Sesión 07: Persistencia. Archivos binarios y serialización.

Objetivos

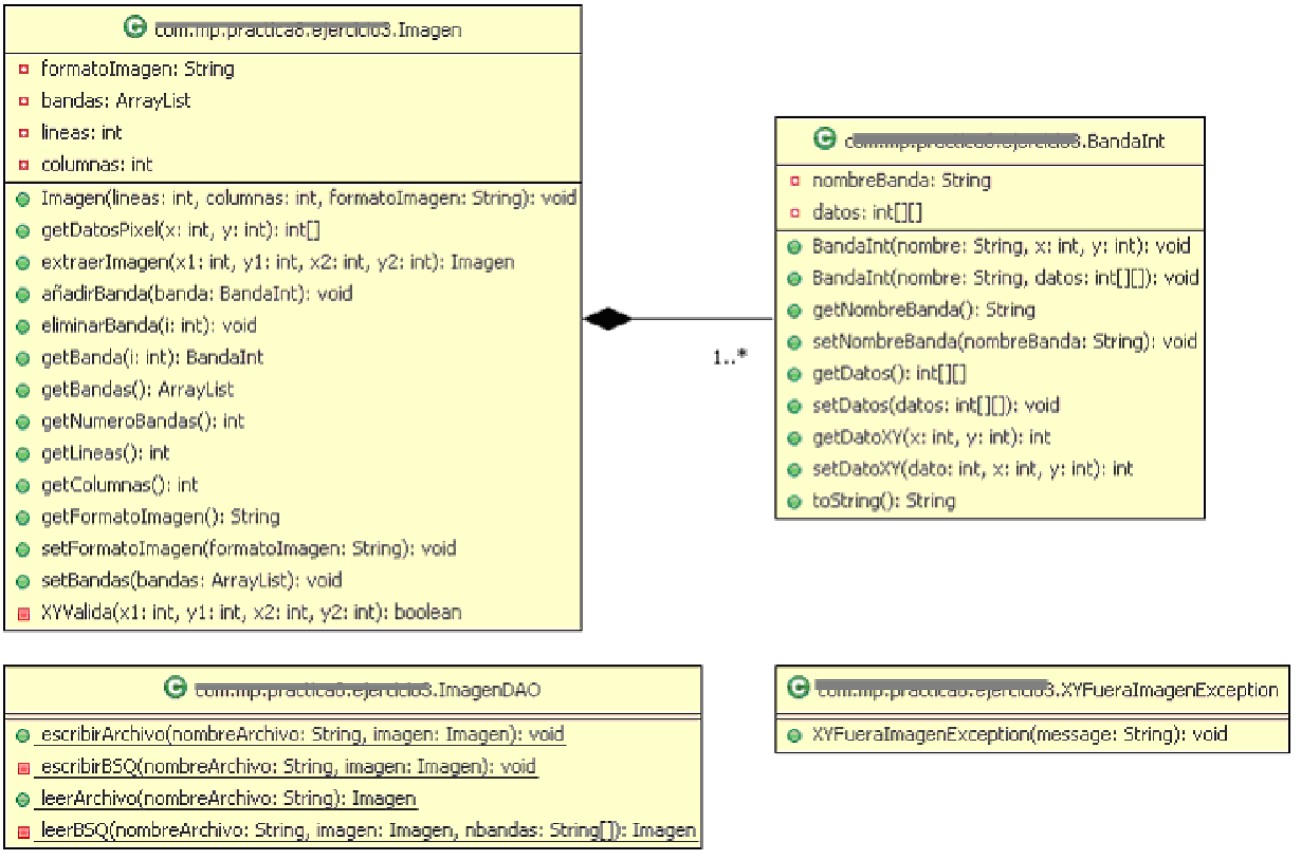
- Conocer y reutilizar de las clases del paquete **java.io**
- Realizar ejercicios utilizando archivos binarios.
- Conocer y utilizar la serialización de objetos.

Requisitos

- Seguir el esquema de nombrado de paquetes **org.mp.sesion0x**. En este paquete, en la carpeta de fuentes **src,** se crearán todos los programas que se proponen en la sesión, dándoles un nombre alusivo a lo que realiza el programa y que se indica en cada ejercicio en negrita.
- Las pruebas JUnit, en el mismo paquete **org.mp.sesion07,** en la carpeta de fuentes **test**.
- Al final de la sesión, el alumno deberá cargar el trabajo realizado a su repositorio indicando la clave correspondiente a la sesión.
- Documentar todas las clases.
- Las sesiones se han diseñado para cubrir una semana de trabajo.

