ESCUELA POLITÉCNICA NACIONAL
ESCUELA DE FORMACIÓN DE TECNÓLOGOS
DESARROLLO DE SOFTWARE

PERÍODO ACADÉMICO: 2025-B **ASIGNATURA:** PROGRAMACIÓN ORIENTA A OBJETOS
DOCENTE: MSc. _____ **PRUEBA 1 BIMESTRE**

NOMBRE DEL ESTUDIANTE: Milka Borja **NOTA:** /

Instrucciones Generales:

- Resolver la presente evalución TEORICA
- Hora de Inicio 12:00– 13:00
- Cualquier intento de copia se retira el examen y la calificación es cero

 Objetivo del ejercicio:

Aplicar conceptos clave de Programación Orientada a Objetos (POO) en Java, incluyendo:
- Atributos privados, Uso de get y set, Encapsulamiento, Constructores, Validación de condiciones.

Preguntas:

1. **¿Qué es una clase en Java y cuál es su propósito?**
   Es una plantilla o molde para crear objetos. Su propósito es definir las propiedades (variables) y comportamientos (métodos) que tendrán los objetos creados a partir de ella.

2. **¿Qué es un método estático y cómo se invoca?**
   Es un método que pertenece a la **clase misma**, no a un objeto individual. Se invoca usando el nombre de la clase, seguido de un punto y el nombre del método (ejemplo: Math.random()), sin necesidad de crear un objeto.

3. **¿Cuál es la diferencia entre un método estático y uno no estático?**
   Estático: Pertenece a la clase. Se puede llamar sin crear un objeto.
   No estático (de instancia): Pertenece a un objeto específico. Se necesita *instanciar* la clase (crear un objeto) para poder llamarlo.

4. **¿Puede un método estático acceder a atributos no estáticos? ¿Por qué?**
   No. Porque los métodos estáticos existen *antes* de que se cree cualquier objeto (instancia). Los atributos no estáticos solo existen *dentro* de un objeto, por lo que el método estático no sabría a qué instancia te refieres.

5. **¿Qué es la sobrecarga de métodos? Dé un ejemplo.**

Es la capacidad de definir varios métodos con el mismo nombre en una clase, siempre y cuando tengan una lista de parámetros diferente (ya sea en número o en tipo de datos). *Ejemplo:* void sumar(int a, int b) y void sumar(double a, double b).

6. **¿Se puede sobrecargar un constructor? ¿Cómo?**
   Sí. Se hace igual que con los métodos: creando varios constructores (con el mismo nombre de la clase) pero con diferentes listas de parámetros
   .
7. **¿Cómo se accede a un atributo privado desde fuera de la clase?** Directamente no se puede. Se debe acceder a través de métodos públicos creados dentro de la misma clase, conocidos como getters (para obtener el valor) y setters (para modificarlo).

8. **¿Qué es un getter y un setter? ¿Por qué son importantes?**
   Getter: Un método público para *obtener* (leer) el valor de un atributo privado (ej: getNombre()).
   Setter: Un método público para *establecer* (modificar) el valor de un atributo privado (ej: setNombre(String nombre)). Son importantes para el encapsulamiento: permiten proteger los datos y añadir lógica (como validaciones) antes de leerlos o modificarlos.

9. **¿Qué ocurre si se intenta acceder directamente a un atributo privado?**
   Se produce un error de compilación. El programa ni siquiera llegará a ejecutarse porque se está violando una regla de acceso.

10. **¿Qué significa instanciar una clase?**
    Es el proceso de crear un objeto (una "instancia") a partir de una clase (la plantilla). Se usa la palabra clave new.

11. **¿Qué es el método main y cuál es su función en una aplicación Java?**
    El método main (public static void main(String[] args)) es el punto de entrada de una aplicación Java. Es el primer método que la JVM (Máquina Virtual de Java) busca y ejecuta cuando corres el programa.

