Programación Orientada a Objetos en Java
Tareas del Bloque 2

Rubén Heradio
rheradio@issi.uned.es
ETSI Informática, Universidad Nacional de Educación a Distancia

Índice

1. Instrucciones para la entrega de los ejercicios

Este bloque se compone de 8 ejercicios. La solución a todos ellos se incluirá en un fichero comprimido ".zip"[1], que contendrá una carpeta por cada ejercicio.Las carpetas se llamarán `Ejercicio1`, `Ejercicio2`, ..., `Ejercicio8`, respectivamente. Cada carpeta incluirá el código fuente correspondiente, es decir, los archivos ".java" que solucionen el problema planteado.

2. Ejercicio 1: Análisis de un String

Realizar un programa que analice un String y contabilice los siguientes aspectos:

- Número total de caracteres.
- Número total de vocales utilizadas.
- Total de veces utilizada la vocal "a" mayúscula o minúscula.
- Total de veces utilizada la vocal "e" mayúscula o minúscula.
- Total de veces utilizada la vocal "i" mayúscula o minúscula.
- Total de veces utilizada la vocal "o" mayúscula o minúscula.
- Total de veces utilizada la vocal "u" mayúscula o minúscula.

3. Ejercicio 2: media y desviación típica

Escribir un programa que rellene un array de 30 `doubles` con números aleatorios y luego calcule la media y la desviación típica, utilizando las siguientes ecuaciones:

$$\text{media} = \sum_{i=0}^{29} \frac{x_i}{30}$$

$$\text{desviación} = \sqrt{\sum_{i=0}^{29} \frac{(x_i - \text{media})^2}{30}}$$

> ☞ ***Pista***: La función `Math.random()` devuelve un número aleatorio entre 0 y 1.

[1] `http://es.wikipedia.org/wiki/Formato_de_compresi%C3%B3n_ZIP`

4. Ejercicio 3: Producto de matrices

Realizar un programa capaz de multiplicar dos matrices[2] de tamaño variable introducidas por el usuario desde teclado. Por ejemplo:

```
Introduzca la primera matriz:
    Numero de filas: 3
    Numero de columnas: 2
    Elemento[0][0]: 0
    Elemento[0][1]: 1
    Elemento[1][0]: 2
    Elemento[1][1]: 3
    Elemento[2][0]: 4
    Elemento[2][1]: 5

Introduzca la segunda matriz:
    Numero de filas: 2
    Numero de columnas: 2
    Elemento[0][0]: 6
    Elemento[0][1]: 7
    Elemento[1][0]: 8
    Elemento[1][1]: 9

La matriz resultado es:
    8  9
   36 41
   64 73
```

El programa se asegurará de que, para que sea posible la realización del producto de dos matrices, el número de columnas de la primera matriz coincida con el número de filas de la segunda matriz.

5. Ejercicio 4: Cajero automático

Realizar un programa que simule un cajero automático de monedas. Los tipos de monedas que dispone el cajero son de 10, 20 y 50 céntimos de euro y 1 y 2 euros. Inicialmente el cajero tiene 100 monedas de cada tipo, que se van consumiendo para proporcionar las cantidades solicitadas. El cajero debe obtener la cantidad solicitada con los tipos de moneda que tenga en cada momento, tratando siempre de utilizar las monedas de mayor valor.

6. Ejercicio 5: Canciones

Escribir una clase `Cancion` con los siguientes atributos:

[2] Puede consultar cómo se realiza el producto de matrices en:

http://es.wikipedia.org/wiki/Multiplicaci%C3%B3n_de_matrices

1. `titulo`: una variable String que guarda el título de la canción.
2. `autor`: una variable String que guarda el autor de la canción.

y los siguientes métodos:

1. `Cancion(String, String)`: constructor que recibe como parámetros el título y el autor de la canción (por este orden).
2. `Cancion()`: constructor predeterminado que inicializa el título y el autor a cadenas vacías.
3. `dameTitulo()`: devuelve el título de la canción.
4. `dameAutor()`: devuelve el autor de la canción
5. `ponTitulo(String)`: establece el título de la canción.
6. `ponAutor(String)`: establece el autor de la canción.

7. Ejercicio 6: Conjuntos

Definir una clase `Conjunto`[3] que contenga:

1. Un array de valores de cualquier tipo (✍utilice la clase `Object`).
2. Los siguientes métodos:
   a) `interseccion`: toma como argumento otro conjunto, y devuelve un nuevo conjunto con la intersección de los dos. Por ejemplo,

   $$\{1,2,3,4\}.\text{interseccion}(\{3,1,5\}) \rightarrow \{1,3\}$$

   b) `union`: toma como argumento otro conjunto, y devuelve un nuevo conjunto con la unión de los dos. Por ejemplo,

   $$\{1,2,3,4\}.\text{union}(\{4,5\}) \rightarrow \{1,2,3,4,5\}$$

   c) `diferencia`: toma como argumento otro conjunto, y devuelve un nuevo conjunto con la diferencia de los dos. Por ejemplo,

   $$\{1,2,3,4\}.\text{diferencia}(\{3,1\}) \rightarrow \{2,4\}$$

[3] para más información, consulte `http://es.wikipedia.org/wiki/Teor%C3%ADa_de_conjuntos`

8. Ejercicio 7: Calculadora

Escribir una clase `Calculadora` que realice las funciones típicas de una calculadora. La calculadora tendrá un método que pedirá al usuario tres valores: operación (+,-,*,/), `operando1`, `operando2`; y a partir de ellos mostrará el resultado de la operación.

