	IES LUIS VIVES - 1º DAW - **EXAMEN 3ª EVALUACIÓN** Módulo "Programación"	Curso 2022/2023 Fecha:15/05/2023
Nombre y Apellidos		Nota:

Parte de Teoría (2 ptos)

En todas las preguntas tipo test hay una o varias respuesta correctas
Cada pregunta completamente bien contestada suma 0,5 ptos
Las preguntas mal contestadas o no contestadas suman 0 ptos.
Una respuesta incorrecta de una pregunta resta el valor de una respuesta correcta de la misma pregunta.

Pregunta 1: En java los métodos finales **:**

a) Pueden modificarse en otra clase que herede de aquella en la que están definidos
b) Puede haber otro método con el mismo nombre en la misma clase con diferente número de parámetros
c) No pueden utilizar atributos no estáticos.
d) Sólo pueden utilizar atributos finales.
e) Ninguna de las anteriores.

Pregunta 2: Dada la clase siguiente

```
public class ClaseA {
      int x=3;
      private int y=2;
      public void metodo(){}
}
```

Indicar cuáles de las siguientes declaraciones de métodos o variables añadidas a la ClaseB serían válidas teniendo en cuenta ésta está declarada como:

class ClaseB extends `ClaseA` {…}

a) double x=33;
b) public int y=11;
c) public int metodo(){return 3;}
d) private void metodo(){}
e) public void metodo(int x){}
f) Ninguna de las anteriores.

Pregunta 3:

Indicar que afirmaciones son ciertas respecto al lenguaje java dadas las siguientes declaraciones:

```
public interface MiItf {
   int metodo1(int a);
}
public class ClaseC implements MiItf{
      public int metodo1(int a) {
            return 0;
      }
}
```

a) La siguiente asignación es correcta List <String> lista= new Vector<String>();
b) La siguiente asignación es correcta MiItf mt= new ClaseC();
c) Se podría crear una clase con public class ClaseD implements MiItf{} sin definir ningún método en la misma
d) Si la clase ClaseC no codificara el metodo1 habría que haberla definido con la palabra abstractClass entre el public y el class
e) Ninguna de las anteriores.

Pregunta 4: Dado el siguiente código:

```
public class GrafThread extends Thread{
       JFrame frame =new JFrame("Test-3 ");
       JButton miBoton;
       JTextField texto;
       Container panel;

    public GrafThread() throws InterruptedException {
       panel = frame.getContentPane();
       miBoton= new JButton("Pulsar");
       panel.add(miBoton,BorderLayout.NORTH);
       texto= new JTextField("0");
       panel.add(texto,BorderLayout.SOUTH);
       miBoton.addActionListener(new ActionListener(){
                    @Override
                    public void actionPerformed(ActionEvent arg0) {
                           texto.setText("XX");
                    }
       });
       frame.setSize(100,100);
       frame.setVisible( true );
    }
    public void run() {
       int i;
       String msg="";
       for (i=1; i<10; i++)
                    try {
                           msg=msg+i;
                           Thread.sleep(1000);
                           texto.setText(msg);
                           System.out.println(msg);

                    } catch (InterruptedException e) {     }
    }

    public static void  main(String args[]) throws InterruptedException {
       GrafThread o= new GrafThread();
         o.start();
         o.join();
         System.out.println("FIN");
         o.frame.dispose();
    }
}
```

Indicar que afirmaciones son ciertas

a) Cuando se pulsa el botón aparece en la zona de texto un XX añadido a lo que ya había.
b) El programa, después de 3,3 segundos de empezar la ejecución, si no se ha pulsado el botón, muestra en la zona de texto :012
c) Si transcurridos 3,3 segundos desde el comienzo de la ejecución se pulsa el botón aparecerá en la zona de texto XX y , 3,3 segundos después de pulsar el botón en la zona de texto tendremos el texto XX456
d) En la consola se escribe lo mismo que aparece en la zona de texto de la ventana
e) A los 9 segundos de empezar la ejecución el programa cierra la ventana y, sin escribir FIN, finaliza.
f) El número 0 nunca se verá en la zona de texto
g) Ninguna de las anteriores.

Solución:

P1: Son ciertas
b)Puede haber otro método con el mismo nombre en la misma clase con diferente número de parámetros

P2:
a)double x=33;
b)public int y=11;
e)public void metodo(int x){}
P3:
a) y b)
P4:
g
.

