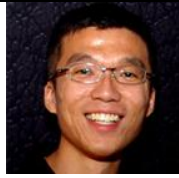
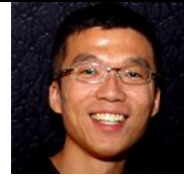


<b>Name:</b> Koh Eng Tat Desmond	
<b>Matric:</b> U077130H	



## **User's Guide** *[also useful for developers]*

### **1. Introduction**

The purpose of FileBot is to create random directory and file structure in a Windows Filesystem by allowing user to specify parameters (see section 2). In addition, FileBot is able to modify existing directories and files (including file attributes) as well. Lastly, FileBot allows user to save an existing directory structure and restore it in other directory for the purpose of creating a 'realistic' structure.

All files and directories will be created in user-specified root folder.

### **2. Parameters used in Generating Directory Structure**

#### **1. Directory Depth**

Has a minimum value of 0 (i.e. only files may be created in the root folder). At least one directory with specified depth will be created.

#### **2. Maximum sub-directories**

For each directory (including the root directory if the Directory Depth (see 2.1) specified is greater than 0), a random number of sub-directories will be created up to the maximum specified.

#### **3. Average Files**

For each directory (including the root folder) a random number of files up to the maximum specified will be created.

#### **4. Creation Date**

All files created will have the specified creation date set explicitly for consistency purposes (this is because if a file with the same name existed before, its creation date might not be correct).

#### **5. Maximum/Minimum File size**

For every file created, its file size will be randomly distributed between the specified values.

## 6. Directory / File naming

### Letters:

Each name will only consists of alphabets (the first character will be upper case)

### Letters/Symbols:

In addition to letters, some symbol characters will be included as well.

### Unicode:

All file and directory name will consist of randomly generated Unicode characters.

## 3. Actions in Modifying Directory Structure

All selected/checked actions will be performed independently on each file with specified probability. e.g. If only "Set as Read-only" and "Set as System File" are selected and the probability set to 10%, the probability of a file being read-only, system file, read-only and system file is 10%, 10% and 1% respectively. User can set the probability to 100% to specify certain actions be performed on all the files.

## 4. Saving/Restore Directory Structure

This feature is best explained with an example:

If a user saved a directory structure of the directory: C:\Documents, and restored it to directory D:\New, all the files and directories in C:\Documents will be 'copied' to D:\New, with the exception that all the file contents will not be the same. All file and directory attributes including the creation/last modified/last accessed time is also preserved. This feature essentially replicates an existing directory and file structure of a directory in other location. Since file contents are not copied, this operation is faster than the traditional 'copy-and-paste' and also preserves file attributes (useful for replicating test data to test synching problems).

## Advanced Settings

FileBot by default will select the application startup directory as the root folder each time it starts. To change the default root directory each time FileBot starts, edit FileBot.exe.config file as follows:

```
<applicationSettings>
  <CodeDroids.FileBot.Properties.Settings>
    <setting name="RootDir" serializeAs="String">
      <value>C:\Your Default Path Here</value>
    </setting>
  </CodeDroids.FileBot.Properties.Settings>
</applicationSettings>
```

## Quick User Guide

The screenshot shows the FileBot application window. It has a title bar with standard Windows window controls. The main interface includes a 'Root Directory' field with a 'Browse' button. Below this are three tabs: 'Generate', 'Modify', and 'Save/Restore'. The 'Generate' tab is active, showing settings for 'Directories' and 'Files'. The 'Directories' section has 'Directory Depth' and 'Maximum sub-directories (within a directory)' both set to 3. The 'Files' section has 'Average files (within a directory)' set to 5, 'Creation Date' set to '20/1/2010 8:03:40 PM', 'Min File Size (KB)' set to 10, and 'Max File Size (KB)' set to 1024. At the bottom, the 'Directory/File Naming' section has three radio buttons: 'Letters' (selected), 'Letters/Symbols', and 'Unicode Characters'. A 'Generate' button is at the bottom right. A marquee progress bar is at the very bottom. Callout boxes provide details for each of these elements.

Root Directory where random files and directories will be created in.

Maximum depth of directory structure that will be created. Minimum value of 0.

All files created will have file size randomly distributed between specified

Type of directory name and filename that will be generated.

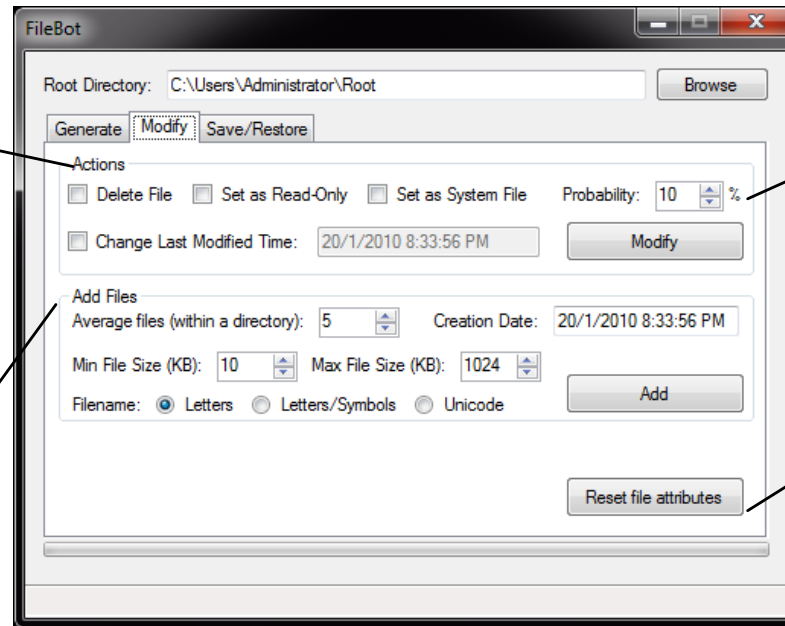
The number of sub-directories within each directory will be randomly distributed between 0 and the number specified here.

