


Dadas estas clases:

```
class A {  
    protected int a;  
    public A() { a=0; }  
    public A(int x) { a= x; }  
    public int getA() { return a; }  
}  
  
class B extends A {  
    private int b;  
    public B() { b=0; }  
    public B(int y) { b=y; }  
    public int getA() { return a+3; }  
    public int getB() { return b; }  
}
```

Si tenemos una referencia 'A objeto = new B(3);', el resultado de llamar a 'objeto.getA()' es

Seleccione una o más de una:

- ☐ a. 1
- ☐ b. Un error de compilación.
- ☒ c. 3 
- ☐ d. 0

Respuesta correcta. 'B' puede acceder a las propiedades protegidas de 'A' y se ejecuta la implementación de 'B'.

La respuesta correcta es: 3


Dadas estas clases:

```
class A {  
    protected int a;  
    public A() { a=0; }  
    public A(int x) { a= x; }  
    public int getA() { return a; }  
}
```

```
class B extends A {  
    private int b;  
    public B() { b=0; }  
    public B(int y) { b=y; }  
  
    public int getB() { return b; }  
}
```

Si tenemos una referencia '**B objeto = new B(3);**', el resultado de llamar a 'objeto.getA()' es

Seleccione una o más de una:

- ☐ a. 3
- ☐ b. Un error de compilación.
- ☒ c. 0 
- ☐ d. 1

Respuesta correcta.

'B' ha heredado getA() de 'A'.


La respuesta correcta es: 0

Dadas estas clases:

```
class A {  
    protected int a;  
    public A() { a=0; }  
    public A(int x) { a= x; }  
    public int getA() { return a; }  
}  
  
class B extends A {  
    private int b;  
    public B() { b=0; }  
    public B(int y) { b=y; }  
    public int getA() { return a+3; }  
    public int getB() { return b; }  
}
```

Si tenemos una referencia '**B objeto = new A(3);**', el resultado de llamar a 'objeto.getA()' es

Seleccione una o más de una:

- ☒ a. 3 
- ☐ b. 1
- ☐ c. 0
- ☐ d. Un error de compilación.

Respuesta incorrecta.

La respuesta correcta es: Un error de compilación.

Si a partir de una clase Vehiculo, creo mediante herencia la clase Coche y la clase Motocicleta, estoy haciendo

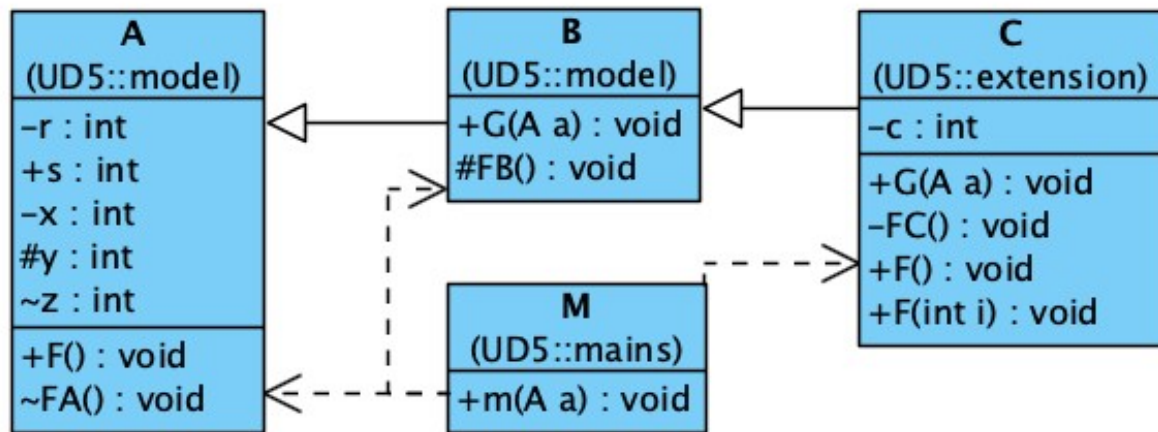
Seleccione una o más de una:

- ☒ a. Una especialización ✓
- ☐ b. Una generalización
- ☐ c. Una excepción
- ☐ d. Una variación

Respuesta correcta

Piensa en las clases como conjuntos que contienen a otros conjuntos (subclases).