Parte de Programación (8 ptos/10 ptos)

Ejercicio 1 (5,5 puntos)

Se desea codificar una aplicación para gestionar una biblioteca.
Se dispone de las clases siguientes:

```
public class Persona {
    private String nombre;
    public Persona(String nombre) {    this.nombre = nombre;     }
    public String getNombre() { return nombre;      }
}
public class Autor extends Persona {
    public Autor(String nombre) {   super( nombre);       }
}
public class PersonalIES extends Persona {
    int penalizacion=0;
    public PersonalIES(String nombre) {  super(nombre);   }
}
public class Alumno extends PersonalIES{
   String grupo;
   public Alumno(String nombre, String grupo) {
      super(nombre);
      this.grupo=grupo;
   }
}
public class Profesor extends PersonalIES {
   String departamento;
   public Profesor(String nombre, String departamento) {
      super(nombre);
      this.departamento=departamento;
   }
}
public enum ErroresDevolucion {
  SINERROR, //No hay ningún error
  PERIODOSUPERADO, //Se ha superado el número de días que el usuario podía tener el libro
  USUARIOSINPRESTAMO; //El usuario no tiene ese libro en préstamo
}
public class Libro {
   String titulo;
   String codigo;
   private int ejemplaresTotales;
   private Vector <Persona> autores;
   Vector <Prestamo> prestamos;

   public Libro(String titulo, String codigo, int ejemplaresTotales) {
        super();
        this.titulo = titulo;
        this.codigo = codigo;
        this.ejemplaresTotales = ejemplaresTotales;
        autores=new Vector <Persona>();
        prestamos=new Vector <Prestamo>();
   }
   public void añadirAutor(String a){
         autores.add(new Autor(a));
   }
   public boolean añadirPrestamo(PersonalIES p, int diaActual){
   // Presta el libro. Devuelve false sino se puede prestar porque todos los ejemplares
están prestados.
      CODIFICAR 1
   }

public ErroresDevolucion devolverPrestamo(PersonalIES p, int dia){
   //Devuelve uno de los valores enumerados de ErroresDevolución. El que corresponde.
```

CODIFICAR 2

```
    }
}
public class Prestamo {
  PersonalIES persona;
  int diaPrestamo;  //Indica el número de día en el que se realizó el préstamo
  public Prestamo(PersonalIES p, int diaPrestamo) {
       super();
       persona = p;
       this.diaPrestamo = diaPrestamo;
  }
}
public class Biblioteca {
  Hashtable <String,Libro> libros;
  Vector <PersonalIES> lectores;
  int diaActual=1;  //Indica el número de día actual. Se irá incrementando cada día
  Teclado t= new Teclado();

  Biblioteca() throws IOException{
```

CODIFICAR 3

```
  }
  Libro buscarLibro(String codigo){
```

CODIFICAR 4

```
  }
  PersonalIES buscarPersona(String nombre){
        for (PersonalIES l:lectores)
              if (l.getNombre().equalsIgnoreCase(nombre))  return l;
        return null;
  }
  boolean prestarLibro(Libro l, PersonalIES p){
        return (l.añadirPrestamo(p, diaActual));
  }
  void prestar () throws IOException{
```

CODIFICAR 5

```
  }
  ErroresDevolucion devolverLibro(Libro l, PersonalIES p){
        return (l.devolverPrestamo(p, diaActual));
  }
  void devolver () throws IOException{
```

CODIFICAR 6

```
  }
  void mostrarPrestamos(){
        for (Libro l:libros)
              System.out.println(l);
  }
  void restarPenalizaciones(){
```

CODIFICAR 7

```
  }
  void altaLibro() throws IOException {
        System.out.println("Título");
        String titulo= t.leerString();
        System.out.println("Código");
        String cod= t.leerString();
        System.out.println("Num Ejemplares");
        int numEjemplares=t.leerInt();
        System.out.println("Autores. Línea en blanco para finalizar");
```

CODIFICAR 8

```
        }
  void altaLector() throws IOException {
        PersonalIES p=null;
        System.out.println("Tipo: A-Alumno P-Profesor");
        String tipo= t.leerString();
        System.out.println("nombre");
        String nombre= t.leerString();
        CODIFICAR 9
     }
  void menu() throws IOException{
      int opc=0;
      do {
        System.out.println("1-Prestar");
        System.out.println("2-Devolver");
        System.out.println("3-Mostrar");
        System.out.println("4-Incrementar dia");
        System.out.println("5-Restar Penalizaciones");
        System.out.println("6-Alta libro");
        System.out.println("7-Alta lector");
        System.out.println("0-Fin");
        opc=t.leerInt();
        switch (opc){
          case 1: prestar(); break;
          case 2: devolver(); break;
          case 3: mostrarPrestamos(); break;
          case 4: diaActual ++; break;
          case 5: restarPenalizaciones(); break;
          case 6: altaLibro(); break;
          case 7: altaLector(); break;
        }
      }while (opc!=0);
  }
  public static void main (String arg[]) throws IOException{
        Biblioteca b=new Biblioteca();
        b.menu();
  }
}
```

