Ejercicios adicionales UT6 (I) –

Colecciones de tamaño flexible: ArrayList. Otras colecciones.

Ejercicio 1.

Define una clase Bola que modela una bola de un determinado color que puede guardarse en una urna. Cada bola tiene un único atributo color, o blanco o negro. El atributo es del tipo enumerado Color. El constructor de esta clase crea una bola asignándole un color de forma aleatoria.

Incluye en la clase los accesores:

- getColor() - devuelve el color de una bola como valor enumerado
- getColorString() - devuelve un String que representa el color de la bola (consulta la teoría acerca del tipo enumerado)
- toString() - devuelve la representación textual de una bola, *por ej.* "*Bola de color NEGRO*"
- esBlanca()
- esNegra()

Diseña una clase Urna que modela una urna en la que se pueden almacenar una serie de bolas (utiliza una clase ArrayList). Además del constructor incluye los métodos:

- public void meterBola(Bola b) – añade una bola a la urna
- public Bola sacarBola() - saca una bola aleatoria mente de la urna. Si la urna está vacía se lanza una excepción. La bola desaparece de la urna y se devuelve. Se utiliza el método obtenerPosicionAleatoria()
- private int obtenerPosicionAleatoria(int hasta) – devuelve una posición aleatoria, el valor aleatorio oscila entre 0 y lo que indique el parámetro
- public int cuantasBlancas() - cuenta las bolas blancas que hay en la urna. Utiliza un *for-each*
- public int cuantasNegras() - cuenta las bolas negra que hay en la urna. Utiliza un for normal
- public void mostrarUrna() - muestra en pantalla la urna con cada bola y su color. Utiliza un iterador.
- public void borrarBlancas() - borra las bolas blancas con un iterador

Añade una clase TestUrna que contiene el main(). Vamos a utilizar esta clase para testear el proyecto. Desde la línea de argumentos del main() se tomará un parámetro que indicará inicialmente cuántas bolas vamos a añadir a la urna. Crea la urna, añade ese nº de bolas y testea los métodos de la clase Urna.

Ejercicio 2 .

La clase Libro representa a un libro de una librería. Un objeto Libro se caracteriza por su título, autor, nº de páginas, y estado (valor del tipo enumerado Estado que indica si está o no prestado).
Completa los mutadores prestar() y devolver() que cambian el estado del libro. Si ya está prestado o se devuelve sin estarlo se emite un mensaje de error. Completa los accesores getEstado() y estaPrestado().

La clase Librería mantiene una relación de libros. Esta relación se modela con un ArrayList. De la librería se guarda también su nombre.

El constructor crea la librería de un determinado nombre que se pasa como parámetro.

La librería también permite:

- public int numeroLibros() - obtener la cantidad de libros que hay en la librería
- public void addLibro(Libro l) - dado un libro, añadirlo a la librería

- public Libro localizarLibro(String titulo) - dado un título de libro, localizarlo y devolverlo, si no está se devuelve *null* (con iterador)
- public boolean hayLibrosDe(String autor) - dado un nombre de autor, indicar (devolviendo *true* o *false*) si hay algún libro de él (con *while* e índices)
- public void borrarLibrosDe(String autor) - borrar los libros de un determinado autor (con *while* e índices). Si no hay ningún libro del autor se indicará con un mensaje
- public void listarPrestados() - mostrar la relación de libros prestados que hay en la biblioteca (con *for-each)*
- public void prestarTitulo(String titulo) - prestar un libro de un determinado título. Si no está en la biblioteca se emite un mensaje y si está ya prestado también.
- public String toString () - representación textual de la librería

Incluye en la clase Libro un atributo adicional *fechaPrestamo* de tipo String (en el constructor inicialízalo a cadena vacía).
Modifica el método prestar() para que cuando se preste un libro se guarde la fecha en la que se hizo el préstamo que será la fecha actual. Utiliza un método privado obtenerFechaPrestamo() que devuelve un String con la fecha del préstamo. **(consulta Moodle y el enunciado para ver cómo trabajar con fechas y horas a partir de la versión 8 de Java).**

Dentro del método obtenerFechaPrestamo() :

- ✗ crea un objeto hoy de la clase LocalDate con la fecha actual
- ✗ obtén un objeto *formateador* de tipo DateTimeFormatter con el patrón "*dd MMM yyyy*" que nos servirá para formatear la fecha
- ✗ formatea el objeto hoy llamando al método format() de LocalDate
- ✗ modifica el método toString() de Libro para que ahora incluya la fecha del préstamo (en caso de que esté prestado el libro).

Añade una clase AppLibrería que contenga el método main() y:

- ✗ acepta como argumento el nombre de la librería (controla los posibles errores en el paso de argumentos)
- ✗ prueba todos los métodos de la clase Libreria

Trabajo con fechas en java 8

La versión java 8 introdujo una nueva API para trabajo con fechas y horas.
Todas las clases están en el paquete **java.time**.

Entre otras clases la nueva API incluye **java.time.LocalDate**, **java.time.LocalTime**, **java.time.LocalDateTime**, y **java.time.format.DateTimeFormatter**.

java.time.LocalDate - Representa a una fecha (año, mes, día).

Para obtener la fecha actual:

```
LocalDate hoy = LocalDate.now();
```

Para obtener una fecha concreta:

```
LocalDate fecha1 = LocalDate.of(2013, Month.AUGUST, 10); // 10 de Agosto de 2013
LocalDate fecha2 = LocalDate.of(2013, 8, 10); // 10 de Agosto de 2013
```

Mostrar día, mes y año separadamente:

```
System.out.println(hoy.getDayOfMonth() + " / "  + hoy.getMonth() + " / " +   hoy.getYear());
```

java.time.LocalTime - Representa a una hora (horas, minutos, segundos)

Para obtener la hora actual:

```
LocalTime ahora = LocalTime.now();
```

Para obtener una hora concreta:

```
LocalTime localTime = LocalTime.of(22, 33); //10:33 PM
```

java.time.LocalDateTime - Combina fecha y hora en una sola clase

Para obtener la fecha y hora actual:

```
LocalDateTime fechaYhora = LocalDateTime.now();
```

Por defecto las clases anteriores usan el sistema de reloj de la zona horaria por defecto.

java.time.format.DateTimeFormatter – Permite formatear una fecha, o una hora.

Se formatea usando el método format() sobre un objeto LocalDate / LocalTime y pasando como parámetro un objeto *formateador* de tipo DateTimeFormatter que incluye el patrón de formato que deseamos para la fecha.

Para formatear una fecha:

```
LocalDate hoy = LocalDate.now();
DateTimeFormatter formateador = DateTimeFormatter.ofPattern("dd MMM yyyy");
String strFechaActual =  hoy.format(formateador);
.....
System.out.println(hoy.format(DateTimeFormatter.ofPattern("dd/MM/yy")));
```

Para formatear una hora:

```
LocalTime ahora = LocalTime.now();
DateTimeFormatter formateador = DateTimeFormatter.ofPattern("H:m:s");
System.out.println(ahora.format(formateador));
…..............................
 System.out.println(ahora.getHour() + ":"  + ahora.getMinute() + ":" +   ahora.getSecond());
```

Parsear fecha y hora

Para parsear una fecha o una hora contenida en un String podemos utilizar el método estático parse() de la clase LocalDate o LocalTime.

```
LocalDate fechaParseada = LocalDate.parse("2015-01-29");
```

```
String str = "1986-04-08 12:30";                                    ese es el formato correcto
DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm");
LocalDateTime dateTime = LocalDateTime.parse(str, formatter);
```