Architecture Design

For KDD-Research Entity Search Tool (KREST)

Version 2.1

Submitted in partial fulfillment of the Masters of Software Engineering degree.

Eric Davis
CIS 895 – MSE Project
Department of Computing and Information Sciences
Kansas State University
# Change Log

<table>
<thead>
<tr>
<th>Version #</th>
<th>Changed By</th>
<th>Release Date</th>
<th>Change Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version 1.0</td>
<td>Eric Davis</td>
<td>01/19/08</td>
<td>Initial Release</td>
</tr>
<tr>
<td>Version 2.0</td>
<td>Eric Davis</td>
<td>03/06/08</td>
<td>Updating Model and Formal Specification</td>
</tr>
<tr>
<td>Version 2.1</td>
<td>Eric Davis</td>
<td>3/17/08</td>
<td>Adding FileLoader class</td>
</tr>
</tbody>
</table>
## Table of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Package View</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>KREST Application Package</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>Controller Package</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>KrestController Class</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>KrestAboutDialog Class</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>FileLoader Class</td>
<td>5</td>
</tr>
<tr>
<td>7</td>
<td>WebCrawler Class</td>
<td>5</td>
</tr>
<tr>
<td>8</td>
<td>SiteVisitor Class</td>
<td>6</td>
</tr>
<tr>
<td>9</td>
<td>ThreadController class</td>
<td>6</td>
</tr>
<tr>
<td>10</td>
<td>HTTPReader Class</td>
<td>7</td>
</tr>
<tr>
<td>11</td>
<td>WebSearcher Class</td>
<td>7</td>
</tr>
<tr>
<td>12</td>
<td>EntitySearcher Class</td>
<td>7</td>
</tr>
<tr>
<td>13</td>
<td>KrestView Class</td>
<td>8</td>
</tr>
<tr>
<td>14</td>
<td>CrawlerObserver Class</td>
<td>8</td>
</tr>
<tr>
<td>15</td>
<td>SearchObserver Class</td>
<td>9</td>
</tr>
<tr>
<td>16</td>
<td>EntityObserver Class</td>
<td>9</td>
</tr>
<tr>
<td>17</td>
<td>KrestModel Class</td>
<td>10</td>
</tr>
<tr>
<td>18</td>
<td>KrestObjectLibrary Class</td>
<td>10</td>
</tr>
<tr>
<td>19</td>
<td>WebObject Class</td>
<td>10</td>
</tr>
<tr>
<td>20</td>
<td>Webpage Class</td>
<td>11</td>
</tr>
<tr>
<td>21</td>
<td>KrestEntity Class</td>
<td>11</td>
</tr>
<tr>
<td>22</td>
<td>AddressEntity Class</td>
<td>11</td>
</tr>
<tr>
<td>23</td>
<td>EmailEntity Class</td>
<td>11</td>
</tr>
<tr>
<td>24</td>
<td>FaxEntity Class</td>
<td>12</td>
</tr>
<tr>
<td>25</td>
<td>PhoneEntity Class</td>
<td>12</td>
</tr>
<tr>
<td>26</td>
<td>ZipEntity Class</td>
<td>12</td>
</tr>
<tr>
<td>27</td>
<td>OverarchingEntity Class</td>
<td>13</td>
</tr>
<tr>
<td>28</td>
<td>Web Crawl Sequence Diagram</td>
<td>14</td>
</tr>
<tr>
<td>29</td>
<td>Web Search Sequence Diagram</td>
<td>15</td>
</tr>
<tr>
<td>30</td>
<td>Entity Search Sequence Diagram</td>
<td>16</td>
</tr>
</tbody>
</table>
# Table of Contents

**Change Log** .................................................................................................................................................. ii  
1. Introduction .......................................................................................................................................................... 1  
   1.1 Background .................................................................................................................................................. 1  
   1.2 References .................................................................................................................................................. 1  
2. KDD-Research Entity Search Tool Architecture .............................................................................................. 1  
   2.1 Package View .............................................................................................................................................. 2  
   2.2 Application Package .................................................................................................................................... 2  
      2.2.1 Class Description .............................................................................................................................. 2  
      2.2.1.1 KrestApplication .......................................................................................................................... 2  
   2.3 Controller Package ....................................................................................................................................... 3  
      2.3.1 Class Description .............................................................................................................................. 3  
      2.3.1.1 KrestController .......................................................................................................................... 3  
      2.3.1.2 KrestAboutDialog ......................................................................................................................... 4  
      2.3.1.3 FileLoader .................................................................................................................................. 5  
      2.3.1.4 WebCrawler ................................................................................................................................. 5  
      2.3.1.5 SiteVisitor .................................................................................................................................. 5  
      2.3.1.6 ThreadController .......................................................................................................................... 6  
      2.3.1.7 HTTPReader ................................................................................................................................. 6  
      2.3.1.8 WebSearcher ................................................................................................................................. 7  
      2.3.1.9 EntitySearcher ............................................................................................................................... 7  
   2.4 View Package ................................................................................................................................................ 7  
      2.4.1 Class Description .............................................................................................................................. 8  
      2.4.1.1 KrestView .................................................................................................................................. 8  
      2.4.1.2 CrawlerObserver ......................................................................................................................... 8  
      2.4.1.3 SearchObserver ........................................................................................................................... 8  
      2.4.1.4 EntityObserver ............................................................................................................................. 9  
   2.5 Model Package ............................................................................................................................................. 9  
      2.5.1 Class Description .............................................................................................................................. 9  
      2.5.1.1 KrestModel ................................................................................................................................. 9  
      2.5.1.2 KrestObjectLibrary ..................................................................................................................... 10  
      2.5.1.3 WebObject ................................................................................................................................ 10  
      2.5.1.4 Webpage ................................................................................................................................... 10  
      2.5.1.5 KrestEntity ................................................................................................................................ 11  
      2.5.1.6 AddressEntity ............................................................................................................................ 11  
      2.5.1.7 EmailEntity ................................................................................................................................. 11  
      2.5.1.8 FaxEntity .................................................................................................................................. 12  
      2.5.1.9 PhoneEntity ................................................................................................................................. 12  
      2.5.1.10 ZipEntity ................................................................................................................................ 12  
      2.5.1.11 OverarchingEntity ................................................................................................................... 12  
   2.6 Sequence Diagrams ..................................................................................................................................... 13  
      2.6.1 User performs a web crawl .............................................................................................................. 13  
      2.6.2 User performs a web search ............................................................................................................ 14  
      2.6.3 User performs an entity search ....................................................................................................... 15
3. Formal Specification........................................................................................................ 16
1. Introduction

