okhttp3.OkHttpClient

```
public class OkHttpClient implements Cloneable, Call.Factory, WebSocket.Factory

public static final class Builder {...}

public Call newCall(Request request)

public WebSocket newWebSocket(Request request, WebSocketListener listener)
```

okhttp3.OkHttpClient.Builder

```
public Builder()

public Builder connectTimeout(long timeout, TimeUnit unit)

public Builder readTimeout(long timeout, TimeUnit unit)

public Builder writeTimeout(long timeout, TimeUnit unit)

public Builder cache(@Nullable Cache cache)

public Builder connectionPool(ConnectionPool connectionPool)

public List<Interceptor> interceptors()
public Builder addInterceptor(Interceptor interceptor)

public List<Interceptor> networkInterceptors()
public Builder addNetworkInterceptor(Interceptor interceptor)

public Builder eventListener(EventListener eventListener)
public Builder eventListenerFactory(EventListener.Factory eventListenerFactory)

public OkHttpClient build()
```

okhttp3.Cache

```
public final class Cache implements Closeable, Flushable

final InternalCache internalCache

final DiskLruCache cache;

public Cache(File directory, long maxSize)

public static String key(HttpUrl url)

public void initialize() throws IOException

public void delete() throws IOException

public void evictAll() throws IOException

public Iterator<String> urls() throws IOException

public long size()

public long maxSize()

public void flush()

public void close()
```

```
public File directory()

public boolean isClosed()

public synchronized int networkCount()

public synchronized int hitCount()

public synchronized int requestCount()
```

okhttp3.Interceptor

```
Response intercept(Chain chain) throws IOException;

interface Chain {...}
```

okhttp3.Interceptor.Chain

```
Request request();

Response proceed(Request request) throws IOException;

Connection connection();

Call call();

int connectTimeoutMillis();

Chain withConnectTimeout(int timeout, TimeUnit unit);

int readTimeoutMillis();

Chain withReadTimeout(int timeout, TimeUnit unit);

int writeTimeoutMillis();

Chain withWriteTimeout(int timeout, TimeUnit unit);
```

okhttp3.Request

```
public HttpUrl url()

public String method()

public Headers headers()

public @Nullable String header(String name)

public List<String> headers(String name)

public @Nullable RequestBody body()

public static class Builder {...}
```

okhttp3.Request.Builder

```
public Builder()

public Builder url(String url)
public Builder url(URL url)

public Builder header(String name, String value)
```

```
public Builder addHeader(String name, String value)

public Builder removeHeader(String name)

public Builder headers(Headers headers)

public Builder cacheControl(CacheControl cacheControl)

    public Builder cacheControl(CacheControl cacheControl) {
        String value = cacheControl.toString();
        if (value.isEmpty()) return removeHeader("Cache-Control");
        return header("Cache-Control", value);
    }

public Builder get()

public Builder post(RequestBody body)

    public Builder post(RequestBody body) {
        return method("POST", body);
    }

public Builder method(String method, @Nullable RequestBody body)

public Request build()
```

okhttp3.CacheControl

```
public static final CacheControl FORCE_NETWORK = new Builder()
                                    .noCache()
                                    .build();

public static final CacheControl FORCE_CACHE = new Builder()
                                    .onlyIfCached()
                                    .maxStale(Integer.MAX_VALUE, TimeUnit.SECONDS)
                                    .build();

public boolean noCache()

public boolean noStore()

public int maxAgeSeconds()

public boolean onlyIfCached()

public String toString()

    @Override public String toString() {
        String result = headerValue;
        return result != null ? result : (headerValue = headerValue());
    }

    private String headerValue() {
        StringBuilder result = new StringBuilder();
        if (noCache) result.append("no-cache, ");
        if (noStore) result.append("no-store, ");
        if (maxAgeSeconds != -1) result.append("max-age=").append(maxAgeSeconds).append(", ");
        if (sMaxAgeSeconds != -1) result.append("s-maxage=").append(sMaxAgeSeconds).append(", ");
        if (isPrivate) result.append("private, ");
        if (isPublic) result.append("public, ");
        if (mustRevalidate) result.append("must-revalidate, ");
        if (maxStaleSeconds != -1) result.append("max-stale=").append(maxStaleSeconds).append(", ");
        if (minFreshSeconds != -1) result.append("min-fresh=").append(minFreshSeconds).append(", ");
        if (onlyIfCached) result.append("only-if-cached, ");
        if (noTransform) result.append("no-transform, ");
        if (immutable) result.append("immutable, ");
```

```
        if (result.length() == 0) return "";
        result.delete(result.length() - 2, result.length());
        return result.toString();
    }

public static final class Builder {...}
```

okhttp3.CacheControl.Builder

```
public Builder noCache()

public Builder noStore()

public Builder maxAge(int maxAge, TimeUnit timeUnit)

public Builder maxStale(int maxStale, TimeUnit timeUnit)

public Builder minFresh(int minFresh, TimeUnit timeUnit)

public Builder onlyIfCached()

public CacheControl build()
```

okhttp3.RequestBody

```
public abstract class RequestBody

public abstract @Nullable MediaType contentType();

public long contentLength() throws IOException

public abstract void writeTo(BufferedSink sink) throws IOException;

public static RequestBody create(@Nullable MediaType contentType, String content)

public static RequestBody create(final @Nullable MediaType contentType,
    final byte[] content)

public static RequestBody create(final @Nullable MediaType contentType,
    final byte[] content, final int offset, final int byteCount)

public static RequestBody create(final @Nullable MediaType contentType,
    final File file)
```