All directories and files created will have the date and time specified here.

Marquee progress bar to indicate program activity.

Actions that are selected will be randomly performed independently on each file with

An average of specified number of files will be added to each directories and subdirectories.



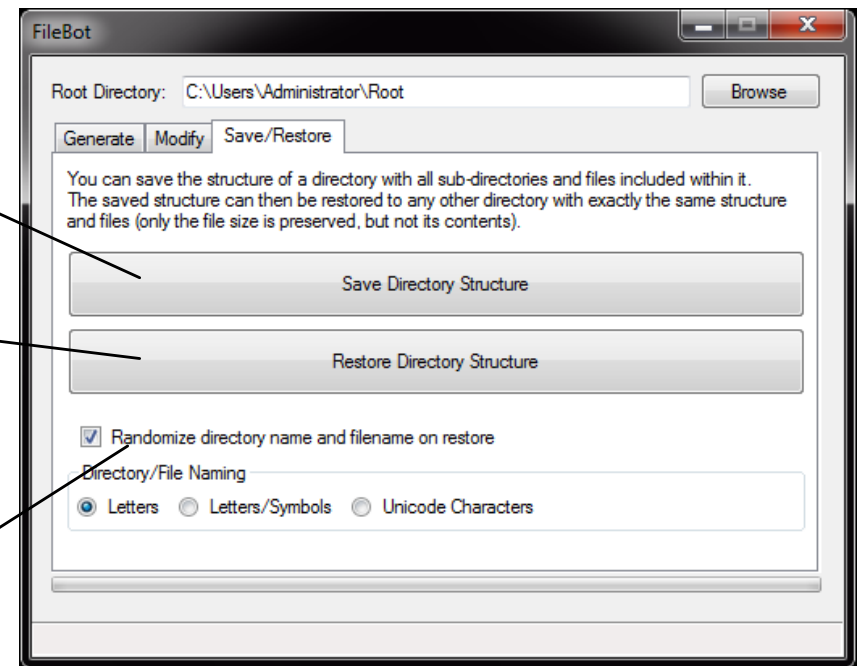
Set to 100 to perform selected action (e.g. change last modified time) on all files and directories.

All files and directories will have all file attributes cleared [i.e. not hidden, not system file etc.] Last modified date/time will be set to date/time action is performed.

When clicked, FileBot will prompt user to select a folder to save its structure (i.e. all files and sub-directories within it).

When clicked, FileBot will prompt user to select a previously saved directory structure and a folder to restore it into. This is similar to a 'copy-and-paste' of the saved directory, just that the file contents will not be the same. Therefore it will be much faster.

User can choose to randomize the names of directories and files being restored.



## **Developers's Guide** *[Refer to User's Guide for explanation for features]*

### **Explanation of Selected Classes and Methods**

#### **Class name:** `RandomGenerator`

This class is used extensively to generate random integers and strings. As such a static object `random` is accessible from the class to be used. Creating other `Random` object is not recommended as the same sequence of integer could be generated by different instances of `Random` object if they are created in about the same time.

The length of the file and directory name is determined by the following statement:

```
random.Next(0, 6) + random.Next(1, 7)
```

Therefore its length varies from 1 character long to 13 characters long, with the mode of the distribution being 7. Filenames are essentially directory names concatenated with 3 letter extension.

#### **Class name:** `Bot`

This class is called by the UI to perform almost all actions. It also contains a very useful and reusable method with the following method signature:

```
TraverseDirectoryTree(string path, bool dirOnly, TraverseDirectoryCallback  
callback)
```

In many operations, the directories have to be traversed recursively, but for each operation, different action has to be performed, hence, this method has a delegate as one of the parameter. This method is used to adding files, changing and resetting file attributes.

#### **Class name:** `FileNode` / `DirectoryNode`

These two classes represents a file and a directory in a file system. They are marked as serializable so that they can be persisted onto disk. More fields can be added to these 2 classes to contain more information about the file or directory as well.

#### **Class name:** `UpdateEventArgs`

As some operations might take some time to complete, in order to update the UI of current progress, certain classes can publish an Update event and make use of this class.

While there is a `progress` field that is meant to represent the progress of action, it is not used in this being yet, reason being it is difficult to know the percentage of work completed so far. This is because many of the operations are either non-deterministic (they are 'random'), or that all directories have to be traversed recursively first (which is unnecessary). This is also a reason why a marquee progress bar is provided to indicate program activity rather than progress of activity completed.

### **Possible Improvements / Known Issues**

There are no data validation checks implemented to check the parameters supplied from the UI are valid (e.g. max value  $\geq$  min value, date time format valid etc.). This is because the application is intended for advanced users for testing purposes. However, it could be made more robust by implementing the checks.

As many of the files and directories are generated non-deterministically, there is no way for user to know the changes that has been made to the filesystem in an efficient way. As such, it will be useful for FileBot to log all operations so that user will be able to review them should it be necessary.

Lastly, for the random generation of Unicode names, there is still a probability that an illegal path name might be generated. In addition, some Unicode characters are character modifier, and hence it might result in an exception being thrown.

## Class Diagram

