DECSAI
Departamento de Ciencias
de la Computación e I.A.

Complementos de Programación

Convocatoria de Septiembre. Curso 2012/2013

4 de Septiembre de 2013

1. **(1 punto)** Indica lo que escribe el siguiente programa en la salida estándar, explicando por qué es así.

```
public class Profesor extends Empleado {
  public static void main(String[] args) {
    Profesor profesor1;
    Profesor profesor2 = new Profesor();
  }

  public Profesor() {
    System.out.println("Profesor");
  }
}

class Empleado extends Persona {
   public Empleado() {
     System.out.println("Empleado");
   }
 }

class Persona {
  public Persona() {
    System.out.println("Persona");
  }
}
```

2. **(1 punto)** Indica cual es la salida de los siguientes programas y explica por qué es así.

   a) Programa 1

```
public class Test {
  public static void main(String[] args) {
    A a = new A();
    a.p(10);
    a.p(10.0);
  }
}

class B {
  public void p(double i) {
    System.out.println(i * 2);
  }
}

class A extends B {
  public void p(double i) {
    System.out.println(i);
  }
}
```

   b) Programa 2

```
public class Test {
  public static void main(String[] args) {
    A a = new A();
    a.p(10);
    a.p(10.0);
  }
}

class B {
  public void p(double i) {
    System.out.println(i * 2);
  }
}

class A extends B {
  public void p(int i) {
    System.out.println(i);
  }
}
```

3. **(1 punto)** Indica cuales son las diferencias entre las excepciones comprobadas y las no comprobadas en Java.

4. **(3 puntos)** La notación *postfijo* permite escribir expresiones aritméticas sin usar paréntesis. Consiste en colocar primero los dos operandos y después el operador. Por ejemplo, la expresión $(1+2)*3$ se escribe en postfijo como $1\ 2+3*$. Para evaluar una expresión aritmética escrita en postfijo podemos hacer uso de una **pila**. La expresión postfijo se escanea de izquierda a derecha. Al leer un token operando, éste se inserta en una pila. Al leer un token operador, se aplica el operador con los dos operandos del tope de la pila (borrándolos de la pila), y el resultado se inserta en la pila. El proceso continúa hasta que la pila quede vacía. El resultado de evaluar la expresión es el operando que se haya extraido en último lugar.

   Escribe un programa que evalúe una expresión postfijo almacenada en un `String`. El valor del string debe leerse como argumento número 0 de la línea de comandos del `main`.

   Para obtener la lista de tokens (operandos y operadores) almacenados en el string `expresion`, se puede utilizar el siguiente código:

```
...
expresion = insertarBlancos(expresion);
String[] tokens = expresion.split(" ");
...

public static String insertarBlancos(String s) {
  String result = "";
  for (int i = 0; i < s.length(); i++) {
    if (s.charAt(i) == '+' || s.charAt(i) == '-' ||
        s.charAt(i) == '*' || s.charAt(i) == '/')
      result += " " + s.charAt(i) + " ";
    else
      result += s.charAt(i);
  }
  return result;
}
```

5. **(4 puntos)** Usa el framework Fork/Join para construir un programa paralelo que encuentre el valor máximo de un array de `int`.

Duración del examen: 2 horas y media.