The purpose of this document is to provide an architectural design of the KDD-Research Entity Search Tool (KREST). The document will illustrate class diagrams and sequence diagrams. The purpose of each class in the diagrams will be given. Also, a formal specification of the web crawler portion of the project will be given in Section 3.

1.1 Background

The purpose of KREST is to provide a multifunctional web search tool that runs as a standalone application. The project allows the user to perform a web crawl, to perform a basic web search over the crawled pages, and to perform an entity search over the crawled pages. The project also allows the user to perform web searches and entity searches based on datasets that can be loaded into the tool.

1.2 References


2. KDD-Research Entity Search Tool Architecture
2.1 Package View

The KREST project will follow the Model-View-Controller (MVC) architecture, with an application class to kick off the project. This allows the screen to be easily updated via changes to the model.

![Package View Diagram](image)

**Figure 1: Package View**

2.2 Application Package

![Application Package Diagram](image)

**Figure 2: KREST Application Package**

2.2.1 Class Description

2.2.1.1 KrestApplication

The KrestApplication class is a very simple class that will be used to start up the program. It will startup the KrestController and makes it visible.
2.3 Controller Package

![Class Diagram](image)

Figure 3: Controller Package

2.3.1 Class Description

2.3.1.1 KrestController
The KrestController class is the class responsible for getting all of the other parts up and running. It is responsible for signaling the web crawls, web searches, and entity searches to begin processing. It also controls displaying the form.
2.3.1.2 KrestAboutDialog

The KrestAboutDialog class is a Dialog that displays information about the KREST application.
2.3.1.3 FileLoader
The FileLoader class is responsible for loading in previously retrieved data into the application.

### FileLoader

| - fileToLoad : string  
| - library : KrestObjectLibrary  
| - parent : KrestController  
| + FileLoader(in newSearchString : string)  
| - readInWebBaseFile()  

**Figure 6: FileLoader Class**

2.3.1.4 WebCrawler
The WebCrawler class is responsible for setting up everything needed for a web crawl, and starting the process to do it.

### WebCrawler

| + siteVisitor : SiteVisitor  
| + debugSwitch : bool  
| + WebCrawler()  
| + beginCrawling(pageAddress : string, in searchString : string, in maxToCrawl : int, in minBacklinks : int, in filePath : string, in maxDepth : int)  
| + stopCrawling()  
| + performReset(in partial : bool)  
| + getMatches() : void  
| + getSiteVisitorThreads() : int  
| + getSiteVisitor() : SiteVisitor  

**Figure 7: WebCrawler Class**

2.3.1.5 SiteVisitor
The SiteVisitor is responsible for visiting individual web pages. Each instance of the SiteVisitor class is a thread that represents a different web page being visited.
### 2.3.1.6 ThreadController

The ThreadController class is responsible for ensuring that only up to the maximum number of specified web crawling threads are running at any one time. The web crawling threads are instances of the SiteVisitor class. The ThreadController maintains tickets to keep track of which threads are allowed to run. If a thread has a ticket, it is allowed to run, otherwise it sleeps while waiting to grab a ticket.

<table>
<thead>
<tr>
<th>ThreadController</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ticketDatabase : int</td>
</tr>
<tr>
<td>+crowdSize : int</td>
</tr>
<tr>
<td>+maxCrowdSize : int</td>
</tr>
<tr>
<td>+ThreadController(in maxCrowdSizes : int)</td>
</tr>
<tr>
<td>+getTicket() : int</td>
</tr>
<tr>
<td>+returnTicket(in ticket : int)</td>
</tr>
<tr>
<td>#findFreeTicket() : int</td>
</tr>
</tbody>
</table>

### 2.3.1.7 HTTPReader

The HTTPReader class is responsible for downloading the text of a given web page. If the given web page does not exist, it will throw an exception.

<table>
<thead>
<tr>
<th>HTTPReader</th>
</tr>
</thead>
<tbody>
<tr>
<td>+HTTP_PORT : int = 80</td>
</tr>
<tr>
<td>+HTTPReader()</td>
</tr>
<tr>
<td>+downloadWWWPage() : String</td>
</tr>
</tbody>
</table>
2.3.1.8 WebSearcher

The WebSearcher class is responsible for setting up everything needed for a web search, and starting the process to do it.

```
WebSearcher
- searchString : string
- matches
+ WebSearcher(in newSearchString : string)
  + beginSearch()
  - findMatches()
```

2.3.1.9 EntitySearcher

The EntitySearcher class is responsible for setting up everything needed for a web crawl, and starting the process to do it.

```
EntitySearcher
- searchString : string
- entityType
- entityMatches
+ EntitySearcher(in newSearchString : string)
  + beginSearch()
  - findMatches()
```

2.4 View Package
2.4.1 Class Description

2.4.1.1 KrestView
The KrestView class is an abstract class that can be implemented by the CrawlerObserver, the SearchObserver, and the EntityObserver classes. It is used to update the display based on changes from the model.

