在 Android 的 Kotlin 中，为 Glide 添加自定义的请求头可以通过实现自定义的 OkHttpClient 来完成。以下是实现步骤：

1. 添加 Glide 和 OkHttp 的依赖

确保在 build.gradle 文件中添加以下依赖：

```
implementation 'com.github.bumptech.glide:glide:4.15.1'
kapt 'com.github.bumptech.glide:compiler:4.15.1'
implementation 'com.squareup.okhttp3:okhttp:4.11.0'
implementation 'com.github.bumptech.glide:okhttp3-
integration:4.15.1'
```

同步项目以确保依赖成功导入。

2. 创建带有自定义请求头的 OkHttpClient

你需要创建一个带有自定义拦截器的 OkHttpClient，拦截器将添加自定义的请求头。

```
import okhttp3.Interceptor
import okhttp3.OkHttpClient
import okhttp3.Request
import okhttp3.Response
import java.io.IOException

val okHttpClient = OkHttpClient.Builder()
    .addInterceptor(object : Interceptor {
        @Throws(IOException::class)
        override fun intercept(chain: Interceptor.Chain):
Response {
            val originalRequest: Request =
chain.request()
            val newRequest = originalRequest.newBuilder()
                .addHeader("Authorization", "Bearer
YOUR_ACCESS_TOKEN")
                .addHeader("Custom-Header",
"CustomHeaderValue")
```

```
                .build()
            return chain.proceed(newRequest)
        }
    })
    .build()
```

3. 配置 Glide 使用自定义 OkHttpClient

实现 Glide 的模块化配置来接入自定义的 OkHttpClient。

创建一个自定义的 AppGlideModule：

```
import android.content.Context
import com.bumptech.glide.Glide
import com.bumptech.glide.GlideBuilder
import com.bumptech.glide.annotation.GlideModule
import
com.bumptech.glide.load.engine.cache.LruResourceCache
import com.bumptech.glide.module.AppGlideModule
import okhttp3.OkHttpClient
import
com.bumptech.glide.integration.okhttp3.OkHttpUrlLoader
import java.io.InputStream

@GlideModule
class CustomGlideModule : AppGlideModule() {
    override fun applyOptions(context: Context, builder:
GlideBuilder) {
        // 设置 Glide 的全局配置（可选）
        val memoryCacheSizeBytes = 1024 * 1024 * 20 // 20
MB

builder.setMemoryCache(LruResourceCache(memoryCacheSizeBy
tes.toLong()))
    }

    override fun registerComponents(context: Context,
glide: Glide, registry: Glide.Registry) {
        // 使用自定义的 OkHttpClient
        val okHttpClient = OkHttpClient.Builder()
```

```
            .addInterceptor { chain ->
                val originalRequest = chain.request()
                val newRequest =
originalRequest.newBuilder()
                    .addHeader("Authorization", "Bearer
YOUR_ACCESS_TOKEN")
                    .build()
                chain.proceed(newRequest)
            }
            .build()

        registry.replace(
            GlideUrl::class.java,
            InputStream::class.java,
            OkHttpUrlLoader.Factory(okHttpClient)
        )
    }
}
```

4. 使用 Glide

Glide 会自动应用 CustomGlideModule 中的配置。加载图片时，不需要做额外的配置：

```
Glide.with(context)
    .load("https://example.com/your_image.jpg")
    .into(imageView)
```

注意事项

1. **安全性**：不要在客户端硬编码敏感信息（如 API 密钥、Token）。可以考虑从安全存储（如 Android Keystore 或加密存储）中动态获取这些信息。
2. **动态 Header**：如果 Header 需要根据不同请求动态变化，可以通过依赖注入或其他机制，将 Header 信息传递到拦截器中。

这样，你就可以为 Glide 添加自定义的请求头了！