UNIVERSIDAD DE MALAGA
DEPARTAMENTO DE LENGUAJES Y
CIENCIAS DE LA COMPUTACION

PROGRAMACION ORIENTADA A OBJETOS
CONVOCATORIA DE SEPTIEMBRE DE 2001
INGENIERIA TECNICA EN INFORMATICA

Apellidos, Nombre: .....Máquina: .....
Especialidad: ..... Curso: .....

1) (*1pto.*) Crear la clase **Respuesta** cuyas instancias mantienen información sobre una cadena de caracteres y dos enteros.

- (*0.25*) Se debe poder crear una instancia conocidos la cadena y los enteros que debe mantener.
- (*0.25*)Además dispone de métodos para conocer cada uno de los objetos que contiene (*getCadena()*, *getX()* y *getY()*).
- (*0.25*) Implementar para esta clase el método `equals(Object)`. Dos respuestas son iguales si lo son las cadenas que contienen.
- (*0,25*) Dar los métodos necesarios para que un objeto de esta clase pueda ser representado como una cadena de caracteres. Su representación debe ser simple:
  `RESPUESTA(cadena, entero1, entero2)`

2) (*2.5ptos.*) Crea la clase **MasterM** que contiene información sobre un número secreto de manera que cualquier usuario de esta clase puede intentar averiguar ese número. Se mantendrá información del número de intentos realizados hasta acertarlo. Los números se representarán como cadenas de caracteres no pudiendo en ningún caso tener cifras repetidas.

- (*0.75*) Escribir el código necesario para crear objetos de la clase **MasterM**. En la creación del objeto, se crea a su vez una cadena que representa a número secreto de longitud dada (por defecto de longitud es de 4 cifras). El número no puede tener cifras repetidas.

Además, debemos proporcionar métodos para:

- (*0.5*) Dada una cadena que representa a un número, verificar que el número es válido. Un número es válido si tiene la misma longitud que el número secreto y no tiene cifras repetidas (*boolean valido(String)*).
- (*0.75*) Dada una cadena que representa a un número, encontrar cuantas cifras coinciden en su posición con el secreto (le llamaremos muertos) y cuantos no coinciden en su posición pero aparecen en el secreto (le llamaremos heridos). Previamente habrá que comprobar que el número dado es válido. Si todo ha ido bien, devolveremos un objeto **Respuesta** con el número dado, los muertos y los heridos e incrementaremos el número de intentos. Si el número no era válido, devolveremos `null`. (*Respuesta intento(String)*).
  Por ejemplo, si el secreto es “36512” y el que se prueba es “16742”, entonces hay dos muertos (el 6 y el 2) y un herido (el 1).
- (*0.25*) Conocer si un número dado coincide con el secreto (*boolean esFinal(String)*).
- (*0.25*) Dar un método que permita conocer el número secreto (*String secreto()*) y otro que permita conocer el número de jugadas que se lleva hasta ese momento (*int numJugadas()*).

3) (*2.5ptos.*) Los objetos de la clase **MasterMP** se comporta igual que los de la clase **MasterM** excepto que recuerda todos los números válidos que se han intentado junto con sus muertos y heridos no permitiendo que el usuario repita un número. En caso de repetición, lo considerara como un número no válido.

Además, dispone de un método que permite mostrar (*void muestra(PrintWriter))* en un **PrintWriter** dado, toda la información almacenada. La representación será parecida a:

```
1 -> 23456 3 1
2 -> 15682 0 2
...
```

que quiere decir, en la jugada 1, el número intentado fue el 23456 que tuvo 3 muertos y 1 herido, etc.

4) (*3ptos.*) Crear una aplicación **PruebaMM** que tome un argumento que será la longitud de los números que va utilizar en el juego del MasterMind.

La aplicación creará un objeto **MasterMP** y permitirá al usuario que intente adivinar el número a través del teclado proporcionando el símbolo de invitación:
`MasterMind>`

En cada momento, el usuario podrá dar uno de entre tres comandos:

- Dar un número para intentar adivinar : `I x...x`
- Pedir que muestre todo lo que lleva almacenado hasta ahora: `M`
- Rendirse: `R`
- Cualquier otro comando será ignorado.