La respuesta correcta es: Una especialización



Usando este diagrama, selecciona la opción correcta. En el código, donde aparecen tres puntos (...) estamos indicando que hay más código que no mostramos.

```

package mains;
import model.A;
import model.B;

public class M {
    public void m(A a) {
        if (a instanceof B) {
            ((B)a).G(new A());
        }
    }
}
  
```

Correcto



```

package mains;
import model.A;
import model.B;

public class M {
    public void m(A a) {
        if (a instanceof B) {
            a.G(new A());
        }
    }
}
  
```

No compila



```
package mains;

import model.*;
import extension.C;

public class M {

    public void m(A a) {
        a = new C();
        A a2 = new B();
        a = a2;
        a.F();
    }

}
```

Ejecuta la implementación de F() en A ⇅



```
package mains;

import model.A;
import extension.C;
```

```
public class M {
    public void m(A a) {
        A c = new C();
        c.F(4);
    }
}
```

No compila



```
package mains;

import model.A;
import model.B;
```

```
public class M {

    public void m(A a) {

        A b = new B();
        b.G(a);
        b.F();
    }
}
```

No compila



```
package mains;
import model.A;

public class M {
    public void m(A a) {
        a.FA();
    }
}
```

No compila




```
package mains;

import model.A;
import model.B;

public class M {

    public void m(A a) {
        B b = new B();
        b.G(a);
        b.F();
    }
}
```

Ejecuta la implementación de F() en A ▾



```
package mains;

import model.A;
import extension.C;

public class M {

    public void m(A a) {
        A c = new C();
        c.F();
    }
}
```

Ejecuta la implementación de F() en C ▾



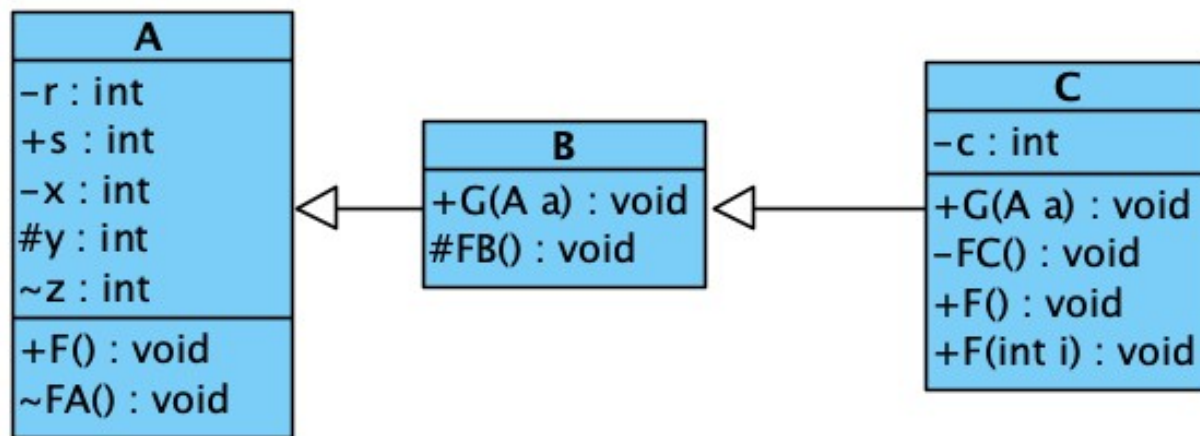
Cuando a partir de una superclase, creamos subclases, decimos que éstas últimas

Seleccione una o más de una:

- ☒ a. heredan las propiedades de la superclase ✓
- ☒ b. refinan el comportamiento de la superclase ✗
- ☒ c. reemplazan el comportamiento de la superclase ✗
- ☐ d. restringen el comportamiento de la superclase
- ☒ e. extienden el comportamiento de la superclase ✓

Respuesta correcta: heredan las propiedades de la superclase y pueden añadir propiedades (extensión de la superclase).