12. **¿Puede una clase tener más de un constructor? ¿Por qué sería útil?**
    Sí (mediante sobrecarga). Es útil para ofrecer diferentes formas de crear un objeto; por ejemplo, un constructor vacío que use valores por defecto, y otro que reciba parámetros para inicializar el objeto con datos específicos.

13. **¿Qué sucede si se llama un método sobre una referencia nula?**
    Se produce un error en tiempo de ejecución (una excepción) llamado NullPointerException. El programa se detendrá bruscamente.

14. **¿Qué beneficio trae el que Java tenga una JVM?**

El principal beneficio es la portabilidad. Gracias a la JVM (Máquina Virtual de Java), el mismo código compilado de Java puede ejecutarse en diferentes sistemas operativos (Windows, Mac, Linux) sin necesidad de cambiarlo. Es el principio de "Escribe una vez, ejecuta en cualquier lugar".

15. **¿Cuál es la diferencia entre JDK y JRE?**
    JRE (Java Runtime Environment): Es lo que necesitas solo para *ejecutar* aplicaciones Java. Incluye la JVM.
    JDK (Java Development Kit): Es lo que necesitas para *desarrollar* (programar) en Java. Incluye todo el JRE y, además, herramientas como el compilador (javac).

1. Determinar si hay un constructor en la clase Addition, seleccione la opción correcta.

1. public **class** Addition{ 2. int a = 5; int b= 3. 5; public void 4. add(){ int c = 5. a+b; 6. } **7.** }	a. Sí, en la línea 1. b. Sí, en la línea 4. c. No hay constructor en esta clase. d. Sí, en la línea 6.

2. Seleccione donde está ubicado el método constructor y cuál es el nombre?

1. public **class** Addition{
2. public static void main(String[] args){
3. Addion add = new Addition();
4. }
5. }
	a. Linea 1, nombre: _____
	b. Linea 2, nombre: _____
	c. Linea 3, nombre: default
	d. Linea 4, nombre: _____

3. Cuál es la estructura correcta de un getter:

a.
```
public TipoDato getNombreAtributo()
{

}
```

b.
```
public TipoDato getNombreAtributo() {    return
nombreAtributo;
}
```

c.
```
public TipoDato getNombreAtributo(String nombre) { return
nombreAtributo;
}
```

d. public TipoDato getNombreAtributo(String nombre) {

}

4. **Cuál es la estructura correcta de un setter en Java:**
   a. public void setNombreAtributo() { this.nombreAtributo = nombreAtributo;
   }
   b. public void setNombreAtributo() { return nombreAtributo;}
   c. public void setNombreAtributo(TipoDato nombreAtributo) {
   this.nombreAtributo = nombreAtributo;
   }
   d. public void setNombreAtributo(TipoDato nombreAtributo) {
   return nombreAtributo;}

5. **¿Cuál es el propósito del siguiente constructor?**

```
public class Persona {
    String nombre;
    int edad;

    public Persona(String n, int e) {
        nombre = n;
        edad = e;
    }
}
```

A) Inicializar atributos
B) Crear métodos
C) Encapsular datos
D) Crear una clase abstracta

6. **¿Qué salida produce este código?**

```
// Clase Persona
public class Persona {
    public String nombre;
    public int edad;

    public Persona(String nombre, int edad) {
        this.nombre = nombre;
        this.edad = edad;
    }
}

// Clase Main
public class Main {
    public static void main(String[] args) {
        Persona p = new Persona("Ana", 30);
        System.out.println(p.nombre);
    }
}
```

A) Error de compilación

B) Ana

C) null

D) 30

7. ¿Cuál es el error en este código?

```
// Clase definida en otro archivo
public class Animal {
    private String tipo;

    public void setTipo(String t) {
        tipo = t;
    }

    public String getTipo() {
        return tipo;
    }
}

// Clase Main
public class Main {
    public static void main(String[] args) {
        Animal a = new Animal();
        a.tipo = "Perro";
    }
}
```

