Nombre:..... ***Legajo:.....***

Recuperatorio Segundo Parcial de Programación Orientada a Objetos

6 de Diciembre de 2010

Ej. 1	*Ej. 2*	*Ej. 3*	*Nota*

- **Condición de aprobación: Tener dos ejercicios calificados como B o B-.**
- **Los ejercicios que no se ajusten estrictamente al enunciado, no serán aceptados.**
- **No es necesario agregar las sentencias import.**
- **Además de las clases pedidas se pueden agregar las que se consideren necesarias.**

Ejercicio 1

Se cuenta con un conjunto de clases que modelan el proceso de envío de mensajes de texto. La clase **PhoneCentral** es la central que administra los teléfonos y los envíos. La clase **CellPhone** modela al teléfono y la clase **Message** al mensaje.

```
public class Message {

	private String from;
	private String to;
	private String text;

	public Message(String from, String to, String text) {
		this.from = from;
		this.to = to;
		this.text = text;
	}
	public String getFrom() {
		return from;
	}
	public String getTo() {
		return to;
	}
	public String getText() {
		return text;
	}
}
```

```
public class CellPhone {

	private String number;
	private PhoneCentral central;

	public CellPhone(String number) {
		this.number = number;
	}
	public void receiveMessage(Message message) {
		System.out.println("FROM: " + message.getFrom() +
				" TO: " + message.getTo() + " MSG: " + message.getText());
	}
	public void sendMessage(String to, String text) {
		central.routeMessage(new Message(number, to, text));
	}
	public String getNumber() {
		return number;
	}
	void setCentral(PhoneCentral central) {
		this.central = central;
	}
}
```

```
public class PhoneCentral {

	private Map<String, CellPhone> phones = new HashMap<String, CellPhone>();

	public void addPhone(CellPhone phone) {
		phones.put(phone.getNumber(), phone);
		phone.setCentral(this);
	}
```

```
	public void routeMessage(Message message) {
		CellPhone phone = phones.get(message.getTo());
		if (phone != null) {
			phone.receiveMessage(message);
		} else {
			throw new UnknownRecipientException();
		}
	}
}
```

Se pide implementar la clase **ContestPhone** que modela un teléfono especial utilizado para votaciones de concursos. Dicho teléfono debe acumular los votos que recibe a través de mensajes. Los votos son válidos sólo si el mensaje coincide con alguna de las opciones de la votación y además la votación está abierta.

```
public class TestPhones {

	public static void main(String[] args) {

		PhoneCentral central = new PhoneCentral();

		CellPhone[] phones = new CellPhone[4];
		for (int i=0; i< 4; i++) {
			phones[i] = new CellPhone("111" + i);
			central.addPhone(phones[i]);
		}

		List<String> options = new ArrayList<String>();
		options.add("A");
		options.add("B");
		options.add("C");
		ContestPhone contestPhone = new ContestPhone("*2020", options);
		central.addPhone(contestPhone);

		phones[0].sendMessage("1111", "Hola");
		phones[1].sendMessage("1110", "Que tal?");

		phones[0].sendMessage("*2020", "B");
		phones[1].sendMessage("*2020", "A");
		phones[2].sendMessage("*2020", "A");
		phones[3].sendMessage("*2020", "hola");

		contestPhone.closeContest();
		contestPhone.printResults();

		phones[0].sendMessage("*2020", "B");
	}
}
```

La siguiente es la salida de la ejecución del programa anterior:

```
FROM: 1110 TO: 1111 MSG: Hola
FROM: 1111 TO: 1110 MSG: Que tal?
FROM: 1110 TO: *2020 MSG: B
FROM: *2020 TO: 1110 MSG: Su voto fue registrado.
FROM: 1111 TO: *2020 MSG: A
FROM: *2020 TO: 1111 MSG: Su voto fue registrado.
FROM: 1112 TO: *2020 MSG: A
FROM: *2020 TO: 1112 MSG: Su voto fue registrado.
FROM: 1113 TO: *2020 MSG: hola
FROM: *2020 TO: 1113 MSG: Opción inválida.
Resultados:
A: 2  |  B: 1  |  C: 0  |
FROM: 1110 TO: *2020 MSG: B
FROM: *2020 TO: 1110 MSG: La votación ya está cerrada.
```

Ejercicio 2

HTML es un lenguaje utilizado en el desarrollo de páginas web. Permite describir la estructura de un documento, así como darle formato. Para hacer esto, utiliza "etiquetas" que permiten indicar el formato a aplicar. A continuación se describen algunas de estas etiquetas:

- **b**: Permite definir un texto en negrita. Ejemplo: `<b>hola</b>`
- **i**: Permite definir un texto en cursiva. Ejemplo: `<i>hola</i>`
- **a**: Permite definir un link, agregando un atributo que indica la página a la cual se desea ir al hacer clic en el link. Ejemplo: `<a href="www.google.com">Ir a la pagina de Google</a>`

Estas etiquetas pueden anidarse para combinar distintos formatos. Por ejemplo, el siguiente código muestra el texto "hola" en negrita y en cursiva: `<b><i>hola</i></b>`. El siguiente código hace lo mismo: `<i><b>hola</b></i>`.

