Facultad de Matemática, Astronomía y Física
Universidad Nacional de Córdoba
Asignatura: Ingeniería del Software I
Segundo Cuatrimestre de 2008

Examen Parcial Nro. 2

Parte teórica

Ej. 1. (a) ¿Cuáles son los factores que afectan el acoplamiento en un diseño funcional y de qué manera hacen que éste se incremente?
(b) ¿Qué tipos de acoplamiento aparecen en el diseño orientado a objetos y cuál de ellos es similar al mencionado en el inciso (a)?

Ej. 2. Describa muy brevemente los 5 pasos que involucra la metodología OMT (object modeling technique) de diseño orientada a objetos

Ej. 3. ¿Qué describe el invariante de representación (o invariante de clase)? ¿Cómo se relacionan el invariante de representación del tipo concreto y el invariante de la clase más abstracta?

Ej. 4. Describa el proceso de codificación dirigido por test. Compárelo con el proceso de codificación incremental básico.

Ej. 5. (a) Describa el método de derivación de test por análisis de valores límites y compárelo con el de particionado por clases equivalencia.
(b) Enumere los distintos niveles de testing y diga para qué se usan cada uno de ellos.

Parte práctica

Ej. 6. Considere el siguiente problema (descripción de un sistema a construir) y produzca un diagrama de clases correspondiente al diseño orientado a objetos del mismo:

> Se desea desarrollar un sistema de cajeros automáticos para un banco. El sistema contará de un gran número de puntos de extracción (i.e., cajeros automáticos), que necesitan acceder a un recurso central que contiene un registro detallado de la información de los clientes del banco. Los clientes utilizan los cajeros automáticos insertando una tarjeta e ingresando su PIN, el cual es codificado y comparado con el correspondiente al registro del usuario.
>
> Luego de identificarse exitosamente, el cliente puede:
>
> - consultar el balance de su o sus cuentas
> - pedir un resumen de cuenta, a ser enviado por correo al usuario
>
> La información sobre las cuentas es almacenada en una base central y puede no estar disponible desde los cajeros. En este caso, los clientes no podrán acceder al balance de sus cuentas. Cuando la base de datos está disponible, los clientes pueden retirar dinero de sus cuentas, tanto como el disponible en las cuentas correspondientes. Además, los clientes pueden tener cuentas especiales, que les permiten retirar dinero en descubierto (retirar más dinero que el disponible en sus cuentas), hasta cierto valor máximo que depende de cada cuenta.
>
> A los usuarios que no logren identificarse correctamente luego de tres intentos, se les retiene la tarjeta, se les bloquea la cuenta, y se informa al banco sobre la retención.

Ej. 7. Considere el siguiente problema y produzca un DFD correspondiente al diseño estructurado para la funcionalidad asociada a la compra de entradas:

> Se desea implementar un sistema simple de venta de entradas de cine a través de una máquina. La sala en la cual se instalará la máquina posee varias salas, y ofrece varias películas diferentes por día, con horarios diferentes entre los fines de semana y los días de semana. Además, las entradas tienen precios diferentes para niños, estudiantes, y público en general. La máquina sólo permite que los usuarios compren entradas para funciones del mismo día.
>
> Cuando un cliente quiere utilizar la máquina para comprar una entrada debe seleccionar la película que quiere ver y en qué horario; en caso de haber asientos disponibles, la máquina preguntará cuántos asientos desea, y permitirá al usuario seleccionar el tipo de entrada (para estudiantes, niños, etc). En caso de existir el número de asientos requerido, la máquina mostrará en la pantalla el monto a pagar, y aceptará el dinero del cliente. La máquina además da vuelto. Si no existen asientos disponibles, la máquina da la posibilidad al cliente de corregir el número de asientos requerido, o cambiar de horario.

Ej. 8. Considere el siguiente fragmento de código C:

```
register int c;                 /* input char */
register int nwhite = 0;        /* whitespace count */
register int nother = 0;        /* other count */
register int i;                 /* counter in a for loop */
int ndigit[10];                 /* digit counts */

/*
 * initialize the ndigit array
 */
for(i = 0; i < 10; i++)
        ndigit[i] = 0;

/*
 * handle input a char at a time
 */
while((c = getchar()) != EOF){
        /* see what it is */
        if (c >= '0' && c <= '9'){
                /* it's a digit -- bump the right count */
                ndigit[c - '0']++;
        }
        else if (c == ' ' || c == '\t' || c == '\n'){
                /* it's whitespace */
                nwhite++;
        }
        else{
                /* it's neither a digit nor whitespace */
                nother++;
        }
}
```

(a) Construya conjuntos de casos de test para este código que cumplan con los criterios de cobertura de ramificaciones.
(b) Construya conjuntos de casos de test para este código que cumplan con los criterios de coberturas de todas las definiciones.

En ambos casos considere como entrada la cadena provista por el usuario para analizar.