Paradigmas de la Programación
Segundo Parcial

Gabriel Infante-Lopez Ezequiel Orbe Luciana Benotti

17 de junio de 2010

Apellido y Nombre: _____

Instrucciones:

- Lea todos los ejercicios antes de iniciar la resolución del exámen.
- Resuelva cada uno de los ejercicios (1, 2, 3 y 4) en hojas distintas.
- Sea conciso y claro. Se quitarán puntos cuando las respuestas sean confusas o irrelevantes.
- Coloque nombre y nro de página en todas las hojas.
- Buena Suerte!

Ejercicio	Pts. Otorgados	Pts. Obtenidos
1	33	
2	10	
3	23	
4	34	
TOTAL	100	
NOTA		
REGULAR	SI	NO
PROMOCION	SI	NO

1. ***(33 Puntos)*** Considere las siguientes funciones:

```
BadFunction = proc {$ X}
                 if {mod X 2} == 0 then
                    {BadFunction X}
                 else
                    {Delay 1000}
                    X
                 end
              end

OddFilter = proc {$ X}
               case X of nil then nil
               [] X|Y|XS then X|{OddFilter XS}
               end
            end

GenerateAllIntegers = proc {$ X Y}
                         if x < Y  then
                            {BadFunction X} | {GenerateAllIntegers X+1 Y}
                         else
                            nil
                         end
                      end
```

La llamada siguiente:

```
{OddFilter {GenerateAllIntegers 0 1000}}
```

no termina.

a) ***(23 Puntos)*** Modifique el código para que la llamada a función termine y devuelva una lista con los primeros 500 nros. pares. Las modificaciones deberán estar sujetas a las siguientes restricciones:

- No se puede modificar la función `BadFunction`.
- La función `GenerateAllIntegers` debe hacer uso de la función `BadFunction`.

b) ***(10 Puntos)*** ¿Cuántos *threads* se construyeron en la definición que propuso? ¿Se podrá redefinir el problema de manera que si el `Delay` de la función `BadFunction`, es lo suficientemente largo, haya tanto threads como números tenga la lista inicial? Justifique.

2. ***(10 Puntos)*** La función `fun{Q A B}` calcula la suma $\sum_{i=A}^{B} i$ iterativamente. Asuma que $A > 0$, $B > 0$ y $B > A$.

   a) ***(5 Puntos)*** Implemente `Q` de forma de que sólo use estado implícito para calcular la suma. Muestre un ejemplo de uso de la función.

   b) ***(5 Puntos)*** Implemente `Q` de forma de que sólo use estado explícito para calcular la suma. Muestre un ejemplo de uso de la función.

3. ***(23 Puntos)*** En la Empresa Itsey Bitsey Machine usan el siguiente sistema para administrar los salarios de sus empleados:

```
% Esta funcion solo es conocida por Eben Scrooge
fun{MakeEmployee Name Amount}
   salary(name: Name amount: {NewCell Amount})
end

% Este procedimiento solo es conocido por Eben Scrooge
proc{ChangeSalary Salary NewAmount}
    {Salary.amount := NewAmount}
end

% Una referencia a esta funcion se le da a cada empleado
fun{CheckSalary Salary}
   {@Salary.amount}
end
```

Cuando un empleado es contratado la jefa del departamento contable, Eben Scrooge, crea una instancia empleado (llamando a `MakeEmployee`). Una referencia a esta instancia asi como también `CheckSalary` se da al empleado para que pueda verificar el monto de su salario en el futuro. Sólo el departamento contable tiene acceso a `ChangeSalary` y `MakeEmployee`. Uno de los empleados ha comenzado a incrementar su salario manipulando el campo `amount` de su salario. Modifique el sistema de forma que los empleados puedan seguir viendo pero no modificando sus salarios.

4. ***(34 Puntos)***

a) ***(10 Puntos)*** En Oz se permite ***herencia múltiple***, esto es, una clase puede heredar de una o más clases. En cambio en Java solo se permite ***herencia simple***, es decir, una clase puede heredar solo de una clase.

(I) Modifique el sistema de objetos de Oz para que solo se soporte herencia simple.

b) ***(16 Puntos)*** En Oz todos los métodos de una clase base son heredados por sus clases derivadas y pueden ser redefinidos en las mismas. Sin embargo, no hay un mecanismo que permita "remover" un método heredado. Suponga que extendemos Oz de forma tal de que podamos declarar métodos como ***phantom***. Un método ***phantom***, es un método que solo se hereda si en la clase derivada hay una definición explícita del mismo. Por ejemplo, si tenemos las siguientes clases:

```
class A
   attr val
   panthom meth f(X)
      ...
   end
   meth g(X)
      {self f(X)}
   end
end

class B from A
   attr val1
   meth g(X)
      A,g(X)
   end
end
```

luego, el método `f` es un método válido de la clase `A` y puede ser invocado en todo objeto de tipo `A`. Sin embargo, dado que `f` no esta declarado en la clase `B`, dicho método no se hereda, y por consiguiente, todo invocación del método `f` que se haga sobre objetos de tipo `B` será inválida.

(I) ***(10 Puntos)*** Tomando como base el sistema de objetos modificado del punto anterior, modifique el mismo para que soporte la definición de métodos ***phantom***.

(II) ***(6 Puntos)*** ¿Hay algún principio general que se viole con esta implementación? Si su respuesta es positiva, de un ejemplo.

c) ***(8 Puntos)*** Ahora considere el siguiente código:

```
Obj={New B ...}
{Obj g(2)}
```

(I) ***(4 Puntos)*** ¿Se presenta algún problema cuando se invoca el método g?. Justifique

(II) ***(4 Puntos)*** Si su respuesta al punto anterior fue positiva, ¿qué modificaciones haría a la definición de las clases para evitar el/los problemas que se presentan?.