<table>
<thead>
<tr>
<th>KrestView</th>
</tr>
</thead>
<tbody>
<tr>
<td>- panel</td>
</tr>
<tr>
<td>- crawler : CrawlerObserver</td>
</tr>
<tr>
<td>- search : SearchObserver</td>
</tr>
<tr>
<td>- entity : EntityObserver</td>
</tr>
<tr>
<td>+ KrestObserver()</td>
</tr>
</tbody>
</table>

Figure 13: KrestView Class

2.4.1.2 CrawlerObserver
The CrawlerObserver class is responsible for updating the screen when the model changes due to web crawling.

<table>
<thead>
<tr>
<th>CrawlerObserver</th>
</tr>
</thead>
<tbody>
<tr>
<td>- panel</td>
</tr>
<tr>
<td>- startCrawlingField</td>
</tr>
<tr>
<td>- logField</td>
</tr>
<tr>
<td>- maxSitesField</td>
</tr>
<tr>
<td>- maxDepthField</td>
</tr>
<tr>
<td>- currentlyCrawlingField</td>
</tr>
<tr>
<td>- crawledURLsTextField</td>
</tr>
<tr>
<td>- queuedSitesTextField</td>
</tr>
<tr>
<td>- crawlProgressBar</td>
</tr>
<tr>
<td>+ CrawlerObserver(in startTextField, in logTextField, in maxDepthTextField, in maxSitesTextField, in currentlyCrawlingTextField, in crawledURLsTextField, in queuedSitesTextField, in crawlProgressBar)</td>
</tr>
<tr>
<td>+ getStartTextField()</td>
</tr>
<tr>
<td>+ getLogTextField()</td>
</tr>
<tr>
<td>+ getMaxSitesTextField()</td>
</tr>
<tr>
<td>+ getMaxDepthTextField()</td>
</tr>
<tr>
<td>+ getCurrentlyCrawlingTextField()</td>
</tr>
<tr>
<td>+ getCrawledURLsTextField()</td>
</tr>
<tr>
<td>+ getQueuedSitesTextField()</td>
</tr>
<tr>
<td>+ getCrawlProgressBar()</td>
</tr>
<tr>
<td>+ updateCurrentlyCrawlingTextField() : string</td>
</tr>
<tr>
<td>+ updateCrawledURLsTextField() : string</td>
</tr>
<tr>
<td>+ updateQueuedSitesTextField() : string</td>
</tr>
<tr>
<td>+ updatedCrawlProgressBar() : int</td>
</tr>
</tbody>
</table>

Figure 14: CrawlerObserver Class

2.4.1.3 SearchObserver
The SearchObserver class is responsible for updating the screen when the model changes due to web searching.
### 2.4.1.4 EntityObserver

The EntityObserver class is responsible for updating the screen when the model changes due to entity searching.

![EntityObserver Class](image)

### 2.5 Model Package

2.5.1 Class Description

#### 2.5.1.1 KrestModel

The KrestModel class is responsible for holding the current KrestObjectLibrary object, and making the appropriate pieces available to other classes.
2.5.1.2 KrestObjectLibrary

The KrestObjectLibrary class is responsible for holding onto all created WebObjects.

```
KrestObjectLibrary
-ojects
+KrestObjectLibrary()
+findObjectByName(name : string) : KrestObjectLibrary
+findObjectsByType(type : int) : KrestObjectLibrary
+getKeys()
+addObject(newObject : WebObject)
+removeObject(objectToRemove : WebObject)
+getAllObjects() : WebObject
```

Figure 18: KrestObjectLibrary Class

2.5.1.3 WebObject

The WebObject class is an abstract class that can be implemented by both the Webpage class and the KrestEntity class. It is used to hold data found based on web crawls and web or entity searches.

```
WebObject
-oename : string
+WebObject(newName : string)
+getName() : string
+setName(newName : string)
```

Figure 19: WebObject Class

2.5.1.4 Webpage

The Webpage class is responsible for holding onto information about a single web site. Each web site explored will have its own Webpage instance.

```
Webpage
-pagetText : string
-backlins : int
+Webpage(newName : string)
+getText() : string
+setText(newText : string)
+getBacklinkCount() : int
+addNewBacklink(newBacklinkName : string)
```

Figure 18: KrestObjectLibrary Class
2.5.1.5 **KrestEntity**

The KrestEntity class is an abstract class that can be implemented by the AddressEntity, EmailEntity, FaxEntity, PhoneEntity, ZipEntity, and OverarchingEntity. It is used to hold data found based on entity searches.

```
KrestEntity
| entityName : string |
| entityPattern : string |
| numberOfOccurrences : int |
| occurrenceList |

+KrestEntity()
+getName() : string
+setName(in newName : string)
+addOccurrence(in websiteFound : string)
+getAllOccurrences()
```

Figure 21: KrestEntity Class

2.5.1.6 **AddressEntity**

The AddressEntity class is responsible for holding onto information about a single address entity. Each address found during an entity search will have its own instance of this class.

```
AddressEntity
| streetAddress : string |
| cityString : string |
| stateString : string |

+AddressEntity(in newStreet : string, in newCity : string, in newState : string)
+getStreet() : string
+getCity() : string
+getState() : string
+setStreet(in newStreet : string)
+setCity(in newCity : string)
+setState(in newState : string)
```

Figure 22: AddressEntity Class

2.5.1.7 **EmailEntity**

The EmailEntity class is responsible for holding onto information about a single email entity. Each email address found during an entity search will have its own instance of this class.

```
EmailEntity
| emailAddress : string |

+EmailEntity(in newEmail : string)
+getEmailAddress() : string
+setEmailAddress(in newEmail : string)
```

