Programación Orientada a Objetos

Curso 2016/2017

Sesión 3

Ejercicios

1. Crea el paquete `tragaperras`. Todos los tipos de datos que se desarrollen en esta práctica se situarán en este paquete.

2. Define el enumerado `Fruta` con los valores fresa, sandía, plátano, melocotón y pera.

3. Define la clase `Premio` que representa los posibles premios que puede conceder una máquina tragaperras.

   Las propiedades de un premio son: el array de frutas que conforman la combinación ganadora y la cantidad de dinero que se concede si sale esta combinación. Estas propiedades pueden ser consultadas pero no se pueden modificar.

   Para construir un premio se deben pasar como parámetro los valores de todas las propiedades.

   No existe ninguna funcionalidad salvo la consulta de sus propiedades.

   Nótese que puesto que un array es un objeto modificable se debe evitar el problema del *aliasing*. Para copiar un array se puede utilizar el *método de clase* `copyOf` de la clase `java.util.Arrays` al que debemos pasar como parámetro el array que queremos copiar y el tamaño. La ejecución del método devuelve un nuevo array que es una copia del que se ha pasado como parámetro.

4. Define la clase `Maquina` que representa el concepto de una máquina tragaperras.

   Las propiedades que caracterizan a una máquina tragaperras y que se pueden consultar son:

   - número de casillas que forma la combinación de frutas que se generará en cada jugada. El valor de esta propiedad no se puede cambiar.
   - precio de la jugada, que puede modificarse en cualquier momento.
   - crédito disponible en la máquina para poder jugar.
   - colección (array) de los premios que concede. La colección de premios no podrá modificarse una vez establecida en la construcción.

   Una máquina se construye estableciendo el número de casillas de la máquina, el precio de la jugada y la colección de premios que puede conceder (argumento variable).

La funcionalidad de esta clase es:

- *incrementarCredito*: incrementa el credito disponible en la cantidad indicada por el parámetro.
- *cobrar*: devuelve el crédito disponible y por tanto deja a cero el crédito disponible en la máquina.
- *jugar*: devuelve la combinación generada en la jugada o un valor `null` si no se ha podido jugar. Para poder jugar se tiene que cumplir que el crédito disponible sea igual o superior al precio de la jugada. Si se puede jugar el algoritmo del método es el siguiente:
    - Decrementar el crédito disponible la cantidad correspondiente al precio de la jugada.
    - Generar aleatoriamente una combinación de frutas. Para ello es necesario crear un array de frutas cuyo tamaño sea el *número de casillas* de la máquina y en cada posición del array establecer una fruta obtenida aleatoriamente. Consulta la nota que se presenta más adelante sobre el uso del generador de números aleatorios.
    - Comprobar si la combinación generada se encuentra entre los premios registrados en la máquina. Para ello puedes utilizar el método de clase `equals` de la clase `java.util.Arrays` que recibe como parámetros los dos arrays que se quieren comparar y devuelve `true` si los arrays son del mismo tamaño y los objetos que contiene son iguales en las mismas posiciones.
    - En el caso de que la combinación generada coincida con uno de los premios registrados, se incrementa el crédito disponible en la cantidad fijada para el premio obtenido.
    - Finaliza retornando la combinación generada.

NOTA: Para generar la combinación aleatoria te puede ser útil la siguiente información

La clase `java.util.Random` es un generador de números aleatorios. Dispone del método de instancia `nextInt` que recibe como parámetro la cota superior para la generación del número aleatorio, es decir, generará un entero entre 0 y la cota superior, no incluida.

Ejemplo:

```
Random generador = new Random();
int index = generador.nextInt(5);
```

`index` puede ser cualquier entero de 0 a 4

Los enumerados disponen de un método de clase denominado `values` que devuelve un array con todos los valores del enumerado.

Ejemplo:

```
public enum DiasLaborables {
   LUNES, MARTES, MIERCOLES, JUEVES, VIERNES
}
DiasLaborables[] dias = DiasLaborables.values();
```

`dias[0]` referencia al primer valor del enumerado (`LUNES`)

5. Implementa la clase `Programa` que contenga el siguiente método `main`:

```
public class Programa {

   public static void main(String[] args) {

      // 1. Declara y construye dos premios

      Fruta[] combinacion1 = {Fruta.FRESA, Fruta.FRESA, Fruta.FRESA};
      Premio premio1 = new Premio(combinacion1, 20);
      Fruta[] combinacion2 = {Fruta.SANDIA, Fruta.FRESA,Fruta.SANDIA};
      Premio premio2 = new Premio(combinacion2, 10);

      /* 2. Crea una máquina con un tamaño de combinación de 3 frutas,
       *    un precio por partida de 0,5 euros
       *    y los dos premios declarados previamente
       */

      Maquina maquina = new Maquina(3, 0.5, premio1, premio2);

      /* 3. Solicita al usuario que introduzca por teclado la cantidad
       *    de crédito para jugar.
       */

      System.out.println("Introduzca el crédito: ");
      Scanner teclado = new Scanner(System.in);
      double credito = teclado.nextDouble();
      teclado.nextLine();
      maquina.incrementarCredito(credito);

      // 4. Jugamos mientras haya crédito disponible

      while (maquina.getCredito() > 0){
            //4.1 Realiza la jugada
            Fruta[] combinacion = maquina.jugar();

            //4.2 Muestra la combinación obtenida y el crédito
            System.out.println(Arrays.toString(combinacion)
                              + " --- " + maquina.getCredito());

            //4.3 Pide al usuario que pulse intro para continuar
            System.out.println("Pulse Intro para volver a jugar");
            teclado.nextLine();
      }
   }
}
```

Observa que:

- Un array se puede inicializar poniendo los valores iniciales entre llaves.
- Para leer de la entrada estándar se puede hacer uso de la clase `java.util.Scanner`.

  ```
  Scanner entrada = new Scanner(System.in);
  ```

  La clase `Scanner` ofrece métodos para leer líneas de texto (`nextLine`) y leer cualquier tipo de datos primitivo (`nextInt`, `nextDouble`, etc.).
- El método de clase `toString` de la clase `Arrays` que devuelve una cadena con la representación textual de un array.