RecompilacionExamesDS-1.pdf

Anónimo

Diseño Software

2º Grado en Ingeniería Informática

Facultad de Informática
Universidad de A Coruña

Dado o seguinte código, cal será o resultado da execución?

```
public class Test {

	public static class Animal {
		public void son() {
			System.out.println("SON ");
		}
	}

	public static class Can extends Animal{
		@Override
		public void son() {
			System.out.println("GUAU ");
		}
	}

	public static void main(String args[]) {
		Animal c = new Can();
		c.son();
	}
}
```

Solución: GUAU

Dado o seguinte código, cal será o resultado da execución?

```
public class Test {

	public static class Animal {
		Public void son(){
			System.out.println("SON ");
		}
	}

	public static class Can extends Animal {

	}

	public static void main(String args[]) {
		Animal c = new Can();
		c.son();
	}
}
```

Solución: SON

Que amosa o seguinte código?

```
public class AccesoEstaticos {

      public static int i;

      public static void main(String[] args) {

            AccesoEstaticos c1 = new AccesoEstaticos();
            AccesoEstaticos c2 = new AccesoEstaticos();
            c1.i = 5;
            c2.i = 10;
            System.out.println("c1.i = " + c1.i);
            System.out.println("c2.i = " + c2.i);
      }
}
```

A. `c1.i = 5 e c2.i = 10`
B. `c1.i = 10 e c2.i = 10`
C. `c1.i = 5 e c2.i = 5`
D. Erro de compilación ao acceder a un atributo estático usando o nome dunha instancia e non o nome da clase.

Solución: B

Dada a clase enumerada Cualificacion que se amosa a continuación, cal é o resultado dunha chamada ao método `Cualificacion.APROBADO.ordinal()`**?**

```
enum Cualificacion {

    SUSPENSO(0), APROBADO(5), NOTABLE(7), SOBRESAINTE(9),
MATRICULA(10);

    private int nota;

    Cualificacion(int nota) {
        this.nota = nota;
    }
}
```

A. Un erro de compilación, xa que ese método non existe en `Cualificacion`.
B. 1
C. 2
D. 2

Solución: B

WUOLAH + BBVA

1/6	BBVA está adherido al Fondo de Garantía de Depósitos de Entidades de Crédito de España. La cantidad máxima garantizada es de 100.000 euros por la totalidad de los depósitos constituidos en BBVA por persona.
Este número es indicativo del riesgo del producto, siendo 1/6 indicativo de menor riesgo y 6/6 de mayor riesgo.	

Te regalamos

15€

1

Abre tu Cuenta Online sin comisiones ni condiciones

2

Haz una compra igual o superior a 15€ con tu nueva tarjeta

3

BBVA te devuelve un máximo de 15€

Cuéntame más

¿Cómo?

WUOLAH + BBVA

Cal das seguintes sentenzas sobre o tipado é falsa?

A. No tipado estático a comprobación de tipos faise en tempo de compilación.
B. No tipado dinámico os erros tardan máis en detectarse, polo que é máis complexo solucionalos.
C. Java presenta tipado estático e forte.
D. O tipado ou é forte ou é débil, non hai valores intermedios.

Solución: D

Cal destes métodos non está na interface `java.util.Iterator<E>` da API de Java?

A. `boolean hasNext()`
B. `E next()`
C. `void set(E e)`
D. `void remove()`

Solución: C

Dado o seguinte código, indica cal das seguintes opcións é certa?

```
public interface Interface {
    int method1();
}

public interface SubInterface extends Interface {
    int method2();
    int method3();
}

public abstract class MyClass {
    public int method3() {
        return 3;
    }
}

public class MySubClass extends MyClass implements SubInterface {
    public int method2() {
        return 2;
    }
}
```

A. O código dá un erro ao compilar porque unha interface non pode extender a outra.
B. O código dá un erro ao compilar porque `MySubClass` non implementa `method1()`.
C. O código dá un erro ao compilar porque `MySubClass` non implementa `method3()`.
D. O código compila sen problemas.

Solución: B

Como se especifica en UML a visibilidade `protected` de Java?

