Examen de Programación Orientada a Objetos (Plan 014)

15 de enero de 2020

Nº matrícula:_____ **Grupo:**_____ **Nombre:** _____

Apellidos: _____

NOTA: El valor de cada una de las preguntas es de 1 punto sobre la nota final del examen.

1. Dadas las siguientes porciones de clases:

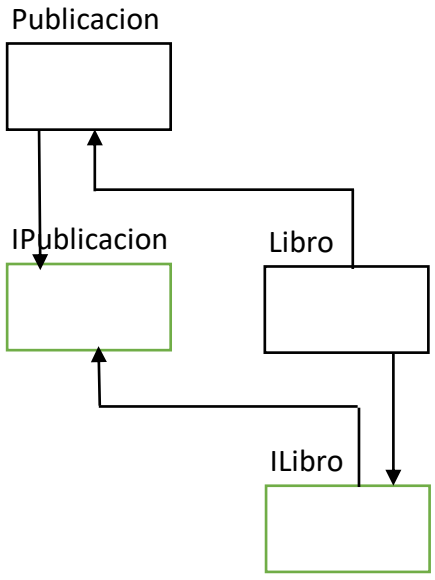
```
public class Punto2D {
    private int x,y;
    …
    public boolean equals(Punto2D p) {
      return p.getX()==x && p.getY()==y;
    }
}

public class Punto3D extends Punto2D {  private int z;  … }
```

   a) Implementar el método `equals` de la clase `Punto3D`.

```
public boolean equals (Punto3D p) {
       return super.equals(p) && p.getZ()==z;
}
```

   b) Completar la frase:
   Se dice que el método `equals` de la clase `Punto3D` `redefine o sobreescribe` al método `equals` de la clase `Punto2D`.

2. Dado el siguiente modelo de clases e interfaces, escribir las cabeceras (*únicamente las cabeceras*) de cada clase e interfaz del modelo. `IPublicacion` e `ILibro` son interfaces; `ILibro` especializa a `IPublicacion`. `Libro` es una clase que especializa a `Publicacion`.




```
public interface IPublicacion

public interface ILibro extends IPublicacion

public class Publicacion implements
IPublicacion

public class Libro extends Publicacion
implements ILibro
```

3. Para conocer el capital (valor total del conjunto de las acciones) de cada accionista de Telefónica, dispondremos una clase `Accionista`. Esta clase contendrá una propiedad o atributo `cantidad` que contenga el número de acciones que posee un accionista, y otra propiedad `valorAccion` que contenga el valor de cada acción. Cuando se modifique el valor de la acción de Telefónica, se podrá realizar empleando una sola instrucción, y se hará efectivo para todos los accionistas instanciados en la aplicación. **Se pide implementar** la clase `Accionista` conteniendo únicamente la declaración de sus propiedades, el método `capital`, que devuelve el valor total del conjunto de acciones de ese accionista y el método `setValorAccion`. *No es necesario implementar constructores ni métodos adicionales a capital que la clase `Accionista` debería tener*.

```
public class Accionista {
    private int cantidad;
    private static double valorAccion;

    private static void setValorAccion(double valor) {
        valorAccion = valor;
    }
    public double capital() {
        return valorAccion*cantidad;
    }
}
```

4. **Responder brevemente/completar**. En una implementación que utilice polimorfismo:
   a) ¿Es obligatorio que exista un método polimórfico?
      Sí, es obligatorio

   b) ¿Pueden existir varios métodos polimórficos?
      Sí, es posible

   c) ¿De existir uno o varios métodos polimórficos deben ser estáticos?
      No es necesario

   d) De existir uno o varios métodos polimórficos ¿Pueden estar definidos en una superclase?
      Sí. Pueden y deben estar definidos en una superclase

   e) ¿Puede existir polimorfismo sin emplear interfaces?
      Sí, es posible

   f) ¿Puede existir polimorfismo sin emplear clases abstractas?
      Sí, es posible

   g) Las instancias de propiedades que se usan de manera polimórfica se declaran usando como tipo: una superclase y se instancian usando como tipo: una clase derivada.

Nº matrícula:_____ **Grupo:**_____ **Nombre:** _____

Apellidos: _____

5. Escribir la cabecera de un interfaz genérico `IConductor` que admita un objeto de tipo `Persona`, o cualquiera de sus clases derivadas, y un objeto de tipo `String` (que supuestamente contiene información de su carnet de conducir). Además, `IConductor` debe incorporar la funcionalidad descrita en otro interfaz `IUsuario`.

```
public interface IConductor <P extends Persona, S String> extends IUsuario
```

Utilizando las propiedades:

```
Persona juan = new Persona("Juan", 23, "B2");
String infCarnet = "…";  …
```

y suponiendo que existe la clase `Conductor` que implementa `IConductor`, se pide declarar e instanciar una propiedad `chofer` de tipo `Conductor`.

```
Conductor<Persona, String> chofer = new Conductor <Persona, String>(juan,
infCarnet);
```

PAGINA A SUCIO PARA USO DEL ALUMNO.
SU CONTENIDO NO SE CONSIDERA PARTE DE LAS RESPUESTAS DEL EXMANEN.

