Examen parcial – Convocatoria de junio de 2005
FUNDAMENTOS DE LA PROGRAMACIÓN

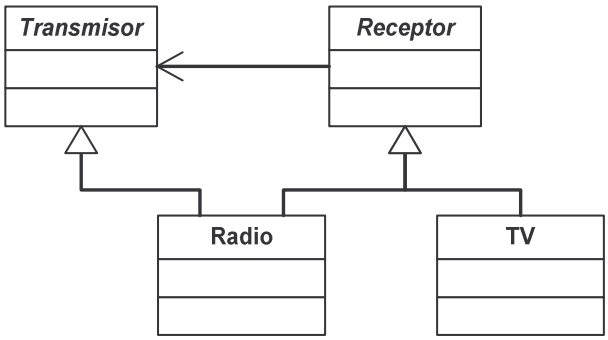
EJERCICIO 1 (1.5 puntos)

Diseñe un conjunto de casos de prueba adecuado para comprobar el funcionamiento de un método encargado de calcular la duración de una llamada telefónica.

Entradas Hora de inicio	Hora de finalización	**Salida** Duración de la llamada	**Comentarios** Justificación del caso de uso
09:55:04	09:55:55	00:00:51	Caso base
10:37:34	10:48:24	00:10:50	Cambio de minuto
19:15:32	20:32:40	01:17:08	Cambio de hora
23:55:30 22/6/05	00:10:30 23/6/05	00:10:00	Cambio de día
00:00:00	00:00:00	00:00:00	Llamada “nula”
10:00:00	09:00:00	Error	Finalización antes del comienzo de la llamada
25:64:98	15:30:45	Error	Hora de inicio incorrecta
15:20:45	25:64:98	Error	Hora final incorrecta
12:54:76	12:76:43	Error	Entradas incorrectas
…	…	…	

EJERCICIO 2 (2.5 puntos)

Declare adecuadamente las clases en Java que se derivan del siguiente diagrama de clases UML:




```
public interface Transmisor
{
  // ...
}
```

```
public abstract class Receptor
{
  Transmisor transmisor;

  // ...
}
```

```
public class Radio extends Receptor implements Transmisor
{
  // ...
}
```

```
public class TV extends Receptor
{
  // ...
}
```

Examen parcial – Convocatoria de junio de 2005
FUNDAMENTOS DE LA PROGRAMACIÓN

EJERCICIO 3 (3 puntos)

Diseñe e implemente un programa en Java capaz de leer una serie de datos almacenada en un fichero de texto y mostrar los dos valores más altos incluidos en el fichero. Por ejemplo, dado un fichero con el conjunto de datos `(1, 2, 4, 8, 1, 3, 6, 9, 0, 7)`, el programa debe indicar que los dos valores más altos son el `9` y el `8`.

NOTA: Cada línea del fichero de texto contiene un único valor.

Solución A: Implementación modular

```
import java.io.*;


// Programa principal

public class DosMayoresModular
{
  public static void main (String[] args)
         throws IOException
  {
    Serie serie;      // Serie de datos
    int   mayores[];  // Valores más altos

    if (args.length<1) {
       System.err.println("USO:");
       System.err.println();
       System.err.println(" java DosMayores <fichero>");
       System.err.println();
       System.err.println("donde <fichero> es el nombre del fichero”
                          + ” que contiene la serie de datos");
       System.exit(-1);
    }

    serie = new Serie(args[0]);

    mayores = serie.mayores();


    if (mayores!=null) {

       if (mayores.length==2) {
          System.out.println("Los valores más altos del fichero son "
                             + mayores[0] + " y " + mayores[1]);
       } else {
          System.out.println("El fichero sólo contiene el valor "
                             + mayores[0]);
       }

    } else {
       System.out.println("El fichero no contiene valores numéricos");
    }

  }
}
```

