Universidad Carlos III de Madrid
Departamento de Ingeniería Telemática
Examen de Programación de Sistemas, Convocatoria Extraordinaria, Junio 2013

NOMBRE:

APELLIDOS:
NIA:
GRUPO:

2ª Parte: Problemas (5 puntos sobre 10)

Duración: 1 hora 45 minutos
Puntuación máxima: 5 puntos
Fecha: 17 de Junio de 2013

PROBLEMA 1 (3 puntos)

El sistema de gestión de personal de la empresa MyCompany está implementado en Java y utiliza la interfaz *MyCompany*, la cual se presenta a continuación:

```
public interface MyCompany {
      static final double BASIC_SALARY = 1000.0;
      public double calculateSalary();
      public String formatSkills();
}
```

Apartado 1 (1 punto)

Programa la clase *Employee*, la cual implementa la interfaz *MyCompany,* y representa a cualquiera de los empleados que trabajan en esta compañía. Ten en cuenta que:

- Esta clase debe tener dos atributos, *name* de tipo *String*, y *years* de tipo *int*, los cuales indican el nombre del empleado, y los años que lleva trabajando en la compañía respectivamente. Los atributos deben ser necesariamente privados.
- El constructor de la clase *Employee* recibe dos argumentos que se utilizan para inicializar los atributos de la clase *Employee* mencionados en el punto anterior.
- La clase *Employee* implementa el método *calculateSalary()*, el cual calcula el salario del empleado sumando al salario básico (*BASIC_SALARY*), el producto de los años que lleva el empleado en la empresa por el entero "100".
- La clase *Employee* implementa el método *getName()*, el cual devuelve el nombre del empleado.
- El método *formatSkills()* no se debe implementar en la clase *Employee* porque no tenemos información para hacerlo, así que debemos obligar a las clases hijas a implementarlo; recuerda que debes indicar esto en la definición de la clase

NOTA: *No pueden añadirse nuevos métodos o atributos en la clase Employee además de los mencionados.*

Apartado 2 (1,5 puntos)

Programa la clase *Developer*, la cual hereda de la clase *Employee*, y representa a cualquiera de los empleados que trabajan como desarrolladores en esta compañía. Ten en cuenta que:

- Esta clase debe incluir dos atributos, *productivity* de tipo *double*, el cual representa un factor de productividad del desarrollador, y *masteredLanguages* donde se recogen en un array de tipo *String* los lenguajes de programación que domina el desarrollador.
- El constructor de la clase *Developer* recibe exactamente el número de argumentos que se necesitan para inicializar todos los atributos de la clase *Developer*, incluidos los heredados de la superclase.
- La clase *Developer* sobrescribe el método *calculateSalary()* para multiplicar el salario del desarrollador, tal y como se calcula en la clase *Employee*, por su factor de productividad.
- La clase *Developer* implementa el método *formatSkills()* el cual devuelve la concatenación de (1) el nombre del desarrollador; (2) el String “ is an expert on ”; (3) los lenguajes que domina el desarrollador separados por espacios; (4) el String “and earns ”; (5) el salario del desarrollador.

NOTAS: *No pueden añadirse nuevos métodos o atributos en la clase Developer además de los mencionados. En el método formatSkills() puedes asumir que el atributo masteredLanguages nunca estará vacío.*

Apartado 3 (0,5 puntos)

Programa la clase *Test,* la cual contiene un método *main*. Dicho método debe permitir crear dos instancias con las siguientes características:

- Juan, el cual lleva diez años en la empresa, tiene un factor de productividad de 1,2 y domina los lenguajes Java y PHP;
- Miguel, el cual lleva cinco años en la empresa, tiene un factor de productividad de 1,8 y domina los lenguajes Java, C y Python;

e imprimir por salida estándar (*System.out*) la información de ambas utilizando el método *formatSkills()*.

PROBLEMA 2 (2 PUNTOS)

El siguiente código representa un nodo de un árbol binario. Se pretende utilizar el árbol para representar funciones reales en la variable real x, cuya definición contenga:

- números reales
- la variable real x
- y los operadores binarios +,-,* (suma, resta y multiplicación de números reales respectivamente)

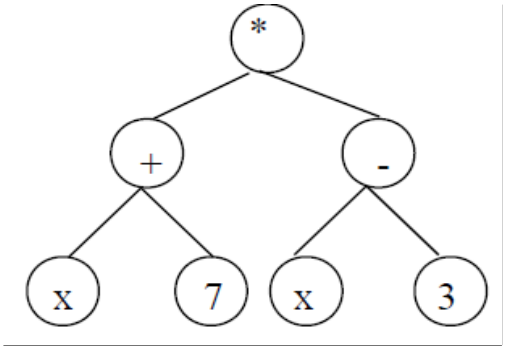
Tanto los números, como los operadores (+,-,*), como la variable x se guardan como cadenas de texto en el árbol.

```
public class BNode {
            private String info;
            private BNode left;
            private BNode right;

            public BNode(String info) {
                  this.info=info;
                  left=null;
                  right=null;
            }


            public String getInfo() {
                  return info;
            }

            public BNode getLeft() {
                  return left;
            }

            public BNode getRight() {
                  return right;
            }

            public void setLeft(BNode left){
                  this.left=left;
            }
            public void setRight(BNode right){
                  this.right=right;
            }

            public void setInfo (String info){
                  this.info=info;
            }

            public void print(){
                  // a programar en el apartado 1
            }


            public float evaluate(float value){
                  // a programar en el apartado 2
            }

}
```

`Para simplificar` la solución, se puede asumir que el árbol está siempre bien construido, esto es:

- Los números reales o la variable x están siempre en las hojas. Por tanto, si se sabe que un nodo guarda un valor real o la variable x, se puede asumir, sin necesidad de comprobarlo, que las referencias a sus hijos valen *null*.
- Los operadores están siempre en nodos internos, y todos los nodos internos tienen exactamente dos hijos. Por tanto, si se sabe que un nodo guarda un número real o la variable x, se puede asumir, sin necesidad de comprobarlo, que ninguna de las referencias a sus hijos vale *null*.
- Todos los nodos guardan una cadena de texto que o bien representa un número real correcto, o la variable x, o bien representa un operador de suma ("+"), resta ("-") o producto ("*"). Esto implica que no es necesario comprobar que pueda haber un dato incorrecto en un nodo.