Las respuestas correctas son: heredan las propiedades de la superclase, extienden el comportamiento de la superclase



Usando como referencia el diagrama de clases, rellena con las palabras clave correctas el código. Si no hay que escribir nada, introduce la cadena VACIO.

```
package model;
```

```
public class A {
```

```
    private int r;
```

```
    public int s;
```

```
    private int x;
```

```
    protected int y;
```

```
    VACIO int z;
```

```
    public void F() {
```

```
    }
```

```
    VACIO void FA() {
```

```
    }
```

```
}
```

```
package model;
```

```
public B extends A {  
    public void G(A a) {  
    }  
    protected void FB() {  
    }  
}
```

```
package extension;
```

```
import model.B;
```

```
public C extends B {  
    private int c;  
    public void G(A a) {  
    }  
    private void FC() {  
    }  
    public void F() {  
    }  
    public void F(int i) {  
    }  
}
```

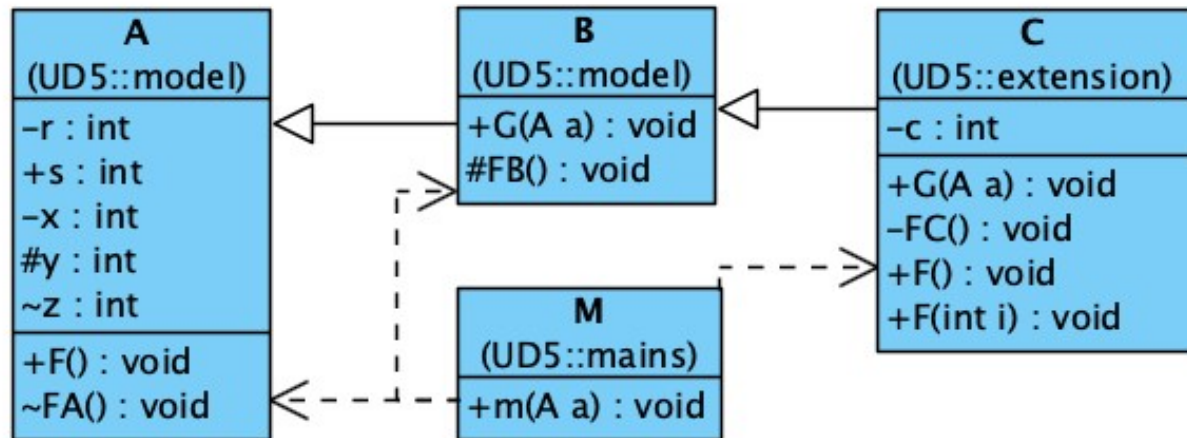
Si mi clase 'MiExcepcion' hereda directamente de la clase 'Exception' y esta hereda a su vez de la clase 'Object' entonces

Seleccione una o más de una:

- ☒ a. 'Object' es una superclase de 'MiExcepcion' ✓
- ☐ b. 'MiExcepcion' es una superclase de 'Object'
- ☒ c. 'MiExcepcion' es una subclase de 'Object' ✓
- ☐ d. 'Object' es una subclase de 'MiExcepcion'

Respuesta correcta. La herencia es transitiva.

