Diagramas UML Actividad#1

Jorge Enrique Celis Cortés

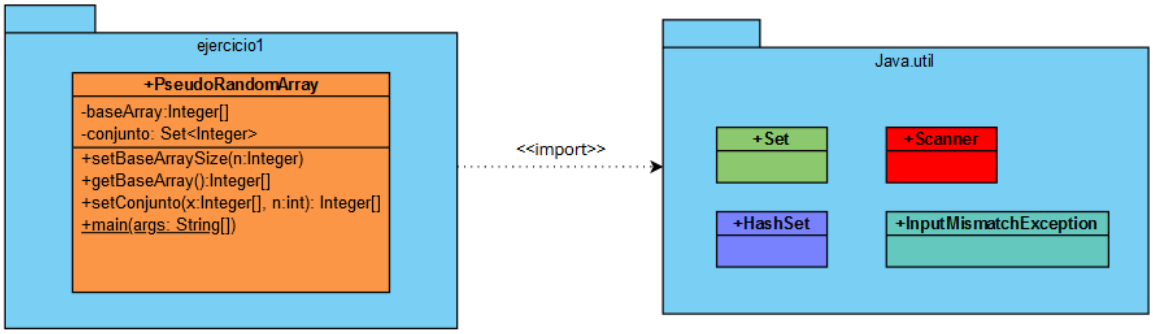
Facultad de Estudios a Distancia, Universidad Militar Nueva Granada

200267: Programa ingeniería informática a distancia

William Frasser Acevedo

7 de febrero de 2023

Ejercicio#1



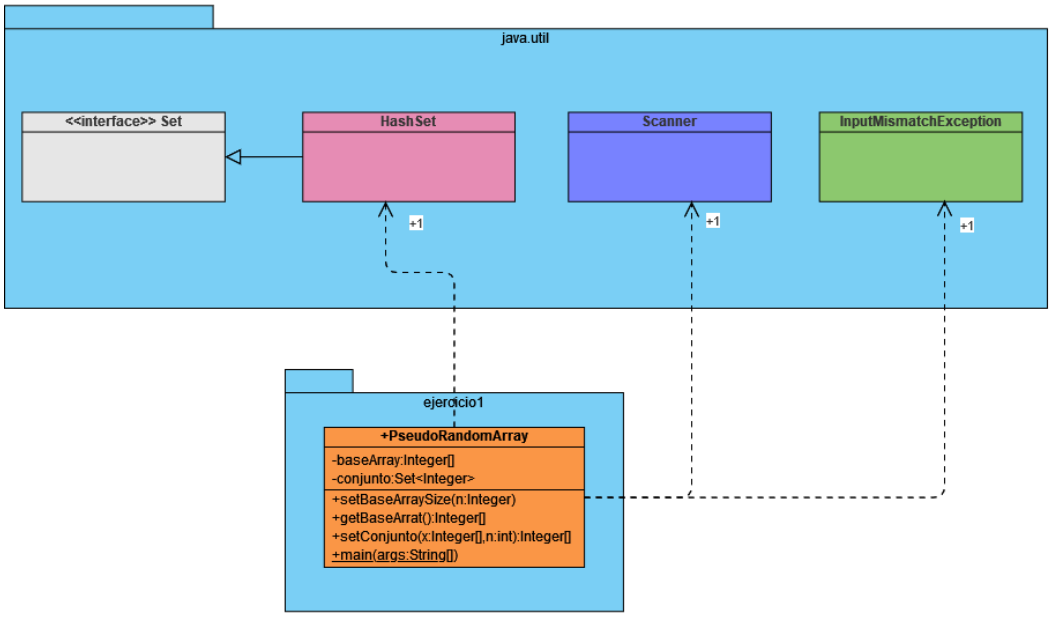
Según lo solicitado en el ejercicio, se creó una clase -la clase PseudoRandomArray -, que implementase:

- Un arreglo de íntegros de tamaño arbitrario cuya única condición de existencia era la de tener elementos aleatorios entre 10 y 300;
- Una función que tomase como argumentos un arreglo de íntegros y un integro indicador y devolviese un nuevo arreglo que poseyese los elementos que presentasen como último digito el integro indicador;
- Implementase la función main() e hiciese una rutina en donde se le solicitaba al usuario dar un número cualquiera, y a partir de este determinar en el arreglo cuáles números poseían como último digito el número cualquiera.

