UNIVERSIDAD PONTIFICIA DE SALAMANCA EN MADRID
Facultad de Informática
Escuela Universitaria de Informática
Departamento de Lenguajes y Sistemas Informáticos e Ingeniería de Software

Cuadernillo de examen

ASIGNATURA : **Laboratorio de Sistemas Operativos Abiertos (Java)**	CÓDIGO: **321**
CONVOCATORIA: **Junio 2002 (2º Parcial)**	PLAN DE ESTUDIOS: **1996**
ESPECIALIDAD: **Sistemas**	CURSO: **3º**
TURNO: **Mañana**	CENTRO: **Escuela**
CARÁCTER: **Anual**	CURSO ACADÉMICO: **2001/2002**

Soluciones propuestas

Preguntas teóricas

1. El modelo de compilación Java.

- Explique detalladamente las distintas fases que llevan desde el programa fuente a la ejecución de un programa en Java.
  (Obtención de uno o varios archivos .class a partir de un programa fuente, la relación de éstos con los nombres y ámbitos de las clases del programa fuente, etc.)
- Concepto y utilidad de *bytecode*.
  (Código intermedio que se ejecuta en el procesador de la máquina virtual Java. Permite la ejecución del mismo programa bajo distintas plataformas, siempre que exista una JVM para dicha plataforma).
- Elementos que intervienen en la ejecución de un programa Java.
  (Utilidad del verificador de bytecode, el cargador de clases y la unidad de ejecución)

Puntuación: 1 punto.

2. Visibilidad de métodos. Explique los distintos tipos de visibilidad de los métodos en una clase Java.
(Acceso amistoso y los modificadores public, private y protected)
Puntuación: 1 punto.

2. Control de excepciones en Java.

- Concepto de excepción.
  (Condición que se presenta en la ejecución de un programa. Motivos por los que se produce una excepción. Procesos que se producen al lanzar una excepción).
- Captura de excepciones: el bloque `try`.
  (Utilidad del bloque try. Captura de excepciones con el bloque catch. Argumentos y funcionamiento del bloque catch. El bloque finally)
- Generación de excepciones.
  (Creación de un objeto de la clase Exception. La cláusula throws. El método throw)

Puntuación: 1 punto.

Preguntas prácticas

Una empresa tiene un **máximo** de 50 empleados de dos tipos diferentes: los vendedores y los administrativos. Todos los empleados tienen un código de empleado, un nombre y un sueldo fijo. Los vendedores mantienen información sobre el volumen de ventas relizados en el mes, mientras que los administrativos guardan información sobre las horas extras trabajadas. El sueldo de los vendedores se compone de un sueldo fijo más las comisiones por ventas (un 10 por ciento de las ventas realizadas en el mes). El sueldo de los administrativos se compone de un sueldo fijo más 30 euros por cada hora extra trabajada.

Codifique en Java:

- Las clases `Empleado`, `Vendedor` y `Administrativo`. Todas tendrán un método `calcularSueldoMensual`. La clase `Vendedor` tendrá un método para aumentar el importe de las ventas del mes. La clase `Administrativo` tendrá un método para aumentar el número de horas extras trabajadas.

- La clase `Nomina`, compuesta de objetos `Empleado`. También contendrá información sobre el mes, el número de empleados que aparecen en la nómina dicho mes y el total de sueldo pagados. Esta clase tendrá un método `totalMes` para averiguar el total de la nómina del més y otro método `listarNomina` que devolverá por cada empleado su código, su nombre, su sueldo fijo y su sueldo neto del mes.
- Codifique un método main para la clase `Nomina` que permita crear y añadir empleados a la nómina del mes, incrementar sus ventas o número de horas extras, devolver un resumen con el total de empleados y el total de sueldos pagados durante dicho mes y devolver un listado de la nómina del mes.