Se cuenta con una interfaz que representa un texto HTML. Dicha interfaz provee un método para obtener el código fuente:

```
public interface HTMLText {
	public String getSource();
}
```

Se tiene además una implementación para textos sin formato (en los cuales el código fuente coincide con el texto a mostrar, ya que no se aplica ninguna etiqueta).

```
public class PlainHTMLText implements HTMLText {

	private String text;

	public PlainHTMLText(String text) {
		this.text = text;
	}
	public void setText(String text) {
		this.text = text;
	}
	@Override
	public String getSource() {
		return text;
	}
}
```

Se quiere ofrecer más funcionalidad para permitir representar textos en negrita, en cursiva y links. Implementar todo lo necesario para que el siguiente ejemplo imprima la salida indicada. Completar además las sentencias indicadas con "..." en el código de ejemplo. **No realizar ninguna otra modifiación en el código de ejemplo.**

```
public class Test {
	public static void main(String[] args) {

		PlainHTMLText text = new PlainHTMLText("Hola");

		HTMLText boldText = ...
		HTMLText italicText = ...

		System.out.println(boldText.getSource());
		System.out.println(italicText.getSource());

		HTMLText boldItalicText = ...

		System.out.println(boldItalicText.getSource());

		text.setText("ITBA");

		System.out.println(boldText.getSource());
		System.out.println(italicText.getSource());
		System.out.println(boldItalicText.getSource());

		HTMLText linkText = ...
		HTMLText linkBoldText1 = ...
		HTMLText linkBoldText2 = ...

		System.out.println(linkText.getSource());
		System.out.println(linkBoldText1.getSource());
		System.out.println(linkBoldText2.getSource());

		text.setText("Ejemplo");

		System.out.println(linkBoldText1.getSource());
		System.out.println(linkBoldText2.getSource());
	}
}
```

La salida esperada es:

```
<b>Hola</b>
<i>Hola</i>
<b><i>Hola</i></b>
<b>ITBA</b>
<i>ITBA</i>
<b><i>ITBA</i></b>
<a href="www.itba.edu.ar">ITBA</a>
<a href="www.itba.edu.ar"><b><i>ITBA</i></b></a>
<b><a href="www.itba.edu.ar">ITBA</a></b>
<a href="www.itba.edu.ar"><b><i>Ejemplo</i></b></a>
<b><a href="www.itba.edu.ar">Ejemplo</a></b>
```

Ejercicio 3

Se cuenta con la siguiente interface de mapa que soporta la operación **`undo()`**.

```
/**
 * Mapa Simple que no admite claves y valores en null, y permite deshacer operaciones
 */
public interface UndoMap<K,V> {

	/**
	 * Agrega un par clave-valor al mapa. Si la clave ya existe, sobreescribe el valor.
	 */
	public void put(K key, V value);

	/**
	 * Devuelve el valor asociado a la clave key. Si la clave no existe retorna null.
	 */
	public V get(K key);

	/**
	 * Elimina el par clave-valor. Si la clave no existe, no hace nada.
	 */
	public void remove(K key);

	/**
	 * Deshace la última modificación realizada al mapa (a través de los métodos put
	 * o remove). Este método puede ser invocado tantas veces como se desee, deshaciendo
	 * tantas operaciones como corresponda.
	 * @throws IllegalStateException cuando no hay una operación para deshacer.
	 */
	public void undo();
}
```

Se pide hacer una implementación de la misma. Considerar el siguiente programa de ejemplo:

```
public class UndoMapTest {

	public static void main(String[] args) {

		UndoMap<String, String> map = new HashUndoMap<String, String>();

		map.put("1", "hola");
		map.put("1", "que paso?");
		map.put("34", "mundo");
		map.put("7", "chau");

		System.out.println(map.get("34"));   //Imprime "mundo"

		map.undo();
		map.undo();

		System.out.println(map.get("34"));   //Imprime "null"
		System.out.println(map.get("1"));    //Imprime "que paso?"

		map.undo();

		System.out.println(map.get("1"));    //Imprime "hola"

		map.put("999", "fin");
		map.remove("1");

		System.out.println(map.get("1"));    //Imprime "null"

		map.undo();

		System.out.println(map.get("1"));    //Imprime "hola"

		map.undo();
		map.undo();
		System.out.println(map.get("1"));    //Imprime "null"
		map.undo();                          // Lanza IllegalStateException
	}
}
```