Programación 2

Tecnicatura en Desarrollo de Aplicaciones Informáticas

Parcialito 1

Si defino un **atributo** en una clase (siguiendo los principios de la Programación Orientada a Objetos) entonces estoy definiendo... (más de una opción puede ser correcta)

- [ ] a. una variable global al sistema
- [x] b. una variable que se puede conocer en todos los métodos de esta clase
- [ ] c. una variable que solo es conocida dentro de los métodos privados de esta clase
- [ ] d. una variable que solo es conocida dentro de los métodos públicos de esta clase
- [ ] e. una variable que se puede acceder desde cualquier clase
- [x] f. una variable de instancia

El encapsulamiento se refiere principalmente a

- Proteger el acceso a los atributos de un objeto
- Reutilizar una clase
- Permitir modificar los atributos de un objeto
- Definir métodos ocultos de una clase

Siempre es conveniente dejar algunos atributos de los objetos de forma pública así se puede acceder a ellos directamente.

Seleccione una:

- ○ Verdadero
- ● Falso

Todos los objetos creados a partir de una clase responden a los mismos mensajes

Seleccione una:

- Verdadero
- Falso

¿Qué se logra con la delegación de responsabilidades?

- a. Una mayor reutilización de atributos.
- b. Una mayor seguridad de los datos y una mejor modularidad del código.
- c. Una mejor distribución de responsabilidades y una mayor flexibilidad en el diseño.

Dadas las siguientes sentencias:

```
Persona p1 = new Persona();
Persona p2;
Persona p3 = new Persona();
Persona p4 = p1;
```

¿Cuántos objetos se instanciaron?

