Universidad de los Andes
Ingeniería de Sistemas y Computación

ISIS 1205 - Algorítmica y Programación 2

Enunciado Adivina Cuál

Cupi2

Objetivos

El objetivo de este ejercicio es que el estudiante comprenda y adquiera práctica en:

- El desarrollo de aplicaciones siguiendo un proceso incremental
- El manejo de estructuras recursivas
- El desarrollo de algoritmos recursivos
- El desarrollo de pruebas unitarias en junit para las clases del ejercicio.

Preparación

1. Visualice el demo.
2. Descomprima el proyecto.
3. Cree el proyecto en eclipse con el contenido del directorio n11_adivinaCual
4. Estudie la documentación del problema disponible en el directorio docs/specs del proyecto
   - La descripción del problema en Descripcion.doc.
   - Los requerimientos funcionales en RequerimientosFuncionales.doc
   - El modelo del mundo en modeloConceptual.jpg
   - El modelo de la interfaz en interfaz.jpg
   - El modelo de pruebas en pruebas.jpg
5. Asegúrese de tener activado el uso de aserciones para la ejecución del programa.
6. Defina una configuración de ejecución del proyecto que verifique las invariantes de las clases del mundo. Para esto, debe indicar como parámetro a la JVM la opción –ea, tal como se ha especificado en ejercicios anteriores
7. Realice el mismo proceso en cada una de las configuraciones de Junit

Proceso de desarrollo

Parte 1 – Trabajando con un árbol inicializado

Para el desarrollo de esta parte, SIEMPRE que ejecute el programa para pruebas debe responder que SI QUIERE EMPEZAR CON UN ÁRBOL LLENO cuando inicie la aplicación.

Aunque en un principio Ud. no verá las imágenes de los animales, el árbol está lleno, la aplicación funciona, pues hace las preguntas y adivina los animales.

Desarrollando y probando métodos analizadores sobre árboles

1. Implemente y pruebe el requerimiento funcional de calcular la altura del árbol, basándose en el cálculo del peso del árbol que ya está implementado. Para esto, complete los métodos correspondientes en: PanelBotones, InterfazAdivinaCual, Juego, Pregunta y PreguntaTest. **NOTA**: Para la ejecución de las pruebas, entienda primero bien los tres escenarios planteados (ver la documentación de los métodos involucrados en docs/api) y cuáles son los resultados esperados en cada caso.
2. Complete la implementación del requerimiento funcional de mostrar el recorrido en Inorden, al cual únicamente le falta mostrarlo en el diálogo correspondiente. Debe entonces desarrollar toda la clase DialogoRecorrido (y su correspondiente PanelRecorrido, que contiene un JTextArea con scroll) y lograr que ésta muestre el recorrido en inorden generado por el mundo.
3. Implemente y pruebe el requerimiento funcional de mostrar el recorrido en Preorden del árbol, basado en la implementación del recorrido en Inorden. De manera similar que con la altura, complete los métodos correspondientes en: PanelBotones, InterfazAdivinaCual, Juego, Pregunta y PreguntaTest
4. Implemente y pruebe el requerimiento funcional de mostrar el recorrido en Postorden del árbol, basado en la implementación del recorrido en Inorden. De manera similar que con la altura, complete los métodos correspondientes en: PanelBotones, InterfazAdivinaCual, Juego, Pregunta y PreguntaTest
5. Pruebe todos los requerimientos anteriores como usuario de la interfaz principal

Visualizando los animales “disponibles”

Como ya se habrá dado cuenta, los animales están en las hojas del árbol. Para poder pintarlos en el DialogoAnimales, es necesario enviarle los animales que están en el árbol (ver el método mostrarDialogo ()). El inconveniente es que el método darAnimales no está implementado…

6. Implemente el método darAnimales de la clase Juego (que funciona de manera similar a los recorridos, pues devuelve una lista de animales..)
7. Implemente el método darHojas de la clase Pregunta, que genera la lista de hojas del árbol
8. Complete las pruebas de darHojas en la clase PreguntaTest
9. Ejecute el programa y ya debe tener la interfaz mostrando las imágenes de los animales existentes en el árbol

Adicionando nuevos animales al árbol

Como ya también se habrá dado cuenta, cuando la aplicación no adivina el animal que Ud. pensó, no pasa nada, pues la pregunta y el nuevo animal no son agregados al árbol. Esto es debido a que el requerimiento R6 no está implementado completamente. Además, como vamos a modificar el árbol, debemos tener mucho cuidado para no generar inconsistencias en el árbol.

10. Estudie el diálogo DialogoAgregarPregunta y su relación con el PanelAgregarPregunta y la InterfazAdivinaCual, con el fin de entender bien el flujo de control y de datos para poder implementarlo completamente. Empiece por seleccionarRespuestaNegativa de la InterfazAdivinaCual y deberá terminar en el método agregarPregunta de (también) InterfazAdivinaCual.
11. Implemente el método esCompleto de la clase Pregunta, que verifica si el árbol tiene la estructura adecuada: Los nodos tienen ó 2 ó ningún hijo.
12. Con base en el método anterior, implemente el método verificarInvariante de la clase Pregunta, que además verifica que las hojas representan animales y no preguntas.
13. Implemente el método agregarPregunta de la clase Pregunta, de acuerdo con la documentación suministrada.
14. Implemente el método buscarAnimal de la clase Pregunta, de acuerdo con la documentación suministrada, que va a ser muy útil para verificar la invariante del juego completo…
15. Verifique que todas las pruebas de la clase PreguntaTest ejecutan correctamente
16. Implemente el método agregarPregunta de la clase Juego

17. Verifique que todas las pruebas de las clases JuegoTest y PreguntaTest ejecutan correctamente
18. Pruebe la funcionalidad de este requerimiento agregando nuevos animales al árbol por medio de la interfaz.

Parte 2 – Trabajando con un árbol vacío

19. Empiece la ejecución del programa contestando que quiere empezar con un árbol vacío
20. Arme su propio zoológico !!!!