Primer Parcial de IIP (ETSInf)
9 de Noviembre de 2022. Duración: 1 hora y 30 minutos

Nota: El examen se evalúa sobre 10 puntos, pero su peso específico en la nota final de IIP es de **3,75 puntos**

1. 5.5 puntos Se quiere diseñar una clase Tipo de Datos denominada `Student` para representar la información de un estudiante de IIP. Cada estudiante tiene asociados los siguientes datos: nombre (desglosado en nombre propio, primer apellido y segundo apellido), documento de identidad (dni o equivalente), nota del primer parcial y nota del segundo parcial.

   Se pide: implementar la clase `Student` con los atributos y métodos que se indican a continuación.

   a) (0.25 puntos) Dos atributos públicos, estáticos y constantes de tipo `double` con el peso de la nota del primer y del segundo parcial. Sus identificadores y valores son, respectivamente:

      - `WEIGHT_1`, con valor `0.40`;
      - `WEIGHT_2`, con valor `0.60`.

      Dichas constantes deberán ser utilizadas en el código cuando se requiera hacer uso de estos pesos.

   b) (0.5 puntos) Seis atributos de instancia y privados, para representar los elementos asociados a un `Student`. Los identificadores y tipos de estos atributos son:

      - `name` para el nombre propio, `surname1` para el primer apellido, y `surname2` para el segundo apellido, todos de tipo `String`;
      - `dni` para el dni o identificador equivalente, de tipo `String`;
      - `grade1` para la nota del primer parcial y `grade2` para la nota del segundo parcial, ambos de tipo `double`.

   c) (0.75 puntos) Un constructor que reciba como parámetros el nombre propio, los dos apellidos y el dni del estudiante, con los que inicialice los correspondientes atributos de la instancia creada, y que deje las notas con los valores por defecto. Se supone como precondición que los valores de los parámetros son correctos.

   d) (0.25 puntos) Métodos modificadores, con el perfil habitual, de los atributos `grade1` y `grade2`. Se supone como precondición que el valor del parámetro de cada modificador es correcto.

   e) (0.75 puntos) Método de nombre `finalGrade`, que devuelva la nota final del estudiante `this`, calculada según la siguiente fórmula

   $$máximo(grade_2, grade_1 \times WEIGHT_1 + grade_2 \times WEIGHT_2)$$

   Para resolver este cálculo puede ser útil el siguiente método de la clase `Math` de Java:

   ```
   static double     max(double a, double b)
                     Returns the greater of two double values.
   ```

   f) (1 punto) Método de nombre `user` que devuelva el usuario de `this`, un `String` formado por las dos primeras letras del nombre, seguido por las dos primeras letras del primer apellido, y seguido por las dos primeras letras del segundo apellido, todo ello en minúsculas. El usuario deberá terminar con el sufijo `"@etsinf.upv.es"`.

   Ejemplo: Para la estudiante Ana Isabel (nombre propio) Pi (primer apellido) del Valle (segundo apellido), el método devuelve `anpide@etsinf.upv.es`.

   Para resolver este cálculo pueden ser útiles los siguientes métodos de la clase `String` de Java:

   ```
   String     substring(int beginIndex, int endIndex)
              Returns a string that is a substring of this string. The substring begins at the
              specified beginIndex and extends to the character at index endIndex - 1.

   String     toLowercase()
              Converts all of the characters in this String to lower case.
   ```

   g) (1 punto) Método `equals`, que sobrescribe el de `Object`, de modo que devuelva `true` sii el objeto parámetro del método es un `Student` con el mismo `dni` que el de `this`.

   h) (1 punto) Método `toString`, que sobrescribe el de `Object` y que devuelva el nombre propio seguido del primer apellido del estudiante, seguido del dni.

   Ejemplos:

   ```
   Ana Isabel Pi. DNI: 12345678Z
   Thomas Pyne. DNI: 87654321X
   ```

Solución:

```
public class Student {

    public static final double WEIGHT_1 = 0.4;
    public static final double WEIGHT_2 = 0.6;

    private String name, surname1, surname2, dni;
    private double grade1, grade2;

    public Student(String n, String s1, String s2, String d) {
        this.name = n;
        this.surname1 = s1;
        this.surname2 = s2;
        this.dni = d;
    }

    public void setGrade1(double g1) {
        this.grade1 = g1;
    }

    public void setGrade2(double g2) {
        this.grade2 = g2;
    }

    public double finalGrade() {
        double wGrade = this.grade1 * WEIGHT_1 + this.grade2 * WEIGHT_2;
        return Math.max(this.grade2, wGrade);
    }

    public String user() {
        String aux = this.name.substring(0, 2) + this.surname1.substring(0,2)
            + this.surname2.substring(0, 2);
        return aux.toLowerCase() + "@etsinf.upv.es";
    }

    public boolean equals(Object o) {
        return o instanceof Student
            && this.dni.equals(((Student) o).dni);
    }

    public String toString() {
        String result = this.name + " " + this.surname1 + ". DNI: " + this.dni;
        return result;
    }

}
```

