Optional: Class-based Interceptors

Besides defining HTTP interceptors as functions (which is the modern, recommended way of doing it), you can also define HTTP interceptors via classes.

For example, the `loggingInterceptor` from the previous lecture could be defined like this (when using this class-based approach):

```
import {
  HttpEvent,
  HttpHandler,
  HttpInterceptor,
  HttpRequest,
} from '@angular/common/http';
import { Observable } from 'rxjs';


@Injectable()
class LoggingInterceptor implements HttpInterceptor {
  intercept(req: HttpRequest<unknown>, handler: HttpHandler): Observable<HttpEvent
<any>> {
    console.log('Request URL: ' + req.url);
    return handler.handle(req);
  }
}
```

An interceptor defined like this, must be provided in a different way than before though.

Instead of providing it like this:

```
providers: [
  provideHttpClient(
    withInterceptors([loggingInterceptor]),
  )
],
```

You now must use `withInterceptorsFromDi()` and set up a custom provider, like this:

```
providers: [
  provideHttpClient(
    withInterceptorsFromDi()
  ),
  { provide: HTTP_INTERCEPTORS, useClass: LoggingInterceptor, multi: true }
]
```