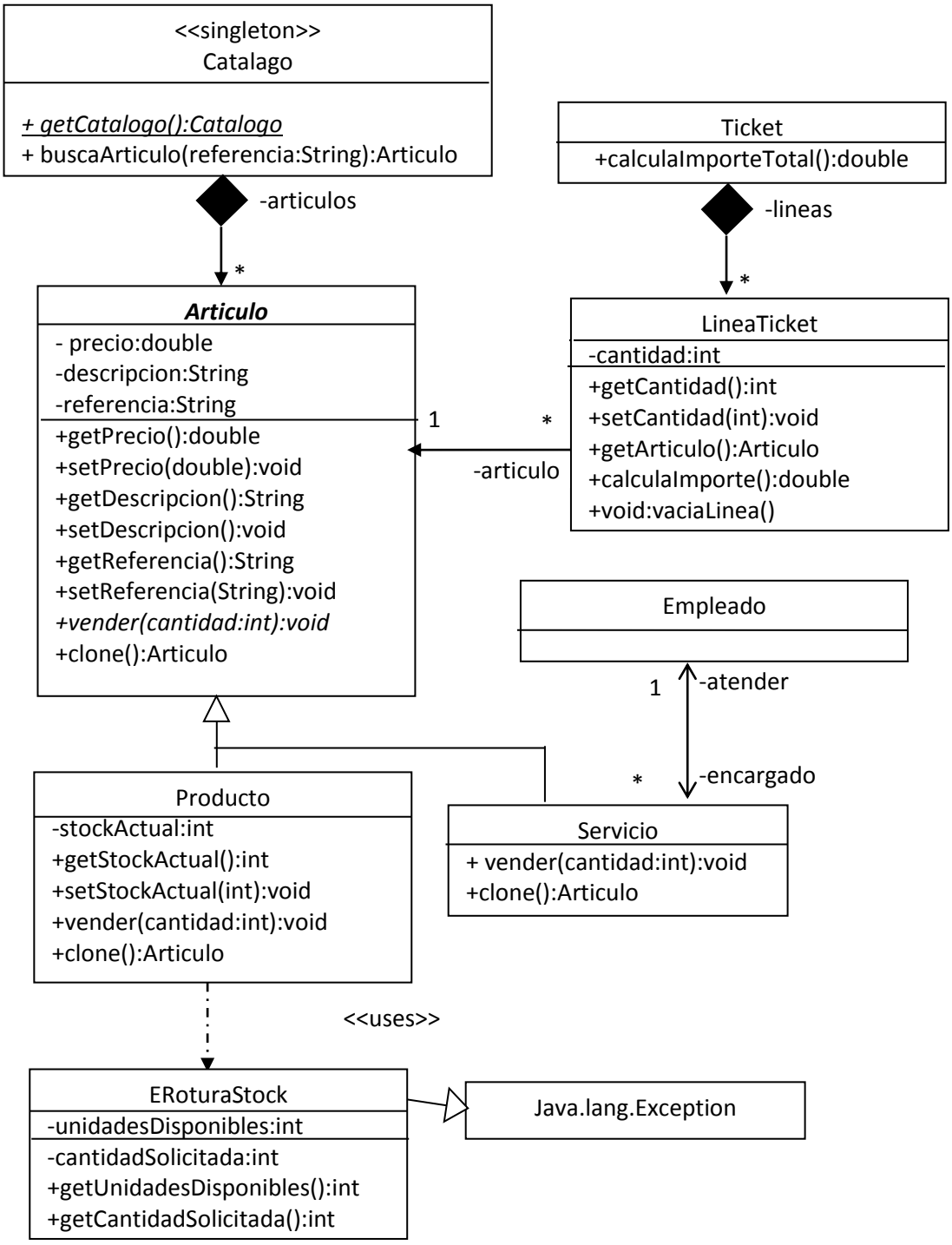
<<singleton>>
Catalago
+ getCatalogo():Catalogo
+ buscaArticulo(referencia:String):Articulo
-articulos
*
Ticket
+calculaImporteTotal():double
-lineas
*
Articulo
- precio:double
-descripcion:String
-referencia:String
+getPrecio():double
+setPrecio(double):void
+getDescripcion():String
+setDescripcion():void
+getReferencia():String
+setReferencia(String):void
+vender(cantidad:int):void
+clone():Articulo
LineaTicket
-cantidad:int
+getCantidad():int
+setCantidad(int):void
+getArticulo():Articulo
+calculaImporte():double
+void:vaciaLinea()
1
*
-articulo
Empleado
1
-atender
*
-encargado
Producto
-stockActual:int
+getStockActual():int
+setStockActual(int):void
+vender(cantidad:int):void
+clone():Articulo
Servicio
+ vender(cantidad:int):void
+clone():Articulo
<<uses>>
ERoturaStock
-unidadesDisponibles:int
-cantidadSolicitada:int
+getUnidadesDisponibles():int
+getCantidadSolicitada():int
Java.lang.Exception

