Examen Parcial de PRED (29/11/2006)

1. **(1.5 puntos)** Dado el siguiente esquema de programa en un lenguaje con estructura de bloques:

```
acción P1 (X1: entero)
     var Y1: entero
     acción P3 (X1: entero)
     … /* punto 1 */...
     facción
     acción P2 (X2: entero)
           var Y3: entero
           acción P1 (X1: entero)
                 … /* punto 3 */...

           facción
     … /* punto 2 */...
     facción
…
facción
```

Contestad a las siguientes preguntas razonando las respuestas:

a) Si en el punto 1 apareciera la instrucción P1(Y1), sería correcta?. En caso de que lo fuera, a qué P1 se llamaría?
b) Si en el punto 2 apareciera la instrucción P3(Y3), sería correcta?.
c) Si en el punto 3 apareciera la instrucción P1(X1), sería correcta?. En caso de que lo fuera, a qué P1 se llamaría y quien sería X1?
d) Si en el punto 1 apareciera la instrucción P2(X1), sería correcta?. En caso de que lo fuera, quien sería X1?.

2. **(1.5 puntos)** Dadas las declaraciones:

```
clase C
            x: entero;
            función F(y: entero) retorna C1
            ...
            ffunción
fclase
clase C1 subclase de C
            x1: entero;
            función F1(y: entero) retorna C1
            ...
            ffunción
fclase
clase C2 subclase de C1
            x2: entero;
            función F(y: entero) retorna C1
            ...
            ffunción
            función F2(y: entero) retorna C
            ...
            ffunción
fclase
clase C3 subclase de C
            x2: entero;
            función F(y: entero) retorna C1
            ...
            ffunción
fclase
```

Supongamos que las reglas que definen la herencia, los tipos y la visibilidad son las de Java y que tenemos las variables `a: C; a1: C1; a2: C2; a3: C3` y supongamos que `v, v1, v2` y `v3` son valores de las clases `C, C1, C2` y `C3`, respectivamente. En este contexto, cuáles de las siguientes secuencias de instrucciones serían correctas y por qué:

a) `a:=v2; a:= a.F(a1.x);`
b) `a:=v1; a1:= a;`
c) `a2:= v1; a2.x1:=3;`
d) `a1:=v3; a2:=a1.F(a2.x);`
e) `a:=v3; a2:=a.F(a2.x);`

Además, en el caso de las llamadas que consideres correctas al método `F`, ¿a qué `F` se llamaría?.

3. (1 punto) Supongamos que tenemos definiciones de funciones con los siguientes tipos:

```
f: t1 x t2 --> t2
f: t2 x t1 --> t1
g: t2 x t2 --> t1
g: t1 x t1 --> t2
```

siendo `t1` subtipo de `t3`, `t2` subtipo de `t3`, `t4` subtipo de `t1` y `t4` subtipo de `t2`. Suponiendo que `a2:t2`, `a3:t3`, `a4:t4`, decid si podemos inferir lo siguiente (y cómo):

```
a) f(g(a4,a4),g(a4,a2)): t3
b) f(g(a4,a4),g(a3,a2)): t4
```

4. (1.5 puntos) En Java (y en la mayor parte de los lenguajes orientados a objeto) la única forma de hacer que una clase sea subtipo (o subclase de otra) es usando el mecanismo de herencia. Por ejemplo si queremos que una clase C2 sea subclase de C1 tendremos que definir C2 diciendo:

```
class C2 extends C1 {...}
```

Supongamos que queremos modificar el lenguaje Java añadiendo una instrucción o declaración que, dadas dos clases ya definidas, nos permita hacer que una sea subclase de la otra. Por ejemplo, si tenemos definidas las clases:

```
class C1 {...}

class C2 {...}
```

queremos extender Java con una nueva declaración o instrucción de la forma:

```
class C2 is subclass of  C1
```

cuyo efecto sería, simplemente, hacer que, a partir de este momento, C2 fuera considerada subclase de C1, sin que C2 herede nada de C1. La cuestión es ¿qué comprobaciones habría que realizar para asegurarnos que esta declaración o instrucción es correcta? Por otra parte, ¿Pueden hacerse estas comprobaciones en tiempo de compilación o en tiempo de ejecución? ¿Por qué?

5. (1.5 puntos) Suponiendo que se ejecuta el siguiente programa P ¿cuales serían los resultados que se escribirían según el paso de parámetros sea por valor, valor/resultado o referencia?. Suponed que en el paso por valor y valor resultado los argumentos y los resultados se pasan de izquierda a derecha.

```
acción P ()
     var i: entero; a: tabla [0..4] de entero;
     acción swap (X,Y:entero)
           var Z: entero
           Z:= X;
           X:= Y;
           Y:=Z;
     facción
     función f() retorna entero
           i:= i+1;
           retorna i
     ffunción
     para i:= 0 a 4 hacer a[i]:= i+1; fpara
     i:=0;
     swap(a[f()],i);
     escribir (i);
     escribir (a);
facción
```

6. (3 puntos) Especificar una abstracción de datos que nos defina una estructura de datos para gestionar una competición contrarreloj. En concreto, la idea es que en esa estructura de datos guardemos, para cada participante en la competición su mejor tiempo. Se considera que cada participante se identifica con un número natural (su dni) y los tiempos son también números naturales. Específicamente, se desea tener las siguientes operaciones:

inic: nos devuelve la estructura vacía
añadir: dado un participante P y un tiempo T, se tienen dos casos. Si P estaba ya en la estructura de datos con un tiempo inferior a T, entonces la operación no hace nada. Si P no estaba en la estructura de datos o estaba con un tiempo mayor que T, entonces T pasa a ser el tiempo de P.
descalificar: dado un participante P, lo elimina de la estructura de datos.
ganador: nos devuelve el participante con un tiempo menor.
tiempo?: dado un participante P nos dice cual es su (mejor) tiempo.