A. Non existe ningún símbolo estándar para representar a visibilidade protexida
B. Represéntase co símbolo # (sustenido)
C. Represeéntase co símbolo ~ (til do eñe)
D. Represéntase co símbolo + (suma)

Solución: B

As clases envoltorio de Java (wrapper classes) son un exemplo do patrón…

A. Composición.
B. Método factoría.
C. Estratexia.
D. Adaptador.

Solución: D

Dadas as seguintes descricións referentes a patróns de deseño, indica cal se refire ao patrón Construtor.

A. Temos unha clase xa existente que conta moedas e devolve a cantidade total en dólares estadounidenses. Un desenvolvedor está a escribir unha nova clase que emprega a xa existente para facer o traballo, pero devolve o resultado en Euros porque é o esixido pola clase cliente que a utiliza.

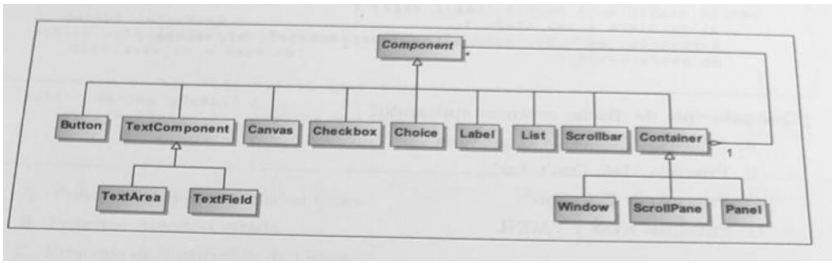
B.
```
CurrencyConverter cc1 = CurrencyConverter.incomingCurrency("USD")
                            .outgoingCurrency("EUR")
                            .build();
cc1.convert(50.00);
```

C.
```
CurrencyConverter cc1 = CurrencyConverter.getInstance();
CurrencyConverter cc2 = CurrencyConverter.getInstance();
assert cc1 == cc2;  // always passes
```

D. Un método dunha clase da interface gráfica do usuario é chamado cada vez que se preme un botón na devandita interface porque a clase que alberga ó método foi rexistrada con anterioridade para ser notificada cada vez que se prema ese botón.

Solución: B

A libraría Abstract Window Toolkit (AWT) de Java ten a seguinte estrutura. Que patrón de deseño representa?



Solución: Composición

Dado o seguinte código que pertence ao ficheiro `Clases.java` indica cal das operacións indicadas son correctas (é dicir, non dan erros de compilación e aparecen marcadas cun OK), ou incorrectas (marcadas cun NON).

```
package paqueteA;

class Clase1 {
    private int privado;
    protected int protexido;
    int paquete;
    public int publico;
}

class Clase2 {
    public void metodoDeAcceso() {
        Clase1 c = new Clase1();
        c.privado   = 1; // Operacion 4
        c.protexido = 2; // Operacion 4
        c.paquete   = 3; // Operacion 4
        c.publico   = 4; // Operacion 4
    }
}
```

A. Operación 1 (NON), Operación 2 (NON), Operación 3 (NON) e Operación 4 (OK)
B. Operación 1 (NON), Operación 2 (NON), Operación 3 (OK) e Operación 4 (OK)
C. Operación 1 (NON), Operación 2 (OK), Operación 3 (OK) e Operación 4 (OK)
D. Operación 1 (OK), Operación 2 (OK), Operación 3 (OK) e Operación 4 (OK)

Solución: C

Dado o seguinte código que compara se dous obxectos da clase Caixa son iguais, sinala a opción verdadeira.

```
public boolean equals(Object o) {
        if(o instanceof Caixa) {
            return (this.hashCode() == o.hashCode());
        } else {
            return false;
        }
}
```

A. É correcta e é a maneira máis rápida de implementar o `equals` se se implementou o `hashCode`, xa que obxectos iguais terán o mesmo `hashCode`.
B. É incorrecta, xa que dous obxectos distintos poden ter o mesmo `hashCode`
C. É incorrecta porque `hashCode` basease no método `equals`, polo que estaríamos a crear unha recursión.
D. E incorrecta porque ó método `equals` hai que pasarlle un obxecto de tipo `Caixa` por parámetro e non de tipo `Object`.

