Escuela Superior Politécnica del Litoral
Programación Orientada a Objetos
Examen Tercera Evaluación
2019 - 1T

Nombre:_____ **Paralelo:**_____

1.- (5 PUNTOS) Califique como Verdadero o Falso cada uno de los enunciados. Escriba la palabra completa.

Enunciado	**Verdadero o Falso**
Si en una clase B que es hija de A se sobrescribe un método, dentro de la definición de la clase B no es posible invocar a la definición original en la clase padre.	
La interface Comparable tiene un único método compareTo que debe ser sobrescrito	
El polimorfismo solo es posible si existe relación de herencia entre las clases.	
Puedo eliminar elementos de un HashSet mientras se recorre con un for each.	
De todos los constructores de una clase padre, sólo el constructor sin parámetros es heredado por la clase hija.	

2.- (5 puntos) Analice el código mostrado a continuación y seleccione la opción que corresponde a la salida en pantalla al ejecutar la clase Test1. Justifique su respuesta.

```
class Padre {
       static void m1() {
            System.out.println("m1");
      }
       void m2() {
            System.out.println("m2");
      }
}
class Hijo extends Padre {
        static void m1() {
              System.out.println("Hijo - m1");
        }
        public void m2() {
             System.out.println("Hijo - m2");
       }
}
class Test  {
       public static void main(String[] args) {
        Padre obj1 = new Hijo();
        obj1.m1();
        obj1.m2();
       }
}
```

Opciones:

a) Error de compilación
b) m1
   Hijo - m2
c) Hijo - m1
   Hijo - m2
d) m1
   m2

Justificación:

3.- (5 puntos) Analice el código mostrado a continuación. ¿Cuál es la salida en pantalla al ejecutarse?. Justifique su respuesta. (EL CÓDIGO COMPILA Y SE EJECUTA CORRECTAMENTE)

```
class Base {
    static int cont;
    Base(){  incrementa();  }
    protected void incrementa(){   cont+=1;    }
    public String toString(){
        return "Este objeto tiene cont="+cont;
    }
}
public class Derivada extends Base{
    Derivada(){  incrementa(2);  }
    public void incrementa(){  cont+=2;      }
    public void incrementa(Integer val){    cont+=val;    }
    public static void main(String[] args) {
        Base obj1 = new Derivada();
        Derivada obj2 = new Derivada();
        obj1.incrementa();
        obj2.incrementa(2);
        System.out.println(obj2);
    }
}
```

Salida:	**Justificación:**

4.- UML (20 puntos)

Awake es un programa concurso donde se inscriben cientos de personas. De cada inscrito se sabe su nombre, ocupación, edad y link de video de Youtube con su audición para el programa. Para cada episodio de entre todos los inscritos se seleccionan 7 participantes. Para estos 7 participantes además de sus datos originales la producción registra su cédula. Cada participante cuenta monedas de 25 centavos de dólar en un periodo de 24 horas. Luego de este periodo, el participante debe dar un estimado en dólares de lo contado. Posterior a esto, la producción del programa hace la cuenta real de las monedas contadas por el participante.

En el comienzo de cada episodio, se debe obtener cuál fue el participante que contó menos monedas y cuál fue el participante que fue menos preciso en su cuenta. Estos participantes son eliminados del concurso.

Ya con 5 participantes, cada episodio de Awake tiene 4 desafíos. Cada desafío tiene un nombre y una categoría: Motricidad Fina, Patrones y Motricidad Gruesa. De cada desafío se almacena el tiempo que le tomó a cada participante completar el desafío. El participante que

completó el desafío en menor tiempo pasa automáticamente al siguiente desafío. El participante que hizo el desafío en el mayor tiempo se irá del concurso. Por cada desafío pasado, los participantes acumulan $1000. Este proceso se repetirá en cada desafío hasta que quede un solo participante, quien es el ganador final de “Awake” que se lleva la cantidad de 10,000 dólares. Por cada episodio se debe saber el ganador, ya que la producción del programa hace difusión en sus redes sociales.

