Certificación de Capacidades de Analista de Conocimiento del Programa 111 mil

Primer Borrador, versión 1.0

Ejercicio 1 - Construir código de acuerdo a especificaciones

Un *grupo de biólogos* requiere un sistema que permita administrar sus *experimentos*. Se dispone de n *laboratorios*. Cada laboratorio tiene sólo un experimento a la vez.

En cada momento puede haber hasta n experimentos en curso, no más de uno por laboratorio. Cuando se quiere iniciar un experimento, el *sistema* selecciona algún laboratorio (cuyo estado sea libre) donde llevarlo a cabo.

A cada experimento se le van registrando observaciones. Una observación puede tener tres posibles resultados: (1) observación **anómala**, (2) observación legítima que **confirma** la hipótesis, (3) observación legítima que **contradice** la hipótesis. Si en cualquier momento el número de observaciones anómalas supera el número de observaciones legítimas, el experimento se cancela al instante.

Un experimento está avanzado si cuenta con más de treinta observaciones legítimas. En caso de que no haya laboratorios libres ni experimentos avanzados, no es posible iniciar un nuevo experimento.

Además, se quiere conocer en todo momento qué cantidad de los experimentos avanzados (dentro del conjunto de experimentos actualmente vigentes) son consistentes con la hipótesis. Un experimento es consistente con la hipótesis si la mayoría de las observaciones son legítimas.

Se pide:

1. Implemente el *ENUM* que tiene los estados de las observaciones llamado *Observaciones*
2. Implemente la clase *Experimento* que permite *agregarObservacion* y responde un bool a la llamada *estaAvanzado.*

Ejercicio 2 - Documentación

Recibimos un módulo con documentación incompleta. Se pide completarla con los valores pedidos dentro de la definición de la clase. En ella se encuentran los indicadores <*completar con …*>. Reemplazar los indicadores con lo pedido.

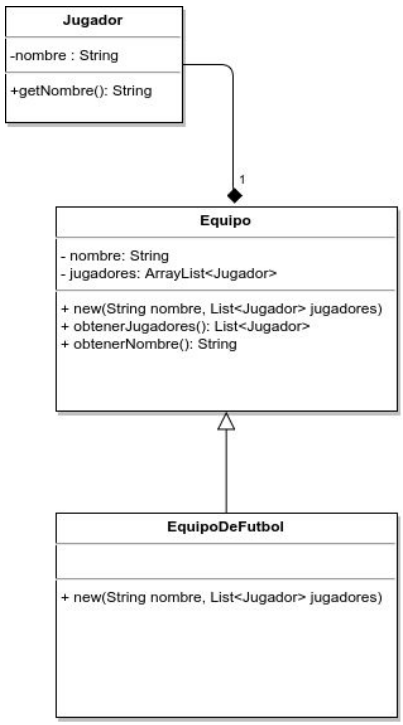
```
1.  public interface Connection extends Wrapper, AutoCloseable {
2.    /**
3.     * Crea un objeto <code>Statement</code> para el envío de sentencias
4.     * SQL a la base de datos.
5.     * Las sentencias SQL sin parámetros son normalmente ejecutadas usando
6.     * objetos <code>Statement</code>.
```

```
7.      *
8.      * Si ha de ejecutar varias veces el mismo Statement se sugiere utilizar
9.      * un PreparedStatement
10.     *
11.     * <completar_con_tag> un nuevo <code>Statement</code> por defecto.
12.     * <completar_con_tag> <completar_con_excepción> si la conexión está cerrada o
13.     * si hay un error en el acceso a la base de datos.
14.     */
15.    Statement createStatement() throws SQLException;
16.
17.    /**
18.     * Crea un objeto <code>PreparedStatement</code> para el envío de
19.     * sentencias parametrizadas a la base de datos.
20.     * Un Statement con o sin parámetros puede ser pre-compilado y almacenado
21.     * en un PreparedStatement para poder utilizarlo eficientemente y
22.     * ejecutarlo multiples veces.
23.     *
24.     * <completar_con_tag> <completar_con_parametro> un Statement SQL
25.     * que puede contener uno  o más símbolos '?'
26.     * para completar luego con los parámetros necesarios.
27.     *
28.     * @return un nuevo <completar_con_tipo> precompilado.
29.     *
30.     * @throws SQLException si la conexión está cerrada o
31.     * si hay un error en el acceso a la base de datos.
32.     * */
33.    PreparedStatement prepareStatement(String sql)
34.        throws SQLException;
35. }
```

¿Qué método debería utilizarse para una consulta que deberá ejecutarse múltiples veces con distintos parámetros?

Ejercicio 3 - Interpretar especificaciones formales

Se tienen las clases que modelan de manera simplificada un equipo de fútbol.



a) Seleccione de la siguiente lista las relaciones que mantienen entre sí

- Asociación.
- Agregación.
- Composición.
- Generalización.
- Realización.
- Dependencia

b) Implemente las clases definidas en el diagrama.

Ejercicio 4 - Depurar estructuras lógicas en código

El siguiente fragmento de código presenta errores de compilación.

```
1.  public class Padre {
2.      protected String nombre;
3.      public Padre(String nombre) {
4.          this.nombre = nombre;
5.      }
6.      public String obtenerNombre(){
7.          return this.nombre;
8.      }
9.  }
10. public class Hijo extends Padre {
11.     public void obtenerNombre(){
12.         System.out.println(this.obtenerNombre());
13.     }
14. }
```

Corrija el error y escriba lo que se imprime en pantalla al instanciar una clase de tipo Hijo.

Ejercicio 5 - Bases de datos

Se tiene una base de datos para administrar el catálogo de libros de la Biblioteca Nacional.

Libro	Autor
@id_libro #autor (FK) fecha_publicacion: DATE título: VARCHAR(20) descripción: VARCHAR(20)	@id_autor nombre: VARCHAR(20) libros_publicados: INTEGER fecha_nacimiento: DATE gano_nobel: BOOLEAN

Se pide:

1. Crear las tablas Libro y Autor.
2. Implementar el índice que acelera las búsquedas por *nombre* de Autor.
3. implementar el índice que acelera las búsquedas por *fecha de publicación* del Libro.