Solución: B

Cal é o resultado de executar o seguinte código?

```
public class Parametros {
    public static void manipularConta(Conta c1) {
        c1.ingreso(500);
        Conta c2 = new Conta(500);
        c1 = c2;
    }

    public static void main(String[] args) {
        Conta c = new Conta(1000);
        manipularConta(c);
        System.out.println("Saldo = " + c.getBalance());
    }
}
```

A. 500
B. 1000
C. 1500
D. Ningún dos anteriores

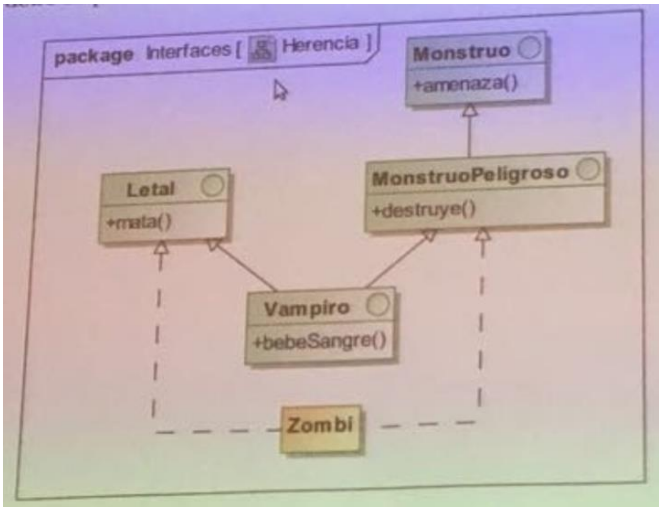
Solución: C

Os métodos construtores en Java... poden definirse privados?

A. Si: Neles baseanse patróns coma o `TypeSafe Enum` ou `Singleton`.
B. Si: Sempre e cando non foran definidos como públicos nunha superclase
C. Non: Un método construtor é un método especial que non acepta especificadores de visibilidade.
D. Non: Porque se non entón sería imposible crear instancias de dita clase.

Solución: A

Que métodos debe implementar a clase `Zombi` para que o código compile correctamente?



A. O código nunca compilará correctamente porque utiliza herdanza múltiple que non está permitida en Java.
B. `mata()` e `destruye()`
C. `mata()`, `destruye()` e `amenaza()`
D. `mata()`, `destruye()`, `amenaza()` e `bebeSangre()`

Solución: C

Dado o seguinte código, as sentenzas que se citan a continuación dan erro de COMPILACIÓN, todas menos unha, cal?

```
public abstract class Student {
    public abstract void goesToGT();
}
interface Graduate {
    public void doIt();
}
public class GettingOut extends Student implements Graduate {
    public void happy() {};
    public void doIt() {};
    public void goesToGT() {};
}
public class Alumni extends GettingOut {
    public void newDo() {};
}
public class GradStudent extends Student {
    public void something() {};
    public void goesToGT() {};
}
```

A. `Student t = new Student();`

B.
```
Student t = new GradStudent();
t.something();
```

C.
```
Student t = new Alumni();
((GradStudent) t).something();
```

D. `Alumni t = new GettingOut();`

Solución: C

Nota: fálase de erro de compilación. A opción correcta daría erro en tempo de execución.

Dada unha clase Animal y unha subclase da mesma denominada Can, ¿cuál sería el resultado de la ejecución del siguiente código?

```
Animal a = new Can();
if (a instanceof Can) System.out.print(" Can ");
if (a instanceof Animal) System.out.print(" Animal");
```