Respuesta: 2

Un [ ] es una señal que recibe un objeto para ejecutar un [ ] determinado

- método
- mensaje
- objeto

- método
- mensaje
- objeto

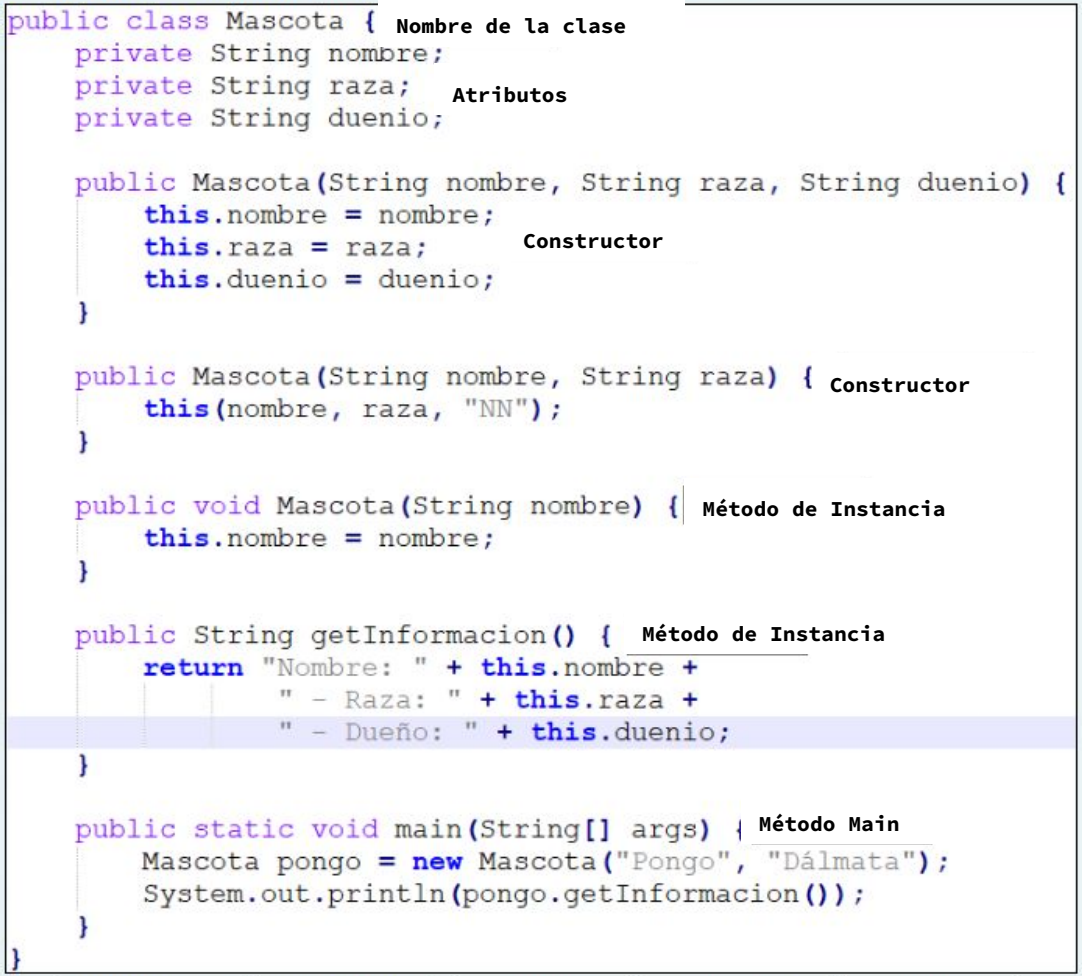
El siguiente código

- Da error en tiempo de ejecución
- Muestra una clase con tres constructores perfectamente válidos
- No es válido porque hay más de un constructor para la clase
- Da error en tiempo de compilación

```
public class Mascotaa

    private String nn

    private String nn

    private String dd


    public Mascota(String nombre, String raza, String duenio) {
        this.nombre = nombre;
        this.raza = raza;
        this.duenio = duenio;
    }

    public Mascota(String nombre, String raza) {
        this(nombre, raza, "NN");
    }

    public Mascota(String nombre, String dueño) {
        this(nombre, "desconocida", dueño);
    }

    //continúa con los métodos de la clase

}
```

MISMA SIGNATURA

Arrastra las definiciones en la parte correspondiente de la imagen del código fuente de una clase en JAVA



método de instancia | Nombre de la clase | constructor | método main | atributos

Una variable de instancia es

- una variable definida en un método
- una variable global al sistema
- un atributo definido en una clase

Dado el siguiente código, ¿qué se imprimirá por consola?

```
public class Mascota {
    private String nombre;
    private String raza;
    private String duenio;
    private LocalDate fechaNacimiento;

    public Mascota(String nombre, String raza, String duenio) {
        this.nombre = nombre;
        this.raza = raza;
        this.duenio = duenio;
    }

    public Mascota(String nombre, String raza) {
        this(nombre, raza, "NN");
    }

    public String getInformacion(){
        return "Nombre: "+this.nombre+
               " - Raza: "+this.raza+
               " - Dueño: "+this.duenio+
               " - Fecha Nacimiento: "+this.fechaNacimiento;
    }

    public static void main(String[] args) {
        Mascota pongo = new Mascota("Pongo", "Dálmata");
        System.out.println(pongo.getInformacion());
    }
}
```

- a. Nombre: Pongo - Raza: Dálmata - Dueño: - Fecha Nacimiento:
- b. Nombre: Pongo - Raza: Dálmata - Dueño: NN - Fecha Nacimiento:
- c. Nombre: Pongo - Raza: Dálmata - Dueño: null - Fecha Nacimiento: null
- d. Nombre: Pongo - Raza: Dálmata - Dueño: NN - Fecha Nacimiento: null

¿Qué significa instanciar una clase? **Puede ser mas de una opción**

- [ ] a. Duplicar una clase
- [x] b. Crear un objeto a partir de una clase
- [ ] c. Eliminar una clase
- [ ] d. Conectar dos clases entre si
- [ ] e. Definir el constructor de una clase

¿Es posible definir métodos de distinta signatura en la misma clase?

- a. ¿Qué es la signatura?
- b. No
- c. Si

Todos los objetos creados a partir de una clase son iguales, ya que la clase define los valores de los atributos y los métodos

Seleccione una:

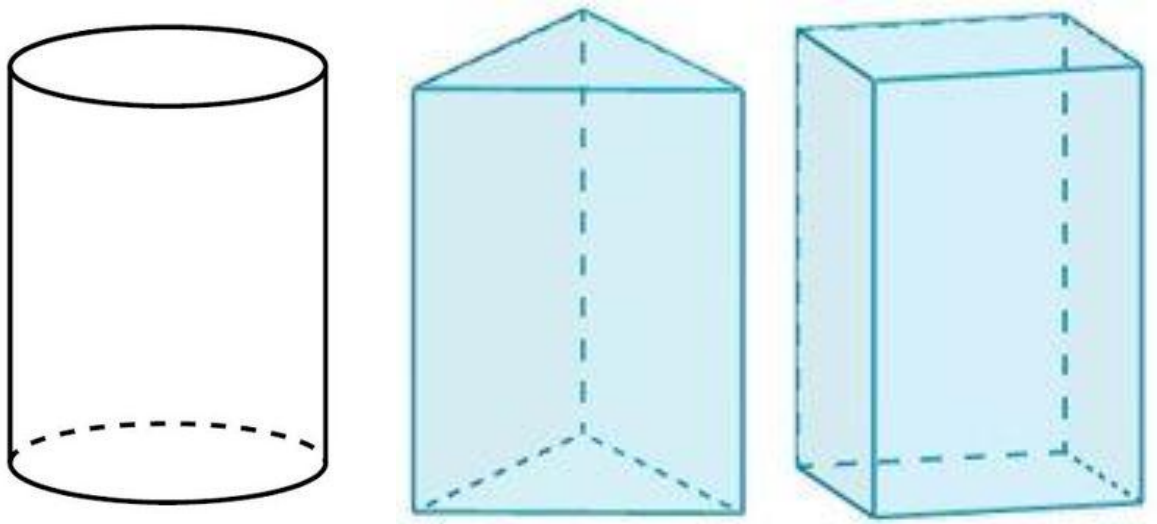
- ○ Verdadero
- ● Falso

¿Qué es el bytecode de JAVA?

