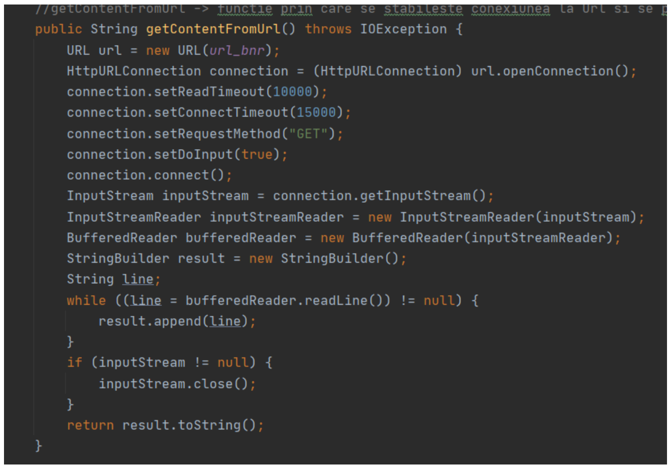
Document tehnic

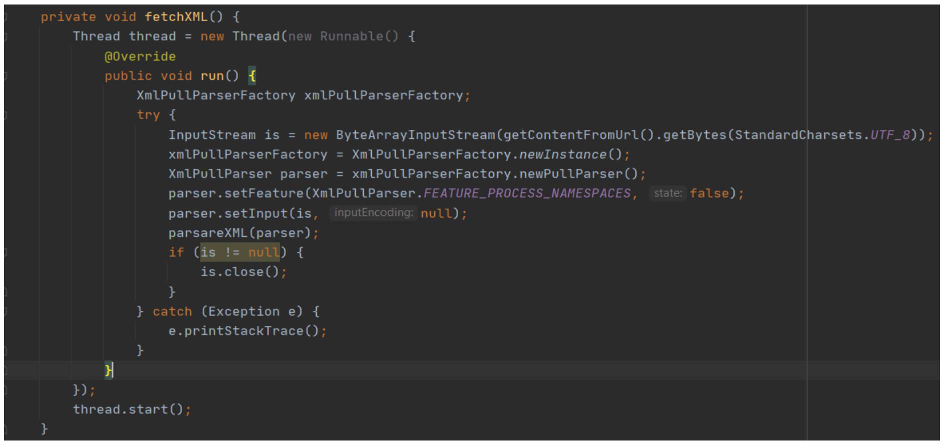
~Curs BNR - Inventar~

- getContentFromURL(){} -> functie care preia continutul unui fisier de tip XML de la URL-ul : “ https://www.bnr.ro/nbrfxrates.xml ” si il transmite ca si input catre parser(URL-ul specific ultimei actualizari a cursului).



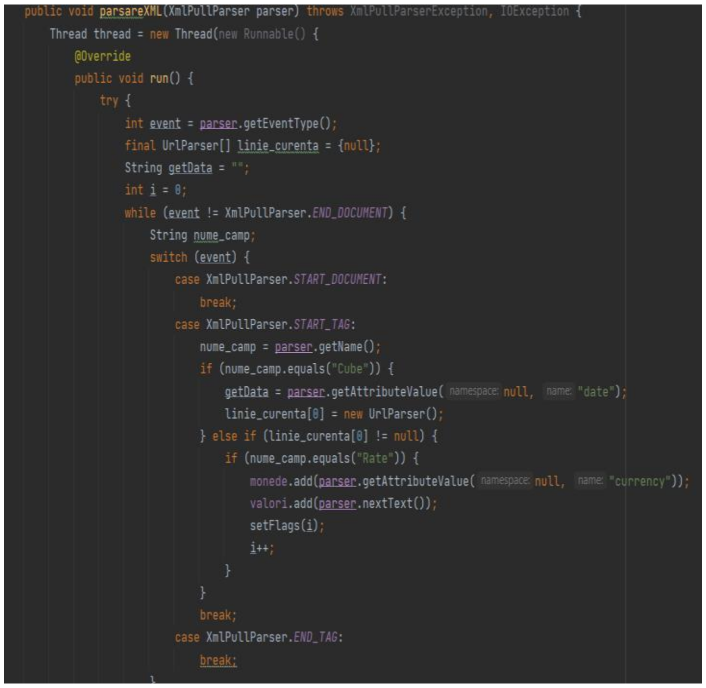
```java
//getContentFromUrl -> functie prin care se stabileste conexiunea la Url si se p
public String getContentFromUrl() throws IOException {
    URL url = new URL(url_bnr);
    HttpURLConnection connection = (HttpURLConnection) url.openConnection();
    connection.setReadTimeout(10000);
    connection.setConnectTimeout(15000);
    connection.setRequestMethod("GET");
    connection.setDoInput(true);
    connection.connect();
    InputStream inputStream = connection.getInputStream();
    InputStreamReader inputStreamReader = new InputStreamReader(inputStream);
    BufferedReader bufferedReader = new BufferedReader(inputStreamReader);
    StringBuilder result = new StringBuilder();
    String line;
    while ((line = bufferedReader.readLine()) != null) {
        result.append(line);
    }
    if (inputStream != null) {
        inputStream.close();
    }
    return result.toString();
}
```

