Nelson Matos 60483

Design Patterns

Memento

Illustrating code snippet:

*private* Document saveFile() *throws* IOException {  
 Document doc = myManager.getDocumentManager().newAutosaveDocument();  
 doc.write();  
 *return* doc;  
}

*private void* restoreDocument(Document document) *throws* IOException, DocumentException {  
 myManager.getProject().restore(document);  
}

Exact location:

* saveFile() -> ganttproject/src/main/java/net.sourceforge.ganttproject/undo/UndoableEditImpl (line 60)
* restoreDocument() -> ganttproject/src/main/java/net.sourceforge.ganttproject/undo/UndoableEditImpl (line 108)

Explanation:

This is an usual and multi-used design pattern in the software engineering companies. Nowadays most of the productivity applications such as excel,

word, etc., let the user save its current work so it can be accessed in the future.  
The two methods above express these actions.

Proxy

Illustrating code snippet:

*/\*\*  
 \* Gets the username used to authenticate to the storage container  
 \*  
 \* @return username  
 \*/  
public* String getUsername();

*/\*\*  
 \* Gets the password used to authenticate to the storage container  
 \*  
 \* @return password  
 \*/  
public* String getPassword();

Exact location:

* getUsername() -> ganttproject/src/main/java/net.sourceforge.ganttproject/document/Document (line 122)
* getPassword() -> ganttproject/src/main/java/net.sourceforge.ganttproject/document/Document (line 129)

Explanation:

This design pattern is usually used when we need to make a bridge between a client and the real subject at hand. This “bridge” (proxy) is helpful because it can perform somewhat like the real subject but in a simplified way. In this case we can see that the proxy controls access of the real document as in, the user needs to specify its name and password to have access to it.

Iterator

Illustrating code snippet:

*private* DefaultMutableTreeTableNode buildTree() {  
  
 DefaultMutableTreeTableNode root = *new* DefaultMutableTreeTableNode();  
 List<HumanResource> listResources = myResourceManager.getResources();  
 Iterator<HumanResource> itRes = listResources.iterator();  
  
 *while* (itRes.hasNext()) {  
 HumanResource hr = itRes.next();  
 ResourceNode rnRes = *new* ResourceNode(hr); *// the first for the resource* root.add(rnRes);  
 }  
 *return* root;  
}

Exact location:

* buildTree() -> ganttproject/src/main/java/net.sourceforge.ganttproject/ResourceTreeTableModel (line 112)

Explanation:

This method makes use of an iterator that helps us traverse a collection of objects without exposing its underlying representation.