Soluciones

1.

a) Es correcta ya que tanto P1 como Y1 son visibles desde el punto 1. Se llamaría al P1 más externo
b) Es correcta ya que tanto P3 como Y3 son visibles desde el punto 2.
c) Es correcta ya que tanto P1 como X1 son visibles desde el punto 3. La llamada a P1 sería recursiva y X1 sería su propio parámetro formal.
d) Es correcta ya que tanto P2 como X1 son visibles desde el punto 1. X1 sería su propio parámetro formal.

2.

a) La instrucción a:=v2; es correcta, ya que a un objeto de clase C siempre se le puede asignar un valor de su subclase. La instrucción a:= a.F(a1.x); también es correcta ya que: el objeto a tiene el método F, el objeto a1 hereda el campo x, la función F devuelve un resultado de clase C1 que es subclase de C (la clase de a). La F llamada sería la de la clase C2, ya que en ese momento a contiene un valor de clase C2.

b) La asignación a:=v1; es correcta, pero la asignación a1:= a; no lo es, ya que a un objeto de clase C1 no se le puede asignar un objeto de clase C.

c) La asignación a2:= v1; es incorrecta ya que a un objeto de clase C2 no se le puede asignar un objeto de clase C1.

d) La asignación a1:=v3; es incorrecta ya que a un objeto de clase C1 no se le puede asignar un objeto de clase C3.

e) La asignación a:=v3; es correcta, pero la asignación a2:=a.F(a2.x); no lo es, ya que a un objeto de clase C2 no se le puede asignar un objeto de clase C1 (el resultado de F)

3.

a)

$$\frac{a4:t4 \quad t4 < t1}{\dfrac{a4:t1 \qquad g: t1 \times t1 \to t2}{g(a4,a4):t2}}$$

Por otra:

$$\frac{a4:t4 \quad t4 < t2}{\dfrac{a4:t2 \qquad a2:t2 \qquad g: t2 \times t2 \to t1}{g(a4,a2):t1}}$$

Finalmente:

$$\frac{\dfrac{g(a4,a4):t2 \qquad g(a4,a2):t1 \qquad f: t2 \times t1 \to t1}{f(g(a4,a4),g(a4,a2)): t1} \qquad t1<t3}{f(g(a4,a4),g(a4,a2)): t3}$$

b) No es posible inferir f(g(a4,a4),g(a3,a2)): t4 ya que a3 tiene tipo t3 y no hay ninguna definición de g que tenga un primer parámetro de tipo t3.

4.) Para que C2 pueda ser considerada subclase de C1 C2 debería de contener todos los campos o atributos y todos los métodos de C1, como se supone que en este nuevo tipo de declaración C2 no puede heredar nada, habría que comprobar que C2 contiene esos campos y métodos. Esa comprobación puede ser estática, porque antes de la ejecución ya se conocen cuáles son los campos y los métodos de cada clase.

Por otra parte, en Java la herencia es simple, lo que quiere decir que una clase C2 no puede ser subclase de dos subclases C1 y C3, salvo que C1 sea subclase de C3 o C3 sea subclase de C1. Esto quiere decir que, si consideramos que esta característica debería de mantenerse, entonces cuando apareciera la declaración o instrucción:

```
class C2 is subclass of C1
```

habría que comprobar que C2 no es subclase de otra clase C3 (salvo que C1 fuera subclase de C3 o C3 fuera subclase de C1). Esta comprobación habría que hacerla en ejecución si la sentencia:

```
class C2 is subclass of  C1
```

fuera una instrucción. En caso contrario, bastaría con hacer la comprobación en tiempo de compilación.

5.

Paso por valor: Se escribiría 1 (valor de i) y 1 2 3 4 5 (valores de a)

Paso por valor-resultado: Se escribiría 2 (valor de i) y 1 1 3 4 5 (valores de a)

Paso por referencia o variable: Se escribiría 2 (valor de i) y 1 1 3 4 5 (valores de a)

6. Una posible especificación, suponiendo que no se pueden dar dos tiempos idénticos, es la siguiente:

```
Especificación compet
tipos compet, nat
operaciones
    inic : -> compet
    añadir: compet x nat x nat -> compet
    descalificar: compet x nat -> compet
    ganador: compet -> nat (parcial)
    tiempo?: compet x nat -> nat (parcial)
axiomas
    ganador(inic)= indefinido
    tiempo(inic,P) = indefinido

    añadir(añadir(C,P',T'),P,T) = añadir(añadir(C,P,T),P',T')
    T ≤ T' => añadir(añadir(C,P,T),P,T') = añadir(C,P,T)

    descalificar(inic,P) = inic
    descalificar(añadir(C,P,T),P)= descalificar(C,P)
    P≠P => descalificar(añadir(C,P,T),P')= añadir(descalificar(C,P',T),P)

    ganador(añadir(inic,P,T)) = P
    T' > T => ganador(añadir(añadir(C,P',T'),P,T)) = ganador(añadir(C,P,T))
    T' < T => ganador(añadir(añadir(C,P',T'),P,T)) = ganador(añadir(C,P',T'))

    tiempo?(añadir(inic,P,T),P) = T
    P≠P' => tiempo?(añadir(C,P,T1),P') = tiempo?(C,P')
    T>T' => tiempo?(añadir(añadir(C,P,T),P,T'),P) = tiempo?(añadir(C,P,T'),P)
    T<T' => tiempo?(añadir(añadir(C,P,T),P,T'),P) = tiempo?(añadir(C,P,T),P)

fespecificación
```