Nº matrícula:_____ **Grupo:**_____ **Nombre:** _____

Apellidos: _____

6. **Completar**. En el patrón Modelo/Vista/Controlador implementado con `java.beans`.

   a) `public class` -Modelo- `implements` `PropertyChangeListener`

   b) La firma del método que recoge el evento es:
   `public void propertyChange (PropertyChangeEvent e)`

   c) El método que lanza un evento, perteneciente a la clase `PropertyChangeSupport` se llama:
   `firePropertyChange`.

7. Deseamos traspasar los datos existentes en un `LinkedList<Integer, String>` a un `HashMap`, sin hacer uso del constructor que lo implementa directamente. Supondremos que existe la siguiente clase `Par`, que nos permitirá definir un par `<Integer, String>`.

```
public class Par<I,S> {
    private I i; private S s;

    public Par(I i, S s) {this.i=i; this.s=s;}
    public S getString() {return s;}
    public I getInteger() {return i;}
}
```

Se pide: completar la implementación del método `listToMap` definido a continuación:

```
public HashMap<Integer,String> listToMap (LinkedList<Par<Integer,String>> lista){
    Par<Integer,String> e;

     ListIterator <Par<Integer,String>> li = lista.listIterator();
     HashMap<Integer,String> hashMap = new HashMap<Integer,String>();

    while (li.hasNext()) {
        e = li.next();
        hashMap.put(e.getInteger(),e.getString());
    }
    return hashMap;
}
```

8. En una implementación del juego del Ajedrez:

   a) ¿Dónde se debe poner el atributo que indica el color de la ficha (de la Torre o del Caballo, por ejemplo)?
   En una superclase -*Figura*- de las clases -*Torre*-, -*Caballo*-, etc.

   b) Suponiendo que el tablero tiene un tamaño SIZE, escribe una sola instrucción que lo declare y lo instancie.
   ```
   private Figura[][] tablero = new Figura[SIZE][SIZE];
   ```

   c) ¿Qué clase debe actuar como modelo en el esquema modelo/vista/controlador? Escribe la cabecera de esa clase.
   ```
   public class Tablero implements PropertyChangeListener
   ```

9. En una simulación, por ejemplo *el juego de la vida*, existe la necesidad de inicializar aleatoriamente una matriz de enteros: `int tablero[][] = new int[100][100]` con un número de casillas que indiquen un estado y otro número diferente de casillas que indiquen otro estado; por ejemplo: `500` casillas con estado `1` (que supondremos que son zorros; `ZORROS=1`) y `6000` casillas con estado `2` (que supondremos que son conejos; `CONEJOS=2`). Inicialmente el tablero está inicializado con valores cero: `VACIO=0`.

   Se pide implementar el método `public void initRandom(int numCasillas, int estado)`, que permitirá realizar la inicialización: `initRandom(500,ZORROS);` `initRandom(6000,CONEJOS)`. El método también deberá informar de que el modelo ha variado.

```
public void initRandom(int numCasillas, int estado) {
    Random r = new Random(); int i = 0;
    do {
        int x = r.nextInt(100); int y = r.nextInt(100);
        if (tablero[x][y]==VACIO) {
            tablero[x][y] = estado; i++;
        }
    } while (i<numCasillas);
    setChanged(); notifyObservers(); //o en su version con java.beans
}
```

Nº matrícula:_____ **Grupo:**_____ **Nombre:** _____

Apellidos: _____

10. En una aplicación de tablero (barquitos, ajedrez, etc.) se desea validar las tiradas (o jugadas), que se codifican usando un `String` con el formato: *"LetraNumero"* (ejemplos: "`B7`" o "`H9`"). Supondremos que cualquier letra es válida (en mayúscula o minúscula) y que cualquier número, del 0 al 9, será válido. Las tiradas, por tanto, tendrán siempre un tamaño de dos caracteres (el primero letra y el segundo número). Supondremos definidas las excepciones: `LongitudException`, `LetraException`, `NumeroException`, que extienden a `ExJugada`, que a su vez extiende a `Exception`. **Implementar el método** estático `void validaJugada`, que valida las tiradas (o jugadas).

```
public static void validaJugada(String jugada) throws ExJugada {
    if (jugada.length()!=2)
        throw new LongitudException();
    else {
        if (!Character.isLetter(jugada.charAt(0)))
            throw new LetraException();
        if (!Character.isDigit(jugada.charAt(1)))
            throw new NumeroException();
    }
}
```

PAGINA A SUCIO PARA USO DEL ALUMNO.
SU CONTENIDO NO SE CONSIDERA PARTE DE LAS RESPUESTAS DEL EXMANEN.