EXAMEN 20_21

Dado el siguiente código:

```
public class Examen {
    private ArrayList<Pregunta> preguntas;
    private ArrayList<Respuesta> respuestas;
    private float nota;

    static void compruebaRespuestas(float rigurosidad,ArrayList<Pregunta> p) {
        if (p.size()==0) {
            //...
        }
        else {
            for(Pregunta pregunta:p) {
                //....
            }
        }
    }
    static void compruebaPreguntas(float rigurosidad,ArrayList<Respuesta> r) {
        if (r.size()==0) {
            //...
        }
        else {
            for(Respuesta respuesta:r) {
                //....
            }
        }
    }
```

```
    static void compruebaPreguntas(float rigurosidad,ArrayList<Respuesta> r) {
        if (r.size()==0) {
            //...
        }
        else {
            for(Respuesta respuesta:r) {
                //....
            }
        }
    }

    public Examen(ArrayList<Pregunta> p,ArrayList<Respuesta> r,float rigu) {
        preguntas=p;
        respuestas=r;
        //Siempre (sin excepción) hay que realizar estas comprobaciones
        //Aunque la colección de preguntas/respuestas esté vacía
        Examen.compruebaRespuestas(rigu,preguntas);
        Examen.compruebaPreguntas(rigu,respuestas);
    }
```

Añade el código necesario para poder construir exámenes a partir de otro examen. Si es imprescindible, añade los consultores o modificadores necesarios. En el proceso de copia se aplica siempre un nivel de rigurosidad fijo de 2.88

Explica, utilizando como referencia solo el código proporcionado, si es posible alterar el estado interno de un examen desde el exterior de la propia clase

Dado el siguiente código:

```
public class Examen {
    private ArrayList<Pregunta> preguntas;
    private ArrayList<Respuesta> respuestas;
    private float nota;

    static void compruebaRespuestas(float rigurosidad,ArrayList<Pregunta> p) {...10 lines }
    static void compruebaPreguntas(float rigurosidad,ArrayList<Respuesta> r) {...10 lines }

    public Examen(ArrayList<Pregunta> p,ArrayList<Respuesta> r,float rigu) {
        preguntas=p;
        respuestas=r;
        //Siempre (sin excepción) hay que realizar estas comprobaciones
        //Aunque la colección de preguntas/respuestas esté vacía
        Examen.compruebaRespuestas(rigu,preguntas);
        Examen.compruebaPreguntas(rigu,respuestas);
    }

    public void corregirExamen(ArrayList<Respuesta> r) {
        //...
    }
    public void evaluarDificultad() {
        //...
    }
```

Crear una nueva clase para un nuevo tipo de exámenes (ExamenLoco) con la misma funcionalidad que un Examen como el proporcionado, salvo por dos hechos:


```
    private float nota;

    static void compruebaRespuestas(float rigurosidad,ArrayList<Pregunta> p) {...10 lines }
    static void compruebaPreguntas(float rigurosidad,ArrayList<Respuesta> r) {...10 lines }

    public Examen(ArrayList<Pregunta> p,ArrayList<Respuesta> r,float rigu) {
        preguntas=p;
        respuestas=r;
        //Siempre (sin excepción) hay que realizar estas comprobaciones
        //Aunque la colección de preguntas/respuestas esté vacía
        Examen.compruebaRespuestas(rigu,preguntas);
        Examen.compruebaPreguntas(rigu,respuestas);
    }

    public void corregirExamen(ArrayList<Respuesta> r) {
        //...
    }
    public void evaluarDificultad() {
        //...
    }
```


Crear una nueva clase para un nuevo tipo de exámenes (ExamenLoco) con la misma funcionalidad que un Examen como el proporcionado, salvo por dos hechos:

- Durante la creación del mismo, las preguntas son desordenadas. Puedes asumir que dispones de un método de instancia (void desordena()) que proporciona esta funcionalidad. En la creación de un examen de este tipo se aplica una rigurosidad representada con el valor -1
- En la mitad de las ocasiones, el proceso de corrección no produce absolutamente ningún efecto (no se realiza ninguna operación)