Sheyan Lalmohammed
NETS 1500
March 19, 2023

**Homework 3**

Part 1 - Theory

1. Java does indeed have a URI Class. In java, the URI class represents objects that are Uniform Resource Identifiers. The URL class represents objects that are Uniform Resource Locators. There are a couple of distinct differences between these two classes within Java, but the two specific differences of concern are the syntax of the two classes and the operations that can be done with objects of each of these classes. When it comes to the syntax related to instances of the two classes, the URI class has a greater amount of flexibility in the input it can take with regard to the scheme that can be given to the class as the URI class can take on any type of scheme which includes non-hierarchical schemes while the URL class must take input schemes that are hierarchical. Furthermore, this is related to the concept of resolution and the way both classes can be used differently to conduct resolution. The URI class has the ability to be used for a relative resolution to resolve URIs to their absolute URIs. On the other hand, the URL class can be used to conduct absolute URL resolution. The interesting part of this is that it is recommended to use the URI class to switch between the values of these two classes. The other big difference between the use of the two classes when it comes to the various ways they interact with a network. The URL class has the ability to connect to the resource which was imputed into the class which the URI class cannot do. Furthermore, the URL class can retrieve data or information from the resource to which it connects while the URI class can only resolve URIs.
2. There is a specific response that comes from the URL class in java if its constructor is given invalid input. In the case that the URL constructor's parameter is given an invalid URL, such as www.cis.upenn.edu, the constructor will throw a MalformedURLException to indicate that it was given an invalid URL. The reasoning behind this is that a URL, such as www.cis.upenn.edu, does not include http:// or https:// which would indicate a protocol to be used by the URL class.
3. The following is the original code for the URLGetter class that includes the new getRedirectedURL function:

```
package networking;
import java.io.IOException;
import java.net.HttpURLConnection;
import java.net.URL;
```

```
import java.util.ArrayList;
import java.util.Scanner;

public class URLGetter {

   private URL url;
   private HttpURLConnection httpConnection;

   public URLGetter(String url) {
      try {
         this.url = new URL(url);
         httpConnection = (HttpURLConnection) this.url.openConnection();
         httpConnection.setInstanceFollowRedirects(false);
      } catch (Exception e) {
         e.printStackTrace();
      }
   }

   public void printStatusCode() {
      try {
         int code = httpConnection.getResponseCode();
         String message = httpConnection.getResponseMessage();
         System.out.println(code + " : " + message);
      } catch (IOException e) {
         throw new RuntimeException(e);
      }
   }

    public ArrayList<String> getContents() {
      ArrayList<String> contents = new ArrayList<>();
      try {
         Scanner in = new Scanner(httpConnection.getInputStream());
         while (in.hasNextLine()) {
            String line = in.nextLine();
            contents.add(line);
         }
      } catch (Exception e) {
         e.printStackTrace();
      }
      return contents;
```

```
        }

        public URL getRedirectURL() {
           URL redirectingURL = null;
           try {
              int code = httpConnection.getResponseCode();
              if (code >= 300 && code < 400) {
                 String tempURL = httpConnection.getHeaderField("Location");
                 if (tempURL != null){
                       redirectingURL = new URL(tempURL)
                 }
              }
           } catch (IOException e) {
              e.printStackTrace();
           }
           return redirectingURL;
        }
}
```