Codificar los métodos no codificados para que:

A.- (0,5 ptos) En el **constructor de la Biblioteca se inicialicen los atributos necesarios.**

Codifica el método **buscarLibro** que recibirá un código de libro y devolverá el libro asociado o null si no está entre los almacenados

B.- (1,25 ptos) Si se selecciona la opción del menú **Prestar**, después de pedir el nombre del alumno o profesor y el código del libro, realice el préstamo del libro a ese alumno o profesor. Es decir, **cree** un **objeto Prestamo** con los datos de ambos y lo **añada al vector de préstamos** de ese libro. Si el **profesor** o el **alumno no existen** en el vector de lectores deberá **mostrar** un mensaje de **error** indicándolo, si el **libro no existe** deberá mostrar un mensaje indicándolo. Si el préstamo lo solicita un **alumno y ese alumno tiene en el campo penalizado un valor mayor de 0 no** se le **prestará** el libro y se mostrará un mensaje con el número de días de penalización de ese alumno. Si **hay más ejemplares** de un libro **prestados** de los **que** figuran en el atributo **ejemplaresTotales** del libro **no se prestará** y se mostrará un mensaje indicándolo. En la codificación del método prestar() se deberá **utilizar** una llamada útil al método `boolean prestarLibro(Libro l, PersonalIES p)`

C.- (1,25 ptos) Si se selecciona la opción del menú **Devolver**, después de **pedir** el nombre del **alumno** o **profesor** y el **código** del **libro**, realice el préstamo del libro de ese alumno a ese alumno o profesor. Es decir, **eliminar**, si todo es correcto, **el objeto préstamo** de ese lector para ese libro. Si la **devolución** la solicita un **alumno** y ese alumno ha **tardado más de 7 días en devolver** el libro se le deberá **sumar** a su campo **penalizado** el **número de días de retraso** (el préstamo es de 7 días cualquier cantidad superior a 7 dias se considera retraso) y se mostrará un mensaje : **PERIODOSUPERADO**. Es decir, si devuelve el libro a los 9 días se le penalizarán 2 días. Si el **profesor o el alumno** no existen en el vector

de **lectores** deberá mostrar un **mensaje de error** indicándolo, si el **libro no existe** deberá mostrar un **mensaje** indicándolo. Si el **usuario no tiene ese libro** en préstamo se mostrará el **mensaje USUARIOSINPRESTAMO**. En la codificación del método devolver() se deberá utilizar una llamada útil al método `boolean prestarLibro(Libro l, PersonalIES p)`

D.- (1,00 ptos) Si se selecciona la opción del menú **Mostrar** se deberá mostrar un **listado** con los datos **de** los **libros y los préstamos** de cada uno de ellos con el formato del ejemplo siguiente:

```
Libro [titulo=El barón rampante, codigo=c1, ejemplaresTotales=4, autores=Italo Calvino,

Libro [titulo=Programación en Java, codigo=c3, ejemplaresTotales=5, autores=José Manuel
Pérez,Juan Velasco,
Prestamo [=Alumno:Ana Torroja, grupo:1DAW, sancionado:0, diaPrestamo=1]
Prestamo [=Profesor:Oscar Alonso, dpto:Informática, sancionado:0, diaPrestamo=2]

Libro [titulo=El libro de las recetas, codigo=c7, ejemplaresTotales=1, autores=Gustavo
Alonso,
Prestamo [=Alumno:Ana Torroja, grupo:1DAW, sancionado:0, diaPrestamo=2]
```

Codificar los métodos **toString** de todas las clases necesarias.

E.- (0,5 ptos) Si se selecciona la opción del menú **Restar Penalizaciones** se **decrementará** en **1 la penalización de todos los lectores que tengan penalización**.

F.- (0.5 ptos) Codificar la opción de **altaLibro** () para añadir un libro con todos sus atributos. Recordad que un libro puede tener varios autores y hay que pedirlos todos por teclado cuando se solicitan los datos del libro en esta opción. Cada **autor de un libro se almacenará en una casilla** diferente del **vector** de **autores** del libro.

G.- (0.5 ptos) Codificar la opción de **altaLector** () para añadir un objeto PersonalIES (puede ser alumno o profesor en función de lo que elija el usuario) con todos sus atributos.

