Łączenie interfejsu GUI z kodem Javy

Czas połączyć układ GUI z kodem źródłowym Javy, aby rozpocząć programowanie działania aplikacji mobilnej Secret Messages. Otwieramy plik *MainActivity.java*, klikając zakładkę w lewym górnym rogu głównego okna zawartości.

Bezpośrednio do deklaracji `public class MainActivity` dodajemy 5 następujących wierszy kodu:

```
public class MainActivity extends AppCompatActivity {
    EditText txtIn;
    EditText txtKey;
    EditText txtOut;
    SeekBar sb;
    Button btn;
```

Trzeba zaimportować klasy do kodu Java, naciskając ALT-ENTER lub OPTION-ENTER w poszczególnych wierszach kodu.

Dodane zmienne wskazują komponenty GUI w wizualnym układzie aplikacji. Możemy połączyć nazwy zmiennych z prawdziwym widżetami, dodając pięć wierszy kodu do metody `onCreate()`, tuż pod instrukcją `setSupportActionBar(toolbar)`:

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
    setSupportActionBar(toolbar);
    txtIn = (EditText)findViewById(R.id.txtIn);
    txtKey = (EditText)findViewById(R.id.txtKey);
    txtOut = (EditText)findViewById(R.id.txtOut);
    sb = (SeekBar)findViewById(R.id.seekBar);
    btn = (Button)findViewById(R.id.button);
```

Funkcja Code Assist dostępna w Android Studio automatycznie uzupełni większość instrukcji – wystarczy wpisać kilka pierwszych znaków, a następnie wybrać pozycję z listy lub nacisnąć klawisz ENTER, aby zaakceptować podpowiadany kod. Takie podejście jest szybsze i pomaga unikać literówek.

Łączenie przycisku Encode z metodą encode()

Skoro mamy już kod połączony z komponentami układu GUI, możemy skopiować metodę `encode()` z komputerowej wersji aplikacji z rozdziału 7. Wkleimy metodę do deklaracji `public class MainActivity`, tuż pod dodanymi wcześniej pięcioma wierszami kodu deklarującymi zmienne, które wskazują komponenty GUI w układzie.

Otwieramy projekt komputerowy *SecretMessagesGUI* w programie Eclipse i zaznaczamy całą metodę `encode()`. Kopiujemy zaznaczony kod i wklejamy go w deklaracji `public class MainActivity` w Android Studio:

```
public class MainActivity extends AppCompatActivity {
    EditText txtIn;
    EditText txtKey;
    EditText txtOut;
    SeekBar sb;
    Button btn;
    public String encode( String message, int keyVal ) {
        String output = "";
        char key = (char) keyVal;
        for ( int x = 0; x < message.length(); x++ ) {
            char input = message.charAt(x);
            if (input >= 'A' && input <= 'Z')
            {
                input += key;
                if (input > 'Z')
                    input -= 26;
                if (input < 'A')
                input += 26;
            }
            else if (input >= 'a' && input <= 'z')
            {
                input += key;
                if (input > 'z')
                    input -= 26;
                if (input < 'a')
                input += 26;
            }
            else if (input >= '0' && input <= '9')
            {
                input += (keyVal % 10);
                if (input > '9')
                    input -= 10;
                if (input < '0')
                input += 10;
            }
            output += input;
        }
        return output;
    }
```

Po umieszczeniu metody `encode()` w kodzie aplikacji wystarczy wywoływać ją przy każdym kliknięciu przycisku Encode/Decode. W tym celu utworzymy metodę obsługi zdarzenia `OnClickListener` dla przycisku Encode/Decode i wywołamy w niej metodę `encode()`.

W metodzie `onCreate()` w pliku *MainActivity.java* rozpoczynamy wpisywanie kodu **`btn.setOnCl`** aż do momentu pojawienia się podpowiedzi, jak pokazano na rysunku 8.9. Z listy wybieramy **`setOnClickListener()`**.