Estructura de datos

Michael Alexander Flórez Muñoz

Gustavo Adolfo Gutiérrez Londoño

Juan Camilo Jaramillo De La Torre

Presentado a: Robinson Arias

Universidad del Quindío

Facultad: Ingeniería

Programa: Ingeniería de Sistemas y Computación

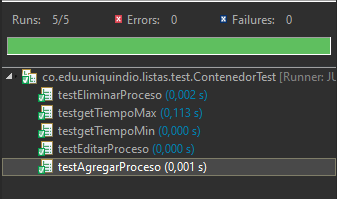
Espacio Académico: Estructura de datos

Grupo: 01-D

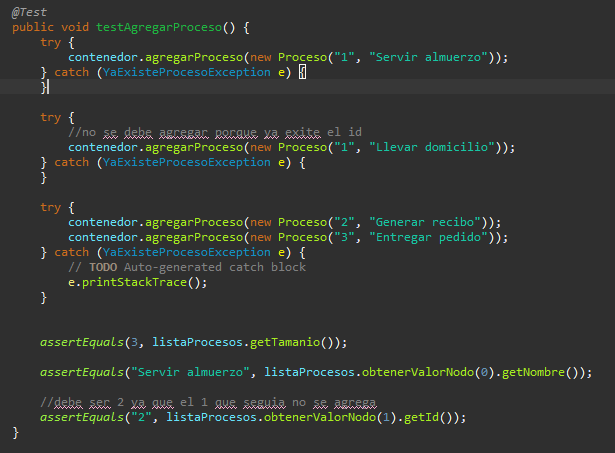
Armenia Quindío

27 abril 2021

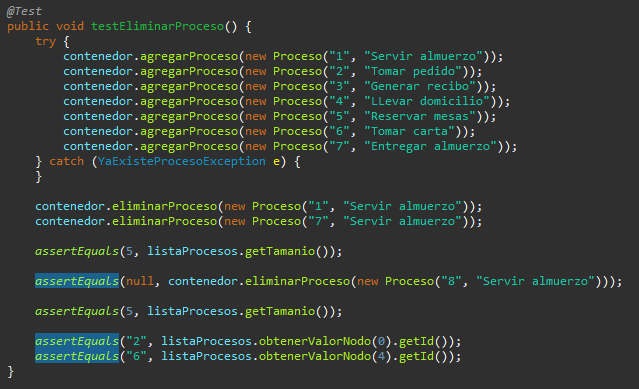
Test de los contenedores



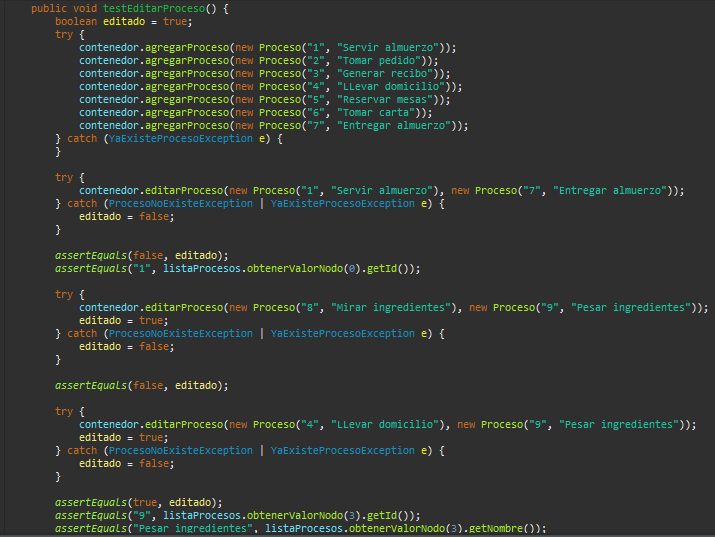
Agregar proceso: En este test sobre agregar el proceso se verifica que se pueda agregar un proceso, a la vez que ese proceso no puede tener un mismo id y esto lo vemos en la imagen cuando se verifica con el assertEquals de que el tamaño de la lista sigue siendo de 3, también verificamos que el proceso con id 1 sigue siendo servir almuerzo y que por ultimo se verifica que en la posición 1 no se haya agregado el proceso repetido.



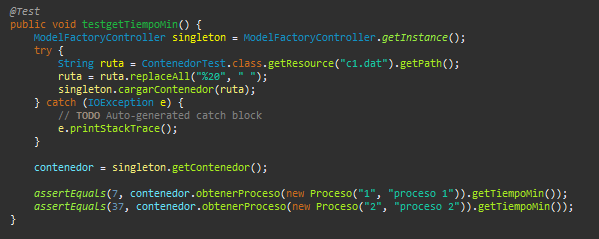
Eliminar proceso: Como ya verificamos que se puede agregar un proceso, agregamos varios procesos para poder testear la función eliminar un proceso en este paso eliminamos 2 procesos anteriormente creados y después verificamos que: el tamaño de la lista concuerde con el tamaño que debería quedar, también verificamos que no se pueda eliminar un proceso que no se haya creado anteriormente y que nuestro tamaño de la lista siga siendo el mismo, por ultimo verificamos que al borrar el proceso con id 1 y id 7 en la lista queden los procesos en la nueva posición indicada.



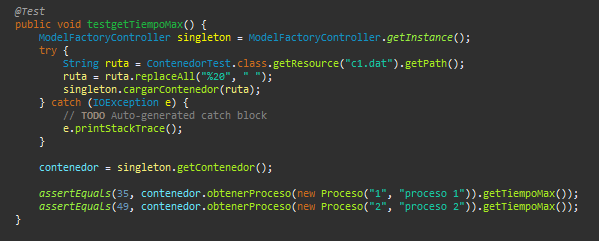
**Editar proceso:** En este test agregamos procesos para poder hacer la verificación de editarlos, creamos una variable booleana la cual nos ayudara para observar si la verificación es o no correcta, empezamos editando un proceso poniéndole un id ya existente lo cual no debería dejar hacer y podemos observar que esperamos que un false como respuesta ya que no deja y es lo que recibimos, verificamos que el id del nodo en la posición 0 siga siendo el mismo ya que no edito y también es lo esperado, verificamos que no se pueda editar un proceso que no exista esperando un false y es lo que recibimos, por ultimo verificamos que realmente podemos editar un proceso existente el cual será el que tiene id 4 y lo editaremos por uno con id 9, verificamos que edite esperando un True como respuesta y a la vez que el id haya cambiado a 9 y su nombre a Pesar ingredientes.



