



NOMBRE:  
APELLIDOS:  
NIA:  
GRUPO:

## 2ª Parte: Problemas (5 puntos sobre 10)

Duración: 120 minutos  
Puntuación máxima: 5 puntos  
Fecha: 3 de Junio de 2013

- No se permite el uso de libros o apuntes, ni tener teléfonos móviles u otros dispositivos electrónicos encendidos. Incumplir alguna de estas normas puede ser motivo de expulsión inmediata del examen
- Rellena tus datos personales antes de comenzar a realizar el examen
- Utiliza el espacio de los recuadros en blanco para responder a los apartados de los problemas

### PROBLEMA 1 (2,5 puntos)

En un sistema de mensajería se define la clase *Mensaje* de la siguiente forma (los puntos suspensivos indican que se dispone de dicho código ya programado, pero no se muestra por brevedad):

```
public abstract class Message {
    private String text;
    private Person from;
    private Person to;
    private int priority;

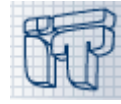
    public Message(String text, Person from, Person to, int priority) {...}
    public String getText() {...}
    public Person getFrom() {...}
    public Person getTo() {...}
    public int getPriority() {...}
    public void dispatch() {...} // envía el mensaje
    public void archive() {...} // archiva el mensaje
    public abstract String format(); // da formato al mensaje como cadena
}
```

Esta clase tiene exclusivamente los métodos y atributos mostrados. No se permite añadir ni modificar nada en ella.

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE  
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

---

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
CALL OR WHATSAPP:689 45 44 70



### Apartado 1 (0,75 puntos)

Programa la clase *Answer*, con los métodos y atributos que sean necesarios. Esta clase hereda de *Message* y representa una respuesta a un mensaje. Ten en cuenta que:

- El constructor debe recibir un texto de respuesta y el objeto de la clase *Message* al cual se responde. Los atributos *from* y *to* se rellenan respectivamente desde *to* y *from* de este último. La prioridad del mensaje de respuesta debe ser igual a la del mensaje al cual se responde.
- El método *format* debe devolver la concatenación de: (1) entre corchetes, lo que devuelva el método *format* del mensaje al cual se responde; (2) un carácter de espacio en blanco; (3) el texto de este mensaje.

```
public class Answer extends Message {  
  
    private Message original;  
  
    public Answer(String text, Message original) {  
        super(text, original.getTo(), original.getFrom(), original.getPriority());  
        this.original = original;  
    }  
  
    public String format() {  
        return "[" + original.format() + "] " + getText();  
    }  
}
```

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE  
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

---

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
CALL OR WHATSAPP:689 45 44 70

Cartagena99



## Apartado 2 (1 punto)

Programa la clase *Question*, con los métodos y atributos que sean necesarios. Esta clase hereda de *Message* y representa un mensaje en que se realiza una pregunta. Ten en cuenta que:

- Debe proporcionar un constructor que reciba los mismos parámetros que el de *Message*.
- El método *format* simplemente devuelve el texto del mensaje.
- No se permite archivar un mensaje *Question* sin responder al mismo. Por tanto, debes sobrescribir el método *archive* para que en la clase *Question* no haga nada.
- Tiene un método *Answer reply(String text)* que crea y devuelve un mensaje de respuesta al mensaje actual con el texto dado. Este método debe también archivar el mensaje actual (el mensaje de pregunta) invocando al método *archive* heredado de la clase *Message* (esto es, no debes usar la redefinición del método en la clase *Question*, sino el programado en *Message*).

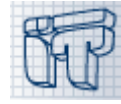
```
public class Question extends Message {  
  
    public Question(String text, Person from, Person to, int priority) {  
        super(text, from, to, priority);  
    }  
  
    public void archive() {  
        // questions cannot be archived without answering them  
    }  
  
    public Answer reply(String text) {  
        Answer answer = new Answer(text, this);  
        super.archive();  
        return answer;  
    }  
  
    public String format() {  
        return getText();  
    }  
}
```

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE  
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

---

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
CALL OR WHATSAPP:689 45 44 70

Cartagena99



### Apartado 3 (0,75 puntos)

Se desea programar una interfaz gráfica de usuario muy simple que, dado un mensaje, recoja un texto del usuario y cree un mensaje de respuesta. La interfaz muestra un cuadro de texto y dos botones (etiquetados como “Aceptar” y “Cancelar”). Cuando se presione el botón de aceptar, se debe crear un mensaje de respuesta con el texto del cuadro de texto y enviarlo. Además, sea cual sea el botón presionado, se debe también borrar el texto del cuadro de texto. La interfaz se programa en una única clase que ejerce de ventana y además es su propio escuchador para los eventos de los botones. Rellena los huecos subrayados en el código de la clase:

```
public class GraphicalInterface extends _____ JFrame _____  
                                     implements _____ ActionListener _____ {  
    private Message original;  
    private JTextArea textArea;  
  
    public GraphicalInterface(Message original) {  
        this.original = original;  
  
        getContentPane()._____setLayout_____ (new FlowLayout());  
        JButton acceptButton = new JButton("Aceptar");  
        JButton cancelButton = new JButton("Cancelar");  
        textArea = new JTextArea(10, 40);  
  
        getContentPane().add(_____ textArea _____);  
        getContentPane().add(_____ acceptButton _____);  
        getContentPane().add(_____ cancelButton _____);  
        acceptButton.addActionListener(_____ this _____);  
        cancelButton.addActionListener(_____ this _____);  
  
        pack();  
        setVisible(_____ true _____);  
    }  
  
    public _____ void actionPerformed(ActionEvent e) _____ {  
        String text = textArea.getText();  
        textArea.setText("");  
        if (e.getActionCommand().equals("Aceptar")) {
```