Las respuestas correctas son: 'MiExcepcion' es una subclase de 'Object', 'Object' es una superclase de 'MiExcepcion'



Usando este diagrama, selecciona la opción correcta. En el código, donde aparecen tres puntos (...) estamos indicando que hay más código que no mostramos.

```

package extension;

import model.B;

public class C extends B {

    ...

    private void FC() {

        FB();

    }

    ...

}
  
```


Correcto



```
package extension;
import model.B;

public class C extends B {
    ...

    private void FC() {
        FA();
    }
    ...
}
```

No compila porque el acceso es de paquete ▾ 

```
package model;

public class B extends A {
    ...


    public void G(A a) {
        FA();
    }
    ...
}
```

Correcto ▾ 

```
package model;

public class B extends A {
    ...

    public void G(A a) {
        int cont = a.r;
    }
    ...
}
```

No compila porque el acceso es privado ▾ 


```
package model;

public class B extends A {

    ...

    public void G(A a) {

        this.y = a.s;

    }

    ...

}
```

Correcto



```
package model;

public class B extends A {

    ...

    public void G(A a) {

        this.r = a.s;

    }

    ...

}
```

No compila porque el acceso es privado



```
package extension;

import model.B;

public class C extends B {

    ...

    public void G(A a) {

        this.c = a.y;

    }

    ...

}
```

Correcto



package model;

public class B extends A {

...

public void G(A a) {

 this.z = a.y;

}

...

}

Correcto



```
package model;
public class A {
    ...

    void FA() {
        r = s = x = y = z;
    }
}
```

Correcto



```
package extension;  
  
import model.B;  
  
public class C extends B {  
  
    ...  
  
    public void G(A a) {  
  
        this.c = a.z;  
  
    }  
  
    ...  
  
}
```

No compila porque el acceso es de paquete



La herencia de implementación...

Seleccione una o más de una:

- ☒ a. nos permite reutilizar código, ya que heredamos tanto la interfaz como la implementación de la superclase. ✓
- ☐ b. sólo nos permite heredar la implementación de la superclase, no su interfaz.
- ☐ c. nos permite reutilizar conceptos, es decir, la interfaz de la clase, pero no código (implementación).

Respuesta correcta

La respuesta correcta es: nos permite reutilizar código, ya que heredamos tanto la interfaz como la implementación de la superclase.

Decimos que estamos haciendo herencia simple de implementación...

Seleccione una o más de una:

- ☐ a. cuando las instancias de las subclases representan a todas las instancias de la superclase.
- ☒ b. cuando una clase sólo tiene una superclase. ✓
- ☐ c. cuando hay instancias de la superclase que no están representadas por alguna subclase.
- ☒ d. cuando una clase sólo tiene una subclase. ✗

Respuesta correcta

La respuesta correcta es: cuando una clase sólo tiene una superclase.

Las subclases...

Seleccione una o más de una:

- ☒ a. pueden añadir atributos y métodos nuevos. ✓
- ☐ b. pueden añadir métodos nuevos pero no atributos.
- ☒ c. pueden modificar la implementación de los métodos heredados. ✓
- ☐ d. no pueden modificar la implementación de los métodos heredados.

Respuesta correcta: La subclase puede extender a la superclase (añadir métodos o atributos nuevos) o modificar su comportamiento (la implementación de los métodos heredados).

Las respuestas correctas son: pueden modificar la implementación de los métodos heredados., pueden añadir atributos y métodos nuevos.

Si 'B' es una subclase de 'A', un método de instancia de 'B'...

Seleccione una o más de una:

- ☒ a. siempre puede acceder a las propiedades de 'A' que tienen visibilidad pública. ✓
- ☐ b. siempre puede acceder a las propiedades de 'A' que tienen visibilidad privada.
- ☒ c. siempre puede acceder a las propiedades de 'A' que tienen visibilidad protegida. ✓
- ☐ d. siempre puede acceder a las propiedades de 'A' que tienen visibilidad de paquete (package).

Respuesta correcta. A las propiedades privadas de 'A' sólo puede acceder 'A'. A las propiedades 'package' sólo las clases que están en el mismo paquete que 'A'.

