Sección 1. Selecciona la mejor opción.

1. Indica el resultado de ejecutar el siguiente código.

```
public static void main(String[] args) {
        int[] x = {7,3,6,1,1};
        updateArray(x);
        printArray(x);
}

public static void updateArray(int[] array) {
        array[3] = 10;
}

public static void printArray(int[] array) {
        for(int i = 0; i<array.length; i++) {
            System.out.print(array[i] + " ");
        }
}
```

a) 7 3 6 1 1
b) 7 3 10 1 1
c) 7 3 6 10 1
d) 7 3 10 1

2. Indica el resultado de ejecutar el siguiente código

```
public static void main(String[] args) {
        double a = 17;
        double b = 0.5;

        int c = xOperation(a,b);
        System.out.println(c);
}

public static int xOperation(int a, int b){
        return a + b;
}

public static double xOperation(double a, int b){
        return a-b;
}

public static int xOperation(double a, double b){
        return (int)(a*b);
}

public static double xOperation(int a, double b){
        return (a-1)*(b+0.5);
}
```

a) 17.0
b) 8
c) 16.0
d) 16
e) 17
f) 8.0

```java
public class Bottle {

    private double capacity; //capacity in liters
    private String owner;
    private static int numberOfBottles;

    public Bottle() {
        this("", 0);
    }

    public Bottle(String owner, double capacity) {
        numberOfBottles++;
        this.owner = owner;
        this.capacity = capacity;
    }

    public void print() {
        System.out.println("Name: " + this.owner);
        System.out.println("Capacity: " + this.capacity + "L");
        System.out.println("# Bottles in the world: " + Bottle.numberOfBottles);
    }
}
```

3. **Elige la opción que mejor representa un setter para la variable `owner`:**

a)
```
public String setOwner() {
    return this.owner;
}
```

b)
```
public void setOwner(String owner) {
    owner = this.owner;
}
```

c)
```
public void setOwner(String owner) {
    this.owner = owner;
}
```

d)
```
public String setOwner(String owner) {
    this.owner = owner;
    return owner;
}
```

4. **Tomando como resultado la clase `Bottle`, y tu elección del método `setOwner`, cuál sería el resultado de ejecutar el siguiente código:**

```
public static void main(String[] args) {

  Bottle cokeBottle = new Bottle("Arturo Curry", 2.5);
  Bottle pepsiBottle = new Bottle("Francisco Castillo", 0.6);

  Bottle copyBottle;
  copyBottle = cokeBottle;
  copyBottle.setOwner("Bruno Díaz");

  copyBottle = pepsiBottle;
  copyBottle.print();
}
```

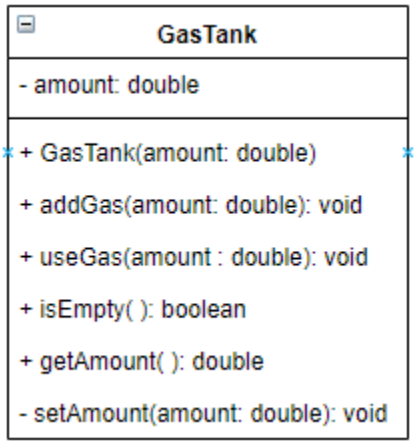
a) Name: Francisco Castillo
Capacity: 0.6L
Bottles in the world: 2

b) Name: Arturo Curry
Capacity: 2.5L
Bottles in the world: 1

c) Name: Francisco Castillo
Capacity: 0.6L
Bottles in the world: 1

d) Name: Bruno Díaz
Capacity: 0.6L
Bottles in the world: 1

Sección 2: Diseña una clase `GasTank` que sirva para representar un tanque de gasolina. Diseña la clase de acuerdo con el siguiente diagrama UML:



- **Constructor:** Este método debe recibir como parámetro de entrada una cantidad **amount**, y llame al método **setAmount** para actualizar la variable de instancia **amount.**
- **addGas**: deberá incrementar la cantidad de gasolina en el tanque en la cantidad recibida como parámetro**.** Asegúrate de validar que sólo se procesen valores positivos.
- **useGas:** deberá reducir la cantidad de gasolina en el tanque en la cantidad recibida como parámetro. Asegúrate de que sólo se procesen valores positivos.
- **isEmpty:** Deberá devolver **true** cuando la cantidad de gasolina en el tanque sea menor a 0.1. De lo contrario, deberá retornar **false.**
- **getAmount:** Getter para la variable amount.
- **setAmount:** Método privado (sólo será usado por el constructor) que actualice la variable **amount** siempre y cuando el parámetro recibido sea mayor o igual a 0.