Para la elaboración de todo lo mencionado, se utilizaron de la librería estándar de java en java.util lo siguiente: la interface Set (un contenedor de objetos no repetitivos), la clase HashSet (implementa la interface Set, pero con un HashMap), la clase Scanner (empleada para obtener input de los usuarios), y la clase InputMismatchException (para manejar errores creados por input inapropiado otorgado por el usuario). Aquí se utiliza la relación entre paquetes <<import>> para hacer alusión al uso de todas las clases antes mencionadas sin afectar en lo mínimo la implementación o definición de estas en el paquete target, aunque se pudo haber hecho un

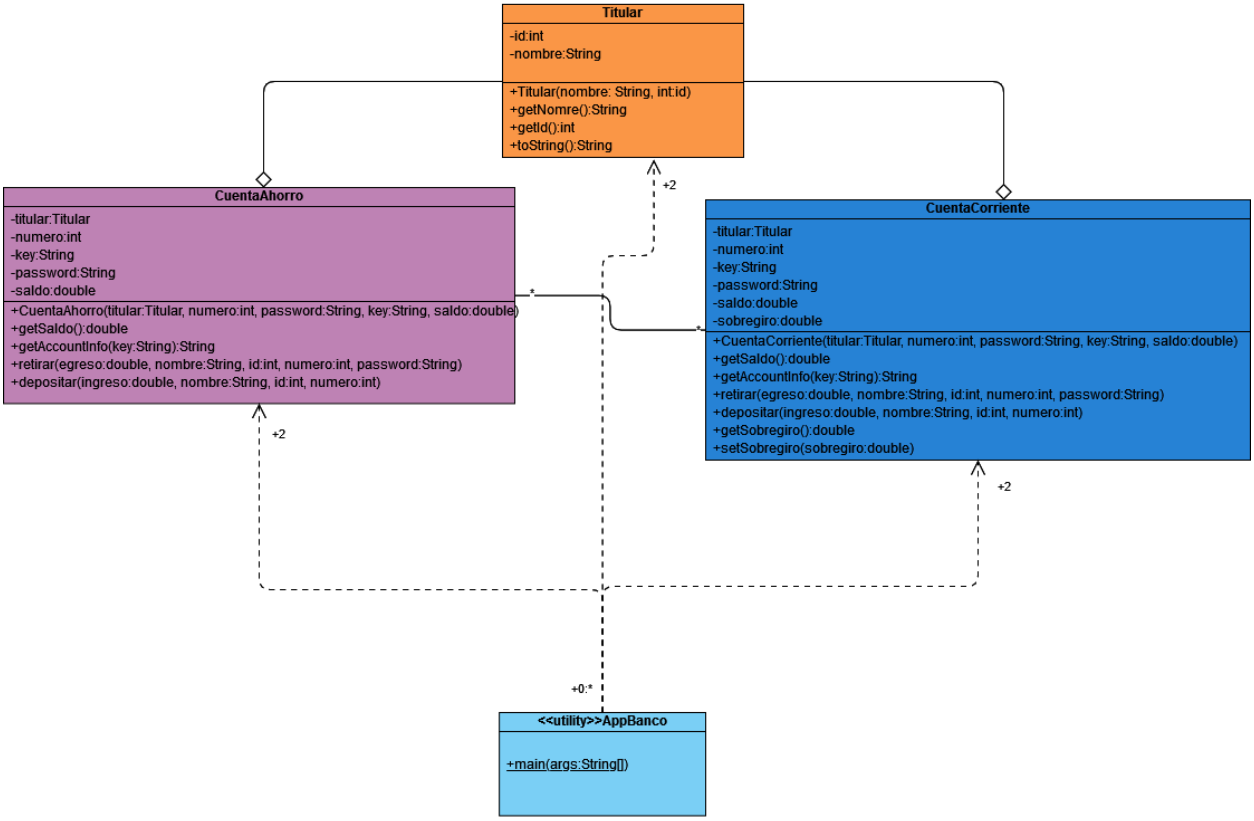
diagrama de dependencia de la clase PseudoRandomArray con las demás clases marcadas en java.util, porque sin estas la susodicha, al menos objetos creados con esta, no tendrían posibilidad de existir o siquiera implementar los requisitos previamente descritos. Si se realizan cambios a la interfaz Set, a la clase HashSet, a la clase Scanner, e inclusive a la clase InputMismatchException, entonces la clase PseudoRandomArray no podría implementarse de la forma en que fue hecha.

Esta alternativa se evidencia en el siguiente diagrama alternativo:



Para más información se sugiere que se dirija a la documentación en el folder nombrado ejercicio1.