okhttp3.FormBody

```
public final class FormBody extends RequestBody

private static final MediaType CONTENT_TYPE =
    MediaType.get("application/x-www-form-urlencoded");

public int size()

public String name(int index)

public String value(int index)

@Override public MediaType contentType() {
    return CONTENT_TYPE;
}

public static final class Builder {...}
```

okhttp3.FormBody.Builder

```
public Builder()

public Builder(Charset charset)

public Builder add(String name, String value)

public Builder addEncoded(String name, String value)

public FormBody build()
```

okhttp3.MultipartBody

```
public final class MultipartBody extends RequestBody

public static final MediaType FORM = MediaType.get("multipart/form-data");

MultipartBody(ByteString boundary, MediaType type, List<Part> parts)

public static final class Part {...}

public static final class Builder {...}
```

okhttp3.MultipartBody.Part

```
public static Part create(RequestBody body)

public static Part create(@Nullable Headers headers, RequestBody body)

public static Part createFormData(String name, String value)

public static Part createFormData(String name, @Nullable String filename,
    RequestBody body)

    public static Part createFormData(String name, @Nullable String filename,
        RequestBody body) {
        if (name == null) {
            throw new NullPointerException("name == null");
        }
        StringBuilder disposition = new StringBuilder("form-data; name=");
        appendQuotedString(disposition, name);

        if (filename != null) {
            disposition.append("; filename=");
            appendQuotedString(disposition, filename);
        }

        return create(Headers.of("Content-Disposition",
                            disposition.toString()), body);
    }

public @Nullable Headers headers()

public RequestBody body()
```

okhttp3.MultipartBody.Builder

```
public Builder()

    public Builder() {
        this(UUID.randomUUID().toString());
    }
```

```
public Builder(String boundary)

public Builder setType(MediaType type)

public Builder addPart(RequestBody body)

public Builder addPart(@Nullable Headers headers, RequestBody body)

public Builder addFormDataPart(String name, String value)

public Builder addFormDataPart(String name, @Nullable String filename,
    RequestBody body)

public Builder addPart(Part part)

    public Builder addPart(Part part) {
        if (part == null) throw new NullPointerException("part == null");
        parts.add(part);
        return this;
    }

public MultipartBody build()
```

okhttp3.Headers

```
private final String[] namesAndValues;

public static Headers of(String... namesAndValues)
public static Headers of(Map<String, String> headers)

public @Nullable String get(String name)

public int size()
public String name(int index)
public String value(int index)

public Set<String> names()
public List<String> values(String name)

public String toString()

    @Override public String toString() {
        StringBuilder result = new StringBuilder();
        for (int i = 0, size = size(); i < size; i++) {
            result.append(name(i)).append(": ").append(value(i)).append("\n");
        }
        return result.toString();
    }

public static final class Builder {...}
```

okhttp3.Headers.Builder

```
public Builder add(String name, String value)

public Builder addAll(Headers headers)

public Builder set(String name, String value)

public Headers build()
```

okhttp3.MediaType

```
public static MediaType get(String string)

public static @Nullable MediaType parse(String string)

    public static @Nullable MediaType parse(String string) {
        try {
              return get(string);
        } catch (IllegalArgumentException ignored) {
             return null;
        }
    }
```

okhttp3.Call

```
public interface Call extends Cloneable

Request request();

Response execute() throws IOException;

void enqueue(Callback responseCallback);

void cancel();

boolean isExecuted();

boolean isCanceled();
```

okhttp3.RealCall

```
final class RealCall implements Call
```

okhttp3.Callback

```
public interface Callback

void onFailure(Call call, IOException e);

void onResponse(Call call, Response response) throws IOException;
```

okhttp3.Response

```
public final class Response implements Closeable

public Request request()

public Protocol protocol()

public int code()

public boolean isSuccessful()

public List<String> headers(String name)

public @Nullable String header(String name)

public @Nullable String header(String name, @Nullable String defaultValue)

public Headers headers()

public @Nullable Response networkResponse()
public @Nullable Response cacheResponse()
public @Nullable Response priorResponse()
```

```
public CacheControl cacheControl()

public long sentRequestAtMillis()
public long receivedResponseAtMillis()

@Override public void close()

public @Nullable ResponseBody body()

public Builder newBuilder()

public static class Builder {...}
```

okhttp3.Response.Builder

```
public Builder()

public Builder request(Request request)

public Builder protocol(Protocol protocol)

public Builder code(int code)

public Builder header(String name, String value)

public Builder addHeader(String name, String value)

public Builder removeHeader(String name)

public Builder headers(Headers headers)

public Builder body(@Nullable ResponseBody body)

public Builder networkResponse(@Nullable Response networkResponse)

public Builder cacheResponse(@Nullable Response cacheResponse)

public Builder priorResponse(@Nullable Response priorResponse)

public Builder sentRequestAtMillis(long sentRequestAtMillis)

public Builder receivedResponseAtMillis(long receivedResponseAtMillis)

public Response build()
```

okhttp3.ResponseBody

```
public abstract class ResponseBody implements Closeable

public abstract @Nullable MediaType contentType();

public abstract long contentLength();

public abstract BufferedSource source();

public final InputStream byteStream()

public final byte[] bytes() throws IOException

public final String string() throws IOException

public static ResponseBody create(@Nullable MediaType contentType, String content)

public static ResponseBody create(final @Nullable MediaType contentType, byte[] content)
```