Pauta Certamen 1, Programación II

Prof. Rodrigo Olivares

Noviembre 8, 2016

---

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

i. Algunos enfoques de la orientación a objeto son:

(a) El enfoque reusable.

(b) El enfoque abstracto.

(c) El enfoque imperativo.

(d) El enfoque procedural.

(e) Ninguna de las anteriores.

ii. En cuanto a la programación orientada a objeto (POO):

(a) Se apoya en el paradigma estructural.

(b) Se base en la interacción de funciones.

(c) Proporciona herramientas para modelar el mundo real.

(d) Es una propiedad de los lenguajes orientados a objetos.

(e) Ninguna de las anteriores.

iii. Una clase es:

(a) Una colección de objetos.

(b) Una herencia del mundo real.

(c) Una categorización de objetos.

(d) Un puntero a memoria.

(e) Ninguna de las anteriores.

iv. Respecto a una clase:

(a) Se declaran utilizando la palabra resevada class.

(b) Debe tener el mismo nombre que el archivo.

(c) Se declaran static los atributos de miembros del objeto.

(d) Todas deben incluir el método main.

(e) Ninguna de las anteriores.

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) Una categorización de la clase.

(b) La instancia de una clase.

(c) Una abstracción del mundo real.

(d) Un método de interación entre clases.

(e) Siempre estático.

vii. El principio de abstracción:

(a) Es una técnica que protege el estado de una entidad.

(b) Es parte del paradigma de orientación a objeto.

(c) En Java, se logra utilizando los modificadores de acceso.

(d) Es absorber el conocimiento de una entidad.

(e) Ninguna de las anteriores.

viii. Respecto los constructores:

(a) Son declarados private

(b) Deben ser métodos de tipo void.

(c) Deben ser normbrados igual que las clases.

(d) El compilador siempre crea el constructor vacío.

(e) Se utiliza para instanciar objetos.

ix. El polimorfismo:

(a) Una funcionalidad implementada con distintos nombres.

(b) El mismo nombre implementa distintas funcionalidades.

(c) Es una característica de Java.

(d) Es una característica de POO.

(e) Un ejemplo es el símbolo $\rightarrow$.

x. El metódo *main*:

(a) Debe ser void.

(b) Debe ser static.

(c) Debe incluir argumentos de entrada.

(d) Debe retornar un valor.

(e) Debe incluirse en un programa.

```java
/********** Jugador.java **********/

import java.util.Random;

public class Jugador {
    private int id;
    private int br;
    private int bg;
    private int gc;
    private char tipo;

    private Random r = new Random();

    public Jugador(int id, char tipo) {
        this.id = id;
        this.tipo = tipo;
        crearHabilidades();
    }

    private void crearHabilidades() {
        br = r.nextInt(100) + 1;
        bg = r.nextInt(100) + 1;
        gc = r.nextInt(100) + 1;
    }

    public void actualizarHabilidades() {
        crearHabilidades();
    }

    public int nivelJuegoIndividual(){
        if (tipo == 'P')
            gc = 0;
        return (int) Math.round(br * 0.2 + bg * 0.35 + gc * .45);
    }
}

/********** Equipo.java **********/

public class Equipo {

    private int id;
    private Jugador[] equipo;
    private final int NUM_JUGADORES = 11;

    public Equipo(int id) {
        this.id = id;
        crearEquipo();
    }

    private void crearEquipo() {
        equipo = new Jugador[NUM_JUGADORES];
        char tipo = ' ';
        for (int i = 0; i < NUM_JUGADORES; i++) {
            if (i == 0) {
                tipo = 'P';
            } else if (i < 4) {
                tipo = 'D';
            } else if (i < 6) {
                tipo = 'L';
            } else if (i < 9) {
                tipo = 'C';
            } else if (i < 11) {
                tipo = 'G';
            }
            equipo[i] = new Jugador(i + 1, tipo);
        }
    }

    public int nivelJuegoColectivo() {
```

```
        int njc = 0;
        for (int i = 0; i < NUM_JUGADORES; i++) {
            njc += equipo[i].nivelJuegoIndividual();
        }
        return njc;
    }

    public void actualizarHabilidades() {
        for (int i = 0; i < NUM_JUGADORES; i++) {
            equipo[i].actualizarHabilidades();
        }
    }

    @Override
    public String toString() {
        return "Equipo " + id + " (nivel de juego " + nivelJuegoColectivo() + ")";
    }
}
```

/********** **Liga.java** **********/

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

public class Liga {

    private Equipo[] liga;
    private ArrayList<Equipo> equiposParticipantes = new ArrayList<>();
    private ArrayList<Equipo> equiposGanadores = new ArrayList<>();
    private final int NUM_EQUIPOS = 16;


    public Liga() {
        crearLiga();
    }

    private void crearLiga() {
        liga = new Equipo[NUM_EQUIPOS];
        for (int i = 0; i < NUM_EQUIPOS; i++) {
            liga[i] = new Equipo(i + 1);
            equiposParticipantes.add(liga[i]);
        }
    }

    public void jugar() {
        Equipo equipoLocal, equipoVisita;
        Random random = new Random();
        int fase = 4;

        do {
            System.out.println(getFase(fase));
            do {
                equipoLocal = equiposParticipantes.remove(random.nextInt(
    equiposParticipantes.size()));
                equipoVisita = equiposParticipantes.remove(random.nextInt(
    equiposParticipantes.size()));

                System.out.print(equipoLocal + " v/s " + equipoVisita + " ");

                if (equipoLocal.nivelJuegoColectivo() > equipoVisita.
    nivelJuegoColectivo()) {
                    equiposGanadores.add(equipoLocal);
                    System.out.println("Ganador: " + equipoLocal);
                } else {
                    if (equipoVisita.nivelJuegoColectivo() > equipoLocal.
    nivelJuegoColectivo()) {
                        equiposGanadores.add(equipoVisita);
                        System.out.println("Ganador: " + equipoVisita);
                    } else {
```

```
                        System.out.println("Empate. Ganador por sorteo: ");
                        if(random.nextBoolean()) {
                             equiposGanadores.add(equipoLocal);
                             System.out.print(equipoLocal);
                        } else {
                            equiposGanadores.add(equipoVisita);
                            System.out.print(equipoVisita);
                        }
                    }
                }
            } while(equiposParticipantes.size() > 0);

            for (int i = 0; i < equiposGanadores.size(); i++) {
                equiposGanadores.get(i).actualizarHabilidades();
                equiposParticipantes.add(equiposGanadores.get(i));
            }
            equiposGanadores.clear();
            fase--;
            System.out.println();
        } while(fase > 0);
    }

    private String getFase(int faseId) {
        String nombreFase = null;
        switch(faseId) {
            case 1: nombreFase = "Final"; break;
            case 2: nombreFase = "Semi-Final"; break;
            case 3: nombreFase = "Cuartos"; break;
            case 4: nombreFase = "Octavos"; break;
        }
        return nombreFase;
    }
}

/********************** LigaImp.java **********************/

public class LigaImp {

    public static void main(String[] args) {
        Liga l = new Liga();
        l.jugar();
    }
}
```