A. Can
B. Animal
C. Can Animal
D. Non se amosaría nada senón que obteríamos un erro de compilación

Solución: C

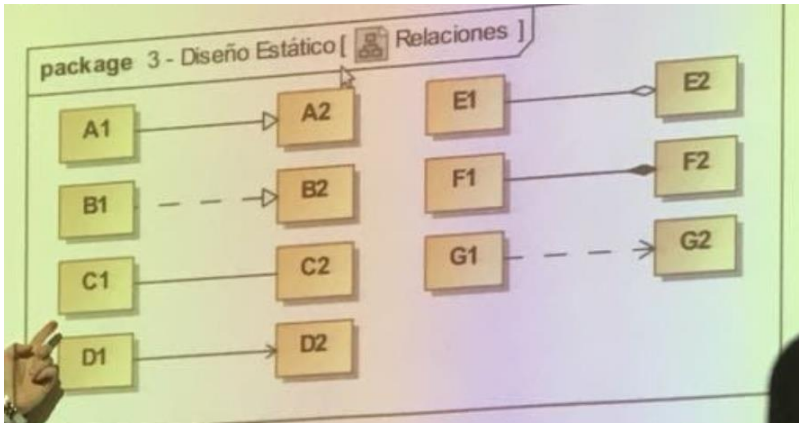
Creouse o seguinte método para imprimir, de maneira xenérica, todos os elementos de calquera lista de elementos de calquera tipo. Sinala cal das seguintes opcións é a correcta.

```
public static void printList(List<Object> l) {
    for (Object e : l) {
        System.out.println(e);
    }
}
```

A. O método non é correcto porque só amosaría coleccións de obxectos `Object`, pero non coleccións de calquera subclase de `Object`.
B. O método non é correcto, deberíase cambiar `<Object>` polo comodín `<? super Object>`.
C. O método é correcto aínda que non cumpre co principio *Get e Put*.
D. O método é correcto e cumpre co principio *Get e Put*.

Solución: A

Identifica as seguintes relacións:

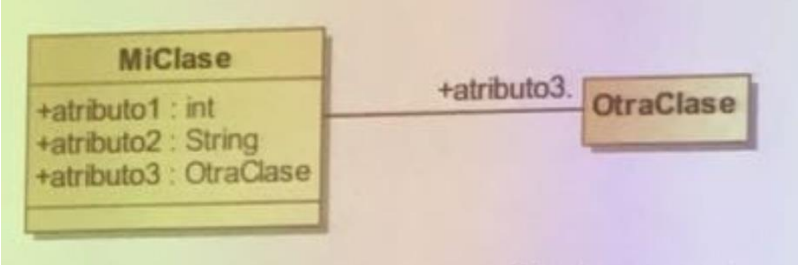


A: Xeneralización B: Realización C: Asociación D: Asociación navegada

E: Agregación F: Composición G: Dependencia

Dado o seguinte código e a seguinte implementación en UML, sinala cal das seguintes opción é a correcta.

```
Class Miclase {
      public int atributo1;
      public String atributo2;
      public OtraClase atributo3;
}
```



A. O diagrama NON é correcto porque o `atributo1` debería representarse polo tipo estándar de UML denominado `integer`.
B. O diagrama NON é correcto porque o `atributo2` debería representarse como unha asociación e non coma un atributo.
C. O diagrama NON é correcto porque o `atributo3` debería representarse só como unha asociación.
D. O diagrama SÍ é correcto.

Solución: C

Dado o seguinte código, indica cal das seguintes sentenzas é verdadeira:

```
class Clase1 {
    public Conta metodo1(int tipo, int saldo) {
        switch (tipo) {
            case 1  : return new FondoGarantido(saldo);
            case 2  : return new FondoRentaFixa(saldo);
            case 3  : return new FondoMixto(saldo);
            case 4  : return new FondoRendaVariable(saldo);
            case 5  : return new FondoCapitalRisco(saldo);
            default : return new FondoGarantido(saldo);
        }
    }
}
```

A. Trátase dun exemplo do patrón método factoría, pero na súa versión parametrizada.
B. Trátase dunha alternativa ao patrón estado, o `metodo1` fai unha cousa ou outra dependiendo do estado que se lle pasa por parámetro.
C. Trátase do patrón estratexia, o parámetro tipo indica a estratexia a seguir nas inversión que crean os clientes.
D. Trátase realmente dun antipatrón, un código que debería evitarse.

Solución: A

Que dúas posibles maneiras existen de representar este código en UML?

```
interface Produto {
    int prezo ();
    String descricion();
}

class Libro implements Produto {
    public String titulo() {...};
    public String isbn() {...};
    public int prezo() {...}
    public String descricion() {...}
}

class UsaProduto {
    Produto p;
}
```