- a. El código fuente de un sistema en java
- b. El código intermedio que se obtiene al compilar el código fuente de un sistema en java
- c. El código nativo de un sistema java para una arquitectura de hardware particular

Figuras 3D

Figura3D



Simplificación en la cual debemos calcular el volumen de la figura

Cilindro-PrismaTriangular-PrismaRectangular

Juego de Dados

Nueva Funcionalidad

---

un jugador es tramposo y utiliza dados cargados que favorecen la salida del **5** y el **6** respectivamente la mitad de las veces que se tira el dado

Dados Cargados

```
public class Dado{
   int valor;
   int caras;

  public int tirar() {
    return (Math.radom()*caras)+1;
}
```

```
public class DadoCargado5 extends Dado{

  public int tirar() {
    if (Math.random()>0.5)
       return super.tirar();
    else
       return 5;
}
```

```
public class DadoCargado6 extends Dado{

  public int tirar() {
    if (Math.random()>0.5)
       return super.tirar();
    else
       return 6;
}
```

Mal uso de la herencia: no voy a crear una clase por cada valor de una variable de instancia que cambie

Dados Cargados

```
public class Dado{
   int valor;
   int caras;

  public int tirar() {
    return (Math.radom()*caras)+1;
}
```

```
public class DadoCargado extends Dado{

  int ladoCargado;
  public int tirar() {
    if (Math.random()>0.5)
       return super.tirar();
    else
       return ladoCargado;
}
```

Dados Cargados

¿Qué otras clases debo modificar?

Como?

Clase Vs Instancia

No puede existir clases iguales

Dos clases iguales son las misma

CUANDO LA DIFERENCIA ES SOLO UNA CONSTANTE, entonces también son la misma clase

Malos ejemplos

PersonaJuan

PersonaPedro ---> “Juan”

----> Existe un nombre, las dos son personas con una variable Nombre

Mal en Dados

Clase padre Dado

DadoCargado5 y DadoCargado6 ---> Los dos son DadoCargado y el valor en realidad es una variable!!!

DESAPARECEN DADOCARGADO5 y DADOCARGADO6

Reconocer fácil el error

NO TIENE QUE HABER CONSTANTES EN EL CÓDIGO

```
return “juan”

return 84;

return “a”;
```

```
if (x>150)

if (nombre.equals(“juan”))
```

```
return “juan”;
reemplazar

return nombreADevolver;
```

La Clase Object

Object

Object es una clase

La clase Object es la superclase de todas las clases

Todas las clases heredan directa o indirectamente de Object

Object

```
public class Object {

      public boolean equals( Object obj );

      public String toString( )

      protected void finalize( )

        ...}
```

Estos métodos pueden ser redefinidos en las subclases si es necesario

equals()

- Retorna true cuando dos objetos tienen iguales valores

- La implementación por defecto compara referencias con ==

equals() Ejemplo Dado

```
public class Dado{
   public boolean equals( Object obj ) { // PARA REDEFINIR
// EL MÉTODO NO PUEDO CAMBIAR LA SIGNATURA DEL MISMO
       Dado other = (Dado) obj; // SI NO ES UN DADO DA ERROR
//MÁS ADELANTE VAMOS A VER COMO SE SOLUCIONA
      return  this.getValor() == other.getVAlor();
    }
}
```

equals() Ejemplo Dado

```
Dado d1 = new Dado(3);

Dado d2 = new Dado(3);

 (d1 == d2);     // returns false

d1.equals(d2);      // returns true
```

toString()

- toString( ) se usa para proveer una representación del objeto como una cadena de caracteres
- Invocada automáticamente cuando utilizamos
  - System.out.println() y “+”
  - String s = “The object is “ + obj;
  - System.out.println(obj);
- La implementación por defecto de la clase Object retorna el nombre de la clase y la ubicación en memoria del objeto

toString() Ejemplo

```
public class Dado {


 public String toString( ) {
   return “Dado con el valor = " + this.getValor();
 }
```

finalize()

Este método se llama justo antes de que un objeto sea basura recolectada (garbage collected).

Es invocado por el recolector de basura en un objeto cuando el recolector de basura determina que no hay más referencias al objeto.

Debemos anular el método finalize() para eliminar los recursos del sistema, realizar actividades de limpieza y minimizar las pérdidas de memoria