Figure 23: EmailEntity Class
2.5.1.8 FaxEntity
The FaxEntity class is responsible for holding onto information about a single fax entity. Each fax number found during an entity search will have its own instance of this class.

```
FaxEntity
- areaCode : string
- faxNumber : string
+ FaxEntity(in newAreaCode : string, in newFaxNumber : string)
+ getAreaCode() : string
+ getFaxNumber() : string
+ getFullFaxNumber() : string
+ setAreaCode(in newAreaCode : string)
+ setFaxNumber(in newFaxNumber : string)
```

Figure 24: FaxEntity Class

2.5.1.9 PhoneEntity
The PhoneEntity class is responsible for holding onto information about a single phone entity. Each phone number found during an entity search will have its own instance of this class.

```
PhoneEntity
- areaCode : string
- phoneNumber : string
+ PhoneEntity(in newAreaCode : string, in newPhoneNumber : string)
+ getAreaCode() : string
+ getPhoneNumber() : string
+ getAreaAndPhoneNumber() : string
+ setAreaCode(in newAreaCode : string)
+ setPhoneNumber(in newPhoneNumber : string)
```

Figure 25: PhoneEntity Class

2.5.1.10 ZipEntity
The ZipEntity class is responsible for holding onto information about a single zip entity. Each zip code found during an entity search will have its own instance of this class.

```
ZipEntity
- zipCode : string
+ ZipEntity(in newZipCode : string)
+ getZipCode() : string
+ setZipCode(in newZipCode : string)
```

Figure 26: ZipEntity Class

2.5.1.11 OverarchingEntity
The OverarchingEntity class is responsible for holding onto information about all entity types. Each street address, email address, fax number, phone number, and zip code found during an overarching entity search will have its own instance of this class.
2.6 Sequence Diagrams

The following three sub-sections show the sequence diagrams for three different user actions: performing a web crawl, performing a web search, and performing an entity search.

2.6.1 User performs a web crawl

**Prerequisites:** KREST is already running.

**Sequence of Events:**
1. User presses the ‘Begin Crawl’ button.
2. The KrestController is notified that the crawl button was pressed, and tells the WebCrawler to begin the crawl.
3. The WebCrawler tells the SiteVisitor to start visiting web pages.
4. SiteVisitor updates the model with the web pages visited.
5. SiteVisitor updates the screen with the latest information via the CrawlerObserver.
6. SiteVisitor notifies that the crawl is complete.
7. WebCrawler updates the screen with the latest information via the CrawlerObserver.

**Post-conditions:** The KrestModel is update with all pages visited, and the screen is updated for the user.
2.6.2 User performs a web search

**Prerequisites:**
1. KREST is already running.
2. A web crawl has already been performed.

**Sequence of Events:**
1. User presses the ‘Begin Search’ button.
2. The KrestController is notified that the search button was pressed, and tells the WebSearcher to begin the search.
3. The WebSearcher queries the KrestModel for all Webpages.
4. The WebSearcher searches through the crawled pages for the search terms.
5. WebSearcher updates the screen with the matching pages via the SearchObserver.

**Post-conditions:** The screen is updated with all matching web pages for the user.
2.6.3 User performs an entity search

**Prerequisites:**
1. KREST is already running.
2. A web crawl has already been performed.

**Sequence of Events:**
1. User presses the ‘Begin Search’ button.
2. The KrestController is notified that the search button was pressed, and tells the EntitySearcher to begin the search.
3. The EntitySearcher queries the KrestModel for all Webpages.
4. The EntitySearcher searches through the crawled pages for the search terms, and extracts corresponding entities.
5. EntitySearcher updates the screen with the entities found via the EntityObserver.

**Post-conditions:** The screen is updated with all found entities.
3. Formal Specification

The project was formally specified and validated using USE 2.3.1. All important classes, attributes, and operations were specified. Invariants and pre and post conditions were also specified. The formal specification is contained below:

```plaintext
model krest

--
-- APPLICATION PACKAGE
--

class KrestApplication
operations
  init()
end
```
class KrestController
attributes
operations
  KrestController()
  initGui()
  crawlButtonActionPerformed()
  resetCrawlerButtonActionPerformed()
  searchButtonActionPerfomed()
  resetTables()
  entitySearchButtonActionPerfomed()
end

class KrestAboutDialog
operations
  KrestAboutDialog()
end

class WebSearcher
attributes
  matches: Set(WebObject)
  searchString: String
operations
  WebSearcher(newSearchString: String)
  beginSearch()
end

class WebCrawler
attributes
  siteVisitor: SiteVisitor
  debugSwitch: Boolean
operations
  WebCrawler()
  beginCrawl(pageAddress: String, searchString: String, maxToCrawl: Integer,
  minBacklinks: Integer, filePath: String, maxDepth: Integer)
  stopCrawling()
  performReset(partial: Boolean)
  getMatches(): Set(WebObject)
  getSiteVisitorThreads(): Integer
  getSiteVisitor(): SiteVisitor
end
class ThreadController
attributes
crowdSize: Integer
maxCrowdSize: Integer
ticketDatabase: Integer
operations
  ThreadController(maxCrowdSizes: Integer)
getTicket(): Integer
returnTicket(ticket: Integer)
findFreeTicket(): Integer
end

class EntitySearcher
attributes
tenentMatches: Set(KrestEntity)
tenityType: Integer
searchString: String
operations
  EntitySearcher(newSearchString: String)
beginSearch()
end