Ejercicios propuestos

1. Escriba un programa que debe procesar un fichero binario que almacena datos de tipo int, podría representa una imagen con datos físicos de un área de un mapa. El archivo está organizado por distintas bandas, todas del mismo tamaño y tipo. Todas las bandas están caracterizadas por número de líneas y píxeles por línea (filas, columnas). Esta información junto con el nombre del archivo esta almacenada en un archivo de texto cabecera. (Vea los tests).



Trabajo autónomo

2. Consulte y estudie mediante la documentación de la API las siguientes clases del paquete ***java.io***

   FileInputStream, DataInputStream
   FileOutputStream, DataOutputStream
   ObjectInputStream, ObjectOutputStream

3. Este ejercicio se refiere al sistema informático de una residencia de vacaciones. En lo que se refiere al apartado de software, se ha decidido empezar con un sistema nuevo. Este se ha concretado, en una primera etapa, en el desarrollo de un sistema de gestión de reserva de habitaciones.

   Después de una investigación preliminar, basada en varias entrevistas y en la revisión de documentación, el dominio de problema se ha concretado en una especificación del software textual con el objetivo de una primera versión del sistema. De la cual se extrae el siguiente resumen:

   - La residencia de vacaciones, cuyo nombre será "**La Mar Salá**", está situada junto al mar. Se estructura en habitaciones. Todas las habitaciones disponen de magnificas vistas al mar Mediterráneo, ver imagen en las página siguiente. Las habitaciones están todas equipadas igual, disponiendo de dormitorio, cuarto de baño y una pequeña salita de estar. Aún está por determinar el número de habitaciones finales que se construirán. Y se identificarán de la forma habitual en los establecimientos hoteleros, con un número cuyo primer dígito identifica la planta y los dígitos que le siguen el número de habitación.
   - De los clientes, a los que se referirá como residentes, se guardarán los datos Dni, Nombre, Fecha nacimiento y Sexo.

   La funcionalidad básica que se pide en este prototipo es:

   - La operación de ingreso en la residencia de un residente consiste en tomarle sus datos y añadirlo a la residencia y en asignarle una habitación mediante una reserva con una fecha de entrada y una fecha de salida posterior.
   - La operación de salida de la residencia de un residente consiste en dar de baja (eliminar) el residente, y en buscar su reserva y asignarle la fecha de salida.
   - La operación de cambio de habitación de un residente consiste en asignar la fecha de salida a la reserva actual del residente y en crear una nueva reserva con el residente, la nueva habitación y como fecha de entrada la fecha de salida.
   - Cada vez que se crea una nueva reserva, esta constará de un número de reserva entero positivo único que se generará autoincrementado, es decir 1, 2, 3,…. Además constará con los datos de residente, habitación, fecha de entrada y fecha de salida.
   - Obtener una serie de listados con información sobre el sistema:
     - Un listado ordenado por el nombre de todos los residentes y las habitaciones que tienen asignadas en una fecha.
     - Un listado de las habitaciones libres en una fecha.
     - Un listado con la edad media en años de los residentes según el sexo.

Restricciones de diseño: Deberá utilizar la clase **ArrayList** o **LinkedList** de java.util para las propiedades

Junto con este documento se ha incluido dos juegos de pruebas para probar el ejercicio.

Ejecute el juego de pruebas **ResidenciaTestA** hasta que sea pasado.

Termine de implementar en ejercicio para que pase el juego de pruebas **ResidenciaTestB**. Haga las modificaciones necesarias hasta que pase el juego de prueba sin modificarlo. Para realizar este apartado debe utilizar JUnit en la perspectiva Java de Eclipse.

Genere el **diagrama de clases** del ejercicio utilizando AmaterasUML

Recuerde: No se puede modificar el código del juego de pruebas para que se adapte a su código. En el paradigma de la programación extrema el juego de pruebas se realizan con frecuencia antes que la propia implementación de las clases.

- Introduzca una **nueva funcionalidad** al ejercicio.

Ahora se pide que, además de la funcionalidad original, el programa permita la persistencia de los objetos mediante archivos. Ver diagrama de clases.

- Diseñe una nueva clase **ResidenciaSER** que permita las operaciones de guardar y recuperar los objetos de la aplicación en archivos. Utilice la interfaz **Serializable** y las clases **ObjectInputStream** y **ObjectOutputStream**.

- Diseñe una nueva clase **ResidenciaXML** que permita las operaciones de guardar y recuperar los objetos de la aplicación en archivos. Utilice la librería **XStream**.

Nota: Lea el tutorial de **dos** minutos: http://x-stream.github.io/tutorial.html