A) No se puede instanciar la clase Animal

B) El atributo tipo es privado

C) Falta el constructor en la clase Animal

D) El método setTipo no existe

8. ¿Qué hace el siguiente método?

```
public static void saludar() {
System.out.println("Hola");
}
```

A) Es un método de instancia
B) Es un método estático
C) Es un constructor
D) Es un getter

9. ¿Cuál de las siguientes firmas de método NO causaría un error de compilación por firma duplicada si se intentara usar para sobrecargar el método operar?

```
public class Calculadora {
    public int operar(int a, int b) {
        return a + b;
    }
```

a. public void operar(int x, int y)
b. public int operar(int a, int b, int c)
c. private int operar(int val1, int val2)
d. public double operar(int a, int b)

10. Dadas las siguientes líneas de código. ¿Qué describe mejor el proceso y el resultado?

```
double valorDecimal = 1500.99; // double tiene 64 bits
int valorEntero = (int) valorDecimal; // int tiene 32 bits
```

A) Es un *Casting* Implícito (Widening), no hay pérdida de información

B) Es un *Casting* Explícito (Narrowing) y valorEntero contendrá 1500, con pérdida de la parte decimal
C) Es imposible realizar esta conversión porque double y int son tipos primitivos incompatibles
D) valorEntero contendrá 1501 debido al redondeo automático.

11. Dada la clase Libro y el siguiente fragmento de código. Cual es el precio final de miLibro:

```
// Clase Libro
public class Libro {
    public double precio;

    public Libro() {
        this.precio = 0.0;
    }

    public Libro(double precio) {
        this.precio = precio;
    }
}

// Metodo que modifica el precio
public void modificarPrecio(Libro libro) {
    libro.precio = 100.00;
}


public class Main {
    public static void main(String[] args) {
        Libro miLibro = new Libro(50.00);
        modificarPrecio(miLibro);
        System.out.println(miLibro.precio);
    }
}
```

A) $5.00, porque la referencia fue cambiada dentro del método.

B) $100.00, porque se pasó una copia de la referencia y se modificó el objeto original.

C) $50.00, porque Java siempre pasa por valor y los cambios internos nunca afectan al objeto original.

D) Error de compilación por mala referencia de objetos.

12. Si la clase Libro tiene un atributo no estático titulo (String) y se define el siguiente método static:

```
public class Libro {
    private String titulo;

    public static void imprimirTitulo() {
        System.out.println("Título: " + this.titulo);
    }
}
```

Al intentar compilar este código, ¿qué error se produciría?

A) Error de encapsulamiento, ya que titulo es private.
B) Error de compilación: Un método static no puede acceder a atributos no estáticos (titulo) porque no tiene un objeto this asociado
C) Ningún error, siempre y cuando se inicialice título en el constructor.
D) Error en el tipo de retorno, ya que los métodos static deben retornar un valor.

13. Dada la clase Producto con un atributo private double precio, y el siguiente constructor:

```
public Producto(double precio) {
    precio = precio;
}
```

Si se crea un objeto Producto p = new Producto(50.0);, ¿cuál será el valor del atributo precio del objeto p?

1. 50.0
2. El valor por defecto para double, que es 0.0
3. Se genera un error de compilación
4. Depende de si se ha llamado a un constructor padre

14. Un desarrollador ha configurado todos los atributos de la clase Libro como private (e.g., - titulo en UML). Sin embargo, intenta llamar directamente al atributo precio desde una clase Main en un paquete diferente:

```
// En clase Main, paquete diferente
Libro miLibro = new Libro();
miLibro.precio = 30.00;
```

¿Qué ocurrirá al compilar la clase Main?

A) Compilación exitosa, porque los atributos primitivos pueden ser accedidos directamente
B) Error de compilación, porque el acceso directo a un atributo private está prohibido fuera de su clase
C) Compilación exitosa, si se usa un *setter* inmediatamente después. D) Error de compilación, solo si el atributo fuera de tipo String

15. Une cada modificador de acceso y su ámbito de acceso