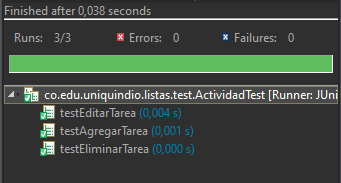
**Get tiempo mínimo:** En este test cargamos los datos de prueba lo cual es por la forma binaria, cargamos el contenedor donde estarán nuestros datos de prueba y donde los procesos tendrán un tiempo mínimo, nuestro tiempo esperado es 7 para el primero proceso ya que la suma de sus tareas obligatorias es 7, y para nuestro proceso 2 nuestro tiempo mínimo esperado es 37 y 37 es el valor obtenido.



**Get tiempo Máximo:** En este test cargamos los datos de prueba lo cual es por la forma binaria, cargamos el contenedor donde estarán nuestros datos de prueba y donde los procesos tendrán un tiempo máximo, nuestro tiempo esperado es 35 para el primero proceso ya que la suma de sus tareas obligatorias y opcionales es 35, y para nuestro proceso 2 nuestro tiempo máximo esperado es 49 y 49 es el valor obtenido.



Test de las actividades



**Agregar tarea:** En este test verificaremos que los requerimientos funcionales de las actividades sean cumplidos, primero creamos una variable booleana la cual nos indicara si el requerimiento llamado se hizo o no se hizo, primero tenemos agregar tarea la cual tendra como parametros la tarea, una posicion y una opcion, dependiendo de la opcion es una manera diferente de agregar, primero agregamos una nueva tarea con la opcion 0 la cual es agregar al final ya que es una cola, esperamos un true como respuesta y es lo que obtenemos, luego intentamos agregar otra actividad con la opcion 0 pero esta vez no dejaria ya que la actividad anterior fue una tarea opcional y lo esperado es un false ya que no agrego y es lo que obtenemos, verificamos que el tamaño de la cola siga siendo el mismo debido a que no agrego nada en este caso sigue siendo 1, luego agregamos varias tareas mas respetando la condicion de que 2 tareas opcionales no pueden ir seguidas el valor esperado es true y es el valor obtenido ,verificamos nuevamente que no se pueda agregar una actividad opcional despues de otra y a su vez que el tamaño no haya cambiado, luego verificamos el otro requerimiento que es agregar una tarea en una posicion indicada en este caso le llega la posicion 1 y la opcion 1 en este caso esperamos un false ya que en la posicion 1 hay una tarea opcional y no dejaria agregar y en este caso recibimos un false como respuesta, y asi intentamos seguir agregando con la opcion 1 en varias posiciones donde haya una tarea opcional y en el resto siempre esperamos un false como respuesta y este es el que obtenemos.

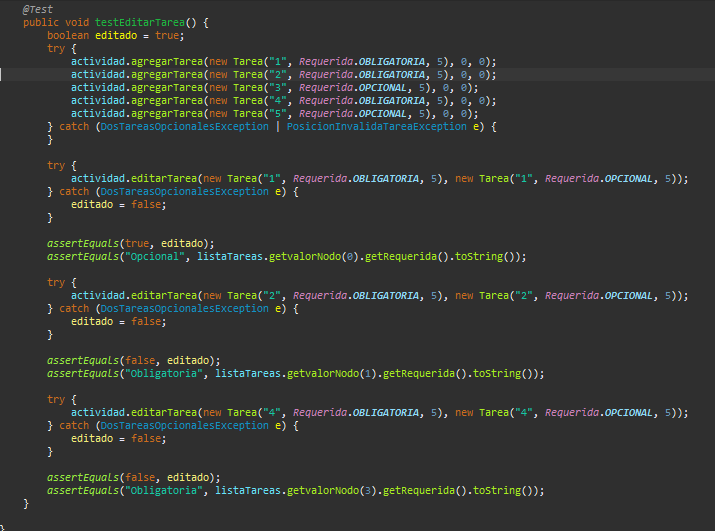




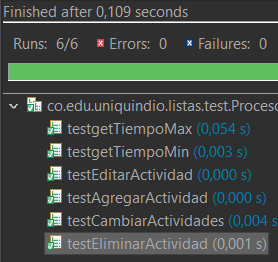
**Eliminar tarea:** En este test verificamos que las tareas se puedan eliminar correctamente pero que no se pueda eliminar una tarea que tenga como consecuencia dejar otras 2 tareas opcionales seguidas, primero creamos una variable booleana la cual nos indicara si el requerimiento llamado se hizo o no se hizo, empezamos agregando varias tareas de prueba para así después poder eliminarlas, luego eliminamos la tarea 1 la cual se podrá eliminar ya que no deja 2 tareas opcionales seguidas en este caso esperamos un true y es lo que obtenemos, y comprobamos que el tamaño de las tareas haya disminuido, luego pasamos a comprobar si deja eliminar la tarea 4 la cual dejaría 2 tareas opcionales y el resultado que esperamos es false y es el resultado obtenido ya que la aplicación no deja hacer esa eliminación y por ultimo comprobamos que el tamaño de la lista siga en 4 ya que no se ha eliminado nada y es el valor que obtenemos.



