Pauta Certamen 1, Programación II

Prof. Rodrigo Olivares

Abril 20, 2017

Instrucciones:

- El puntaje máximo del certamen es 100%, siendo el 60% el mínimo requerido para aprobar.
- Responda cada pregunta en la hoja indicada, agregando su nombre. Si no responde alguna pregunta, debe entregar la hoja con su nombre e indicar que **no responde**.
- El certamen es **individual**. Cualquier intento de copia, será sancionado con nota **1,0**.

1. *30pts.* De las siguentes afirmaciones, encierre en un círculo la o las alternativas correctas.

i. El paradigma de la orientación a objeto se basa en:
(a) El uso de objetos y su intercacción.
(b) El uso de clases como categorización de instancias.
(c) El uso de un enfoque imperativo.
(d) El uso de un enfoque reusable.
(e) Principios como herencia y abstracción.

ii. En cuanto a la programación orientada a objeto:
(a) Se apoya en el paradigma estructural.
(b) Se apoya en el paradigma orientación a objetos.
(c) Divide el programa en pequeñas unidades de código.
(d) Proporciona herramientas para modelar el mundo real.
(e) Es una herramienta del lenguaje JAVA.

iii. Una clase es:
(a) Un arreglo de objetos.
(b) Una absorción del mundo real.
(c) Una herramienta de programación.
(d) Un punto a memoria.
(e) La instancia de un objeto.

iv. Respecto a una clase:
(a) Se declaran utilizando la palabra resevada *classe*.
(b) Puede no tener el mismo nombre que el archivo.
(c) Puede no tener el mismo nombre que el constructor.
(d) Puede o no tener atributos finales.
(e) Puede o no tener métodos estáticos.

v. Respecto a una clase:
(a) Puede o no tener atributos.
(b) Puede o no tener métodos.
(c) Puede o no tener constructor.
(d) Puede o no tener un nombre.
(e) Puede o no tener un tipo.

vi. Un objeto es:
(a) Un tipo de dato de la clase.
(b) La instancia de una clase.
(c) Una abstracción del mundo real.
(d) Un sub-conjunto de atributos y métodos de la clase.
(e) Siempre estático.

vii. El principio de ocultamiento:
(a) Es una técnica que protege el estado de una entidad.
(b) Es indispensable en el paradigma de orientación a objeto.
(c) En Java, se logra utilizando los modificadores de acceso.
(d) Es encapsular el conocmineto de una entidad.
(e) Ninguna de las anteriores.

viii. Respecto a la herencia, las clases:
(a) Heredan sólo los métodos privados.
(b) Heredan el compotamiento completo de la clase padre.
(c) Heredan sólo el comportamiento que se desea utilizar.
(d) En Java, se implementan con la palabra implements.
(e) En Java, se implementan con la palabra extends.

ix. El polimorfismo:
(a) El mismo nombre implementa distintas funcionalidades.
(b) Una funcionalidad implementada con distintos nombres.
(c) Es una característica de JAVA.
(d) Es una característica de POO.
(e) Un ejemplo es el símbolo %.

x. El metódo *main*:
(a) Puede no ser void.
(b) Debe ser static.
(c) Puede no llevar argumentos de entrada.
(d) Debe retornar un valor.
(e) Debe incluirse en un programa.

2. *70pts.* Como la asignatura Programación 2 tiene demasiados alumnos, el profesor dividió el curso en 2 grupos (A y B) equitativos, para tomar evaluaciones (cantidades semejantes en ambos grupos). En la asignatura se realizarán 3 evaluaciones. En la primera evaluación ingresan primero los del grupo A y luego del B. En la segunda evaluación es a la inversa. Para la tercera evaluación, el profesor decidió que para ser justo, los integrantes de los grupos serán reodenados de forma aleatoria, por lo cual le ha solicitado a Ud que desarrolle un programa que, seleccione randómicamente y sin repetición, los alumnos que formaran parte del grupo A y B y luego muéstre los dos grupos en la salida estándar.

   Considere:

   - La lista total de los alumnos es un valor aleatorio mayor a 40 y menor a 100.
   - La lista contiene alumnos (agregue la propiedad "identificador" de tipo entero que almacenará un valor aleatorio, no repetible en entre los alumnos de la lista).
   - Desarrolle el programa bajo el paradigma de orientacin a objeto.

