Von Kenneth Quilon (NetID: vvq1) and Alex Loh (NetID: al644)

The program works in three parts: parsing the torrent file (TorrentFile), contacting the tracker (TrackerConnection), and downloading from a peer (PeerConnection). Each part works independently of each other, and are called sequentially by the main method, which is located in RUBTClient, as shown in the diagram. Also, the parts individually call functions from either Bencoder2 or Functions as helper methods to do miscellaneous tasks like de-bencoding data or parsing byte arrays into other forms of data.

Bencoder2.java:

This class holds all of the methods needed to bencode and de-bencode byte array data into Objects of type Map, List, Integer, or String - a necessary step when communicating with the tracker to get a list of all peers. This class is a copy of that provided on the Sakai assignment page.

BencodingException.java:

This class represents an exception thrown by Bencoder2.java when it discovers an error while de-bencoding data. This class is a copy of that provided on the Sakai assignment page.

Functions.java:

This class holds all of the methods that perform miscellaneous tasks that are necessary to but wouldn't be part of another class's responsibility, usually transforming some form of data to another or generating data. For example, the 'encodeToSHA1' method provides functionality for taking a byte array and converting it to a SHA1 hash of the byte array, which is used by TorrentFile.java and PeerConnection.java.

PeerConnection.java:

This class represents the connection to a peer while torrenting. It only downloads the whole file from one peer with a specific IP address through the function 'getFileFromPeer' - later, this class will be changed to allow uploading and downloading from multiple peers.

RUBTClient.java:

This class is where the main() method is called, with first argument being the torrent file and second argument being the saved file's filename. It runs in three distinct parts: First, it generates the peerID used by this client and reads in the torrent file. Next, it contacts the tracker using the info from the torrent file to get a list of peers. Lastly, it contacts a specific peer's IP, if it can find that peer, downloads the file, and saves it with the filename specified in the arguments. Each part is generally represented by a different class each holding responsibility for that particular task: TorrentFile, TrackerConnection, and PeerConnection respectively.

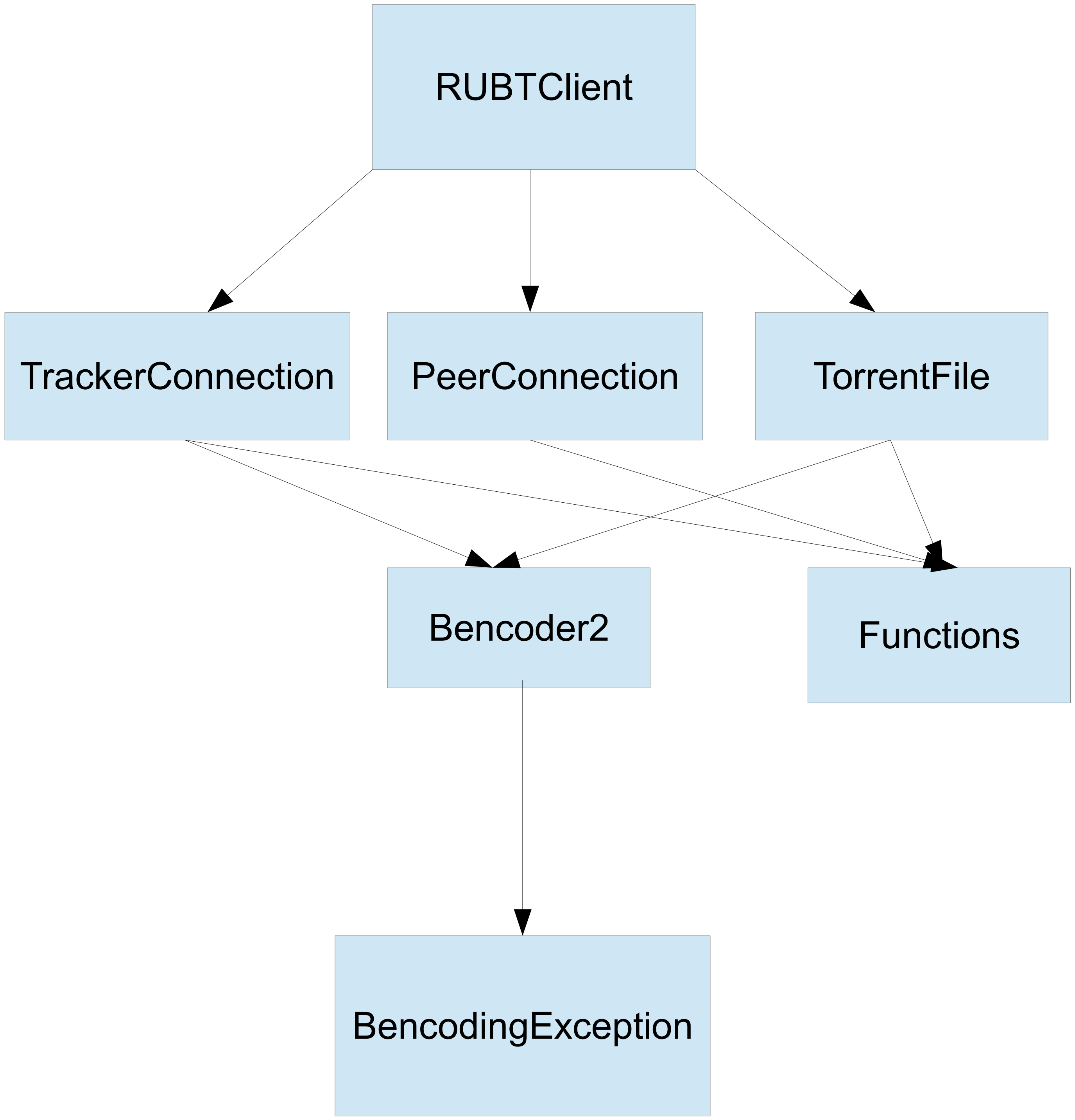
TorrentFile.java:

This class deals with obtaining and parsing data from the torrent file into fields that we need for the next steps, like the downloaded file's size and the IP of the tracker. We can either parse the torrent file into dictionaries containing the data, or parse it into specific fields that we expect the torrent file to have, which are set as data members and can be accessed via getters - both forms of the data will be needed in our program.

TrackerConnection.java:

This class contacts the tracker to obtain its response, which contains the information we need to contact the peers. Two methods provide functionality for contacting the tracker: getTrackerResponse() gets the raw byte array data from the tracker, while getPeersFromTrackerResponse() parses that byte array data into a list of peers.

(an arrow to a box means that that the class it's pointing from depends on the functionality of the class it's pointing to)


RUBTClient
TrackerConnection
PeerConnection
TorrentFile
Bencoder2
Functions
BencodingException