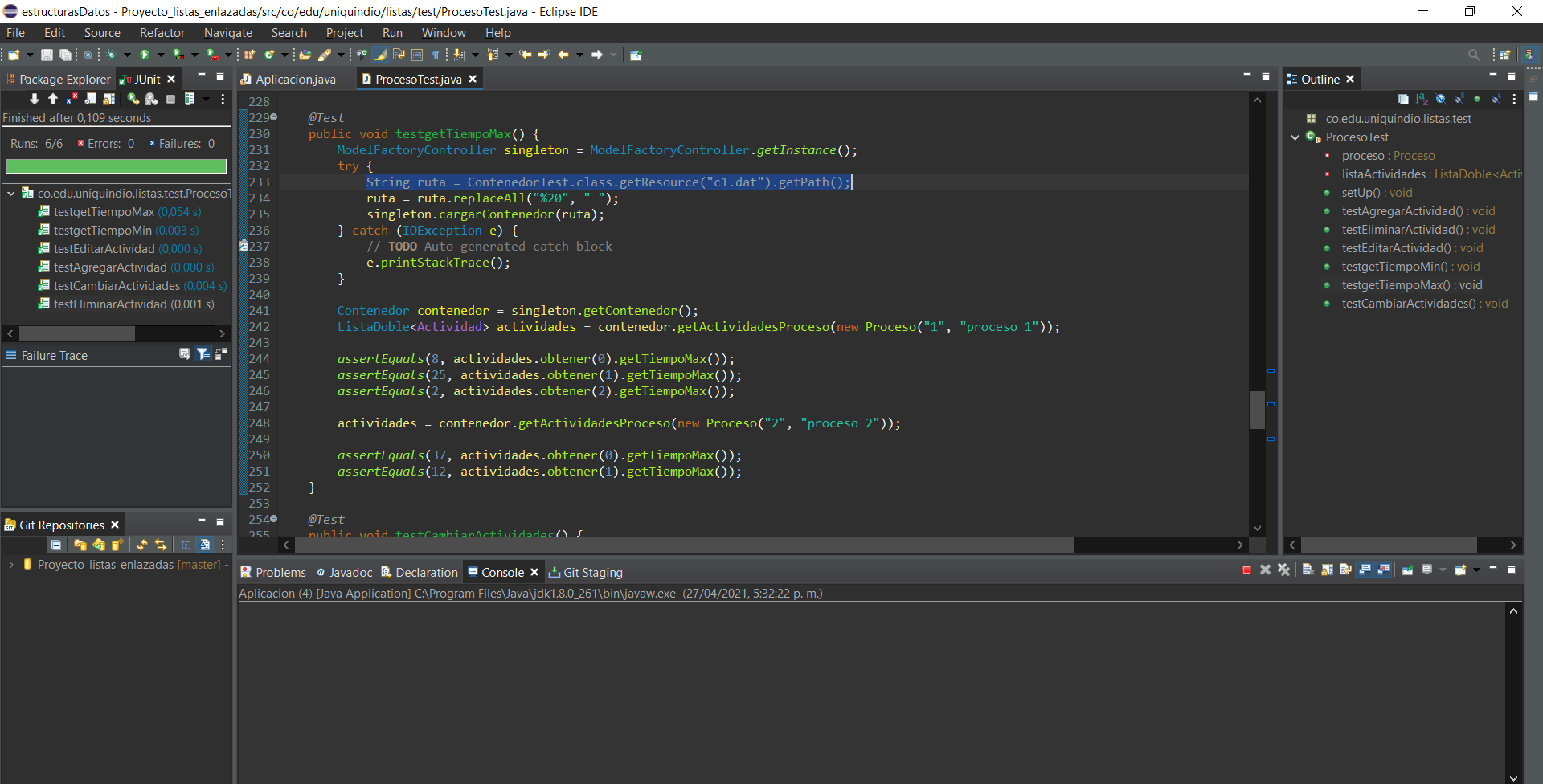
**Editar tarea:** En este test verificamos que las tareas se puedan editar correctamente pero que no se pueda editar una tarea que tenga como consecuencia dejar otras 2 tareas opcionales seguidas, primero creamos una variable booleana la cual nos indicara si el requerimiento llamado se hizo o no se hizo, empezamos agregando varias tareas de prueba para así después poder editarlas, primero intentamos editar la tarea 1 que es obligatoria por opcional, el resultado esperado es true ya que no dejaría 2 tareas opcionales seguidas y true es el valor obtenido, también verificamos que haya editado correctamente esperando Opcional como valor, y obteniendo opcional como resultado, luego intentamos editar la tarea 2 que es obligatoria por opcional pero esta vez no dejaría ya que editamos la tarea 1 anteriormente por opcional y 2 tareas opcional seguidas no dejaría, en este caso esperamos un false como respuesta y es false el valor que obtenemos y a su vez verificamos que el valor del nodo 1 siga siendo obligatorio y no haya cambiado