Ejercicio#2

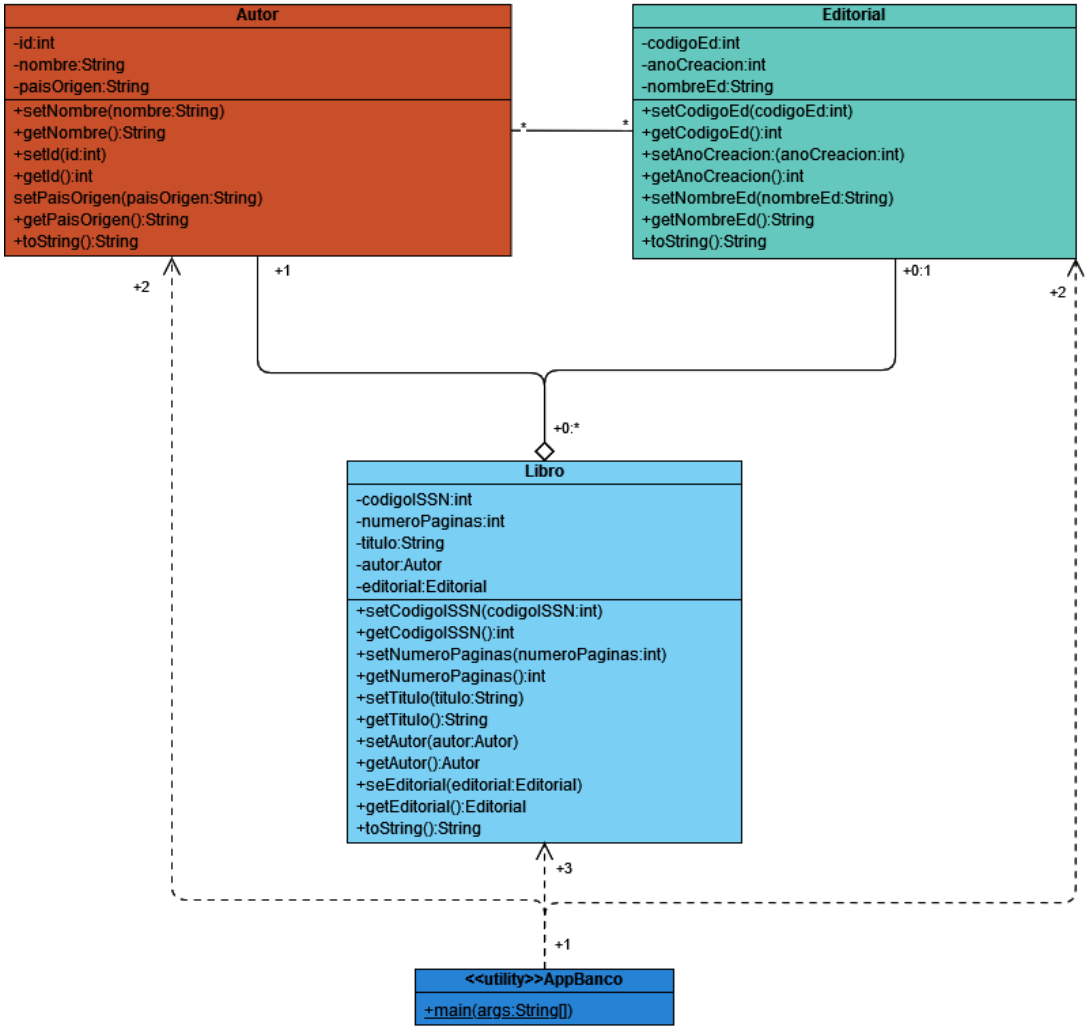


Para la implementación de este ejercicio se crearon tres clases: Titular, CuentaAhorro, y CuentaCorriente. A pesar de que instancias de las clases CuentaAhorro y CuentaCorriente son parecidas en términos de atributos y métodos, ambos objetos son completamente independientes y rara vez relacionados, ya que ningún atributo o método es usado entre ambas, y en la vida real, no se necesitan entre sí de ningún modo formal o recurrente; tal vez para depositar un saldo inicial o para sentar una deuda bancaria, pero en la mayoría de los casos estos son usos establecidos por otras entidades no solicitadas en la resolución de este ejercicio.

A su vez, instancias de la clase Titular no dependen o siquiera presentan un tiempo de vida enlazado estrictamente a la vigencia de una cuenta, mientras que las cuentas necesitan, en el

caso de este ejercicio, de un titular para poder hacer una realización apropiada de su existencia; además, los objetos de tipo Titular necesitan saber de la existencia de cuentas creadas a su nombre, por lo que la relación entre instancias de la clase Titular e instancias de CuentaAhorro o CuentaCorriente es de agregación.

Ejercicio#3



En este ejercicio se necesitaron de tres clases: la clase Autor, la clase Editorial, y la clase Libro, para simular y abstraer objetos reales como los libros que tienen que haber sido escritos por un autor o revisados por una editorial.

Una editorial puede estar asociada a varios autores, y un autor puede estar relacionado a varias editoriales, pero no necesariamente las editoriales dependen de los autores o los autores de las editoriales, ya que las editoriales son capaces de ser autoras de sus propios libros (como los compendios y diccionarios), y los autores son capaces de publicar y editar sus propios libros utilizando sus propios recursos, por lo que la relación entre dos objetos tipo Autor y tipo Editorial es simplemente asociativa e inclusive inexistente.

En cambio, los libros necesitan haber sido escritos por alguien, y no necesariamente debe haber una editorial para su publicación, pero como los objetos de tipo Autor y tipo Editorial tienen un ciclo de vida independiente del libro (un autor puede morir o dejar un libro sin terminar, y una editorial no requiere de libros para existir), y un autor o editorial necesitan saber de la existencia de un libro hecho por ellos, la relación que se presenta entre objetos Libro y objetos Autor o Editorial es de agregación.

Finalmente, debido a los requerimientos del ejercicio, se tiene la clase pública de utilidad AppLibro que sirve para probar qué tan bien fue hecha la implementación de las clases antes mencionadas, dando a entender una relación de dependencia con las otras clases en este ejercicio.

Para más información diríjase a la documentación en el folder ejercicio3.