3.- (10puntos) Considere el siguiente método

```
static void f(int k, int[] A, String S) {
    int j = 1 / k;
    int len = A.length + 1;
    char c;
    try {
        c = S.charAt(0);
        if (k == 10) j = A[3];
    } catch (ArrayIndexOutOfBoundsException ex) {
        System.out.println("array error");
        throw new NullPointerException();
    } catch (ArithmeticException ex) {
        System.out.println("error aritmético");
    } catch (StringIndexOutOfBoundsException ex) {
        System.out.println("error al obtener caracter de cadena!");
    } finally {
        System.out.println("Proceso Terminado!");
    }
    System.out.println("AP");
}
```

a) **(9 puntos)** En la clase **Test,** para cada una de las llamadas al método f en la tabla 1, diga que se muestra en pantalla al invocarlo (si es que algo se muestra) y cuáles excepciones son lanzadas (si es que hay).

```
public class Test{
    public void main(String[] args){
        int[] X = {3,4,5}
        //invocacion metodo f
    }
}
```

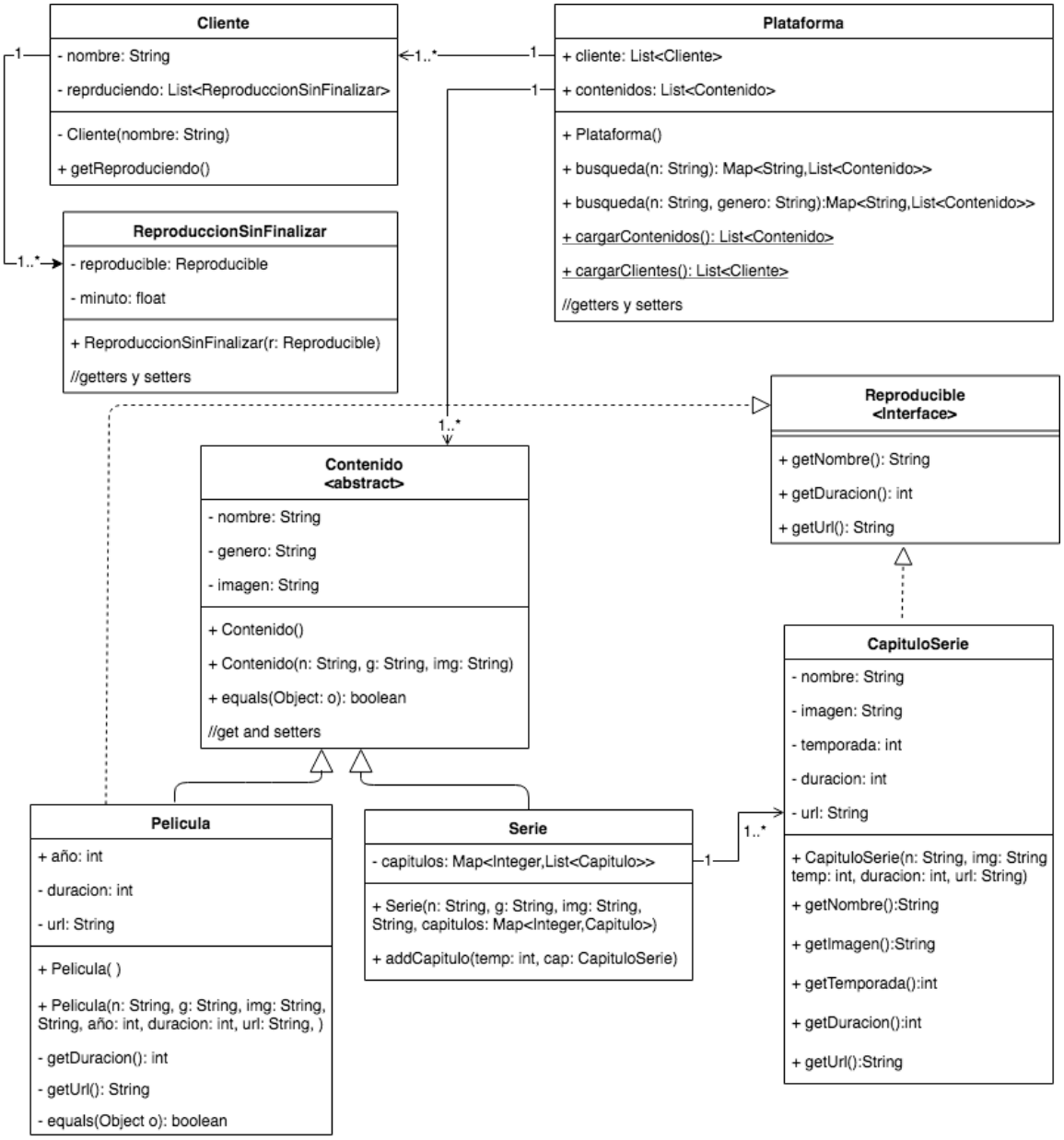
Invocación	**Respuesta**
f(0, X, "hi");	
f(10, X, "");	
f(10, X, "bye");	

Tabla 1

b) **(1 punto)** ¿Por qué en el método **`f(int k, int[] A, String S)`** no es obligatorio incluir una cláusula throws para la excepción “NullPointerException”?