1. Dado el diagrama UML, el siguiente código en java contiene un error de compilación si el método que lo contiene declara lanzar la excepción ERoturaStock.

```
Articulo a = new Articulo();
a.setDescripcion("Dinner");
a.vender(10);
```

2. Dado el diagrama UML, el siguiente código de Java contiene un error de compilacion si el método que lo contiene declara lanzar la excepción ERoturaStock.

```
Articulo a = new Producto();
a.setDescripcion("Table");
a.setStockActual(20);
a.vender(10);
```

3. Viendo el diagrama, podemos decir que diferentes objetos de tipo Ticket pueden compartir objetos de tipo LineaTicket a través de la relación del UML.

4. La clase ERoturaStock es una clase de tipo excepción y en el UML tiene una relación de dependencia con Producto porque usa la información proporcionada por esta última.

5. Si un método en la clase base indica que lanza la excepción IOException el método sobrescrito de la clase derivada puede indicar que lanza la excepción FileNotFoundException.

6. Si un método en la clase base indica que lanza la excepción IOException el método sobrescrito de la clase derivada puede indicar que lanza la excepción Exception.

7. El nuevo método podrá indicar en su clausula throws Excepciones que no aparece en el método sobrescrito.

8. El constructor de Catalogo puede ser público.

9. El método que devuelve una instancia de tipo Catalogo puede ser un método de instancia.

10. Podemos tener varias instancias de Catalogo en nuestra aplicación.

11. Podemos tener varias referencias de tipo Catalogo en nuestra aplicación.

12. Llamadas sucesivas al método getCatalogo() devolverían la misma instancia de Catalogo.

13. Catalogo solo puede tener métodos de clase.

14. El constructor de servicio podría no tener parámetros.

15. El constructor de Empleado podría no tener parámetros.

16. Existe una referencia en artículo que apunta a Producto. .

17. La relación entre el singleton y Articulo es una relación entre clases.

18. La relación entre Ticket y LineaTicket es bidireccional.

19. Un método que tiene como parámetro un Articulo puede recibir una instancia de Producto y estaremos usando el principio de sustitución.

20. El método getArticulo puede devolver un objeto de tipo Articulo, de tipo producto o de tipo servicio.

21. Catalogo solo podrá contener objetos de tipo Producto o objetos de tipo Servicio, nunca de tipo Articulo.

22. Si observamos la implementación de la relación entre Empleado y servicio, no podríamos diferenciar si es una asociación bidireccional o son dos agregaciones.

23. En el constructor de LineaTicket debería existir un objeto de tipo ticket para indicar a que ticket pertenece.

24. Tanto producto como servicio pueden refinar el método vender de Articulo.

25. Producto y Servicio reemplazarán el método vender heredado.

26. El método vender en Articulo puede ser final.

27. El enlace con el método vender se realiza en tiempo de compilación.

28. Un método de Ticket puede imprimir la descripción de los productos que contiene sin realizar downcasting.

29. Un método de Ticket puede imprimir el Stock de los productos que contiene sin realizar downcasting.

30. Un método de Ticket puede vender una unidad de todos los productos que contiene sin realizar donwncasting.

31. Es importante el orden en el cual aparecen las Excepcoines en los catch.

32. Los catch deben ir desde las excepciones mas especificas a las mas generales.

33. Los métodos virtuales son métodos abstractos.

34. Un método polimórfico puro es un método abstracto.

-

35. Para implementar la relación entre Servicio y Empleado existirá una referencia de Empleado a servicio llamada encargado y una referencia de Servicio a Empleado llamada atender.

36. Un servicio podrá ser atendido por muchos empleados y u Empleado podrá atender muchos servicios.

37. Cuando un empleado atiende un servicio se creará un objet nuevo de tipo Servicio con los datos del nuevo servicio que atenderá el empleado, esto es lo que se conoce como inclusión por valor.