Algoritmos y Programación III (75.07) – Integrador – 5/7/2013

Nombre:	Padrón: Turno (Mié/JT/JN):
Corrigió:	Nota:

1) **Ejercicio conceptual.** Una universidad pública argentina define el siguiente circuito para la aprobación de planes de estudio:

   Paso 1: El Consejo Directivo (CD) de la Facultad pide a la Comisión Curricular (CC) de la carrera respectiva (hay una CC por cada carrera de grado, con representantes de docentes, alumnos y graduados) que elabore un plan, dándole un plazo. Dentro de ese plazo, la CC elabora el plan y lo pasa al CD.

   Paso 2: Si al cabo del plazo fijado por el CD, la CC no hubiese elaborado un plan, o no hubiese llegado a un acuerdo sobre el mismo, debe pasarle todos los antecedentes al CD. El CD pedirá a la Comisión de Enseñanza (CE) de la Facultad, la adaptación de otro plan preexistente en un plazo menor.

   Paso 3: El plan recibido por el CD, proveniente de la CC o de la CE, pasará a consideración del CD, que podrá incorporarle cambios. De haberlos, devolverá el plan a la CC para que opine en un plazo breve.

   Paso 4: Una vez que la CC haya opinado, o se le haya vencido el plazo para hacerlo, el CD volverá a analizar el plan, pudiendo aceptar las propuestas de la CC o mantener el plan como estaba.

   Paso 5: El plan aprobado por el CD se girará al Consejo Superior (CS) de la Universidad para su aprobación o rechazo. El CS procederá a aprobarlo o devolverlo al CD para que le introduzca cambios. De volver al CD, se repite todo desde el paso 3.

   Se pide:

   a. Haga un diagrama de estados de UML, con todos los eventos que provocan transiciones de estados. Use su criterio para elegir los estados relevantes y los nombres de los mismos.

   b. Modelar en UML (diagrama de secuencia, con objetos y mensajes) la determinación del estado de los planes de estudio de todas las carreras de la universidad. Mantenga el nivel de abstracción adecuado, sin indicar cuestiones de implementación de métodos de clases ajenas a este escenario.

   c. Escriba 3 de las pruebas automatizadas necesarias para probar el comportamiento modelado en el diagrama de secuencia, usando SUnit, e incluyendo al menos una prueba positiva y una negativa.

   d. Escriba el código Smalltalk que haga funcionar las pruebas del punto d, pero sin escribir nada de los métodos de otras clases (dicho de otra manera, los métodos de otras clases debe suponerlos implementados).

2) ¿En qué consiste la práctica de integración continua? ¿Qué ventajas ofrece? (una carilla)

3) La **cohesión** es un atributo de calidad del diseño de módulos (paquetes, clases y métodos), a los cuales se les suele exigir una cohesión alta. Asimismo, definimos que un módulo tiene alta cohesión cuando sus responsabilidades están bien establecidas y focalizadas. A juicio de quien redactó este parcial, las interfaces de iteradores de Java y C# tienen algún problema de cohesión. ¿Puede usted identificar alguno? ¿Qué cambio les propondría a los responsables de la plataforma Java/.NET para mejorar la cohesión en el caso detectado? ¿Cómo cambiaría el uso de iteradores? ¿En qué sentido resulta mejor o más cómodo el enfoque provisto actualmente? (máximo 1 carilla, usando Java o C#, no necesariamente ambos casos)

En Java se utiliza la interfaz Iterator<T>, en la cual hay 2 métodos:

next() : T
avanza al siguiente elemento de la colección y devuelve una referencia al mismo

hasNext() : boolean
indica si hay más elementos en la colección

Ejemplo de uso:

```
Iterator<String> i = listaNombres.iterator( );
while ( i.hasNext( ) ) {
        String x = i.next ( );
        // trabajo con x
}
```

En C# se utiliza la interfaz IEnumerator<T>, en la cual hay un método y una propiedad:

MoveNext( ) : bool
avanza al siguiente elemento, devolviendo true si tuvo éxito o false si no hay más elementos

Current : T
propiedad que devuelve una referencia al elemento actual en la colección

Ejemplo de uso:

```
IEnumerator<String> i = listaNombres.GetEnumerator();
while ( i.MoveNext( ) ) {
        String x = i.Current;
        // trabajo con x
}
```

Algoritmos y Programación III (75.07) – Integrador -- 26/7/2013

Nombre:	Padrón:	Turno (Mié/JT/JN):
Corrigió:	Nota:	

1) **Ejercicio conceptual.** Para la implementación de un sistema de control de transporte de pasajeros, se establecen las siguientes definiciones:

   Itinerario: un viaje completo a ser realizado por una persona para llegar de un punto a otro del país; un itinerario puede estar compuesto por uno o más tramos.

   Tramo: una parte de un viaje entre dos puntos, que se realiza en un solo medio de transporte.

   Medios de transporte: pueden ser colectivo, tren urbano, subte, ómnibus interurbano, tren interurbano. Cada medio tiene un sistema tarifario diferente; algunos cobran un precio fijo, otros un valor dependiente de la distancia y otros tienen una tabla de tarifas entre destinos.

Se pide (tenga en cuenta que la calidad de los nombres que elija va a ser evaluada):

   a. Modelar en UML (diagrama de clases) el problema recién descripto. Tanto en las clases Tramo como Itinerario, deben tener en su comportamiento algún método que permita calcular costo y tiempo de viaje.

   b. Suponiendo que todas las clases del sistema existen, a excepción de la clase Itinerario, se pide escribir las pruebas automatizadas necesarias para los métodos que calculen el costo y tiempo neto de un itinerario, sabiendo que ambos se obtienen de la simple suma de los costos y tiempos de los tramos.

   c. Hacer el diagrama de secuencia (UML) del cálculo del costo de un itinerario.

   d. Escribir, de la clase Itinerario, los atributos y los métodos que permite calcular el costo y el tiempo.

2) Explique con un ejemplo la necesidad de contar con un mecanismo de exclusión o exclusión mutua. Sobre el mismo ejemplo, muestre cómo funcionarían los casos de sincronización de métodos y bloques, las llaves de lectura y escritura y las actualizaciones optimistas.

3) ¿Qué son las pruebas alfa y las pruebas beta? ¿Se pueden automatizar? (máximo media carilla)