Un ejemplo de diálogo con el usuario podría ser:

```
C:\java>java PruebaMM 4

MasterMind> I 2345
1 -> 2345 0 2
MasterMind> I 2344
2344 no es un número válido
MasterMind> I 1643
2 -> 1643 2 1
MasterMind> I 2345
2345 no es un número válido
MasterMind> M
1 -> 2345 0 2
2 -> 1643 2 1
MasterMind> I 1642
3 -> 1642 2 2
MasterMind> R
El número era el 1624
```

NOTA.- En la cuarta jugada, 2345 no es válido pues ya se intentó antes.
Si el usuario acierta el número secreto, el programa termina.

(*1pto.*) Se obtendrá por la claridad, documentación y correcta elección de las estructuras, tanto de control como de los datos.

PROGRAMACION ORIENTADA A OBJETOS

NOTAS PARA LA REALIZACION DEL EXAMEN

(Leer detenidamente antes del examen)

1. Limpiar inicialmente el contenido del directorio **C:\JAVA**. Si no está creado, crearlo (Desde una ventana de MSDOS).
2. Realizar todas las clases y ficheros necesarios en el directorio **C:\JAVA**
3. Todos los ficheros con extensión .java deben estar identificados con Nombre, Curso y Máquina

Al finalizar el examen, el profesor correspondiente pasará por cada puesto. Para cuando lo haga, el directorio **C:\JAVA** deberá contener **EXCLUSIVAMENTE:**

Las clases: `Respuesta.java` `MasterM.java` `MasterMP.java` `PruebaMM.java`

SOLO ESTO será salvado por el profesor. Revisar que los nombres coinciden. Cualquier otro fichero será borrado.
Además, recogerá la hoja del examen debidamente cumplimentada. Es obligatorio entregar la hoja, aún cuando no se entregue el ejercicio práctico.

- ✍✍ **TODAS LAS PREGUNTAS INDICAN LOS PUNTOS QUE VALEN**
- ✍✍ **HAY 1pto QUE SE CONCEDE POR PRESENTACIÓN, CLARIDAD, ETC.**
- ✍✍ **PUEDEN CREARSE OTROS MÉTODOS ADEMÁS DE LOS PEDIDOS, SI SE CONSIDERA NECESARIO.**

PODEIS:

- ?? Consultar la página **http://192.168.198.3/~pacog/index.html** en donde se encuentran los apuntes.
- ?? Consultar el API que se encuentra en c:\jdk1.3\docs\api\index.html.
- ?? Consultar el tutorial que se encuentra en c:\jdk1.3\tutorial\index.html

NO PODEIS:

- ?? Utilizar otra documentación electrónica o impresa.
- ?? Intercambiar documentación entre vosotros.
- ?? Utilizar soportes magnéticos.
- ?? Copiar vuestro ejercicio al final del examen. Si alguien desea una copia de su ejercicio, que se pase posteriormente por el despacho del profesor a retirarla.