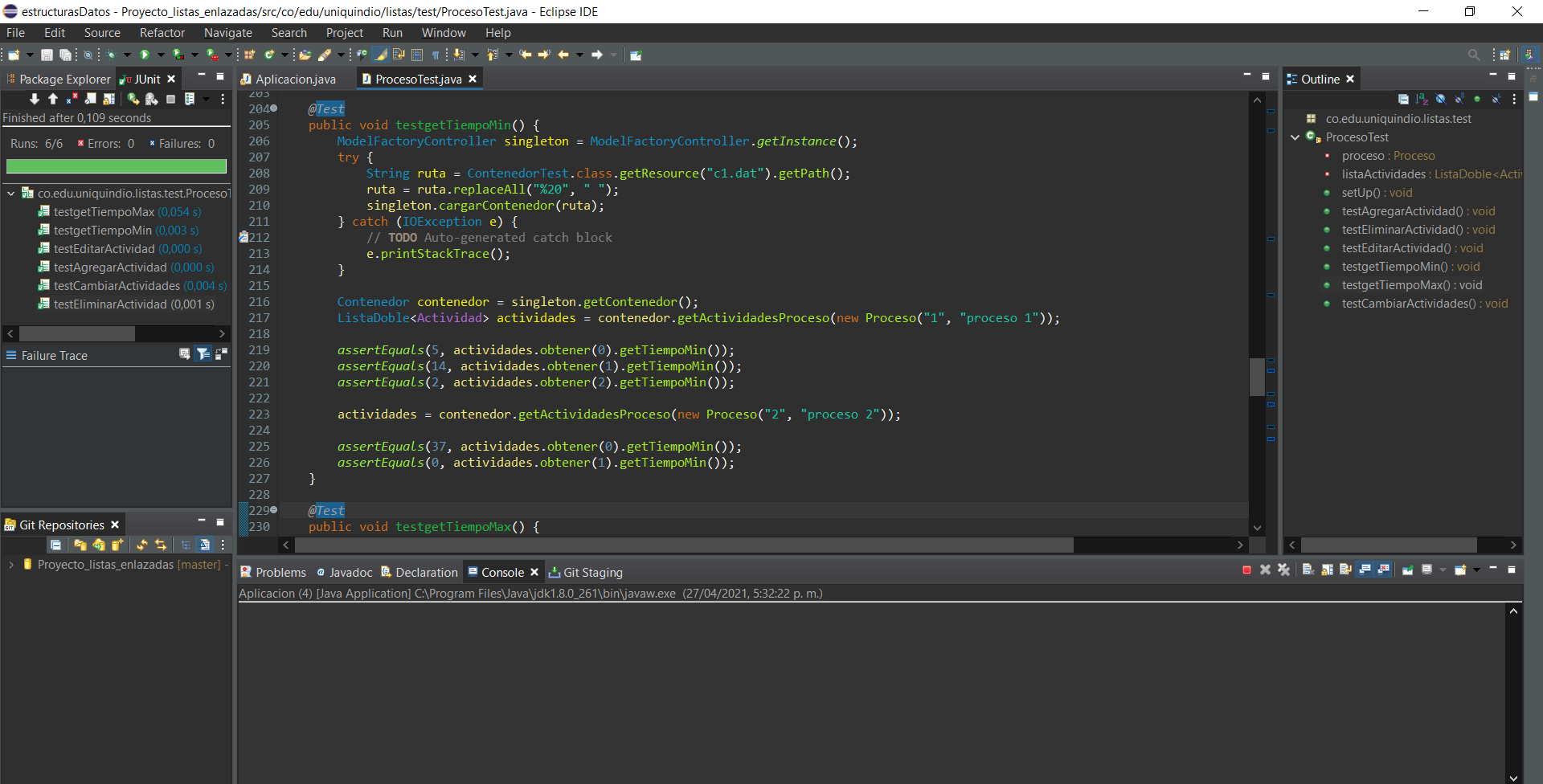
Test de los Procesos



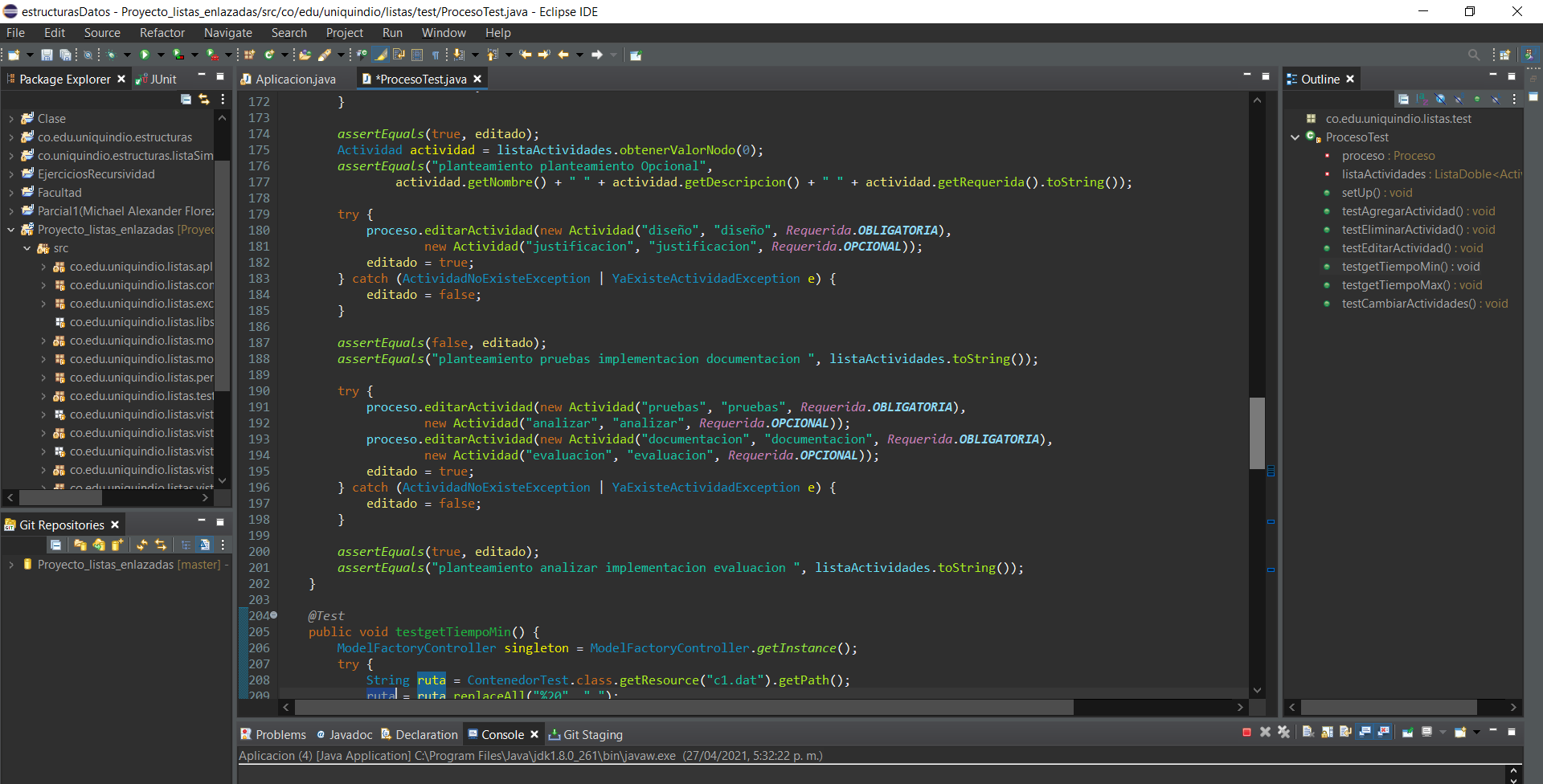
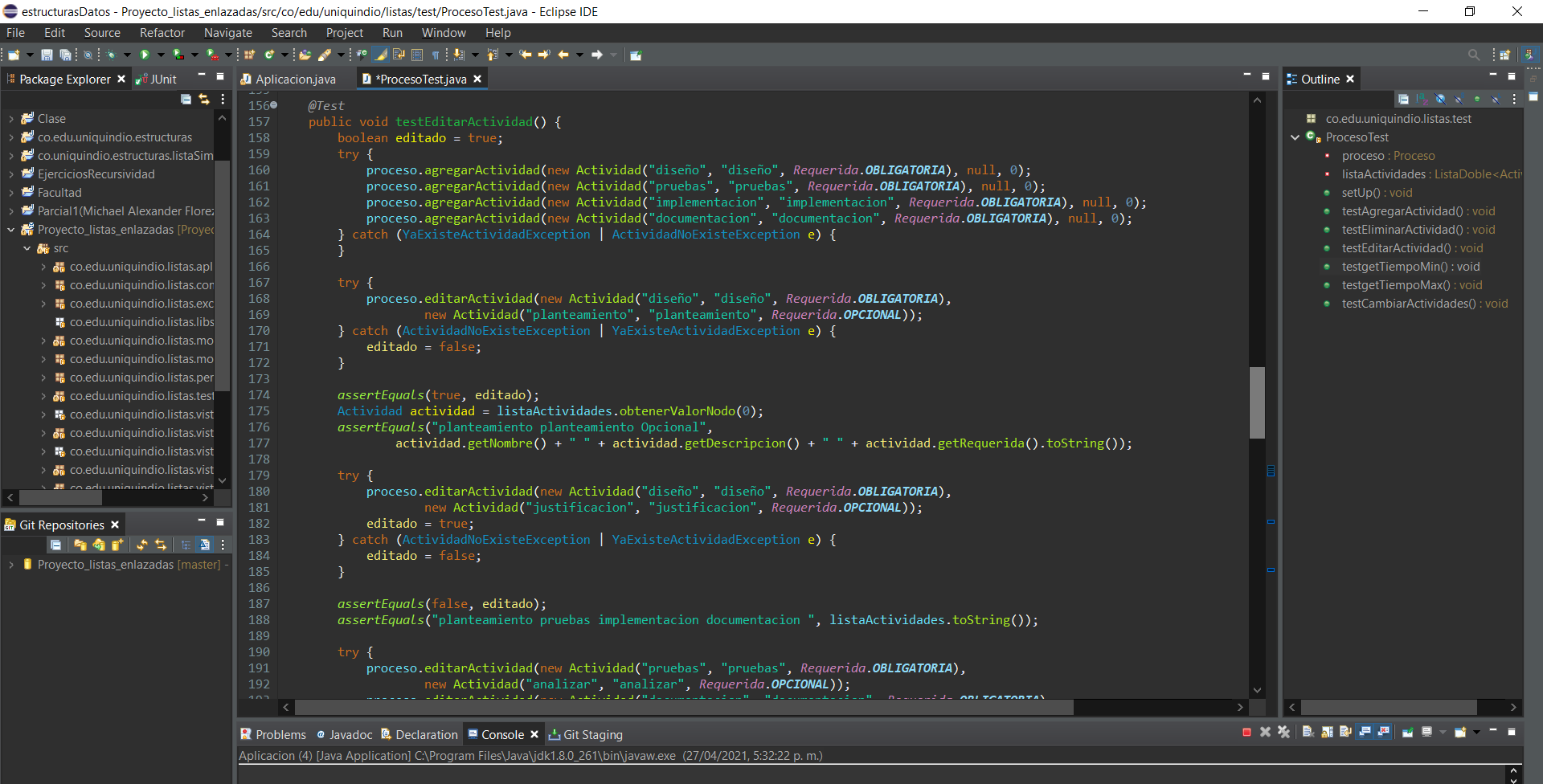
**Get tiempo Máximo:** En este test lo que hacemos es cargar un contenedor que se encuentra dentro del proyecto en un archivo con extensión “.dat”, después de esto, sacamos el contenedor en una variable y creamos una lista doble para almacenar las actividades de dicho contenedor, luego de esto accedemos a las actividades para conseguir el tiempo máximo de cada una y lo comparamos con el tiempo que debe de retornar el método, como se puede observar en la imagen anterior todo funciona bien.



