

CONCURSO PROGRAMACIÓN

HASTA 0,5 PUNTOS EXTRA

LA ÚNICA REGLA... ES QUE NO HAY REGLAS!

3 DEC - CASTELAO

3 EQUIPOS DE DESARROLLADORES COMPETIRÁN HASTA EL FINAL POR HACERSE CON LA VICTORIA

ATENCIÓN! Pueden producirse bugs e incluso llegar a oírse algún taco. No recomendable para menores de edad.

20.-

What is the meaning of the keyword 'this', and how can the keyword be used? Check all that apply.

1. It refers to the object on which a method or constructor has been called (sometimes called the "implicit parameter").
2. It is used to call one constructor from another.
3. It can be used to access or set an object's field values
4. It is used when one object wants to access data from a second object.
5. It is used in conjunction with the 'that' keyword when the programmer wants to write a parallel "this and that" algorithm.
6. It is required when a class has more than one constructor.
7. It can be used to call the object's methods.

21.-

What is abstraction?

1. When alcohol was illegal in the United States in the early 20th century.
2. The ability to focus on a problem at a high level without worrying about the minor details.
3. When things are confusing and vague rather than simple and understandable.
4. The opposite of addition, as seen in mathematics.
5. A Java keyword that optimizes programs that do not use classes or objects.

How do objects provide abstraction?

1. By giving us more powerful pieces of data that have sophisticated behavior without having to manage and manipulate the data directly.
2. When one object converts into another type of object, it becomes abstract.
3. Public fields provide abstraction because they let the client directly view an object's data.
4. Objects provide abstraction through the `.provideAbstraction()` method.
5. Objects provide abstraction when they declare a lot of useful data inside them.

22.- Qué falla en la siguiente sobreescritura del método equals?

```
public class Circulo() {  
  
    private int radio;  
  
    public int equals(Object obj) {  
  
        // Reflexivo  
        if (this == obj) {  
            return 1;  
        }  
        // Comparado con null siempre es false  
        if (obj == null) {  
            return 0;  
        }  
  
        // Tienen que ser de la misma clase  
        if (getClass() != obj.getClass()) {  
            return 0;  
        }  
  
        // Comparamos el estado  
        Circulo other = (Circulo) obj;  
        if (this.radio != other.radio) {  
            return 1;  
        }  
  
        return 1;  
    }  
}
```

23.- Esta sobreescritura del método equals tiene 2 problemas. Cuales?

```
public class Circulo() {  
  
    private int radio;  
  
    public boolean equals(Circulo obj) {  
  
        // Reflexivo  
        if (this == obj) {  
            return true;  
        }  
  
        // Tienen que ser de la misma clase  
        if (getClass() != obj.getClass()) {  
            return false;  
        }  
  
        // Comparamos el estado  
        Circulo other = (Circulo) obj;  
        if (this.radio != other.radio) {  
            return false;  
        }  
  
        return true;  
    }  
}
```

25.- Compilará el siguiente código?

```
class Prueba {  
    protected String nombre;  
    protected int ID;  
    public String getIdent() { return nombre; }  
    public int getIdent() { return ID; }  
}
```

26.- Verdadero o falso:

- 1.- Una clase abstracta no se puede instanciar.
- 2.- Los métodos de una clase abstracta no tienen implementación.
- 3.- En la declaración de una interface sólo pueden aparecer declaraciones de métodos pero no sus implementaciones.
- 4.- Todas las propiedades declaradas en una interface son `public static final`.

27.- El siguiente código da un error de compilación. ¿ Qué puede estar pasando?

```
final String s1 = new String("Hola");  
String s2 = new String(" Mundo!");  
s1 = s1 + s2;
```


28.- Tenemos la siguiente clase:

```
public abstract class Vehiculo {  
    private int peso;  
    public final void setPeso(int p) { peso = p; }  
    public abstract int getVelocidadActual();  
}
```

1.- Puede tener descendencia esta clase?

2.- Se pueden sobrescribir todos sus métodos?

Razona tus respuestas.

29.- Tenemos las siguientes clases:

```
public class Animal {  
    private String nombre;  
    private int edad;  
    private String color;  
  
    public Animal() {  
        System.out.println("Soy un animal sin nombre");  
    }  
  
    public Animal(String nombre, String color) {  
        this();  
        this.nombre = nombre;  
        this.color = color;  
    }  
  
    public String getNombre() { return nombre; }  
    public int getEdad() { return edad; }  
}
```

```
public class Gato extends Animal {  
  
    public Gato() {  
        System.out.println("Soy un gato sin nombre");  
    }  
  
    public Gato(String nombre, int edad, String color) {  
        super(nombre, color);  
        System.out.println("Soy un gato llamado " + getNombre() + " de " + getEdad() + " años");  
    }  
}
```

Que se imprimirá en pantalla si ejecutamos: `Gato miGato = new Gato("Arañazos", 5, "Blanco");`

30.- Tenemos la siguiente jerarquía:

Mamifero hereda de Animal y Gato hereda de Mamifero.