IMPORTANTE: APAGAD LOS DISPOSITIVOS ELECTRÓNICOS DE COMUNICACIÓN

```
public class Respuesta {
     private int m, h;
     private String n;
     public Respuesta(String on, int om, int oh) {
          n = on;
          m = om;
          h = oh;
     }
     public String getCadena() {
          return n;
     }
     public int getX() {
          return m;
     }
     public int getY() {
          return h;
     }
     public String toString() {
          return "Res("+n+":"+m+","+h+")";
     }
     public boolean equals(Object o) {
          Respuesta r = (Respuesta)o;
          return n.equals(r.getCadena());
     }
}
```

```
import java.util.*;
public class MasterM {
 static private Random r = new Random();
 static final protected String pozo = "0123456789";
 protected static final int LON=4;
 protected String numSel;
 protected int longitud;
 protected int numJugadas;



 public boolean valido(String n) {
      if (n.length()!=longitud)
           return false;
      for(int i=0;i<longitud;i++) {
           if (pozo.indexOf(n.charAt(i)) < 0)
                return false;
           if (i != n.lastIndexOf(n.charAt(i)))
                return false;
      }
      return true;
 }

 public MasterM() {
      this(LON);
 }

 public MasterM(int t) {
      longitud = t;
      inicializa();
 }

 protected void inicializa(){
```

```
     int i=longitud;
     numSel="";
     while (i>0) {
          char n = pozo.charAt(r.nextInt(10));
          if (numSel.indexOf(n) < 0) {
               numSel+=n;
               i--;
          }
     }
     numJugadas = 0;
}

public String numeroSeleccionado() {
     return numSel;
}

protected Respuesta intentoValido(String numPen){
     int muertos=0;
     int heridos=0;
     for(int i=0;i<longitud;i++) {
          if (numSel.charAt(i)==numPen.charAt(i))
               muertos++;
          else if (numSel.indexOf(numPen.charAt(i))>=0)
               heridos++;
     }
     return new Respuesta(numPen,muertos,heridos);
}

public Respuesta intento(String numPen) {
     Respuesta res=null;
     if (valido(numPen)) {
          res = intentoValido(numPen);
          numJugadas++;
```

```
        }
        return res;
    }

    public int numJugadas() {
              return numJugadas;
    }

    public boolean esFinal(String numPen) {
        return numSel.equals(numPen);
    }
}
```

```
import java.util.*;
import java.io.*;
public class MasterMP extends MasterM {
 protected List lr;

 public MasterMP(int n) {
      super(n);
      lr = new ArrayList();
 }

 public MasterMP() {
      super();
      lr = new ArrayList();
 }

 public Respuesta intento(String numPen) {
      Respuesta res = super.intento(numPen);
      if (res!=null)
           if (lr.contains(res)){
                res=null;
                numJugadas--;
           }
           else
                lr.add(res);
      return res;
 }

 public void muestra(PrintWriter pw) {
      Iterator it = lr.iterator();
      int i=1;
      while (it.hasNext()) {
           Respuesta r = (Respuesta)it.next();
           pw.println(i+" -> "+r.getCadena()+" "+r.getX()+" "+r.getY());
```

```
            i++;
        }
    }
}
```

```
import java.io.*;
import java.util.*;
class PruebaMM {
 private static final String MERINDO="R";
 private static final String INTENTO="I";
 private static final String MUESTRA="M";
 static public void main(String [] args) {
      int lon=0;
      MasterMP m;
      Respuesta r;
      String c,c1,c2;
      boolean seguir;
      BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
      PrintWriter pw = new PrintWriter(System.out, true);
      try {
           lon = Integer.parseInt(args[0]);
      } catch (ArrayIndexOutOfBoundsException e) {
           System.err.println("Uso: PruebaMM longitud");
           System.exit(0);
      }
      m = new MasterMP(lon);

      seguir = true;
      while (seguir) {
           c="";
           try {
                System.out.print("MasterMind> ");
                c = br.readLine();
                while (c==null) {
                     System.out.print("MasterMind> ");
                     c= br.readLine();
                }
           } catch (IOException e) {
```

```
                    System.out.println("Error de E/S");
                    System.exit(0);
               }
               StringTokenizer st = new StringTokenizer(c," ");
               if (st.hasMoreTokens()) {
                    c1 = st.nextToken();
                    if (c1.equalsIgnoreCase(INTENTO)) {
                         if (st.hasMoreTokens()) {
                                   c2=st.nextToken();
                                   r=m.intento(c2);
                                   if (r==null)
                                        System.out.println(c2+" -> Número no válido");
                                   else {
                                        System.out.println(m.numJugadas()+" -> "+c2+" "+r.getX()+"
    "+r.getY());
                                        seguir = !m.esFinal(c2);
                                   }
                         }
                    }    else if (c1.equalsIgnoreCase(MUESTRA))
                         m.muestra(pw);
                    else if (c1.equalsIgnoreCase(MERINDO)) {
                         System.out.println("El número era el "+m.numeroSeleccionado());
                         seguir = false;
                    }
               }
          }
     }
}
```