

Pregunta 1: Sobre los objetos, se puede decir (indica la respuesta **falsa**)

- a. Los objetos son especificados por las clases.
- b. Los objetos representan instancias de las clases.
- c. Los objetos se comunican con las clases con métodos.
- d. Una clase puede manejar objetos.

(c). Según el libro base "Podemos comunicarnos con objetos invocando sus métodos". Los objetos se comunican con otros objetos (que son instancias de clases), pero no con las clases directamente.

Pregunta 2: Dado este fragmento de código, ¿cuál sería el resultado de compilar/ejecutar e

Número de Línea	Código
4	public static void main(String[] args) {
5	int valor = 5;
6	cambiarValor(valor);
7	System.out.println(valor);
8	}
9	public static void cambiarValor(int valor) {
10	private int valor = valor * 2;
11	}

- a. 5
- b. 10
- c. Error en la línea 10
- d. Error en la línea 6

Respuesta:(C)

El resultado es error en la línea 10 porque dentro de un método no pueden declararse variables locales precedidas de modificadores de acceso (public/private/protected) , se puede como "final".

Pregunta 3: Dado el siguiente código, ¿cuál será su salida?

Número de Línea	Código
4	class Cantante { public static String cantar() { return "la"; } }
5	public class Tenor extends Cantante {
6	public static String cantar() { return "fa"; }
7	public static void main(String[] args) {
8	Tenor t = new Tenor();
9	Cantante s = new Tenor();
10	System.out.println(t.cantar() + " " + s.cantar());
11	}
12	}

- a. fa fa
- b. fa la
- c. la la
- d. la fa

(B) fa la esto es lo que da al compilar y ejecutar

Pregunta 4: ¿Cuál de los siguientes condicionales compilaría sin errores?

```
int[] array = new int[15];  
    // EL CÓDIGO IRÍA AQUÍ  
array[j] = j;
```

- a. for (int j=0; j<array.length;j++)
- b. for (int j=0; j<array.length();j++)
- c. for (int j=0; j<array.size;j++)
- d. for (int j=0; j<array.size();j++)

(A) La "a" compila sin errores. Y el array j contiene 0,1,2....13,14

Pregunta 5: Tienes que hacer una clase que almacena objetos únicos. No es necesario que estén ordenados. ¿Qué interfaz sería la más apropiada implementar en esta clase?

- a. Set
- b. List
- c. Map
- d. Vector

Set no permite elementos duplicados pueden estar o no ordenados.

List permite elementos duplicados y colecciones ordenadas.

Pregunta 6: Cuando varios componentes de un software colaboran para completar una misma tarea se dice que entre ellos hay ...

- a. una clase clara y bien definida.
- b. una instancia clara y bien definida.
- c. una interfaz clara y bien definida.
- d. un proceso claro y bien definido.

(C) Cuando varios componentes de un software colaboran para completar una misma tarea se dice que entre ellos hay una interfaz clara y bien definida

Pregunta 7: Dado el siguiente código, ¿cuál será su salida?

Número de Línea	Código
4	class Vehiculo {
5	public void imprimirSonido() {
6	System.out.print("Vehiculo");
7	}
8	}
9	class Coche extends Vehiculo {
10	public void imprimirSonido() {
11	System.out.print("Coche");
12	}
13	}
14	class Bicicleta extends Vehiculo {
15	public void imprimirSonido() {
16	System.out.print("Bicicleta");
17	}
18	}
19	public class Test {
20	public static void main(String[] args) {
21	Vehiculo v = new Coche();
22	Bicicleta b = (Bicicleta) v;
23	v.imprimirSonido();
24	b.imprimirSonido();
25	}
26	}

- a. Fallo de compilación.
- b. Lanza una excepción en tiempo de ejecución.
- c. Imprime "VehiculoCoche".
- d. Imprime "BicicletaBicicleta".

(B) ; falla en la línea 22 en tiempo de ejecución.

Pregunta 8: ¿Qué pasará si se compila / ejecuta este código?

Número de Línea	Código
4	class Padre {}
5	class Hijo extends Padre {}
6	class Hijo2 extends Padre {}
7	public class CEx{
8	public static void main(String[] args){
9	Padre p=new Padre();
10	Hijo h=(Hijo) p;
11	}
12	}

- a. El código compilará y se ejecutará sin errores.
- b. El código daría un error a compilar.
- c. El código daría un error a ejecutar.
- d. El código no daría ningún error; sin embargo, h no tendría el tipo deseado.