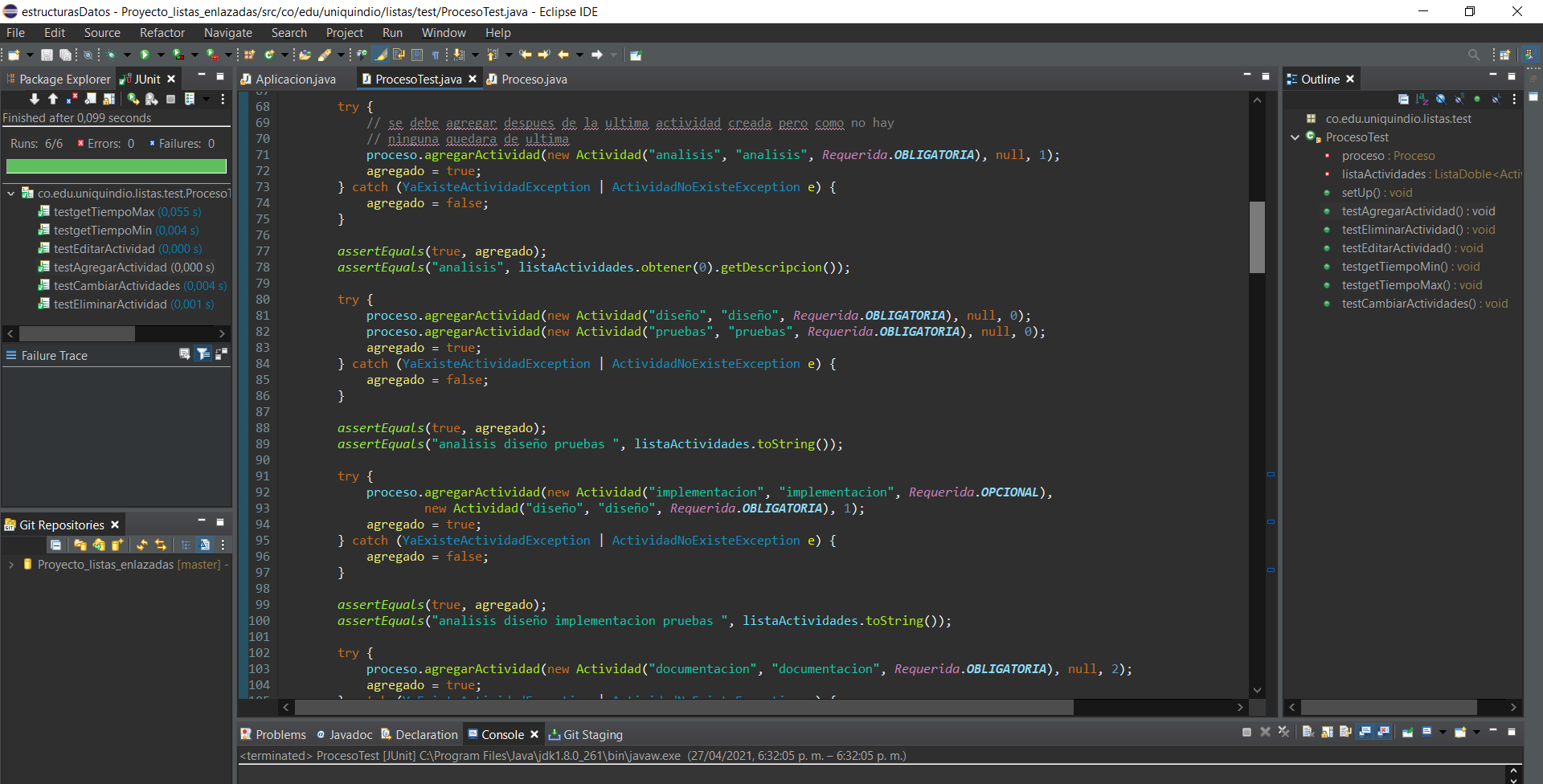
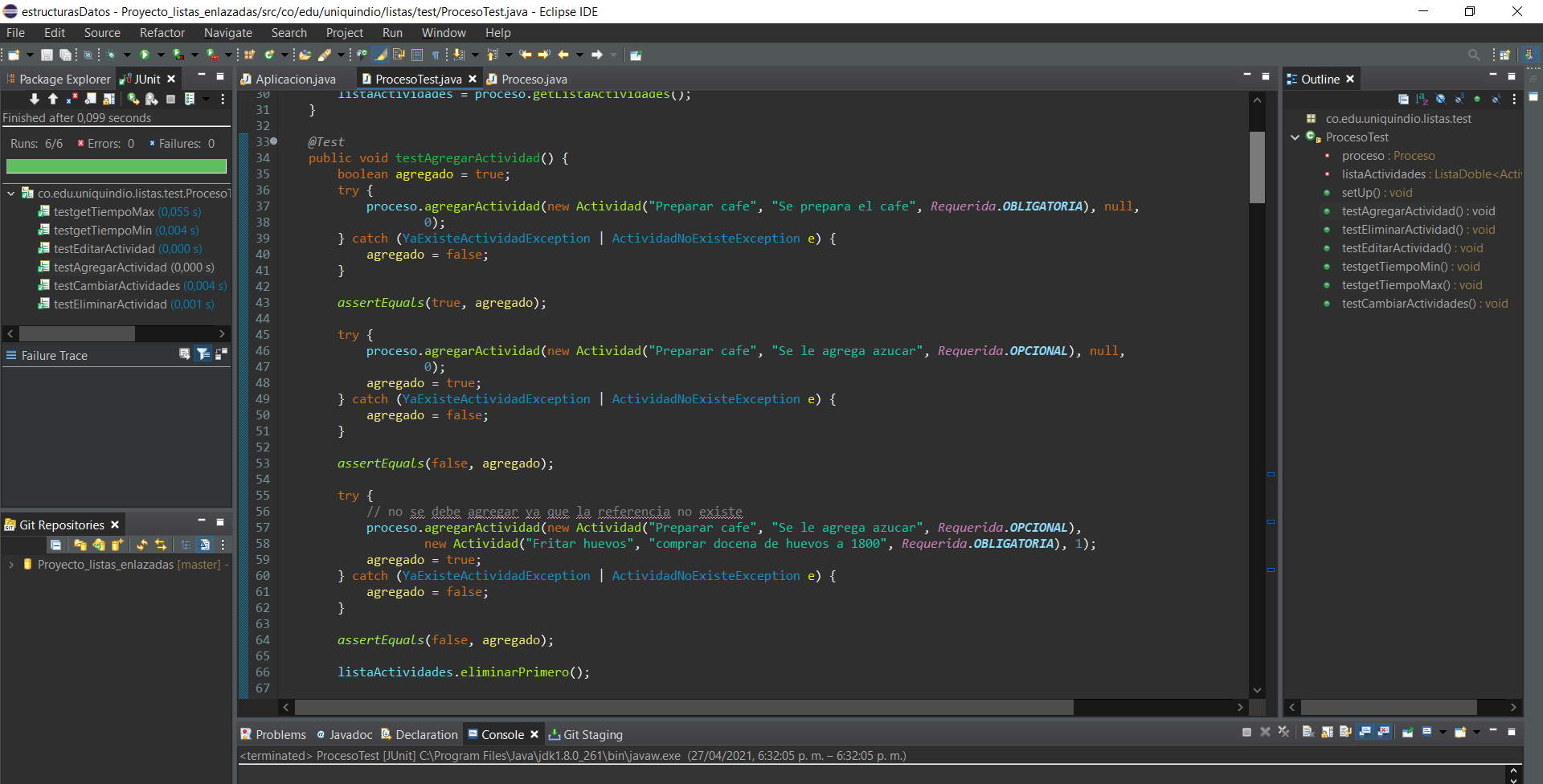
**Get Tiempo Mínimo:** En este método lo que hacemos es cargar nuevamente un contenedor y seguidamente obtener la lista de actividades de un proceso determinado, luego comparamos los valores de tiempo mínimo que retorna el método con los valores que son realmente, con ello confirmamos que dichos métodos funcionan bien. Primero verificamos los valores de tiempo mínimo de las actividades de un primer proceso y después verificamos los mismos valores pero de un segundo proceso.