2. 2.5 puntos **Se pide**: implementar la clase Programa `StudentGrades`, suponiendo que se ubica en el mismo paquete que la clase `Student` de la pregunta anterior, con un método `main` que realice las siguientes acciones.

   a) (0.25 puntos) Crear dos objetos de la clase `Student`: uno referenciado por una variable `student1` para representar a la estudiante de nombre *Ana Isabel*, primer apellido *Pi*, segundo apellido *del Valle* y dni *12345678Z*, y otro referenciado por `student2` para representar al estudiante de nombre *Thomas*, primer apellido *Pyne*, segundo apellido *Roure* y dni *87654321X*.

   b) (0.5 puntos) Mostrar por pantalla la representación como `String` del `student1` seguida de su usuario.

   c) (0.5 puntos) Mostrar un mensaje por pantalla para solicitar la nota del primer parcial de `student1` y leerla por teclado. Ídem para la nota del segundo parcial.

   d) (0.25 puntos) Actualizar las notas de `student1` con las introducidas anteriormente.

   e) (0.25 puntos) Actualizar las notas de `student2` con los valores 8.5 y 9.75 para el primer y segundo parcial, respectivamente.

   f) (0.75 puntos) Calcular la nota final de ambos estudiantes y comprobar si la nota de la primera estudiante es mayor que la del segundo. Escribir en el terminal la frase `Is the first student's final grade the highest?`, seguido de la respuesta `true` o `false`.

Solución:

```
import java.util.Scanner;
import java.util.Locale;

public class StudentGrades {
    private StudentGrades() { }

    public static void main(String[] args) {
        Scanner kb = new Scanner(System.in).useLocale(Locale.US);
        Student student1 = new Student("Ana Isabel", "Pi", "del Valle", "12345678Z");
        Student student2 = new Student("Thomas", "Pyne", "Roure", "87654321X");

        System.out.println(student1 + " " + student1.user());

        System.out.print(" First midterm grade: ");
        double gr1 = kb.nextDouble();
        System.out.print(" Second midterm grade: ");
        double gr2 = kb.nextDouble();

        student1.setGrade1(gr1);
        student1.setGrade2(gr2);

        student2.setGrade1(8.5);
        student2.setGrade2(9.75);

        System.out.println("Is the first student's final grade the highest? "
            + (student1.finalGrade() > student2.finalGrade()));
    }
}
```

3. [2 puntos] Se dispone de la clase `Point` que, como la clase `Punto` vista en el tema 4, define un punto en un espacio bidimensional real (con dos atributos representando su abscisa y su ordenada), con la funcionalidad que se muestra en parte, a continuación, en su documentación:

Constructor Summary

Constructors

Constructor	Description
`Point(double px, double py)`	Crea un `Point` con abscisa `px` y ordenada `py`.

Method Summary

All Methods | **Instance Methods** | **Concrete Methods**

Modifier and Type	Method	Description
`double`	`getX()`	Devuelve la abscisa del `Point this`.
`double`	`getY()`	Devuelve la ordenada del `Point this`.
`void`	`setX(double px)`	Actualiza la abscisa del `Point this` a `px`.
`void`	`setY(double py)`	Actualiza la ordenada del `Point this` a `py`.
`java.lang.String`	`toString()`	Devuelve un `String` que representa el `Point this` en el formato tipico matemático, i.e., (abscisa,ordenada).

Se pide: indicar qué se muestra por pantalla tras la ejecución del siguiente código.

```
public class Exercise3 {
    private Exercise3() { }

    public static void main(String [] args) {
        Point a = new Point( 0.0, 10.0);
        Point b = new Point( 5.0,  0.0);
        Point c = new Point(-5.0,  0.0);

        System.out.println("Triangle with points: " + a.toString()
            + " " + b.toString() + " " + c.toString());

        swap(a, b);
        swap(b, c);
        swap(c, a);

        System.out.println("Triangle with points: " + a.toString()
            + " " + b.toString() + " " + c.toString());
    }

    public static void swap(Point p, Point q) {
        double temp = p.getX(); p.setX(q.getX()); q.setX(temp);
        temp = p.getY(); p.setY(q.getY()); q.setY(temp);
    }
}
```

Solución:

```
Triangle with points: (0.00, 10.00) (5.00, 0.00) (-5.00, 0.00)
Triangle with points: (0.00, 10.00) (-5.00, 0.00) (5.00, 0.00)
```