File Genx – Design Documentation	
The purpose of this document is to describe each component in the software design.	
Component	**Description**
MainWidget	An instance of this class is the starting point of the application. **Attributes:** **GLADEPATH**: a static constant str object that holds the path to the GUI .glade file used to build the GUI using the Gtk.Builder class; **builder:** an instance of the Builder class, used to build the Glade GUI layout. Created with the object. **handler:** an instance of the Handler class, used to connect signals to the layout. Created with the object. **res:** an instance of the Res class, used to bind resources (strings, images, etc) to the GUI layout. Created with the object. **Methods:** **bindRes():** bind the resources to the layout using **res**. This method accesses **res.resources** dictionary and makes calls to **builder.get_object()** to set the resources. **bindHandler():** connect the event listeners to the GUI layout. **launch():** get the main window from **builder** and display it. **getBuilder():** returns **builder**.
Builder	Subclass of Gtk.Builder class. **Methods:** **getInputs():** read the inputs in the GUI and return a dictionary. The dictionary has template: { “option-name”, “value” }
Gtk.Builder	Gtk.Builder is a class form the Gtk library. For more information about it, please refer to Gtk documentation.
Handler	Handler is a singleton class, whose methods are signal listeners. These methods are not declared in the UML diagram because it is not possible to know the exact number and names of event listeners since that is something GUI layout dependent. However, Handler will always have a static method that will construct it called **getInstance().** Thus Handler instances should be created using **Handler.getInstance()**. **Attributes:**

	uniqueInstance: static variable that holds the unique instance **builder:** Builder instance passed as argument to the constructor.
Res	Res is a singleton class. Thus, just like **Handler**, this class have a static **getInstance()** method. **Attributes:** **uniqueInstance:** static variable that holds the unique instance. **PATH:** str object that holds the path to the resources JSON file. **Strings:** dictionary that holds str type resources. The template is: { “object-name”: [“method-name”, “string-value”] }. Where object-name is the name of the object in the GUI layout, method-name is the Gtk.Widget method that changes the text of that widget. **Methods:** **loadResources():** get the JSON resources file and populate the attributes.
Generator	An instance of this class is created when the user clicks the generate button on the GUI. Thus an event listener from a Handler instance creates an object of Generator class. **Attributes:** **options:** holds the options for generation that the user have input. The dictionary has template: { “option-name”, “value” }. **contentGenerator:** an instance of the ContentGenerator class that is created when the Generator object is created. **Methods:** **generate():** instantiate **File** object and call its write() method using **contentGenerator**. This repeats according to **options**. **configureCotentGenerator():** set up the **contentGenerator** according to **options**. **translateInputs(dict inputs):** translate inputs dictionary to a more simple to deal with dictionary that will be stored in **options**.
File	An instance of this class is used as interface to create and persist files. **Attributes:** **suffix:** a static int that starts with value 0 and is incremented every time a call to **write()** occurs. This variable is used to control the

	generate file name. **path**: a string that holds where the new file will be created. **Methods:** **write(str prefix, str content):** try to create and persist file with name **prefix-suffix.extension** populated with **content**.
ContentGenerator	Abstract class. **Methods:** **factory(str type):** instantiate and return a **ContentGenerator** object using a subclass constructor. Parameter **type** can be “web” for **WebContentGenerator** object or “rand” for **RandomContentGenerator** object. **generate()**: abstract method. Generate content. **configure(dict options)**: abstract method. Configure the generator attributes
WebContentGenerator	Subclass of **ContentGenerator**. Used to get web content from some source. **Attributes:** **source:** a str object that is the web source URL. **SOURCES:** a static const dictionary that holds source name as key and source URL as value: { “source-name”, “source-URL” }. **n:** an int that describes how much of **mode** must be generated. **mode:** an str that describes how to count: “char” per character, “par” per paragraph, “line” per line. **Methods:** **generate():** try to connect to web source and get content. Return a str object. **configure(dict options):** set **source**, **n** and **mode**.
RandomContentGenerator	Subclass of **ContentGenerator**. Used to generate random content. **Attributes:** **n:** an int that describes how much of **mode** must be generated. **mode:** an str that describes how to count: “char” per character, “par” per paragraph, “line” per line.

	Methods: **generate():** generates random content. Return a str object. **configure(dict options):** set **n** and **mode**.