Examen parcial – Convocatoria de junio de 2005
FUNDAMENTOS DE LA PROGRAMACIÓN

```
// Serie de datos almacenada en un fichero

public class Serie
{
  String fichero;

  public Serie (String fichero)
  {
    this.fichero = fichero;
  }

  public int[] mayores ()
         throws IOException
  {
    FileReader     file;   // Lector asociado al fichero
    BufferedReader lector;    // Lectura línea a línea
    String         line;      // Línea del fichero
    int            total;     // Número de valores del fichero
    int            valor;     // Valor leído del fichero
    int            max1;      // Mayor valor
    int            max2;      // Segundo valor mayor

    file    = new FileReader(fichero);
    lector  = new BufferedReader(file);

    total = 0;
    max1 = Integer.MIN_VALUE;
    max2 = Integer.MIN_VALUE;

    line = lector.readLine();

    while (line!=null) {
          try {
              valor = Integer.parseInt(line);
              total++;
              if (valor>max1) {
                 max2 = max1;   // El antiguo máximo
                 max1 = valor;  // pasa a ser el segundo mayor valor
              } else if (valor>max2) {
                 max2 = valor;
              }
          } catch (Exception error) {
              // Pasar a la siguiente línea (número incorrecto)
          }
          line = lector.readLine();
    }

    if (total>2) {
       return new int[] { max1, max2 };
    } else if (total==1) {
       return new int[] { max1 };
    } else {
       return null;
    }

  }
}
```

Examen parcial – Convocatoria de junio de 2005
FUNDAMENTOS DE LA PROGRAMACIÓN

Solución B: Implementación monolítica

```
import java.io.*;

public class DosMayores
{
  public static void main (String[] args)
         throws IOException
  {
    FileReader     fichero;   // Lector asociado al fichero
    BufferedReader lector;    // Lectura línea a línea
    String         line;      // Línea del fichero
    int            total;     // Número de valores del fichero
    int            valor;     // Valor leído del fichero
    int            max1;      // Mayor valor
    int            max2;      // Segundo valor mayor

    fichero = new FileReader(args[0]);
    lector  = new BufferedReader(fichero);

    total = 0;
    max1 = Integer.MIN_VALUE;
    max2 = Integer.MIN_VALUE;

    line = lector.readLine();

    while (line!=null) {
          try {
              valor = Integer.parseInt(line);
              total++;
              if (valor>max1) {
                 max2 = max1;   // NOTA: El antiguo máximo
                 max1 = valor;  // pasa a ser el segundo mayor valor
              } else if (valor>max2) {
                 max2 = valor;
              }
          } catch (Exception error) {
              // Pasar a la siguiente línea
          }
          line = lector.readLine();
    }


    if (total>2) {
       System.out.println("Los valores más altos del fichero son "
                          +max1+" y "+max2);
    } else if (total==1) {
       System.out.println("El fichero sólo contiene el valor "+max1);
    } else {
       System.out.println("El fichero no contiene valores numéricos");
    }

  }
}
```

Examen parcial – Convocatoria de junio de 2005
FUNDAMENTOS DE LA PROGRAMACIÓN

EJERCICIO 4 (3 puntos)

Implemente un **servidor TCP** que nos permita acceder de forma remota a los datos medidos por un `Sensor`. El servidor recibirá peticiones a través del puerto 2206.

El servidor deberá atender **cada petición en una hebra independiente**, de forma que varios clientes puedan consultar simultáneamente los datos medidos por el sensor.

Otros miembros de nuestro equipo de trabajo ya se han encargado de diseñar e implementar el protocolo de comunicación necesario, por lo que nosotros sólo tenemos que delegar en el método `atenderCliente` cada vez que alguien se conecte a nuestro servidor:

```
public class ProtocoloSensor
{
  public void atenderCliente (InputStream entrada, OutputStream salida)…
}
```

donde `entrada` corresponde al *stream* mediante el que el servidor recibe datos y `salida` representa el *stream* utilizado para enviar datos desde el servidor hasta el cliente.

```
public class Servidor
{
  public static void main (String args[]) throws IOException
  {
    ServerSocket  servidor;
    Socket        cliente;
    HebraServidor hebra;
    servidor = new ServerSocket(2206);
    while (true) {                  // ¡OJO! Bucle infinito
      cliente = servidor.accept();
      hebra = new HebraServidor(cliente);
      hebra.start();
    }
  }
}

class HebraServidor extends Thread
{
  Socket socket;
  public HebraServidor (Socket socket)
  {
    this.socket = socket;
  }
  public void run()
  {
    ProtocoloSensor protocolo = new ProtocoloSensor();
    try {
       protocolo.atenderCliente ( socket.getInputStream(),
                                  socket.getOutputStream());
    } catch (IOException error) {
    }
  }
}
```