class SiteVisitor
attributes
  MAX_THREADS: Integer
  threadLimiter: ThreadController
debugSwitch: Boolean
searchString: String
crawlCounter: Integer
pagesToCrawl: Integer
pagesVisited: Integer
maxCrawl: Integer
threadCount: Integer
fileName: String
pageAddress: String
keepProcessing: Boolean
threadList: Set(SiteVisitor)
maxDepth: Integer
currentDepth: Integer
pageDatabase: Set(Webpage)
pageToFetch: Webpage
operations
  SiteVisitor(pageAddr: String, searchStr: String, maxToCrawl: Integer, filePath:
String, maxSearchableDepth: Integer, curDepth: Integer)
  start()
  run()
stopAllThreads()
resetCrawler(partial: Boolean)
getMatches(): Set(Webpage)
getThreadCount(): Integer
getCrawlCount(): Integer
getQueueCount(): Integer
loadPage(page: String): String
extractHyperTextLinks(page: String)
containsSearchString(page: String): Boolean
alreadyVisited(pageAddr: String): Boolean
markAsVisited(pageAddr: String)
end

class HTTPReader
attributes
    HTTP_PORT: Integer
operations
    HTTPReader()
    downloadWWWPage(): String
end

--
-- MODEL PACKAGE
--

class KrestModel
attributes
    library: KrestObjectLibrary
    name: String
operations
    KrestModel()
    setName(newName: String)
    getName(): String
    addObject(webObject: WebObject)
    removeObject(webObject: WebObject)
    getData(): KrestObjectLibrary
    addObserver(observer: KrestView)
end

class KrestObjectLibrary
attributes
    objects: Set(WebObject)
operations
    KrestObjectLibrary()
    findObjectByName(name: String): WebObject
findObjectsByType(type: Integer): KrestObjectLibrary
getKeys(): Set(String)
addObject(newObject: WebObject)
removeObject(objectToRemove: WebObject)
getAllObjects(): Set(WebObject)
end

class WebObject
attributes
   name: String
operations
   WebObject(newName: String)
   getName(): String
   setName(newName: String)
end

class Webpage < WebObject
attributes
   pageText: String
   backlinksCount: Integer
   backlinks: Set(String)
operations
   Webpage(newName: String)
   getText(): String
   setText(newText: String)
   getBacklinkCount(): Integer
   addNewBacklink(backlinkName: String)
end

class KrestEntity < WebObject
attributes
   entityName: String
   entityPattern: String
   numberOfOccurrences: Integer
   occurrenceList: Set(KrestEntity)
operations
   KrestEntity()
   getName(): String
   setName(newName: String)
   addOccurrence(websiteFound: String)
   getAllOccurrences(): Set(KrestEntity)
end

class AddressEntity < KrestEntity
attributes
   streetAddress: String
cityString: String
stateString: String
operations
AddressEntity(newStreet: String, newCity: String, newState: String)
getStreet(): String
city(): String
getState(): String
setStreet(newStreet: String)
setCity(newCity: String)
setState(newState: String)
end

class PhoneEntity < KrestEntity
attributes
areaCode: String
phoneNumber: String
operations
PhoneEntity(newAreaCode: String, newPhoneNumber: String)
getAreaCode(): String
getPhoneNumber(): String
getAreaAndPhoneNumber(): String
setAreaCode(newAreaCode: String)
setPhoneNumber(newPhoneNumber: String)
end

class FaxEntity < KrestEntity
attributes
areaCode: String
faxNumber: String
operations
FaxEntity(newAreaCode: String, newFaxNumber: String)
getAreaCode(): String
getFaxNumber(): String
getFullFaxNumber(): String
setAreaCode(newAreaCode: String)
setFaxNumber(newFaxNumber: String)
end

class ZipEntity < KrestEntity
attributes
zipCode: String
operations
ZipEntity(newZipCode: String)
getZipCode(): String
setZipCode(newZipCode: String)
end
class OverarchingEntity < KrestEntity
attributes
    phoneAreaCode: String
    phoneNumber: String
    faxAreaCode: String
    faxNumber: String
    streetString: String
    cityString: String
    stateString: String
    zipString: String
    emailAddress: String
operations
    OverarchingEntity(phoneAreaCodeString: String, phoneNumberString: String,
                        faxAreaCodeString: String, faxNumberString: String,
                        newStreetString: String, newCityString: String,
                        newStateString: String, newZipCodeString: String,
                        newEmailAddress: String)
    getPhoneAreaCode(): String
    getPhoneNumber(): String
    getFaxAreaAndNumber(): String
    getAreaAndPhoneNumber(): String
    getFaxAreaCode(): String
    getFaxNumber(): String
    getStreetAddress(): String
    getCity(): String
    getState(): String
    getZipCode(): String
    getEmailAddress(): String
    setPhoneAreaCode(newAreaCode: String)
    setPhoneNumber(newPhoneNumber: String)
    setFaxAreaCode(newAreaCode: String)
    setFaxNumber(newFaxNumber: String)
    setStreetAddress(newStreetAddress: String)
    setCity(newCity: String)
    setState(newState: String)
    setZipCode(newZipCode: String)
    setEmailAddress(newEmailAddress: String)
end

class EmailEntity < KrestEntity
attributes
    emailAddress: String
operations
    EmailEntity(newEmail: String)
    getEmailAddress(): String
    setEmailAddresses(newEmail: String)
class KrestView
attributes
crawler: CrawlObserver
search: SearchObserver
entity: EntityObserver
operations
KrestObserver()
end

class CrawlObserver
attributes
operations
CrawlerObserver()
updateCurrentlyCrawlingField(): String
updateCrawledURLsTextField(): String
updateQueuedSitesTextField(): String
updateCrawlProgressBar(): Integer
end

class EntityObserver
attributes
operations
EntityObserver()
updateEntitySearchResults(results: KrestObjectLibrary)
end

class SearchObserver
attributes
operations
SearchObserver()
updateWebSearchResults(results: KrestObjectLibrary)
end