Como ejemplo, la función `f(x)=(x+7)*(x-3)` se representaría mediante el siguiente árbol



Apartado 1 (1 PUNTO)

Programa el método `public void print()`, que imprime en la salida estándar la función real de variable real partiendo de su representación arbórea. Cada operación de suma, resta o producto debe aparecer entre paréntesis, pero no los operandos que sean directamente un número real o la variable x. Por ejemplo, para la función anterior el método imprimirá ((x+7.0)*(x-3.0))

Apartado 2 (1 PUNTO)

Escribir el método `public float evaluate(float value)` que devuelve el resultado de evaluar la función real de variable real del árbol cuya raíz es el nodo sobre el cual se invoca, esto es calcule el valor `f(value)`. A modo de ejemplo, si se invoca sobre la raíz del árbol que se muestra arriba y con value=3 obtendremos `f(3)=(3+7)*(3-3)=0`

Nota: Para transformar una cadena de caracteres a float se puede utilizar el método estático parseFloat de la clase Float cuya cabecera es:

```
public static float parseFloat(String)
```

Soluciones problema 1

Apartado 1

```
public abstract class Employee implements MyCompany {

	private String name;
	private int years;

	public Employee (String name, int years){
		this.name = name;
		this.years = years;

	}

	public double calculateSalary(){
		return (100*years)+BASIC_SALARY;
	}

	public abstract String formatSkills();

	public String getName(){
		return this.name;
	}


}
```

Apartado 2

```
public class Developer extends Employee {

	private String[] masteredLanguages;
	private double productivity;

	public Developer(String name, double productivity, int years, String[] masteredLanguages) {
		super(name, years);
		this.masteredLanguages = masteredLanguages;
		this.productivity = productivity;
	}

	public double calculateSalary(){
		return super.calculateSalary()*productivity;
	}


	public String formatSkills(){
		String formattedMasteredLanguages = new String();
		for (int i = 0; i < masteredLanguages.length; i++){
			formattedMasteredLanguages = formattedMasteredLanguages +
masteredLanguages[i]+ " ";
		}
		return getName() + " is an expert on " + formattedMasteredLanguages + " and earns " +
calculateSalary();
	}

}
```

Apartado 3

```
public class ClaseTest {

	public static void main (String args[]){
		Developer employee1 = new Developer("Juan", 1.2, 10, new String[]{"Java", "PHP"});
		Developer employee2 = new Developer("Juan", 1.8, 5, new String[]{"Java", "C", "Python"});
		System.out.println(employee1.formatSkills());
		System.out.println(employee2.formatSkills());

	}
}
```

Soluciones problema 2

Apartado 1 (1 PUNTO)

```
public void print(){
	// a programar en el apartado 1
	if	(getRight()==null && getLeft()==null) {
		System.out.print(getInfo());
	}
	else{
		System.out.print("(");
		getLeft().print();
		System.out.print(getInfo());
		getRight().print();
		System.out.print(")");
	}
}
```

Apartado 2 (1 PUNTO)

```
public float evaluate(float value){
	// a programar en el apartado 2
	if	(getRight()==null && getLeft()==null) {
		if (getInfo().equals("x")) return value;
		else return Float.parseFloat(getInfo());
	}
	else{
		if (getInfo().equals("+")) return getLeft().evaluate(value)
		+getRight().evaluate(value);
		else if (getInfo().equals("-")) return
		getLeft().evaluate(value)-getRight().evaluate(value);
		else return
getLeft().evaluate(value)*getRight().evaluate(value);
	}
}
```

O bien otra versión sería la siguiente utilizando la característica nueva de los switch con String a partir de JSE 7.0 o posterior, quedaría un código más elegante

```
public float evaluate(float value){
	// a programar en el apartado 2
	if	(getRight()==null && getLeft()==null) {
		if (getInfo().equals("x")) return value;
		else return Float.parseFloat(getInfo());
	}
	else{
		float res=0.0f;
		switch (getInfo()){
		case "+":
			res= getLeft().evaluate(value)+getRight().evaluate(value);
			break;
		case "-":
			res= getLeft().evaluate(value)-getRight().evaluate(value);
			break;
		case "*":
			res= getLeft().evaluate(value)*getRight().evaluate(value);
```

```
                    break;
                }
                return res;
        }
}
```

Para la construcción del árbol que se corresponde con la función real de variable real de ejemplo y la prueba de los dos apartados, se tiene:

```
public static void main (String args[]){
        BNode hi=new BNode("+");
        hi.setLeft(new BNode("x"));
        hi.setRight(new BNode("7"));
        BNode hd=new BNode("-");
        hd.setLeft(new BNode("x"));
        hd.setRight(new BNode("3"));
        BNode arbol= new BNode("*");
        arbol.setLeft(hi);
        arbol.setRight(hd);
        arbol.print();
        System.out.println();
        System.out.println(arbol.evaluate(3.0f));
}
```