
 Clase3 o = **new** Clase3();  
 Clase2 o2 = **new** Clase3();  
 }  
}  
  
  
**class** Clase1 {  
 **public** Clase1() { **System**.out.print(1); }  
}  
**class** Clase2 **extends** Clase1 {  
 **public** Clase2() { **System**.out.print(2); }  
}  
**class** Clase3 **extends** Clase2 {  
 **public** Clase3() { **System**.out.println(3); }  
}

2. ¿Qué salida produce este código? Justifique su respuesta.

**public** **class** PruebasConstructores2 {   
 **public** **static** void main(**String** [ ] args) {  
 **System**.out.println("Pruebas 2");  
 ClaseC o = **new** ClaseC(1);  
 ClaseB o2 = **new** ClaseC(1);  
 ClaseB o2 = **new** ClaseC();  
 ClaseA o3 = **new** ClaseA();  
 }  
}  
  
**class** ClaseA {  
 **private** ClaseA() { **System**.out.print(10); }  
}  
**class** ClaseB **extends** ClaseA {  
 **public** ClaseB() { **System**.out.print (20); }  
}  
  
**class** ClaseC **extends** ClaseB {  
 **public** ClaseC(int A) { **System**.out.print (30); }   
}

### Ejercicios, parte 2

1. ¿Cuál será la salida por consola cuando el siguiente código java sea compilado y ejecutado? Justifique su respuesta.

**public** **class** C1 {  
 **public** C1() { **System**.out.print(1); }  
}  
**public** **class** C2 **extends** C1 {  
 **public** C2() { **System**.out.print(2); }  
}  
**public** **class** C3 **extends** C2 {  
 **public** C3() { **System**.out.print(3); }  
}  
**public** **class** Ctest {  
 **public** **static** void main (**String** [] a) {  
 C3 c = **new** C3();  
 }  
}

a) 3

b) 23

c) 32

d) 123

e) 321

f) Falla la compilación

g) Se lanza una excepción en tiempo de ejecución

### Cuestionario

1. En el constructor ¿En qué casos necesario invocar explícitamente al constructor de la clase base?
2. ¿En toda clase es necesario definir un constructor?
3. ¿Una clase abstracta debe tener constructor?
4. ¿Una interfaz puede tener constructor?
5. En una jerarquía de clases ¿En que orden se ejecutan los constructores?
6. ¿Qué ocurre si olvidamos poner super como primera línea de la subclase?
7. Los constructores no se heredan entre jerarquías de clases. Verdadero o Falso. Justifique

## Modificadores de visibilidad

Complete la siguiente tabla con el comportamiento de los modificadores de visibilidad en los atributos de una clase:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Modificadores de los atributos | | | |
| Visibilidad | private | package | protected | public |
| Misma clase |  |  |  |  |
| Mismo paquete |  |  |  |  |
| Subclase en mismo paquete |  |  |  |  |
| Subclase en otro paquete |  |  |  |  |
| Cualquier clase que no sea subclase en otro paquete |  |  |  |  |

Brindar un ejemplo de cada una de las pruebas necesarias para comprobarlo.

## Colecciones

Es sabido que el Java Collection Framework tiene herramientas muy útiles para el programador. Llene esta tabla de acuerdo a los resultados que obtenga, experimentando con él:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | ArrayList | LinkedList | TreeSet | HashMap |
| Agregar |  |  |  |  |
| Eliminar |  |  |  |  |
| Recorrido |  |  |  |  |
| Orden de los elementos |  |  |  |  |
| Escenario de uso ideal |  |  |  |  |
| Admite duplicados |  |  |  |  |

## Excepciones

Realice los siguientes pasos, completando con el código dentro de cada ítem:

1. Dentro de un método, lance una excepción del tipo ArithmeticException
2. Cambie la excepción por una IOException
3. ¿Qué diferencia encontró? ¿Por qué?
4. Cree una excepción propia, heredando de RuntimeException.
5. Muestre cómo utilizarla, afectando de la menor manera posible al contexto en que se llama al método que la lanza
6. Muestre cómo tratar una excepción registrando un mensaje de error
7. Muestre cómo tratar una excepción relanzándola con un cambio de clase
8. ¿Para qué sirve el bloque **finally**?
9. ¿Qué parte tiene más prioridad en un bloque **try-catch-finally**?
10. Si hubiera un **return** dentro del bloque **catch**, y existiera un **finally**… ¿se ejecutan ambos? ¿en qué orden?