(C) El código da un error al ejecutar en la línea 10.

Pregunta 9: ¿Cuál sería el resultado de ejecutar el método `goo()`?

Número de Línea	Código
4	<code>public void goo() {</code>
5	<code> foo f = new foo();</code>
6	<code> System.out.println(f);</code>
7	<code>}</code>
8	<code>public class foo {</code>
9	<code> String f = "22";</code>
10	<code> public String toString(){</code>
11	<code> return("44");</code>
12	<code>}</code>
13	<code> public foo(){}</code>
14	<code>}</code>

- a. null
- b. 22
- c. 44
- d. Un error de ejecución

(C) lo que se está imprimiendo es el objeto `foo=f` no la variable `String f`, y el objeto retorna 44.

Pregunta 10: ¿Cuál de las siguientes definiciones es correcta para una clase abstracta?

- a. `abstract Animal {abstract void ladrar();}`
- b. `class abstract Animal {abstract void ladrar();}`
- c. `abstract class Animal {abstract void ladrar();}`
- d. `abstract class Animal {abstract void ladrar() {System.out.println("RRRRRRR");}}`

(C) La respuesta correcta es la C es la única que no da fallo al compilar.

Pregunta 11: ¿Cómo podemos detectar que el usuario ha hecho click en un botón en una interfaz Swing?

- a. Implementando `public void actionPerformed(ActionEvent e)` de la interfaz `ActionListener`
- b. Implementando `public void actionPerformed(ActionEvent e)` de la interfaz `ActionListener`
- c. Implementando `public boolean actionPerformed(ActionEvent e)` de la interfaz `ActionListener`
- d. Implementando `public void actionPerformed(Event e)` de la interfaz `ActionListener`

(B)

Pregunta 12: Las sentencias de código que podrían lanzar una excepción se protegen de la siguiente manera:

```
try {  
    // aquí se protege una o más sentencias  
}  
  
_____  
    // aquí se informa y se recupera de la excepción  
}
```

¿Qué habría que colocar en el hueco correspondiente?

- a. `catch (exception Exception)`
- b. `catch (Event exception)`
- c. `catch (Exception exception)`
- d. `catch (event Exception)`

(C)

`try {`

`} catch (ExceptionType name) {}`

Pregunta 13: Las clases de un sistema corresponden a las X y los métodos a las Y. Donde X y Y son ...

- a. X = verbos, Y = sustantivos
- b. X = sustantivos, Y = verbos
- c. X = sustantivos, Y = sustantivos
- d. X = verbos, Y = verbos

Pregunta 14: ¿Cómo se puede crear una nueva instancia de la clase Vector y añadir un elemento? (indica la respuesta falsa)

- a. `Vector<Integer> v = new Vector<Integer>();v.add(99);`
- b. `Vector<Integer> v = new Vector<Integer>(99);v.add(99);`
- c. `Vector<Integer> v = new Vector<Integer>(99, 99);v.add(99);`
- d. `Vector<Integer> v = new Vector<Integer>(99);v.add(99, 99);`

(D) `add` se utiliza para añadir un elemento en la última posición del vector, para añadir en una posición determinada se utiliza `v.insertElementAt (valor, posición en el vector);`

-se crea un vector cuya dimensión inicial es 10.

`Vector vector=new Vector ();`

La dimensión del vector se duplica si se rebasa la dimensión inicial, por ejemplo, cuando se pretende guardar once elementos.

-se le pasa la dimensión inicial.

`Vector vector=new Vector (20);`

Si se rebasa la dimensión inicial guardando 21 elementos, la dimensión del vector se duplica.

-Cuando creamos un vector u objeto de la clase Vector, podemos especificar su dimensión inicial, y cuanto crecerá si rebasamos dicha dimensión.

`Vector vector=new Vector (20, 5);`

Tenemos un vector con una dimensión inicial de 20 elementos. Si rebasamos dicha dimensión y guardamos 21 elementos la dimensión del vector crece a 25.

-Hay dos formas de añadir elementos a un vector. Podemos añadir un elemento a continuación del último elemento del vector, mediante la función miembro `addElement`.