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE  
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

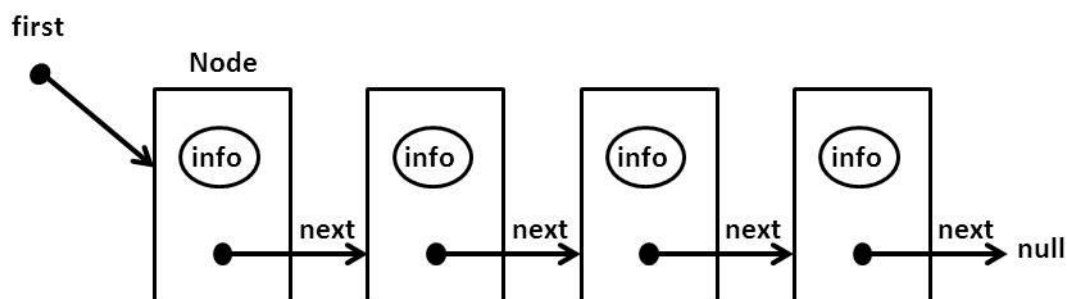
---

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
CALL OR WHATSAPP:689 45 44 70



## PROBLEMA 2 (2,5 puntos)

Se pretende crear una *cola de datos* (Queue) mediante el uso de una *Lista Enlazada* (LinkedList) como la que se muestra en la siguiente figura:



```
public class Node {
    public Object info;
    public Node next;

    public Node() {...}
    public Node(Object info, Node next) {...}
}

public class LinkedList {
    public Node first;
    public int size = 0;

    public LinkedList () {...}
}
```

Se pide (los apartados está en las siguientes páginas):

*Nota: en el examen se comunicó a los alumnos que se puede asumir también que Node tiene métodos get y set para acceder a sus atributos.*

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE  
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

---

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
CALL OR WHATSAPP:689 45 44 70



### Apartado 1 (0,75 puntos)

Implementa el siguiente método de la clase `LinkedList` que permite insertar un dato (`data`) en la lista enlazada *justo en la posición posterior a* `position` y actualiza el tamaño (`size`) de la lista:

- `public void insertAt (Object data, int position)`

**NOTAS:** La numeración de las posiciones de la lista comienza en 1. Si `position` es menor o igual que 0 entonces el método inserta al principio de la lista. Si `position` es mayor o igual que `n` entonces el método inserta al final de la lista.

```
public void insertAt(Object data, int position) {
    if ((first == null) || position <= 0) {
        Node tmp = new Node(data, first);
        first = tmp;
    } else {
        Node aux = first;
        for (int i = 1; i < position && aux.getNext() != null; i++) {
            aux = aux.getNext();
        }
        Node tmp = new Node(data, aux.getNext());
        aux.setNext(tmp);
    }
    size++;
}
```

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE  
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

---

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
CALL OR WHATSAPP:689 45 44 70

Cartagena99



## Apartado 2 ( 0,75 puntos)

Modifica la declaración de la clase `LinkedList` para que implemente la interfaz `Queue` que se define a continuación.

```
public interface Queue {  
    public void enqueue(Object o);  
    public boolean isEmpty();  
    public int size();  
}
```

Debes añadir código a los tres métodos de la interfaz.

**NOTAS:** Se ha eliminado el método `dequeue` de la interfaz para simplificar el problema. Para la implementación del método `enqueue` puedes utilizar el método `insertAt` del apartado anterior independientemente de que lo hayas implementado o no en dicho apartado.

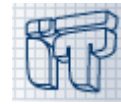
```
public class LinkedList implements Queue {  
  
    (...)  
  
    public void enqueue(Object info) {  
        insertAt(info, size);  
    }  
  
    public boolean isEmpty(){  
        return (first == null);  
    }  
  
    public int size() {  
        return size;  
    }  
}
```

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE  
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

---

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
CALL OR WHATSAPP:689 45 44 70

Cartagena99



### Apartado 3 (1 punto)

Implementa el método `public Queue invert()` de la clase `LinkedList` que permita devolver una nueva instancia de una *cola* que contenga una copia de los elementos de la lista enlazada colocados *en orden inverso*.

**NOTAS:** Para hacerlo utiliza la siguiente pila auxiliar de datos suponiendo que sus métodos están ya implementados

<b>Constructor Summary</b>	
<u>Stack</u> ()	Creates an empty Stack.
<b>Method Summary</b>	
boolean	<u>empty</u> () Tests if this stack is empty.
Object	<u>pop</u> () Removes the object at the top of this stack and returns that object as the value of this function.
Object	<u>push</u> (Object item) Pushes an item onto the top of this stack.

```
public Queue invert() {
    Stack stack = new Stack();
    Queue queue = new LinkedList();
    Node aux = first;
    while (aux != null) {
        stack.push(aux.getInfo());
        aux = aux.getNext();
    }
    while (!stack.isEmpty()) {
        queue.enqueue(stack.pop());
    }
    return queue;
}
```

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE  
 LLAMA O ENVÍA WHATSAPP: 689 45 44 70

---

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
 CALL OR WHATSAPP:689 45 44 70

