Examen de Programación I
(Ingeniería Informática)

Septiembre 2007

Primera parte (4 puntos, 40% nota del examen)

1) Indicar razonadamente por qué el siguiente algoritmo es incorrecto, y proponer un cambio al diseño que lo haga correcto:

```
método sumaImpares (entero[0..n-1] val) retorna entero
   {Pre: n>=0, val contiene n números enteros}
      var
        suma:=0; i:=0;
      fvar
      {Invariante Inv:
         suma= suma de los valores val[j] impares, para 0<=j<i}
      mientras i<n hacer
         {se cumplen Inv y 0<=i<n}
         suma:= suma+val[i];
         i++;
      fpara
      {se cumplen Inv e i=n}
      retorna suma;
   {Post: valor retornado=suma de los valores val[j] impares,
          para 0<=j<n}
fmétodo
```

2) Convertir la siguiente clase descrita mediante pseudocódigo a una clase en Java

```
clase Motor
  atributos
   real potenciaActual;
   real pMax;
   real pMin;
  fatributos
  método cambiaPotencia(real nueva)
   {Pre:}
      si
         nueva está en [pMin,pMax] -> potenciaActual := nueva
         nueva>pMax                -> potenciaActual := pMax
         nueva<pMin                -> potenciaActual := pMin
      fsi
   {Post: potenciaActual=nueva, limitada por
          arriba a potMax y por abajo a potMin}
  fmétodo
 fclase
```

3) La siguiente especificación describe un método al que se le pasa una tabla de números enteros y retorna otra tabla con los mismos números pero en orden inverso. Escribir para ella un diseño iterativo.

```
método invierte (entero[0..n-1] v) retorna entero[0..n-1]
   {Pre: n>=0, v={v0,v1, v2, ..., vn-2, vn-1} }
     invierte
   {Post: valor retornado={vn-1,vn-2,..., v2, v1, v0} }
fmétodo
```

4) Se pide especificar y diseñar un método recursivo llamado `esDivisible`, al que se le pasa una secuencia `s` de números enteros y un entero `num`, y que retorna un booleano que indica si `num` es divisible entre alguno de los números de la parte derecha de `s`, o por el contrario no es divisible entre ninguno. Si la parte derecha de s está vacía, retorna `false`.

   La secuencia `num` sigue la interfaz de la secuencia vista en clase. Para el diseño recursivo considerar:

   - El *caso directo* es el caso en el que la parte derecha de la secuencia está vacía (es decir, estamos al final de la secuencia); para este caso retornar `false`.
   - El *caso recursivo* es el caso contrario, en el que la parte derecha de la secuencia no es vacía; para ese caso si `num` es divisible entre el elemento *actual* de la secuencia retornar `true`, o si no es divisible *avanzar* la secuencia y luego retornar el valor retornado por la llamada recursiva al mismo método `esDivisible`.

Examen de Programación I
(Ingeniería Informática)

Septiembre 2007

Segunda parte (6 puntos, 60% nota del examen)

Se desea escribir un programa Java que gestione las calificaciones de los alumnos de una asignatura. Para ello se dispone de una clase ya realizada llamada `Alumno` que posee la siguiente interfaz.

Constructor Summary
Alumno(java.lang.String nombre) Constructor que crea un alumno con el nombre que se le da como argumento

Method Summary	
java.lang.String	**nombre**() Retorna el nombre del alumno
double	**notaEjercicios**() Retorna la nota de los ejercicios
double	**notaExamen**() Retorna la nota del examen
double	**notaFinal**() Retorna la nota final calculada con un peso del 60% para el examen, 30% para las prácticas y 10% para los ejercicios
double	**notaPracticas**() Retorna la nota de prácticas
void	**ponNotaEjercicios**(double nota) Pone la nota de ejercicios
void	**ponNotaExamen**(double nota) Pone la nota del examen
void	**ponNotaPracticas**(double nota) Pone la nota de prácticas

Todas las notas son un valor numérico entre 0 y 10. La calificación final de un alumno se obtiene mediante la media ponderada de las calificaciones obtenidas en el examen, en las prácticas y en los ejercicios, tal y como se especifica en el método `notaFinal()`.

Se pide:

1) (1 punto) Escribir un método de la clase `Alumno` denominado `aprobado()` que devuelva un booleano que indique si el alumno está aprobado o no. Para que un alumno se considere aprobado su nota final debe ser igual o superior a 5.0 y además debe tener aprobadas (nota mayor o igual que 5.0) las prácticas y los ejercicios.

2) (1 punto) Escribir un método de la clase `Alumno` denominado `calificacionFinal()` que retorne un `String` con la calificación en texto de un alumno (`SUSPENSO`;

APROBADO; NOTABLE; SOBRESALIENTE). La calificación final es "SUSPENSO" si el alumno no está aprobado (según el método aprobado()) y en caso contrario es "APROBADO" si la nota está en el rango [5.0,7.0), "NOTABLE" si está en [7.0,9.0), y "SOBRESALIENTE" en [9.0,10.0].

A continuación se desea escribir una clase denominada GestionAsignatura con operaciones para obtener listados de calificaciones de los alumnos de la asignatura. Dicha clase debe poseer dos atributos privados, num de tipo entero que es un contador que indica el número de matriculados en la asignatura y asignatura que es una tabla que contiene los alumnos de la asignatura en el orden en que han sido añadidos entre las posiciones [0..num-1]. Es decir:

```
atributos
    entero num;
    Alumno[0..99] asignatura;
fatributos
```

Se pide

3) (0.5 puntos) Declarar en Java los atributos de la clase GestionAsignatura tal y como se describen en el pseudocódigo anterior. Escribir un método constructor para la clase GestionAsignatura que ponga a cero el número de alumnos matriculados en la asignatura.

4) (0.5 puntos) Escribir un método de la clase GestionAsignatura denominado buscar(...) que toma como argumento el nombre de un alumno y devuelve la posición que ocupa en la tabla si se encuentra en ella y num si no se encuentra.

5) (1 punto) Escribir un método de la clase GestionAsignatura llamado añadir(...) que toma como argumento el nombre de un alumno, su calificación de examen, su nota de prácticas y su nota de ejercicios y, si la tabla no está llena y el alumno no está en la tabla, crea el alumno con los datos que se han suministrado y lo añade metiéndolo en la posición *num* de la tabla e incrementando luego ese contador; en este caso el método retorna la constante booleana true. Si la tabla está llena o el alumno está en la tabla no hace nada y retorna false.

6) (2 puntos) Escribir un método de la clase GestionAsignatura denominado estadística() que imprima por la pantalla el porcentaje de alumnos con la calificación APROBADO, de alumnos con la calificación NOTABLE, de alumnos con SOBRESALIENTE y de alumnos con SUSPENSO, de acuerdo al formato:

```
SUSPENSO            porcentaje%
APROBADO            porcentaje%
NOTABLE             porcentaje%
SOBRESALIENTE       porcentaje%
```