`v.addElement("uno");`

Podemos también insertar un elemento en una determinada posición, mediante

`insertElementAt:`

`v.insertElementAt ("tres", 2);`

Pregunta 15: Sobre una variable local que se declara dentro del bloque "try", se puede decir que ...

- a. es visible dentro de los bloques "catch" y "finally".
- b. es visible dentro del bloque "catch" pero no del bloque "finally".
- c. es visible dentro del bloque "finally" pero no del bloque "catch".
- d. no es visible dentro de los bloques "catch" y "finally".

Las variables locales de un bloque **Try** no se encuentran disponibles en un bloque **Catch** porque se trata de bloques independientes. Si se desea utilizar una variable en más de un bloque, se debe declarar la variable fuera de la estructura **Try...Catch...Finally**.

PARTE PRÁCTICA [6,5 PUNTOS]:

Una empresa de alquiler de automóviles tiene a su disposición un conjunto de vehículos indicados en la siguiente tabla. Se quiere diseñar e implementar un programa que almacene y gestione la información relacionada con estos vehículos.

Tipo de vehículo	Características
Motos	marca, matrícula, número de identificación, número de kilómetros, estado actual de depósito de gasolina.
Coches (turismos)	marca, matrícula, número de identificación, tipo (normal / familiar), número de puertas, número de kilómetros, tipo de motor (gasolina / gasoil), estado actual del depósito.
Coches (deportivos)	marca, matrícula, número de identificación, capacidad de motor, número de kilómetros, turbo o no, número de puertas, número de asientos, estado actual del depósito de gasolina.
Coches (4x4)	marca, matrícula, número de identificación, número de kilómetros, tipo de motor (gasolina / gasoil), número de asientos, estado actual de depósito.
Monovolúmenes	marca, matrícula, número de identificación, número de kilómetros, número de puertas, puertas laterales, número de asientos, tipo de motor (gasolina / gasoil), capacidad de carga, estado actual del depósito.
Furgonetas	marca, matrícula, número de identificación, número de kilómetros, capacidad de carga, altura, estado actual del depósito de gasoil.

Se pide:

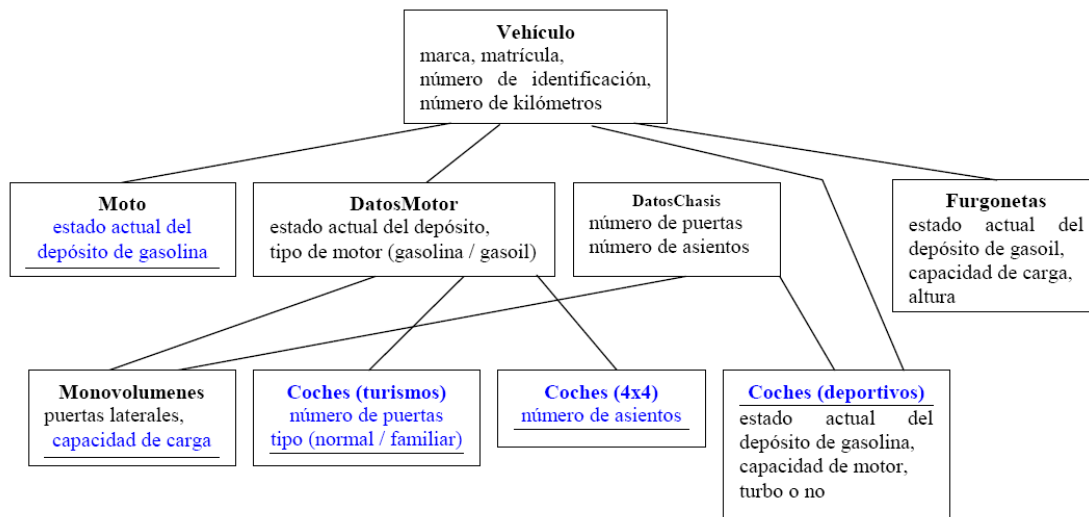
- 1) **[1,5 puntos]** Identificar la estructura y las relaciones de herencia y de uso de las clases necesarias para almacenar y gestionar esta información.
- 2) **[1,5 puntos]** Dibujar un esquema de la organización de estas clases en el diseño global.
- 3) **[2,0 puntos]** Implementar la especificación de las clases.
- 4) **[1,5 puntos]** Se quiere declarar un array de objetos para almacenar todos los vehículos, independientemente del tipo.
 - a) ¿Cómo declararías el array?
 - b) ¿Qué métodos se necesitan para acceder a un vehículo concreto?
 - c) ¿Cómo se almacenan las diferencias entre los distintos tipos de vehículos?