--
-- ASSOCIATIONS
--
-- CONTROLLER PACKAGE
--

association Dialog between
  KrestController[1]
  KrestAboutDialog[0..1] role dialog
end

association Searcher between
  KrestController[1]
  WebSearcher[1] role searcher
end

association Crawler between
  KrestController[1]
  WebCrawler[0..1] role crawler
end

association Entity between
  KrestController[1]
  EntitySearcher[1] role entity
end

association Threads between
  WebCrawler[1]
  ThreadController[1] role threads
end

association Visitor between
  WebCrawler[1]
  SiteVisitor[1..*] role visitor
end

association Reader between
  SiteVisitor[1]
  HTTPReader[1] role reader
end

--
-- MODEL PACKAGE
--

association Library between
  KrestModel[1]
  KrestObjectLibrary[1] role library
end
association Objects between
  KrestObjectLibrary[1]
  WebObject[0..*] role objects
end

--
-- VIEW PACKAGE
--

association CrawlerView between
  KrestView[1]
  CrawlObserver[1] role crawlerview
end

association SearcherView between
  KrestView[1]
  SearchObserver[1] role searchview
end

association EntityView between
  KrestView[1]
  EntityObserver[1] role entityview
end

--
-- CONSTRAINTS
--

canstraints

--
-- All WebSearcher matches must have unique names
--

cantext ws : WebSearcher
  inv UniqueNamesWebSearcherMatches:
    ws.matches->forAll(p1,p2 | p1 <> p2
      implies p1.name <> p2.name)

--
-- Every ThreadController has a current crowd size, a max crowd size,  
-- and a tickets in the database count >= 0
--

cantext tc : ThreadController
inv PositiveCrowdSize:
   tc.crowdSize >= 0
inv PositiveMaxCrowdSize:
   tc.maxCrowdSize >= 0
inv PositiveDatabaseTicketsCount:
   tc.ticketDatabase >= 0

--
-- All EntitySearcher matches must have unique names, and entity type
-- must be >= 0
--

context es : EntitySearcher
inv UniqueNamesEntitySearcherMatches:
   es.entityMatches->forAll(p1,p2 | p1 <> p2
      implies p1.entityName <> p2.entityName)
inv PositiveEntityType:
   es.entityType >= 0

--
-- Every SiteVisitor has a MAX_THREADS value that must be >= 0,
-- crawlCounter that must be >= 0, a pages left
to crawl counter that must be >= 0, a pages visited counter that must be >= 0, -
-- a max number of pages to
crawl that must be >= 0, a current thread count that must be >= 0, a maximum
-- search depth value that must
-- be >= 0, a current depth count that must be >= 0 and <= the max search depth,
-- a page database that only
-- contains unique webpages
--

context sv : SiteVisitor
inv PositiveMaxThreads:
   sv.MAX_THREADS >= 0
inv PositiveCrawlCounter:
   sv.crawlCounter >= 0
inv PositivePagesToCrawl:
   sv.pagesToCrawl >= 0
inv PositivePagesVisited:
   sv.pagesVisited >= 0
inv PositiveMaxCrawlCount:
   sv.maxCrawl >= 0
inv PositiveThreadCount:
   sv.threadCount >= 0
inv PositiveMaxSearchDepth:
   sv.maxDepth >= 0
inv PositiveCurrentDepth:
    sv.currentDepth >= 0
inv CurrentDepthNotGreaterThanMaxDepth:
    sv.currentDepth <= sv.maxDepth
inv UniqueWebpagesOnly:
    sv.pageDatabase->forAll(p1,p2 | p1 <> p2
                            implies p1.name <> p2.name)

--
-- Every HTTPReader has a HTTP_PORT value between 0 and 65535
--

context hr : HTTPReader
inv PositivePortValue:
    hr.HTTP_PORT >= 0
inv PortValueLessThanMax:
    hr.HTTP_PORT <= 65535

--
-- All KrestObjectLibrary objects must have unique names
--

context lib : KrestObjectLibrary
inv UniqueNamesKrestObjectLibrary:
    lib.objects->forAll(p1,p2 | p1 <> p2
                         implies p1.name <> p2.name)

--
-- Every Webpage object has a positive number of backlinks
--

context wp : Webpage
inv PositiveBacklinks:
    wp.backlinksCount >= 0

--
-- All KrestEntities must have unique names, and the number of occurrences must
-- be positive
--

context ent : KrestEntity
inv UniqueNamesKrestEntities:
    ent.occurrenceList->forAll(p1,p2 | p1 <> p2
                                 implies p1.entityName <> p2.entityName)
inv PositiveOccurrenceCount:
    ent.numberOfOccurrences >= 0
---
-- All WebObjects must be either a Webpage or KrestEntity, but not both
--

context wo: WebObject
inv IsOneOfItsSubtypes:
  wo.oclIsKindOf(Webpage) or wo.oclIsKindOf(KrestEntity)
inv MutualExclusion1:
  if wo.oclIsKindOf(Webpage) then not wo.oclIsKindOf(KrestEntity) else
  wo.oclIsKindOf(KrestEntity) endif

---
-- OPERATIONS
--

---
-- Any added objects to the KrestModel must be new objects
--

context KrestModel::addObject(webObject: WebObject)
  pre  cond1 : library.objects->excludes(webObject)
  post cond2 : library.objects = library.objects@pre->including(webObject)
  post cond3 : (library.objects - library.objects@pre)->size() = 1

---
-- Deleting an object from the KrestModel must remove it while the other objects remain unchanged
--

context KrestModel::removeObject(webObject: WebObject)
  pre  cond1 : library.objects->includes(webObject)
  post cond2 : library.objects = library.objects@pre->excluding(webObject)
  post cond3 : (library.objects@pre - library.objects)->size() = 1

---
-- Finding an object by name in the KrestObjectLibrary
--

context KrestObjectLibrary::findObjectByName(name: String): WebObject
  post cond1 : result = objects->any(c1 | c1.name = name)

