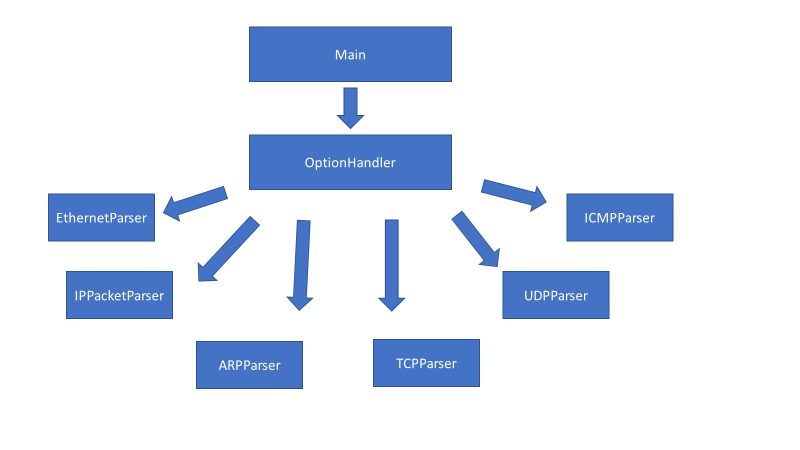
**Intro:**

In this project both a packet parser and a packet generator were designed using java. The packet parser was built using a modular approach, the top part being Main. Main instantiates the real top class which is called OptionHandler which runs the user’s options. The OptionHandler instantiates a class for each different kind of header parsing. In all 6 classes were instantiated ARParser, UDPParser, TCPParser, ICMPParser, IPPacketParser, and EthernetParser. For packet generation a class was created that is called PacketGenerator.

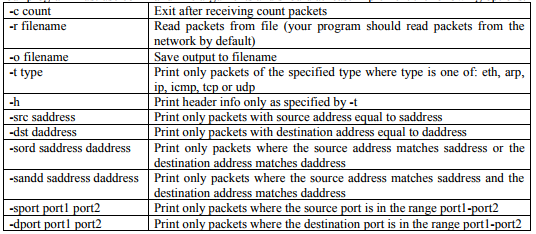
Packet Parser Hierarchy



**Option Handler:**

OptionHandler receives the users options from main and using a library extracted from org.apache.commons.cli parses the options. All the possible options consist of the following:

Packet Parsing Options



In addition, the options can be viewed from the program by passing the -help option. If the user does not pass in a file to read packets from, the class will look for network interfaces on the user’s computer, it will then prompt the user to choose from one. If a read file is provided the class will open it up and begin looking for bytes to parse. When no output file is provided the class will default to printing output to the console. If an output file is specified, then the class will set all output to be sent to the specified file.

A switch statement is used to parse for diverse types of packets. When the user does not specify what kind of packet to parse for, the class defaults to TCP packet parsing. At the beginning of all the packet parsing switch statements, the counter option is checked to see if it has been specified. If it has then a count is kept of every packet that has the specified header, if it hasn’t then packets will continue to be parsed indefinitely until the user types ctrl + C to close the program.

The packets are retrieved though a function that keeps track of whether they are coming from a file or a network interface card. If the type is set to ethernet, the length of the packet is checked to see that it is long enough to parse for this type. Then based on whether the user has specified header only or not, the OptionHandler will choose whether to print header only or header plus payload. If the type is set to arp, the length is again checked. The packet will first go through the ethernet parser. The type of packet is retrieved from the parser, and compared to 0x0806 which is for arp. The packet is then passed through the ARPParser instantiation. Since this kind of packet only has a header that is all that is printed. If the