SOLUCIÓN:

1. Dados las características de los vehículos, se pueden agrupar los datos para ver los siguientes elementos comunes:

	Motos	Coches (turismos)	Coches (deportivos)	Coches (4x4)	Monovolúmenes	Furgonetas
Datos comunes	marca, matrícula, número de identificación, número de kilómetros	marca, matrícula, número de identificación, número de kilómetros	marca, matrícula, número de identificación, número de kilómetros	marca, matrícula, número de identificación, número de kilómetros	marca, matrícula, número de identificación, número de kilómetros	marca, matrícula, número de identificación, número de kilómetros
		estado actual del depósito		estado actual del depósito	estado actual del depósito	
		tipo de motor (gasolina / gasoil)		tipo de motor (gasolina / gasoil)	tipo de motor (gasolina / gasoil)	
			número de puertas		número de puertas	
			número de asientos		número de asientos	
Datos específicos	estado actual del depósito de gasolina	número de puertas tipo (normal / familiar)	estado actual del depósito de gasolina, capacidad de motor, turbo o no	número de asientos	puertas laterales, capacidad de carga	estado actual del depósito de gasoil, capacidad de carga, altura

Con herencia múltiple, se tienen las siguientes relaciones de herencia:



2. Se puede ver la especificación de estas clases en Java a continuación:

```

class Vehiculo {
    private String marca;
    private String matrícula;
    private int id;
    private int km;
}

class Moto extends Vehiculo {
    private int depGasolina;
}

class Furgonetas extends Vehiculo {
    private int depGasoil;
    private int capCarga;
    private int altura;
}

class DatosMotor extends Vehiculo {
    private int deposito;
    private boolean gasolina;
}

class CochesTurismos extends DatosMotor {
    private int puertas;
    private boolean normal;
}

class Coches4x4 extends DatosMotor {
    private int asientos;
}

class DatosChasis {
    private int puertas;
    private int asientos;
}

class CochesDeportivos extends DatosChasis {
    private int depGasolina;
}
  
```

```

        private int capMotor;
        private boolean turbo;
    }

    //en este caso, como no se puede tener la herencia múltiple en Java
    //hay que usar una versión de la clase DatosChasis
    class Monovolumenes extends Vehiculo {
        DatosChasis dc;
        private int puertasLaterales;
        private int capCarga;
    }

```

3.

a. Se declara el array como la clase base, `Vehiculo`, así, como todas los vehículos están basados en esta clase se puede guardar cualquier tipo de vehículo allí:

```

Vehiculo vehiculos[] = new Vehiculo[5];
vehiculos[0] = new Moto("Yamaha", "QHW1278", 123, 50, 100);
vehiculos[1] = new CocheTurismo("Renault", "nht1980", 987, 20050, 100, true, 4, true);
vehiculos[2] = new Coche4x4("Jeep", "mfr3645", 454, 1809, 58, false, 2);
vehiculos[3] = new CocheDeportivo("Jaguar", "bhy4567", 480, 23, 3, 2, 98, 4500, true);
vehiculos[4] = new Monovolumen("Citroën", "mnb0987", 788, 34500, 1, 5, 4, 8);

```

b. Se puede acceder a cada objeto dentro del array a través del número de su posición en el array. Pero para poder acceder a los datos dentro de cada objeto hay que tener o un método virtual general para devolver todos los datos (el mismo método en cada clase) o un método virtual para cada tipo de datos, como por ejemplo, para indicar el contenido del depósito de gasolina:

```

class Vehiculo {
    //código no incluido
    public int getDepGas(){}
}

class Moto extends Vehiculo {
    //código no incluido
    public int getDepGas(){
        return depGasolina;
    }
}

```

c. Se puede almacenar las diferencias entre los tipos de dos maneras:

1. Usando el constructor de cada tipo de vehículo como se ha hecho arriba.
2. Incorporando métodos virtuales para cada tipo de datos, como por ejemplo:

```

class Vehiculo {
    //código no incluido
    public void setDepGas(){}
}

class Moto extends Vehiculo {
    //código no incluido
    public void setDepGas(int g){
        depGasolina = g;
    }
}

```