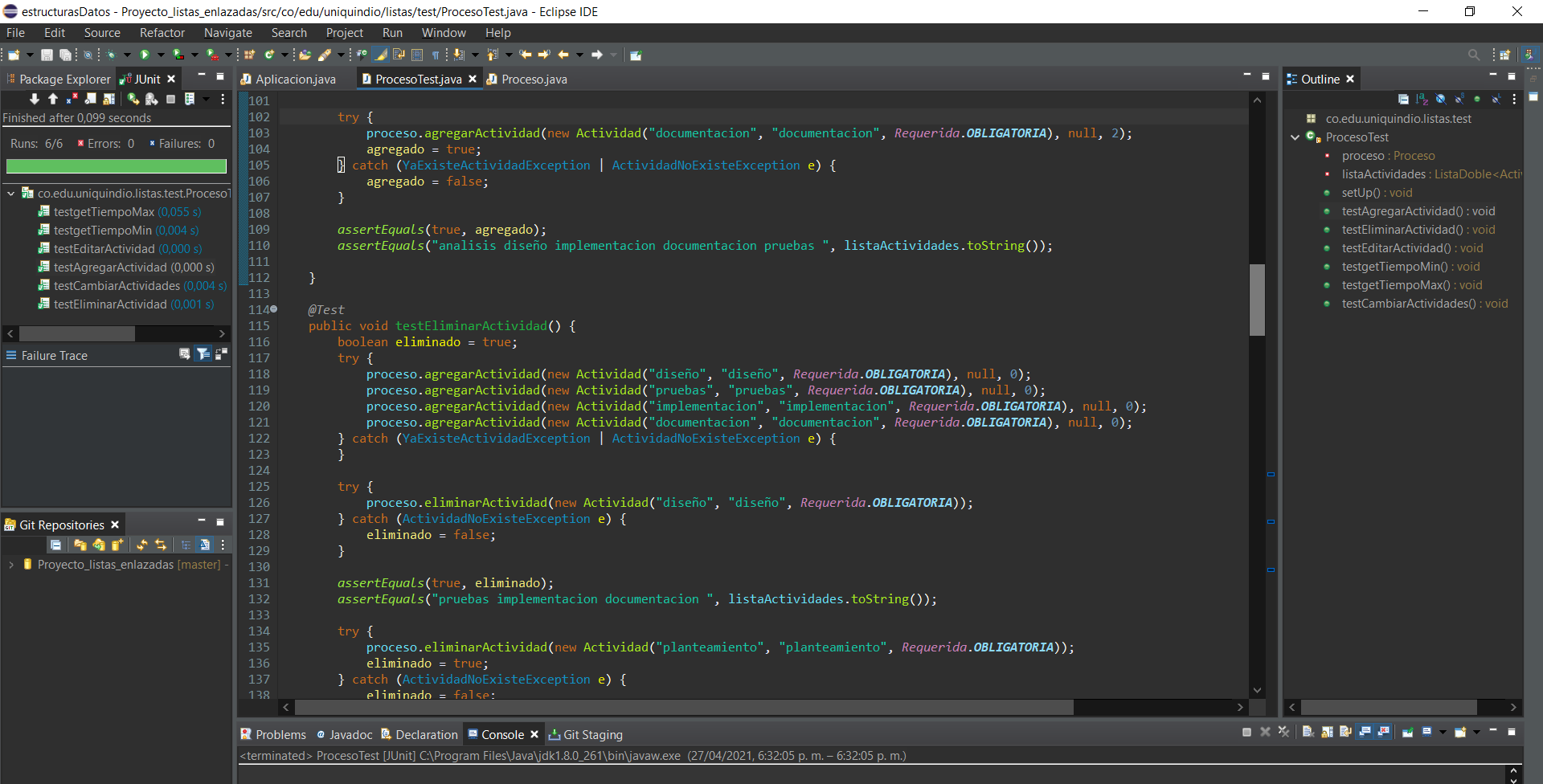


**Editar Actividad:** En este test inicialmente creamos una variable booleana que llamamos editado y la inicializamos en true, seguido de esto lo que hacemos es agregar actividades a un proceso anteriormente creado( agregamos 4 actividades con sus respectivos datos), después de tener las actividades agregadas al proceso lo que hacemos en intentar editar una actividad, llamamos al método editarActividad y le enviamos una actividad que ya existe en la lista de actividades del proceso y una actividad nueva, verificamos con la variable editado que efectivamente se pudo realizar la edición de la actividad, creamos una nueva variable para almacenar una actividad del proceso( en este caso la actividad 0 de la lista) y verificamos que la actividad si halla sido editada, con esto verificamos la funcionalidad correcta del método. Después, lo que hacemos es tratar de editar una actividad que ya no existe en la lista de actividades del proceso y lo que vemos es que evidentemente la variable editado se encuentra false porque el método entra en el catch ya que el método editarActividad nos lanza una exception por no existir la actividad que se quiere modificar en la lista, seguidamente lo que hacemos es verificar la lista de actividades con un método toString vemos que ninguna actividad ha sido modificada, finalmente lo que hacemos es intentar editar dos actividades y verificamos inicialmente con la variable booleana que efectivamente si se editaron y después verificamos con la lista que las dos actividades si fueron editadas.