Cuando el usuario introduzca una `Z` como valor de la operación el programa parará. Si se introduce cualquier otro carácter se debe producir una excepción definida por el usuario imprimiendo un mensaje de error.

9. Ejercicio 8: Simulador del juego "Piedra, Papel o Tijera"

"Piedra, papel o tijera" es un juego infantil donde dos jugadores cuentan juntos *1... 2... 3... ¡Piedra, papel o tijera!* y, justo al acabar, muestran al mismo tiempo una de sus manos, de modo que pueda verse el arma que cada uno ha elegido: Piedra (un puño cerrado), Papel (todos los dedos extendidos, con la palma de la mano mirando hacia abajo) o Tijera (dedos índice y corazón extendidos y separados formando una 'V'). El objetivo del juego es vencer al oponente, seleccionando el arma que gana a la que ha elegido él. La figura 1 resume la precedencia entre las armas:

- La piedra aplasta o rompe a la tijera (gana la piedra)
- La tijera corta al papel (gana la tijera)
- El papel envuelve a la piedra (gana el papel)
- Si los jugadores eligen la misma arma, hay empate

Realizar un programa que simule aleatoriamente 5 partidas del juego, produciendo una salida del estilo de la figura 2.

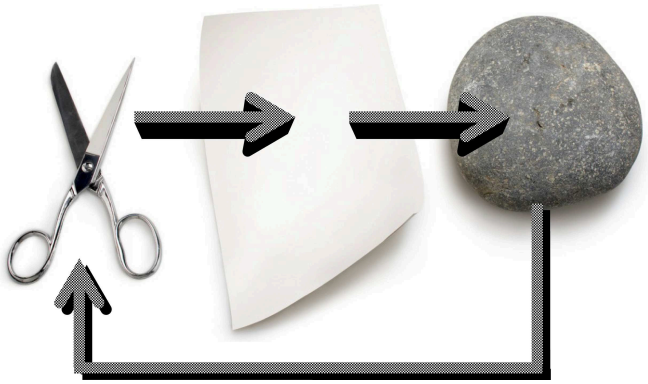

Figura 1: Tijera gana a Papel, Papel gana a Piedra, Piedra gana a Tijera

```
------------------------------
Partida 1:
------------------------------
Jugador 1:
        Tijera
Jugador 2:
        Tijera
Resultado: Empate

------------------------------
Partida 2:
------------------------------
Jugador 1:
        Papel
Jugador 2:
        Papel
Resultado: Empate

------------------------------
Partida 3:
------------------------------
Jugador 1:
        Tijera
Jugador 2:
        Papel
Resultado: Gana Jugador 1

------------------------------
Partida 4:
------------------------------
Jugador 1:
        Piedra
Jugador 2:
        Piedra
Resultado: Empate

------------------------------
Partida 5:
------------------------------
Jugador 1:
        Piedra
Jugador 2:
        Papel
Resultado: Gana Jugador 2
```

Figura 2: Resultado de ejecutar el simulador PiedraPapelTijera

☞ ***Pista***: La Figura 3 representa un posible diseño del simulador, que contiene una clase principal (Simulador) y tres clases auxiliares (Jugada, Jugador y Arbitro). Las flechas indican relaciones de dependencia entre clases. Así, $X \rightarrow Y$ significa que la clase X usa algún elemento de la clase Y.

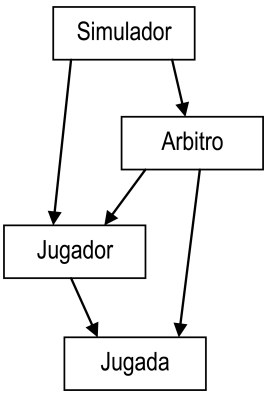

Figura 3: Diseño del programa

A continuación, se resume la funcionalidad de las clases:

1. Clase Jugada: implementa las armas que puede escoger un jugador. Incluye el método generarJugada, que devuelve un valor aleatorio de entre {Piedra, Papel, Tijera}.
2. Clase Jugador: simula el comportamiento de un jugador. Puede elegir un arma (mediante un método jugar, que a su vez utiliza a generarJugada), devolver el arma elegida (método consultarJugada) e imprimirla por pantalla (método imprimirJugada).
3. Clase Arbitro: decide, de entre dos jugadores, cual de ellos ha ganado la partida (o si hay empate).
4. Clase simulador: contiene el método función principal (main), que controla la ejecución de los dos jugadores y el árbitro.