¿Cómo seré evaluado en la pregunta 2?			
Tópico	**Logrado**	**Medianamente logrado**	**No logrado**
Construir entidades	*15pts* Crea la clase atómica Alumno con sus atributos/métodos. Atributo: identificación del alumno. Métodos: para fijar y obtener el atributo id, para mostrar su información, para comparar con otra instancia Alumno.	*8pts* Crea la clase atómica Alumno sin el atributo o sin todos los métodos necesarios para el problema..	*0pts* No crea la clase atómica Alumno.
Construir clase Lista y sus métodos	*25pts* Define e implementa correctamente la clase Lista, sus atributos y métodos: Atributos: listas, tamaño, límites, etc. Métodos: llenar, generar identificador sin repetir, mostrar, etc.	*10pts* Define algunos atributos o algunos métodos, pero no todos los necesarios para el problema.	*0pts* No define ni los atributos ni métodos.
Construir clase principal	*10pts* Define la clase con el método principal.	*5pts* Define el método principal en la misma clase.	*0pts* No define el método principal.
Paradigma Orientación a Objetos	*20pts* Resuelve el problema utilizando el POO.	*7pts* Utiliza parte del POO para resolver el problema.	*0pts* No utiliza el POO para dar solución al problema.
Total máximo puntaje pregunta 2	*70pts*	*30pts*	*0pts*

```java
/* Clase Principal - OrdenarListaAlumno */

public class OrdenListaAlumno {

    public static void main(String[] args) {
        ListaAlumno la = new ListaAlumno();

        System.out.println("\n Llenando Lista \n");
        la.llenarLista();

        System.out.println("\n Mostrando Lista \n");
        la.mostrarLista();

        System.out.println("\n Dividiendo Lista \n");
        la.dividirListas();

        System.out.println("\n Mostrando Grupos \n");
        la.mostrarGrupos();
    }
}

/* Clase atomica - Alumno */

public class Alumno {

    private int id;

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    @Override
    public String toString() {
        return String.format("AlumnoID: %d", id);
    }

    @Override
    public boolean equals(Object o) {
        if(o instanceof Alumno) {
            Alumno alumno = (Alumno) o;
            if (alumno.getId() == this.id) {
                return true;
            }
        }
        return false;
    }
}

/* Clase Lista - ListaAlumno */

import java.util.ArrayList;
import java.util.Random;

public class ListaAlumno {

    private ArrayList<Alumno> listaAlumnos = new ArrayList<>();
    private ArrayList<Alumno> listaAlumnosGrupoA = new ArrayList<>();
    private ArrayList<Alumno> listaAlumnosGrupoB = new ArrayList<>();
    private Random rnd;
    private int tamanio;
    private final int MIN_LISTA = 40;
    private final int MAX_LISTA = 100;
    private final int MAX_ID = 100;
```

```
    public ListaAlumno() {
        rnd = new Random();
        tamanio = rnd.nextInt(MAX_LISTA - MIN_LISTA) + MIN_LISTA + 1;
    }

    public void llenarLista() {
        Alumno a;
        for (int i = 0; i < tamanio; i++) {
            listaAlumnos.add(doAlumnoSinRepeticion());
        }
    }

    public void dividirListas() {
        Alumno a;
        for (int i = 0; i < tamanio; i++) {
            a = listaAlumnos.remove(rnd.nextInt(listaAlumnos.size()));
            if (rnd.nextBoolean()) {
                listaAlumnosGrupoA.add(a);
            } else {
                listaAlumnosGrupoB.add(a);
            }
        }
    }

    private Alumno doAlumnoSinRepeticion() {
        Alumno a = new Alumno();
        int id;
        do {
            id = rnd.nextInt(MAX_ID) + 1;
            a.setId(id);
            for(int i = 0; i < listaAlumnos.size(); i++) {
                if (listaAlumnos.get(i).equals(a)) {
                    id = -1;
                    break;
                }
            }
        } while (id < 0);
        return a;
    }

    public void mostrarLista() {
        System.out.println("Curso:");
        for(int i = 0; i < listaAlumnos.size(); i++) {
            System.out.println((i + 1) + " " + listaAlumnos.get(i));
        }
    }

    public void mostrarGrupos() {
        System.out.println("Grupo A:");
        for(int i = 0; i < listaAlumnosGrupoA.size(); i++) {
            System.out.println((i + 1) + " " + listaAlumnosGrupoA.get(i));
        }
        System.out.println("Grupo B:");
        for(int i = 0; i < listaAlumnosGrupoB.size(); i++) {
            System.out.println((i + 1) + " " + listaAlumnosGrupoB.get(i));
        }
    }
}
```