Las respuestas correctas son: siempre puede acceder a las propiedades de 'A' que tienen visibilidad protegida., siempre puede acceder a las propiedades de 'A' que tienen visibilidad pública.

Dadas estas clases:

```
class A {  
  
    public A() {}  
  
    public A(int x) {}  
  
}
```

```
class B extends A {  
  
    public B() {}  
  
    public B(int y) {}  
  
}
```

Seleccione una o más de una:

- ☒ a. B.B(int) invoca implícitamente a A.A() ✓
- ☐ b. B.B(int) invoca implícitamente a A.A(int)
- ☒ c. B.B() invoca implícitamente a A.A() ✓
- ☐ d. B.B(int) invoca implícitamente a A.A() y luego a A.A(int)

Respuesta correcta. El único constructor de la superclase que se invoca implícitamente desde la subclase es el constructor por defecto.

Las respuestas correctas son: B.B() invoca implícitamente a A.A(), B.B(int) invoca implícitamente a A.A()

Dadas estas clases:

```
class A {  
  
    public A() {}  
  
    public A(int x) {}  
  
}
```

```
class B extends A {  
  
    public B() {}  
  
    public B(int y) {}  
  
}
```

cuando hacemos 'new B(3)'...

Seleccione una o más de una:

- ☐ a. Primero se ejecuta A.A(int) y luego B.B(int)
- ☒ b. Primero se ejecuta A.A() y luego B.B(int) ✓
- ☐ c. Primero se ejecuta B.B(int) y luego A.A(int)
- ☐ d. Primero se ejecuta B.B(int) y luego A.A()

Respuesta correcta. Primero se ejecuta el constructor de la superclase.

La respuesta correcta es: Primero se ejecuta A.A() y luego B.B(int)

Dadas estas clases:

```
class A {  
    private int a;  
    public A() { a=0; }  
    public A(int x) { a= x; }  
}
```

```
class B extends A {  
    private int b;  
    public B() { b=0; }  
    public B(int y) { b=y; }  
}
```

Si tenemos una referencia 'A objeto;'...

Seleccione una o más de una:

- ☐ a. podemos asignarle cualquier tipo de objeto. : 'objeto = new Object();'
- ☐ b. sólo podemos asignarle objetos de tipo 'A': 'objeto = new A();'
- ☒ c. podemos asignarle objetos de tipo 'A' directamente ('objeto = new A();') y objetos de tipo 'B' mediante upcasting explícito: 'objeto = (A) new B();' ✓
- ☒ d. podemos asignarle objetos de tipo 'A' y 'B'. : 'objeto = new A();' y : 'objeto = new B();' ✓

Respuesta correcta. El upcasting siempre se puede hacer, implícita o explícitamente.

Las respuestas correctas son: podemos asignarle objetos de tipo 'A' y 'B'. : 'objeto = new A();' y : 'objeto = new B();', podemos asignarle objetos de tipo 'A' directamente ('objeto = new A();') y objetos de tipo 'B' mediante upcasting explícito: 'objeto = (A) new B();'

Dadas estas clases:

```
class A {  
    private int a;  
    public A() { a=0; }  
    public A(int x) { a= x; }  
    public int getA() { return a; }  
}
```

```
class B extends A {  
    private int b;  
    public B() { b=0; }  
    public B(int y) { b=y; }  
    public int getA() { return 1; }  
    public int getB() { return b; }  
}
```

Si tenemos una referencia 'A objeto = new B(3);', el resultado de llamar a 'objeto.getA()' es

Seleccione una o más de una:

- ☒ a. 1 ✓
- ☐ b. Un error de compilación.
- ☐ c. 0
- ☐ d. 3

Respuesta correcta. Se ejecuta la implementación de la subclase, por que el objeto 'en tiempo de ejecución' al que apunta 'objeto' es de tipo subclase.

La respuesta correcta es: 1