## Static

Investigue, explique y ejemplifique los siguientes conceptos:

1. Importación static
2. Atributos static
3. Métodos static
4. Explique claramente qué significa declarar una clase como static y resuelva el ejemplo siguiente analizando en profundidad la respuesta escogida y explicando el porqué de su elección.

Teniendo en cuenta:

**public** **class** Line {  
 **public** **static** **class** Point { }  
}  
  
**class** Triangle {   
 // inserte código aquí  
}

¿Qué código, insertado en la línea señalada en el fragmento de arriba, crea una instancia de la clase Punto?

1. Point p = new Point();
2. Line.Point p = new Line.Point();
3. La clase Point no puede ser instanciada en la línea señalada.
4. Line l = new Line() ; l.Point p = new l.Point();

## Final

La palabra reservada “final” es aplicable para determinados usos. Identifique dónde es posible aplicar este concepto, defina el alcance de cada uso y dé ejemplos de los mismos.

## Clases abstractas

Realice los siguientes pasos, completando con el código dentro de cada ítem:

1. Cree una clase, y agregue un método abstracto (a través de la palabra clave abstract). ¿Qué error ocurrió? ¿Por qué?
2. Cree una clase como abstracta sin incluir ningún método abstracto, y verifique que no puede crear una instancia de esa clase.
3. Cree una clase como abstracta incluyendo sólamente un método abstracto, y verifique que tampoco puede crearse una instancia de esa clase.
4. Cree una nueva clase y haga que extienda la clase abstracta creada en (2). Instánciela. ¿Qué puede decir de lo que ha ocurrido?
5. Agregue un par de métodos abstractos a la clase abstracta utilizada en (4) y observe que errores ocurrieron. ¿Cuales son las dos maneras de solucionar estos errores (sin eliminar los métodos creados)?

## Interfaces

1. Todos los métodos en una interfaz son automáticamente públicos. ¿Es esto verdad? ¿Cuál es la utilidad?
2. Una interfaz no debería tener atributos, pero puede tener constantes. ¿Por qué sería necesario?
3. En Java 1.8 se incorporó el concepto de Métodos por Defecto, dentro de las interfaces. ¿Por qué se hizo esto? Proporcione un ejemplo de su autoría.

## Clases abstractas e interfaces

1. ¿Cómo se puede extender de más de una clase en Java?
2. ¿Cuántas interfaces se pueden implementar en una clase en Java?
3. ¿Cuántas clases se pueden extender desde una clase en Java?
4. ¿Qué diferencia hay entre implementación y extensión?

## Herencia

1. La herencia es una relación entre clases… ¿de qué tipo? ¿en qué sentido/s se puede leer? Proporcione un ejemplo de su autoría.
2. ¿Por qué es necesaria la herencia? Algunos dicen que para reutilizar código. Sin embargo, esto no es así.
3. En Java todas las clases heredan implícitamente, salvo que se explicite. ¿De quien heredan? ¿Por qué? ¿Qué permite esto?
4. Diferencie claramente y brinde ejemplos de uso de sobrecarga y sobreescritura de métodos.

## Polimorfismo

1. ¿Qué se entiende por polimorfismo?
2. ¿Qué sucedería en la POO si no existiera el concepto de Polimorfismo?
3. Proporcione un ejemplo en el que asigne polimórficamente una referencia. Cambie la asignación, y muestre en qué afecta al código preexistente, si correspondiese.
4. Defina parámetro formal y parámetro real.
5. Proporcione un ejemplo en el que se reciba un parámetro de un supertipo, y cómo se invocaría dicho método desde una clase cliente del mismo. ¿Cambiará el método original si el parámetro real fuera de una u otra subclase del parámetro formal?
6. Proporcione un ejemplo en el que se declare una lista (List) de un tipo polimórfico. ¿Qué ventajas encuentra contra declararla de un tipo específico?