

Paradigmas de la Programación – Recuperatorio del Segundo Parcial

17 de Junio de 2015

Apellido y Nombre: _____

1. [10 pt.] Seleccione todas las respuestas correctas entre las diferentes opciones que se dan para completar cada oración:
 - a) La diferencia entre `extends` e `implements` en Java consiste en que...
 - 1) `extends` crea relaciones de herencia, mientras que `implements` crea relaciones de implementación.
 - 2) `extends` se aplica para heredar de clases, mientras que `implements` se aplica para heredar de interfaces.
 - 3) `extends` sólo permite herencia simple, mientras que `implements` permite herencia múltiple.
 - b) Muchos lenguajes proveen sintaxis amigable para referirse a algunas clases dentro de la jerarquía de clases, que son:
 - 1) las clases hermanas de la clase desde las que se las refiere.
 - 2) la clase madre de la clase desde que se la refiere.
 - 3) las clases ancestro de la clase desde la que se está trabajando.
 - 4) la propia clase desde la que se está trabajando.
 - 5) las clases descendientes de la clase desde la que se está trabajando.
 - c) Señale cuáles de las siguientes expresiones son `true` en Prolog, es decir, en qué casos se encuentra una unificación exitosa para la expresión:
 - 1) `'Libro' = libro`
 - 2) `Revista = libro`
 - 3) `Juan = leer(Juan)`
 - 4) `convertir(booleano) = booleano`
 - 5) `leer(odisea,Juan) = leer(Iliada,maria)`
 - 6) `leer(odisea(juan),Maria) = leer(Juan,iliada(maria))`
 - 7) `leer(odisea,Juan,iliada) = leer(Iliada,maria)`
 - 8) `leer(odisea(juan),iliada(maria)) = leer(X,Y)`
2. [20 pt.] Los actores se comunican mediante mensajes desordenados. Explique qué información y métodos debería añadir a un modelo basado en actores para poder procesar los mensajes recibidos en el orden en el que fueron enviados.

3. [30 pt.] Reescriba el siguiente código en Java en un lenguaje declarativo, por ejemplo Haskell o pseudocódigo declarativo, de forma que su semántica denotacional sea idéntica.

```
class P01 {  
  
    <T> T f1(List<T> list) {  
        if (list.isEmpty()) throw new NoHayElemento("lista_vacia");  
        List<T> elements = list.tail();  
        List<T> result = list;  
        while (elements.nonEmpty()) {  
            result = elements;  
            elements = elements.tail();  
        }  
        return result.head();  
    }  
}
```

4. [20 pt.] El siguiente ejemplo en C++ causa un error de compilación. Por qué? cómo puede solucionarse?

```
class trabajador  
{  
    public:  
        void hora_de_levantarse( )  
        { ..... }  
};  
class estudiante  
{  
        void hora_de_levantarse( )  
        { ..... }  
};  
class ayudante_alumno : public trabajador, public estudiante  
{  
  
};  
  
int main()  
{  
    ayudante_alumno obj;  
  
    obj.hora_de_levantarse( )  
}
```

5. [20 pt.] En el siguiente código java, inserte en el `main` los mecanismos de manejo de excepciones necesarios para capturar la excepción de forma adecuada.

```
public class RepartirComensales
{
    public static void main(String [] args)
    {
        Mesa mesa = new Mesa(1);
        System.out.println("vamos a llenar la mesa 1...");
        mesa.aniadirComensal("Ana");
        mesa.aniadirComensal("Juan");
        mesa.aniadirComensal("Maria");
        mesa.aniadirComensal("Pedro");
        mesa.aniadirComensal("Juana");
        mesa.aniadirComensal("Esteban");
        mesa.aniadirComensal("Lola");
    }
}

public class Mesa
{
    private int numeroDeComensales;
    private int numeroDeMesa;
    public Mesa(int numeroDeMesa)
    {
        this.numeroDeMesa = numeroDeMesa;
    }
    public void aniadirComensal(string comensal) throws ExcepcionMesaLlena
    {
        if(numeroDeComensales > 5)
        {
            throw new ExcepcionMesaLlena(numeroDeComensales)
        }
        else
        {
            numeroDeComensales += 1;
        }
    }
}
```