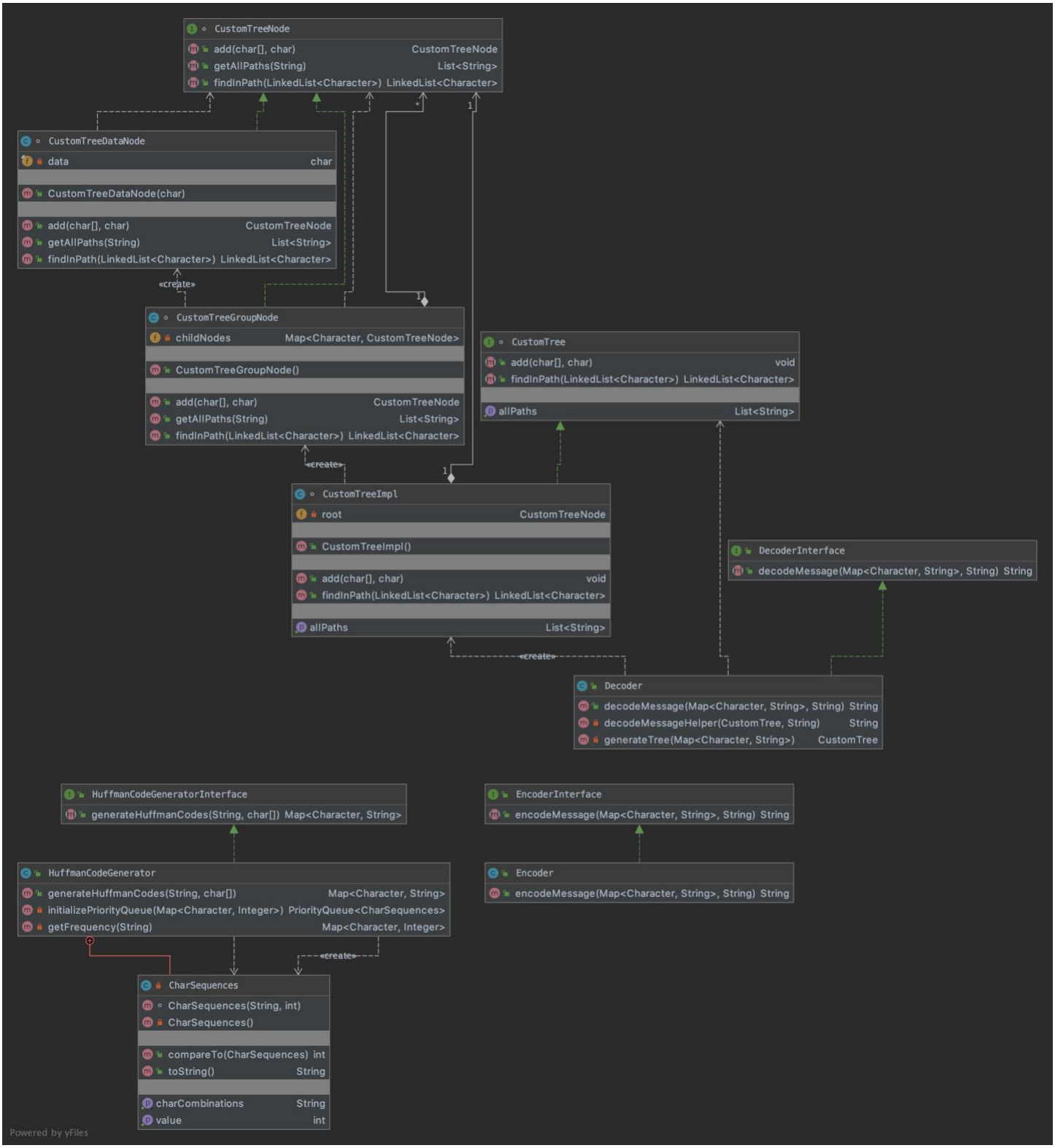
HW4 CODING TREES

1. UML Diagram



The above UML diagram represents the final class design and association between various elements like Decoder, Encoder, Huffman code generator and n-children trees. Encoder, Decoder and Huffman code generator interface will make the code more reusable and efficient. All the features that required encoding and decoding are as part of methods in the respective classes.

Decoder is implemented by using a custom tree which is n-children tree. There is a group node in the tree that contains a dictionary mapping of tress at every branch. Moreover, the leaf node will contain the actual character data. All the tree interfaces and classes are package private to avoid outside access.

Finally, many use cases have been handled in the Driver class which allows you to execute either hardcoded sample cases for encoding, decoding, generating prefix codes or give a custom input the run the program.

2. Testing Plan

The following conditions are tested as part of the Testing

Huffman code generator: (Takes plain text and gives code dictionary)

1. Test whether code is distinct for each character in the plain text
2. Test whether no code is prefix of another

Encoder: (Takes plain text and code dictionary as input and outputs encoded string)

1. Test whether correct code is applied for each character in the plain text
2. Test whether code dictionary is correct for given plain text. There might be cases where there is no code for character in plain text. If so, exception should be thrown.

Decoder: (Takes encoded string and code dictionary as input and outputs decoded message)

1. Test whether encoded string is correct such that there exists a leaf node(character) for the character sequences of encoded string.
2. Test whether correct decoding tree is formulated before starting to decode.
3. Test whether code dictionary is correct for given encoded string. There might be cases where there is no code for character in encoded string. If so, exception should be thrown.

General:

1. Test end-to-end process. Eg. Take a plain text, get Huffman code, encode it. Upon decoding you should get the same plain text. This confirms the correctness of the application.

3. Test execution