Los **profesores no tienen penalización** pero el sistema está preparado para que si en algún momento la tuvieran que tener se pudiera implementar.

Los **errores** mostrados deben tener un mensaje **adaptado al tipo de error** y **no** uno **genérico**.

Ejercicio 2 (1.5 puntos)

Dada la clase MiLista siguiente:

```
public class MiLista <Tipo> {
	protected Vector <Tipo> pila;

	MiLista (){
		pila = new Vector<Tipo>( );
	}
	int insertar (Tipo o){
		pila.add(o);
		return 0;
	}
	Tipo obtener (){
		Tipo elemento=null;
		if (pila.size()>0)
		  elemento=pila.remove(pila.size()-1);
		return elemento;
	}
	int tamano(){
		return pila.size();
	}
	public static void main (String arg[]){
		MiLista <String>ls= new MiLista <String>();
		ls.insertar("uno");
		ls.insertar("dos");
		System.out.println(ls.obtener());
```

```
        }
}
```

A.- (1,25 ptos) Codificar una clase **ListaConMezcla que herede de la anterior** e implemente el interfaz ItfMezclar siguiente:

```
public interface ItfMezclar {
        void mezclar (int n);
}
```

El método mezclar debe intercambiar el valor de 2 elementos de la lista tantas veces como el número recibido como parámetro. Se **contarán sólo los intercambios efectivos**. No se deberá mezclar nada si el número de elementos en la lista es menor de 2.

B.- (1,25 ptos) Dada la clase Carta siguiente

```
public class Carta {
   Palo p;
   int numero;
   public Carta(Palo p, int numero) {
        this.p = p;
        this.numero = numero;
   }
   public String toString() {
       return "Carta [p=" + p + ", numero=" + numero + "]";
   }
}
```

donde

```
public enum Palo {   OROS, COPAS, ESPADAS, BASTOS;  }
```

Crear un método main que cree un objeto llamado baraja de tipo ListaConMezcla en el que se inserten las 40 cartas de la baraja española

Después se deberá llamar al método mezclar (1000) y obtener las 3 primeras cartas de la baraja que serán mostradas en pantalla.

Todos las clases de este ejercicio deben estar creadas en un **paquete** llamado **pklistaconmezcla**

Ejercicio 3(1.5 puntos)

Realizar una aplicación gráfica que :

- Muestre una ventana en pantalla que tenga un botón.
- Cada 3 segundos la ventana desaparecerá y aparecerá en otro sitio aleatorio de la pantalla
- Cada vez que se pulsa el botón, este se deshabilitará y se contará como que se ha pulsado una vez más.
- Después de haber mostrado 5 ventanas como la indicada, se mostrará en consola el número de veces que se pulsó el botón y finalizará la aplicación.

Se puede suponer que la pantalla tiene un tamaño de 1500 x 800 pixeles.

SOLUCION

EJER 1

A- CODIFICAR 3:

```
  Biblioteca() throws IOException{
       String linea;
       int numEjem;
       BufferedReader bf= new BufferedReader(new FileReader( NOMFICHLIBROS ));
       libros= new Vector<Libro>();
       while (bf.ready()){
             linea= bf.readLine();
             String campos[]=linea.split(";");
             numEjem=Integer.parseInt(campos[2]);
             Libro l= new Libro(campos[0], campos[1], numEjem);
             for (int i=3; i<campos.length; i++)
                   l.añadirAutor(campos[i]);
             libros.add(l);
       }
       bf.close();
       BufferedReader bf2= new BufferedReader(new FileReader( NOMFICHLECTORES ));
       lectores= new Vector<PersonalIES>();
       PersonalIES p;
       while (bf2.ready()){
             linea= bf2.readLine();
             String campos[]=linea.split(";");
             if (campos[0].equals("A"))  p= new Alumno(campos[1], campos[2]);
             else p= new Profesor(campos[1], campos[2]);
             lectores.add(p);
       }
       bf2.close();
  }
```

B.- **Clase Biblioteca** CODIFICAR 4:

```
  void prestar () throws IOException{
         System.out.println("Nombre persona");
         String nombre=t.leerString();
         PersonalIES p=buscarPersona(nombre);
         if (p==null) System.out.println("Persona no encontrada");
         else {
           System.out.println("Código libro");
           String cod=t.leerString();
           Libro l=buscarLibro(cod);
           if (l==null)System.out.println("Libro no encontrado");
           else
             if (p instanceof Profesor || p.penalizacion==0){
                 if (!prestarLibro(l, p))
                      System.out.println("No hay ejemplares libres");
                 else
                   System.out.println("Libro prestado");
             }
             else System.out.println("Sancionado "+p.penalizacion+ " dias");
         }

  }
```