5.- Desarrollo (55 PUNTOS)

Se le ha pedido a usted que ayude en el desarrollo de una plataforma de **Streaming**. En base al diagrama de clases que se presenta a continuación:

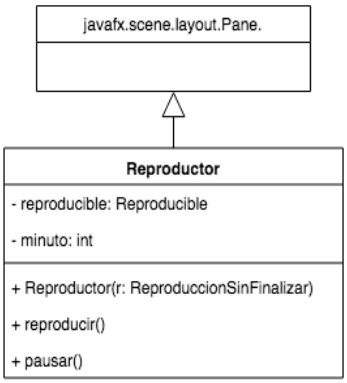


a. (5 puntos) ¿Qué es sobre-escritura? ¿En la clase Película escriba la firma de los métodos han sido sobrescritos?
b. (5 puntos) ¿Qué es sobrecarga? En las clases Plataforma y Película qué métodos han sido sobrecargados
c. (14 puntos) Implemente la clase Película. Considere que una película es igual a otra si sus atributos nombre y año son iguales.
d. **En la clase Plataforma realice lo siguiente:**
    i. **(6 puntos)** Implemente el método cargarContenido(): List<Contenido>. Este método lee el archivo serializado contenidos.dat que contiene un ArrayList con los contenidos de la plataforma y lo retorna.
    ii. **(8 puntos)** Implemente el método búsqueda(nombre: String): Map<String, List<Contenido>> que recibe un String y retorna un HashMap de los contenidos cuyos nombres contiene el parámetro. donde las claves son los géneros de los contenidos y el valor asociado es un ArrayList<Contenido> con los contenidos de ese género.

e. Parte Gráfica

La variable de instancia **reproduciendo** en la clase **Cliente** tiene objetos de tipo Reproducible que son las ***Peliculas*** o ***CapituloSerie*** que el cliente ha empezado a ver pero que no ha terminado.

Para esta parte se adicionan las siguientes clases al UML original:



La clase **VistaContinuarViendo** muestra la información de las películas y series que el usuario no ha terminado de reproducir (ver imagen 1). Cuando el usuario da click sobre alguna de las películas o series mostradas aparece el reproductor para ese contenido (ver imagen 2)

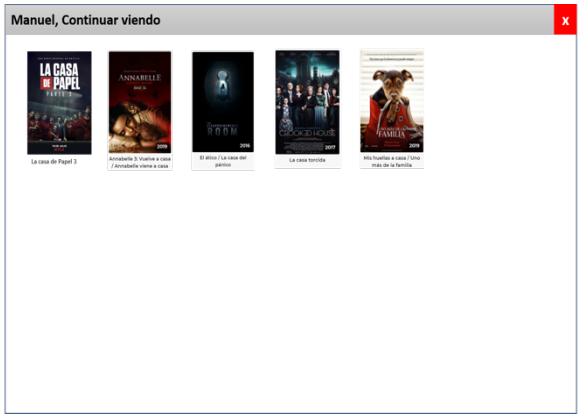


Imagen 1

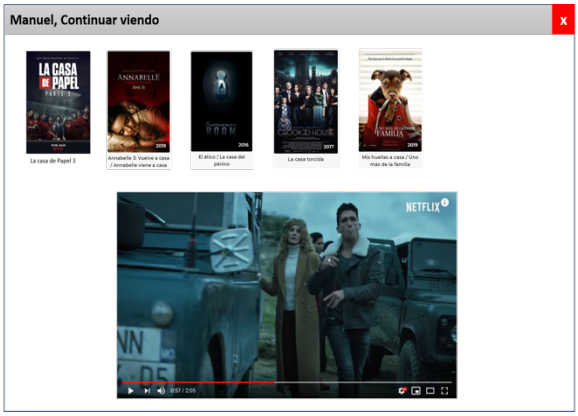


Imagen 2

El constructor de esta clase recibe una instancia del Cliente que está usando actualmente la aplicación.
La clase **Reproductor** extiende de Pane. Esta clase crea un contenedor con un reproductor multimedia. El constructor de esta clase recibe un objeto de tipo ReproduccionSinFinalizar que es el que se va a reproducir. **ESTA CLASE YA EXISTE**

(17 puntos) Implemente la clase **VistaContinuarViendo** para que haga lo descrito anteriormente:

- Cree la clase VistaContinuarViendo con sus respectivos atributos
- Cree su constructor
- Muestre las películas y capítulos que el usuario no ha terminado de ver
- Fije la(s) accion(es) correspondiente para cargar el reproductor cuando se da click sobre una de las películas o capítulos de serie que no se han terminado de ver.