---
-- Any added objects to the KrestObjectLibrary must be new objects
--
context KrestObjectLibrary::addObject(newObject: WebObject)
pre  cond1 : objects->excludes(newObject)
post cond2 : objects = objects@pre->including(newObject)
post cond3 : (objects - objects@pre)->size() = 1

--
-- Deleting an object from the KrestObjectLibrary must remove it while the other objects remain unchanged
--

context KrestObjectLibrary::removeObject(objectToRemove: WebObject)
pre  cond1 : objects->includes(objectToRemove)
post cond2 : objects = objects@pre->excluding(objectToRemove)
post cond3 : (objects@pre - objects)->size() = 1

--
-- Getting all objects from the KrestObjectLibrary returns all objects
--

color KrestObjectLibrary::getAllObjects(): Set(WebObject)
post cond1 : result = self.objects

--
-- Getting the name from the WebObject returns its name
--

color WebObject::getName(): String
post cond1 : result = self.name

--
-- Setting the name for the WebObject sets its name
--

color WebObject::setName(newName: String)
post cond1 : self.name = newName

--
-- Getting the page text from the Webpage returns its text
--

color Webpage::getText(): String
post cond1 : result = self.pageText

--
-- Setting the pageText for the Webpage sets its text
--

context Webpage::setText(newText: String)
    post cond1 : self.pageText = newText

--
-- Getting the backlink count from the Webpage returns its count
--

context Webpage::getBacklinkCount(): Integer
    post cond1 : result = self.backlinksCount

--
-- Any added backlinks to the Webpage must be new objects
--

context Webpage::addNewBacklink(backlinkName: String)
    pre cond1 : backlinks->excludes(backlinkName)
    post cond2 : backlinks = backlinks@pre->including(backlinkName)
    post cond3 : (backlinks - backlinks@pre)->size() = 1

--
-- Getting the name from the KrestEntity returns its name
--

context KrestEntity::getName(): String
    post cond1 : result = self.entityName

--
-- Setting the name for the KrestEntity sets its entityName
--

context KrestEntity::setName(newName: String)
    post cond1 : self.entityName = newName

--
-- Any added occurrence of an entity will be a new occurrence, and will increment
-- the number of occurrences
--

context KrestEntity::addOccurrence(websiteFound: String)
    pre cond1 : self.occurrenceList.entityName->excludes(websiteFound)
    post cond2 : occurrenceList.entityName = occurrenceList.entityName@pre->including(websiteFound)
    post cond3 : (occurrenceList - occurrenceList@pre)->size() = 1
    post cond4 : (numberOfOccurrences - numberOfOccurrences@pre) = 1
Getting all KrestEntity occurrences returns the list of occurrences

context KrestEntity::getAllOccurrences(): Set(KrestEntity)
    post cond1 : result = self.occurrenceList

Creating a new AddressEntity sets the values passed in

context AddressEntity::AddressEntity(newStreet: String, newCity: String, newState: String)
    post cond1 : streetAddress = newStreet
    post cond2 : cityString = newCity
    post cond3 : stateString = newState

Getting the Street from AddressEntity returns the Street string

context AddressEntity::getStreet(): String
    post cond1 : result = self.streetAddress

Getting the City from AddressEntity returns the city string

context AddressEntity::getCity(): String
    post cond1 : result = self.cityString

Getting the stateString from AddressEntity returns the state string

context AddressEntity::getState(): String
    post cond1 : result = self.stateString

Setting the street for AddressEntity stores the new street

context AddressEntity::setStreet(newStreet: String)
    post cond1 : streetAddress = newStreet
-- Setting the city for AddressEntity stores the new city

context AddressEntity::setCity(newCity: String)
    post cond1 : cityString = newCity

--

-- Setting the state for AddressEntity stores the new state

context AddressEntity::setState(newState: String)
    post cond1 : stateString = newState

--

-- Creating a new PhoneEntity sets the values passed in

context PhoneEntity::PhoneEntity(newAreaCode: String, newPhoneNumber: String)
    post cond1 : areaCode = newAreaCode
    post cond2 : phoneNumber = newPhoneNumber

--

-- Getting the Area Code from PhoneEntity returns the area code

context PhoneEntity::getAreaCode(): String
    post cond1 : result = self.areaCode

--

-- Getting the Phone Number from PhoneEntity returns the phone number

context PhoneEntity::getPhoneNumber(): String
    post cond1 : result = self.phoneNumber

--

-- Getting the Area Code and Phone Number from PhoneEntity returns the area code concatenated with the phone number

context PhoneEntity::getAreaAndPhoneNumber(): String
    post cond1 : result = self.areaCode.concat(self.phoneNumber)

--

-- Setting the area code for PhoneEntity stores the new area code
context PhoneEntity::setAreaCode(newAreaCode: String)
    post cond1 : areaCode = newAreaCode

--

-- Setting the phone number for PhoneEntity stores the new phone number
--

context PhoneEntity::setPhoneNumber(newPhoneNumber: String)
    post cond1 : phoneNumber = newPhoneNumber

--

-- Creating a new FaxEntity sets the values passed in
--

context FaxEntity::FaxEntity(newAreaCode: String, newFaxNumber: String)
    post cond1 : areaCode = newAreaCode
    post cond2 : faxNumber = newFaxNumber

--

-- Getting the Area Code from FaxEntity returns the area code
--

context FaxEntity::getAreaCode(): String
    post cond1 : result = self.areaCode

--

-- Getting the Fax Number from FaxEntity returns the phone number
--

context FaxEntity::getFaxNumber(): String
    post cond1 : result = self.faxNumber