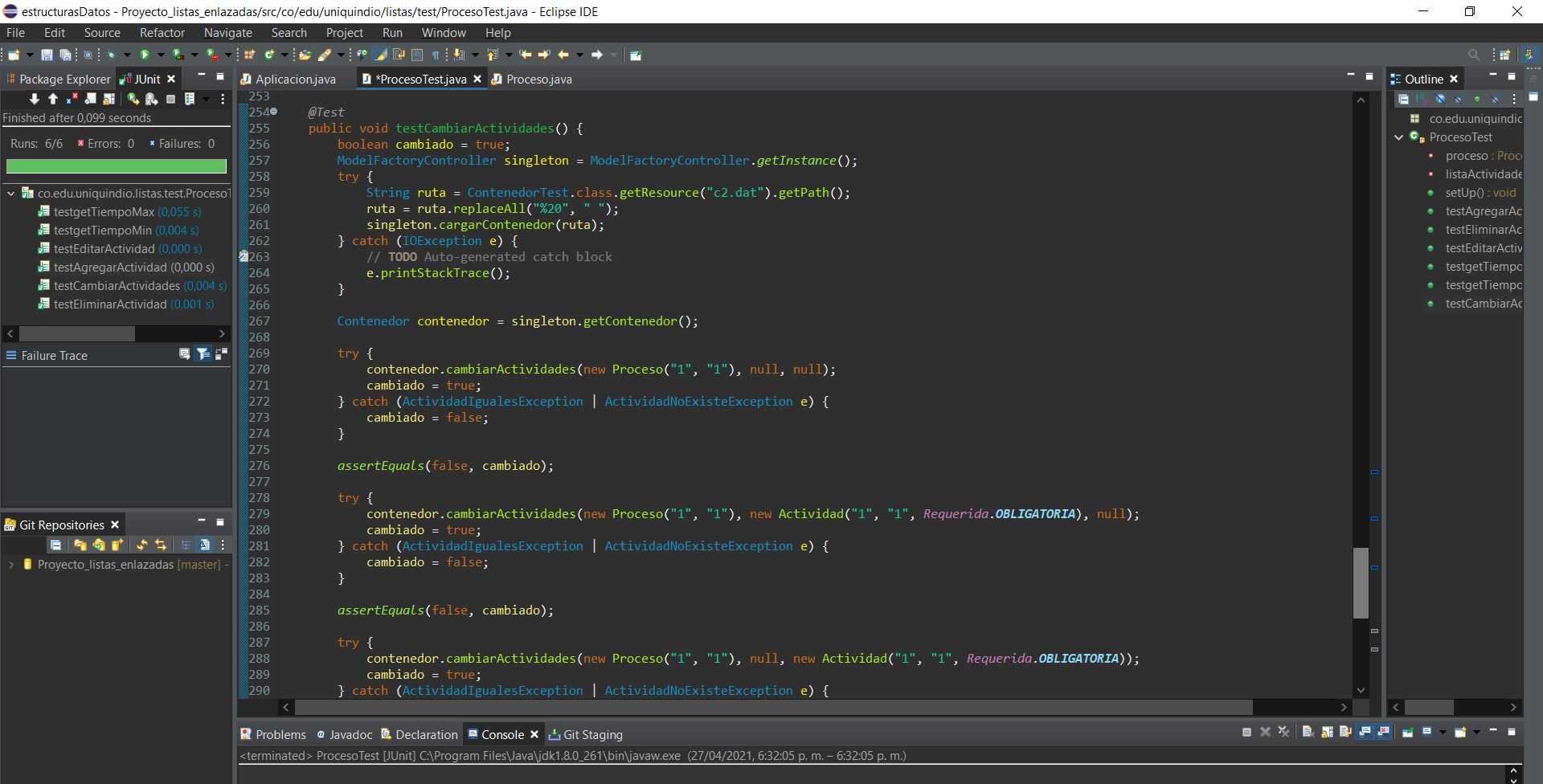


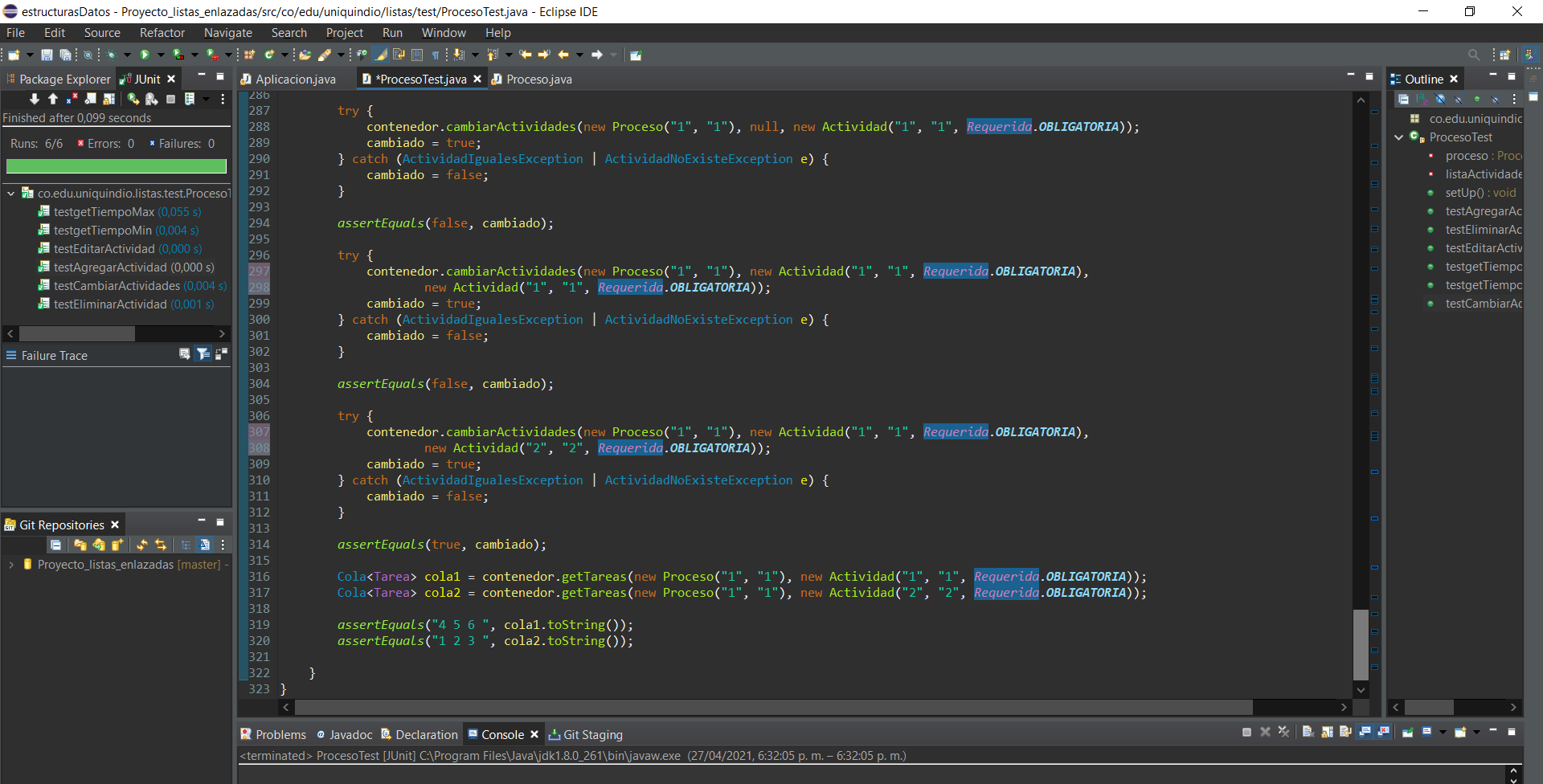
**Agregar Actividad:** En este test lo que hacemos es crear una variable booleana y la inicializamos en true para utilizarla posteriormente, luego lo que hacemos es llamar al método agregarActividad de la clase proceso y le enviamos una actividad con todos sus datos, le enviamos un null que seria la actividad anterior y una opción que en este caso es la opción 0 para agregar al final de la lista, luego verificamos con la variable booleana inicialmente creada que la actividad ha sido agregada correctamente, luego intentamos agregar la misma actividad pero el método nos lanza una exception por lo que entra en el catch y la variable agregado debe estar en false ya que no se pudo agregar la actividad, seguidamente verificamos que dicha variable se encuentra en false, luego intentamos agregar una actividad con la opción de agregar después de una actividad determinada y enviamos como actividad determinada una actividad que no existe en la lista por lo que nuevamente se lanza una exception y la variable agregado nuevamente debería de estar false, esto lo verificamos y notamos que evidentemente la variable esta false, luego eliminamos la primera actividad e intentamos agregar una actividad después de la última creada pero como la última actividad creada ha sido eliminada lo que hace el programa es agregarla al final de la lista, esto lo verificamos a continuación, no solo con la variable booleana si no también con la lista de actividades, luego agregamos dos actividades al final y seguidamente verificamos que se hayan agregado correctamente no solo con la variable booleana sino que también lista, después agregamos una actividad después de una actividad determinada y le enviamos la actividad que se quiere agregar y la actividad que se tomará como referencia para agregar la nueva actividad y verificamos nuevamente con la variable y con la lista que la actividad se ha agregado correctamente, finalmente agregamos una actividad con la opción de agregar después de la ultima actividad creada y lo verificamos con la variable y con la lista, ya con esto notamos que los métodos para agregar actividades funcionan bien.





**Cambiar Actividad:** En este método lo que hacemos primero es crear una variable booleana e inicializarla en true, seguidamente cargamos un contenedor que se encuentra guardado en un archivo con extensión “.dat” llamamos al singleton para que cargue dicho archivo enviándole la ruta, después obtenemos el contenedor y lo que hacemos es intentar cambiar dos actividades nulas por lo que el método nos lanza una exception y cambiamos el valor de la variable booleana creada inicialmente a false, verificamos que la variable si se encuentre en false y evidentemente se encuentra en false, luego intentamos cambiar una actividad que existe con una actividad nula y si se logra cambiar la actividad, cambiamos el valor de la variable booleana a true pero como hay una actividad nula entonces evidentemente la variable booleana se debería encontrar false por lo que miramos que si este false y lo verificamos con el método assertEquals, hacemos el mismo procedimiento anterior pero mandamos la actividad null en diferente posición es decir ( actividad, null ) y luego ( null, actividad ) verificamos que la variable este en false y evidentemente lo está, posteriormente intentamos cambiar dos actividades pero le enviamos la actividad dos veces por lo que se lanza una exception de ActividadIgualesException y la variable booleana se encuentra nuevamente en false porque no se pudo hacer el cambio, lo verificamos. Finalmente enviamos dos actividades existentes y además de eso distintas y verificamos que ahora la variable booleana si se encuentra en true porque si se pudo hacer el cambio de las actividades, creamos dos colas y almacenamos las tareas de las actividades que se cambiaron y por últimos las verificamos con un toString.





**Eliminar Actividad:** En este test lo que hacemos es crear una variable booleana e inicializarla en true y posteriormente agregar 4 actividades a un proceso, luego intentamos eliminar una actividad y le enviamos una actividad que a existe en la lista y si el método lanza una exception lo que hacemos es cambiar la variable booleana a false pero como en este caso se pudo eliminar la actividad, entonces esta variable debe encontrarse en true, verificamos que la variable se encuentre en true y además de esto verificamos también en la lista con un toString, después lo que hacemos es intentar eliminar una actividad que no existe por lo que el método lanza una exception y la variable booleana debe cambiarse a false, verificamos esto con el método assertEquals, finalmente intentamos eliminar dos actividades existentes en la lista y posteriormente verificamos que la variable este en true y verificamos con un toString que se hayan eliminado las dos actividades de la lista.