Puntuación: 2 puntos.

```
/**
 * Clase abstracta Empleado. De ella derivan las clases
 * Vendedor y Administrativo
 */
abstract class Empleado{
    private String codigo = "000000";
    private String nombre = "anonimo";
    protected double sueldoFijo = 0;

   /**
    * Contructor de la clase Empleado
    * Recibe el código y el nombre del empleado
    */
    Empleado(String cod, String nom){
        codigo = cod;
        nombre= nom;
    }

   /**
    * Devuelve el código, el nombre y el sueldo del empleado
    */
    public String toString(){
        return this.codigo + " " + this.nombre + " " + this.sueldoFijo;
    }

    abstract public double calcularSueldoMensual();
}

/**
 * Clase Administrativo
*/
class Administrativo extends Empleado{
    private int horasExtras = 0;
    final int PRECIO_HORA = 30; // Precio por hora extra

   /**
    * Constructor de la clase Adminstrativo
    * Recibe el código, el nombre, el sueldo y las horas extras del empleado
    * Utiliza el constructor de la superclase
    */
    Administrativo (String cod, String nom, double s, int h){
        super(cod,nom);
        sueldoFijo = s;
        horasExtras = h;
    }

   /**
    * Al método toString de la superclase añade
    * el tipo de empleado (A) y el número de horas extras
    */
```

```
    public String toString(){
        return "(A) " +  super.toString() + " " + this.horasExtras ;
    }

   /**
    * Incrementa en h el número de horas extras del empleado
    */
    public void aumentarHorasExtras(int h){
        horasExtras += h;
    }

   /**
    * Calcula el sueldo mensual a partir de las
    * horas extras y el sueldo fijo
    */
    public double calcularSueldoMensual(){
        return sueldoFijo + horasExtras * PRECIO_HORA;
    }
}

class Vendedor extends Empleado{
    private double ventasMes = 0;
    final double COMISION = 10.0; //Comisión por ventas

   /**
    * Contructor de la clase Vendedor
    * Recibe el código, el nombre, el sueldo y las ventas del empleado
    * Utiliza el constructor de la superclase
    */
    Vendedor (String cod, String nom, float s, float v){
        super(cod,nom);
        sueldoFijo = s;
        ventasMes = v;
    }

   /**
    * Al método toString de la superclase añade
    * el tipo de empleado (V) y sus ventas mensuales
    */
    public String toString(){
        return "(V) " +  super.toString() + " " + this.ventasMes ;
    }

   /**
    * Incrementa en v las ventas del empleado
    */
    public void aumentarVentas(double v){
        ventasMes =+ v;
    }

  /**
    * Calcula el sueldo mensual a partir de las
    * ventas y la comisión por ventas
    */
    public double calcularSueldoMensual(){
        return sueldoFijo + sueldoFijo * (COMISION / 100);
    }
}

public class Nomina {
    String mes = null;
    Empleado nom[] = new Empleado[50];
```

```
    int numEmpleados = 0;
    double totalSueldosMes = 0;

    public Nomina(String mes) {
        this.mes = mes;
    }

    public void añadirEmpleado(Empleado e){
        nom[numEmpleados] = e;
        numEmpleados++;
    }

    public void sumarTotalMes(){
        for(int i=0;i<=this.numEmpleados-1;i++)
            totalSueldosMes += nom[i].calcularSueldoMensual();
    }

    public void listarNomina(){
        System.out.println(this);
    }

    public String toString(){
        String cad = null;
        cad = this.mes + '\n';
        for (int i=0;i<=this.numEmpleados-1;i++)
            cad += nom[i] + " " + nom[i].calcularSueldoMensual() + '\n';
        sumarTotalMes();
        cad += "Total: " + this.totalSueldosMes;
        return cad;
    }

    public static void main(String args[]){
        Nomina n = new Nomina("Enero");
        Empleado e = new Vendedor("222222","Ana González",1300,1000);
        n.añadirEmpleado(e);
        e = new Vendedor("111111","Pepe Pérez",1200,800);
        n.añadirEmpleado(e);
        e = new Administrativo("3333333","Juan de los Palotes",1500,0);
        n.añadirEmpleado(e);

        n.listarNomina();
    }
}
```