Es válido hacer esto?

```
Animal animal = new Mamifero();  
Gato gato = (Mamifero)animal;
```

Y hacer?

```
Animal animal = new Gato();  
Gato gato = (Gato)animal;
```

Razona brevemente tus respuestas.

31.-

```
public class ParameterMystery {  
    public static void main(String[] args) {  
        String x = "java";  
        String y = "tyler";  
        String z = "tv";  
        String rugby = "hamburger";  
        String java = "donnie";  
  
        hamburger(x, y, z);  
        hamburger(z, x, y);  
        hamburger("rugby", z, java);  
        hamburger(y, rugby, "x");  
        hamburger(y, y, "java");  
    }  
  
    public static void hamburger(String y, String z, String x) {  
        System.out.println(z + " and " + x + " like " + y);  
    }  
}
```

Cual será la salida de :

```
hamburger(z, x, y);  
hamburger(y, y, "java");
```

32.-

| | |
|---|---|
| <pre>public class A extends B { public void method2() { System.out.print("a 2 "); method1(); } } public class B extends C { public String toString() { return "b"; } public void method2() { System.out.print("b 2 "); super.method2(); } }</pre> | <pre>public class C { public String toString() { return "c"; } public void method1() { System.out.print("c 1 "); } public void method2() { System.out.print("c 2 "); } } public class D extends B { public void method1() { System.out.print("d 1 "); method2(); } }</pre> |
|---|---|

Que imprimirá el siguiente código?

```
C [] elements = {new A(), new B()};
for (int i = 0; i < elements.length; i++) {

    System.out.println(elements[i]);
    elements[i].method1();
    elements[i].method2();
}
```

33.- Qué valdrá “satelites” después de ejecutar el siguiente bloque de código?

```
public class Planeta {  
    private String nombre;  
    public static int satelites = 0;  
  
    public Planeta(String nombre) {  
        this.nombre = nombre;  
        satelites++;  
    }  
  
    public static void main(String[ ] args) {  
  
        Planeta tierra = new Planeta("Tierra");  
        Planeta luna = new Planeta("Lunea");  
        new Planeta("Jupiter");  
    }  
}
```

34.-

| | |
|---|--|
| <pre>public class Ice extends Fire { public void method1() { System.out.print("Ice 1 "); } } public class Rain extends Fire { public String toString() { return "Rain"; } public void method1() { super.method1(); System.out.print("Rain 1 "); } }</pre> | <pre>public class Fire { public String toString() { return "Fire"; } public void method1() { method2(); System.out.print("Fire 1 "); } public void method2() { System.out.print("Fire 2 "); } } public class Snow extends Rain { public void method2() { System.out.print("Snow 2 "); } }</pre> |
|---|--|

Qué se imprimirá?

Fire[] elements = {new Fire(), new Snow()}; 

```
for (int i = 0; i < elements.length; i++) {
    System.out.println(elements[i]);
    elements[i].method1();
    elements[i].method2();
}
```

35.- Compilará la siguiente clase? Razona tu respuesta.

```
public class Planeta extends Astro {  
    private String nombre;  
    public Astro(String nombre) {  
        this.nombre = nombre;  
    }  
}
```


36.- Si quiero usar la clase Scanner del paquete java.util, que sentencia debo incluir en mi programa para poder usar esta clase?

```
import java.util.Scanner;
```

37.- Si quiero utilizar una clase que está contenida en una librería de terceros en un archivo .jar ¿Qué puedo hacer si quiero que Java encuentre dicha clase?

incluir esa clase o archivo al classpath

38.- Razona brevemente si es correcto el siguiente código:

```
public class Vertebrado { };  
public class Mamifero { };
```

```
public class Perro extends Vertebrado, Mamifero {  
}
```

una clase solo puede tener un padre

39.- Es válido el siguiente código? Razona tu respuesta.

```
public abstract class Vehiculo {  
  
    private int combustible;  
  
    public void setCombustible(int combustible) {  
        this.combustible = combustible;  
    }  
}  
  
public class VehiculoTest {  
  
    public static void main(String[ ] args) {  
        Vehiculo miVehiculo = new Vehiculo( );  
        miVehiculo.setCombustible(100);  
    }  
}
```

no se puede instanciar un abstract

40.- Que imprimirá por pantalla el siguiente código?

```
public class Circle {  
  
    private int radio;  
  
    public Circle(int radio) { this.radio = radio; }  
  
    public boolean equals(Object obj) {  
  
        if (this == obj) {  
            return true;  
        }  
  
        if (obj == null) {  
            return false;  
        }  
  
        if (getClass() != obj.getClass()) {  
            return false;  
        }  
  
        return true;  
    }  
  
    public static void main(String [ ] args) {  
  
        Circle a = new Circle(5);  
        Circle b = new Circle(7);  
  
        System.out.println(a == b);  
        System.out.println(a.equals(b));  
    }  
}
```

false

true