Sección 3. Resuelve el siguiente caso.

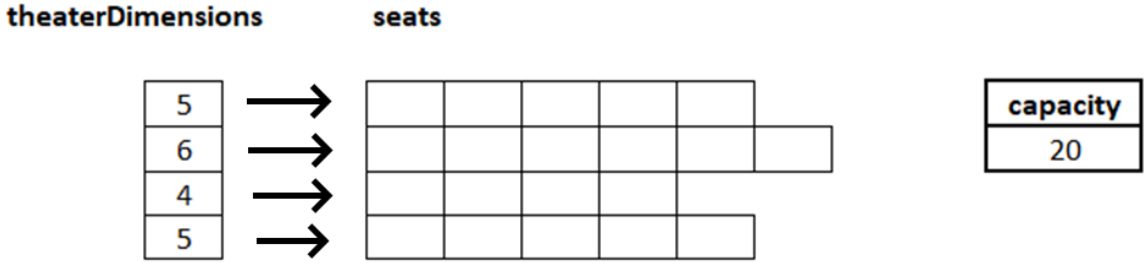
Cineplus, una nueva cadena de complejos de cine, ha decidido abrir su primera sucursal en la ciudad de Monterrey. Su concepto único está basado en ofrecer boletos de cine a un costo variable dependiendo de la disponibilidad de asientos en cada función.

Diseña y codifica una clase llamada `CinemaShow` que permita modelar una función de cine en un complejo de Cineplus. La clase deberá contar con los atributos definidos a continuación. Elige los tipos de datos que mejor se adecúan para representar la siguiente información. Utiliza las mejores prácticas y conceptos de la programación orientada a objetos. Considera si cada método debe ser estático o de instancia.

- Variable `movieName` que sirva para almacenar el nombre de la película que se va a proyectar.
- Variable `movieDate` que sirva para representar la fecha en la que se proyectará la película.
- Variable `movieTime` que sirva para representar la hora en la que se proyectará la película.
- Variable `capacity` que permita almacenar el aforo de la sala de cine (cantidad de personas que caben en la sala).
- Variable `soldTickets` que mantenga un registro actualizado de la cantidad de asientos vendidos de la función al momento.
- Arreglo bidimensional `seats` que sirva para llevar el control de los asientos vendidos.

Adicionalmente, la clase deberá contener las siguientes acciones:

- Método constructor que reciba e inicialice las variables: `movieName`, `movieDate` y `movieTime`.
- Métodos setter diferentes para las variables: `movieName`, `movieDate` y `movieTime`.
- Métodos getter diferentes para las variables: `capacity` y `soldTickets`.
- Método `initializeSeats` que reciba como parámetro de entrada un arreglo de enteros `theaterDimensions`, y que inicialice el arreglo `seats`. Adicionalmente, deberá calcular el aforo de la sala y almacenarlo en la variable de instancia `capacity`.
  El arreglo recibido `theaterDimensions` representará la cantidad de asientos en cada fila de la sala. Por ejemplo:



- Método `double assignPrice(int soldTickets, int totalTickets)` que calcule y retorne el precio de un boleto de acuerdo con la siguiente fórmula:

$$Precio = 50 + ((número\ de\ boleto)/(asientos\ totales)) \times 50$$

  Por ejemplo, el boleto #36 se vendería en:

$$Precio = 50 + (36/100) \times 50 = \$68.00$$

- Método `boolean sellSeat(int row, int column)` que reciba como parámetro de entrada dos enteros: `row` y `column`, calcule el precio del boleto (ver método `assignPrice`), y lo almacene en la fila y columna recibida. Posteriormente deberá actualizar la variable `soldTickets`.
- Sobrecarga el método `boolean sellSeat()` para que asigne automáticamente un espacio vacío en la sala.

El arreglo `seats` no debe ser retornado directamente por un método getter. ¿Cuál es el riesgo de implementar un getter para dicha variable y cómo pudiera mitigarse? Justifica tu respuesta.