Clase Libro CODIFICAR 1:

```
   public boolean añadirPrestamo(PersonalIES p, int diaActual){
// devuelve false si todos los ejemplares del libro están prestados.
         boolean ok=false;
         if (prestamos.size()<ejemplaresTotales){
           prestamos.add(new Prestamo(p,diaActual));
           ok=true;
         }
         return ok;
   }
```

C.- **Clase Biblioteca** CODIFICAR 5:

```
void devolver () throws IOException{
```

```
        System.out.println("Nombre persona");
        String nombre=t.leerString();
        PersonalIES p=buscarPersona(nombre);
        if (p==null) System.out.println("Persona no encontrada");
        else {
          System.out.println("Código libro");
          String cod=t.leerString();
          Libro l=buscarLibro(cod);
          if (l==null) System.out.println("Libro no encontrado");
          else {
             ErroresDevolucion ed=devolverLibro(l, p);
             if (ed != ErroresDevolucion.SINERROR)
                   System.out.println(ed);
          }
        }
  }
Clase Libro   CODIFICAR 2:
   public ErroresDevolucion devolverPrestamo(PersonalIES p, int dia){
         //Devuelve uno de los valores enumerados de ErroresDevolución
         ErroresDevolucion ok=ErroresDevolucion.USUARIOSINPRESTAMO;
         Prestamo prEncontrado=null;
         for (Prestamo pr:prestamos)
            if (pr.persona.getNombre().equalsIgnoreCase(p.getNombre())){
                  ok=ErroresDevolucion.SINERROR;
                  prEncontrado=pr;
                  if ( dia- pr.diaPrestamo > 7 ){
                         ok=ErroresDevolucion.PERIODOSUPERADO;
                         if (pr.persona instanceof Alumno   )
                                pr.persona.penalizacion=dia- pr.diaPrestamo-7;
                  }
            }
         if (prEncontrado!=null) prestamos.remove(prEncontrado);
         return ok;
   }
```

D.- Clase Libro

```
   public String toString() {
       String s="Libro [titulo=" + titulo + ", codigo=" + codigo;
       s+= ", ejemplaresTotales=" + ejemplaresTotales;
       s+=", autores=";
       for (Persona a:autores)   s+=a+",";
       s+="\n";
       for (Prestamo pr:prestamos) s+=pr+"\n";
       return s;
   }
Clase Prestamo
  public String toString() {
       return "Prestamo [=" + persona + ", diaPrestamo=" + diaPrestamo + "]";
  }
Clase Persona
  public String toString() {
       return "" + nombre + "";
  }
Clase PersonaIES
  public String toString() {
       return "PersonalIES [penalizacion=" + penalizacion + "]";
  }
Clase Profesor
  public String toString() {
return "Profesor:"+ getNombre() +", dpto:" + departamento + ", sancionado:"+ penalizacion;
  }
Clase Alumno
  public String toString() {
       return "Alumno:"+ getNombre() +", grupo:" + grupo + ", sancionado:"+ penalizacion;
  }
```

E.- CODIFICAR 6:

```
void restarPenalizaciones(){
      for (PersonalIES p:lectores)
            if (p.penalizacion>0)
                  p.penalizacion--;
}
```

EJER 2

A.-

```
public class PilaConMezcla <Tipo>extends Pila<Tipo>{
   void mezclar (int n){
        int p1, p2, c=n;
        Tipo t1, t2;
        if (pila.size()>2)
          while (c>0){
            p1=(int)(Math.random()* this.tamano() );
            p2=(int)(Math.random()* this.tamano() );
            if (p1!=p2){
              c--;
              t1=pila.elementAt(p1);
              t2=pila.elementAt(p2);
              pila.remove(p1);
              pila.insertElementAt(t2, p1);
              pila.remove(p2);
              pila.insertElementAt(t1, p2);
            }
          } //while
   } //mezclar
} //class
```

B.-

```
public static void main (String ar[]){
        PilaConMezcla <Carta>baraja= new PilaConMezcla <Carta>();
        for (int i=1;i<10; i++)
          for (Palo palo:Palo.values())
            baraja.insertar(new Carta(palo, i));
        baraja.mezclar(100);
        baraja.mostrar();
        System.out.println("Extraemos 3 cartas:");
        System.out.println(baraja.obtener());
        System.out.println(baraja.obtener());
        System.out.println(baraja.obtener());
    }
```