```java

public class GrantTypeAuthenticationProvider implements AuthenticationProvider {

    private final OAuth2AuthorizationService authorizationService;
    private final OAuth2TokenGenerator< ? extends OAuth2Token > tokenGenerator;

    public GrantTypeAuthenticationProvider( OAuth2AuthorizationService authorizationService ,
                                            OAuth2TokenGenerator< ? extends OAuth2Token > tokenGenerator ) {
        this.authorizationService = authorizationService;
        this.tokenGenerator = tokenGenerator;
    }


    @Override
    public Authentication authenticate( Authentication authentication ) throws AuthenticationException {

        GrantTypeAuthenticationToken customCodeGrantAuthentication = (GrantTypeAuthenticationToken) authentication;

        // Ensure the client is authenticated
        OAuth2ClientAuthenticationToken clientPrincipal = getAuthenticatedClientElseThrowInvalidClient( customCodeGrantAuthentication );
        RegisteredClient registeredClient = clientPrincipal.getRegisteredClient();

        // Ensure the client is configured to use this authorization grant type
        if (!registeredClient.getAuthorizationGrantTypes()
                             .contains( customCodeGrantAuthentication.getGrantType() )) {
            throw new OAuth2AuthenticationException( OAuth2ErrorCodes.UNAUTHORIZED_CLIENT );
        }

        //TODO: Validate the code parameter

        // Generate the access token
        OAuth2TokenContext tokenContext = DefaultOAuth2TokenContext.builder()
                                                                   .registeredClient( registeredClient )
                                                                   .principal( clientPrincipal )
                                                                   .authorizationServerContext( AuthorizationServerContextHolder.getContext() )
                                                                   .tokenType( OAuth2TokenType.ACCESS_TOKEN )
                                                                   .authorizationGrantType( customCodeGrantAuthentication.getGrantType() )
                                                                   .authorizationGrant( customCodeGrantAuthentication )
                                                                   .build();

        OAuth2Token generatedAccessToken = this.tokenGenerator.generate( tokenContext );
        if (generatedAccessToken == null) {
            OAuth2Error error = new OAuth2Error( OAuth2ErrorCodes.SERVER_ERROR ,
                                                 "The token generator failed to generate the access token." ,
                                                 null );
            throw new OAuth2AuthenticationException( error );
        }
        OAuth2AccessToken accessToken = new OAuth2AccessToken( OAuth2AccessToken.TokenType.BEARER ,
                                                               generatedAccessToken.getTokenValue() ,
                                                               generatedAccessToken.getIssuedAt() ,
                                                               generatedAccessToken.getExpiresAt() ,
                                                               null );


```

```
55
56
57          // Initialize the OAuth2Authorization
58          OAuth2Authorization.Builder authorizationBuilder = OAuth2Authorization.withRegisteredClient( registeredClient )
59                                                                                 .principalName( clientPrincipal.getName() )
60                                                                                 .authorizationGrantType( customCodeGrantAuthentication.getGrantTyp());
61          if (generatedAccessToken instanceof ClaimAccessor) {
62              authorizationBuilder.token( accessToken ,
63                                          ( metadata ) -> metadata.put( OAuth2Authorization.Token.CLAIMS_METADATA_NAME ,
64                                                                                                ( (ClaimAccessor) generatedAccessToken ).getClaims() ) );
65          } else {
66              authorizationBuilder.accessToken( accessToken );
67          }
68          OAuth2Authorization authorization = authorizationBuilder.build();
69
70
71          // Save the OAuth2Authorization
72          this.authorizationService.save( authorization );
73
74          return new OAuth2AccessTokenAuthenticationToken( registeredClient ,
75                                                           clientPrincipal ,
76                                                           accessToken );
77      }
78
79      @Override
80      public boolean supports( Class< ? > authentication ) {
81          return GrantTypeAuthenticationToken.class.isAssignableFrom( authentication );
82                  ;
83      }
84  }
85
```