--

-- Getting the Area Code and Fax Number from FaxEntity returns the area code
-- concatenated with the fax number
--

context FaxEntity::getFullFaxNumber(): String
    post cond1 : result = self.areaCode.concat(self.faxNumber)

--

-- Setting the area code for FaxEntity stores the new area code
--
context FaxEntity::setAreaCode(newAreaCode: String)
    post cond1 : areaCode = newAreaCode

--
-- Setting the fax number for FaxEntity stores the new phone number
--

context FaxEntity::setFaxNumber(newFaxNumber: String)
    post cond1 : faxNumber = newFaxNumber

--
-- Creating a new ZipEntity sets the value passed in
--

context ZipEntity::ZipEntity(newZipCode: String)
    post cond1 : zipCode = newZipCode

--
-- Getting the Zip Code from ZipEntity returns the zip code
--

context ZipEntity::getZipCode(): String
    post cond1 : result = self.zipCode

--
-- Setting the zip code for ZipEntity stores the new zip code
--

context ZipEntity::setZipCode(newZipCode: String)
    post cond1 : zipCode = newZipCode

--
-- Creating a new EmailEntity sets the value passed in
--

context EmailEntity::EmailEntity(newEmail: String)
    post cond1 : emailAddress = newEmail

--
-- Getting the Email Address from EmailEntity returns the email adress
--

context EmailEntity::getEmailAddress(): String
    post cond1 : result = self.emailAddress
- Setting the Email Address for EmailEntity stores the new email address

context EmailEntity::setEmailAddrses(newEmail: String)
  post cond1 : emailAddress = newEmail

- Creating a new OverarchingEntity sets the values passed in

context OverarchingEntity::OverarchingEntity(phoneAreaCodeString: String,
  phoneNumberString: String, faxAreaCodeString: String, faxNumberString:
  String, newStreetString: String, newCityString: String, newStateString: String,
  newZipCodeString: String, newEmailAddress: String)
  post cond1 : self.phoneAreaCode = phoneAreaCodeString
  post cond2 : self.phoneNumber = phoneNumberString
  post cond3 : self.faxAreaCode = faxAreaCodeString
  post cond4 : self.faxNumber = faxNumberString
  post cond5 : self.streetString = newStreetString
  post cond6 : self.cityString = newCityString
  post cond7 : self.stateString = newStateString
  post cond8 : self.zipString = newZipCodeString
  post cond9 : self.emailAddress = newEmailAddress

- Getting the Area Code from OverarchingEntity returns the area code

context OverarchingEntity::getPhoneAreaCode(): String
  post cond1 : result = self.phoneAreaCode

- Getting the Phone Number from OverarchingEntity returns the phone number

context OverarchingEntity::getPhoneNumber(): String
  post cond1 : result = self.phoneNumber

- Getting the Area Code and Phone Number from OverarchingEntity returns the area code concatenated with the phone number

context OverarchingEntity::getAreaAndPhoneNumber(): String
  post cond1 : result = self.phoneAreaCode.concat(self.phoneNumber)
-- Getting the Area Code from OverarchingEntity returns the area code
--

custom OverarchingEntity::getFaxAreaCode(): String
    post cond1 : result = self.faxAreaCode

--
-- Getting the Fax Number from OverarchingEntity returns the phone number
--

custom OverarchingEntity::getFaxNumber(): String
    post cond1 : result = self.faxNumber

--
-- Getting the Area Code and Fax Number from OverarchingEntity returns the
-- area code concatenated with the fax number
--

custom OverarchingEntity::getFaxAreaAndNumber(): String
    post cond1 : result = self.faxAreaCode.concat(self.faxNumber)

--
-- Getting the Street from OverarchingEntity returns the Street string
--

custom OverarchingEntity::getStreetAddress(): String
    post cond1 : result = self.streetString

--
-- Getting the City from OverarchingEntity returns the city string
--

custom OverarchingEntity::getCity(): String
    post cond1 : result = self.cityString

--
-- Getting the stateString from OverarchingEntity returns the state string
--

custom OverarchingEntity::getState(): String
    post cond1 : result = self.stateString

--
-- Getting the zip code from OverarchingEntity returns the zip code string
--
context OverarchingEntity::getZipCode(): String
    post cond1 : result = self.zipString

--
-- Getting the email address from OverarchingEntity returns the email address
-- string
--

context OverarchingEntity::getEmailAddress(): String
    post cond1 : result = self.emailAddress

--
-- Setting the phone area code for OverarchingEntity stores the new area code
--

context OverarchingEntity::setPhoneAreaCode(newAreaCode: String)
    post cond1 : phoneAreaCode = newAreaCode

--
-- Setting the phone number for OverarchingEntity stores the new phone number
--

context OverarchingEntity::setPhoneNumber(newPhoneNumber: String)
    post cond1 : phoneNumber = newPhoneNumber

--
-- Setting the fax area code for OverarchingEntity stores the new area code
--

context OverarchingEntity::setFaxAreaCode(newAreaCode: String)
    post cond1 : faxAreaCode = newAreaCode

--
-- Setting the fax number for OverarchingEntity stores the new phone number
--

context OverarchingEntity::setFaxNumber(newFaxNumber: String)
    post cond1 : faxNumber = newFaxNumber

--
-- Setting the street for OverarchingEntity stores the new street
--

context OverarchingEntity::setStreetAddress(newStreetAddress: String)
    post cond1 : streetString = newStreetAddress
context OverarchingEntity::setCity(newCity: String)
    post cond1 : cityString = newCity

context OverarchingEntity::setState(newState: String)
    post cond1 : stateString = newState

context OverarchingEntity::setZipCode(newZipCode: String)
    post cond1 : zipString = newZipCode

context OverarchingEntity::setEmailAddress(newEmailAddress: String)
    post cond1 : emailAddress = newEmailAddress