- fetchXML(){} -> functie care creeaza o noua instanta de parser, caruia ii atribui input-ul dorit si care apeleaza functia de parsare necesara.

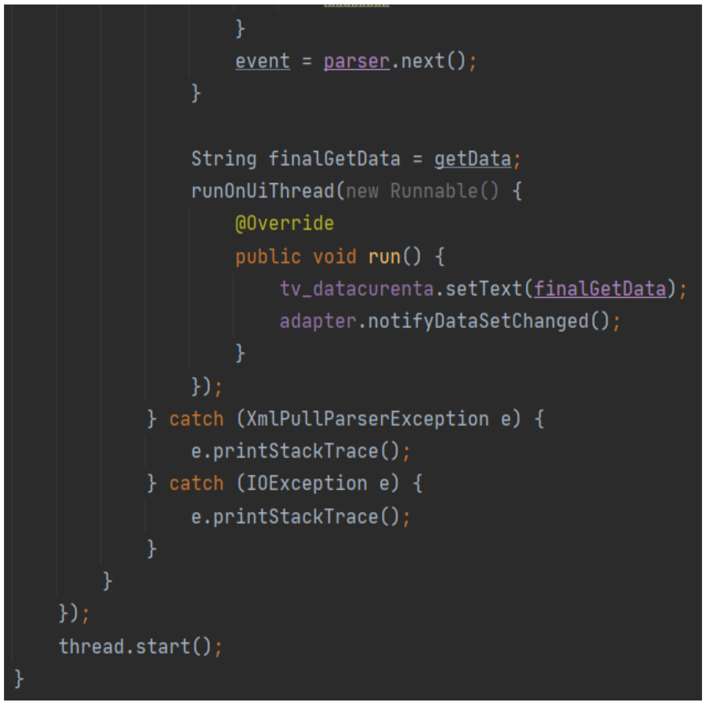


```java
private void fetchXML() {
    Thread thread = new Thread(new Runnable() {
        @Override
        public void run() {
            XmlPullParserFactory xmlPullParserFactory;
            try {
                InputStream is = new ByteArrayInputStream(getContentFromUrl().getBytes(StandardCharsets.UTF_8));
                xmlPullParserFactory = XmlPullParserFactory.newInstance();
                XmlPullParser parser = xmlPullParserFactory.newPullParser();
                parser.setFeature(XmlPullParser.FEATURE_PROCESS_NAMESPACES, state: false);
                parser.setInput(is, inputEncoding: null);
                parsareXML(parser);
                if (is != null) {
                    is.close();
                }
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    });
    thread.start();
}
```

- parsareXML(){} -> functie care realizeaza parsarea propriu-zisa si adaugarea datelor necesare (monele si valoarea corespunzatoare lor) in lista ce urmeaza sa fie afisata.

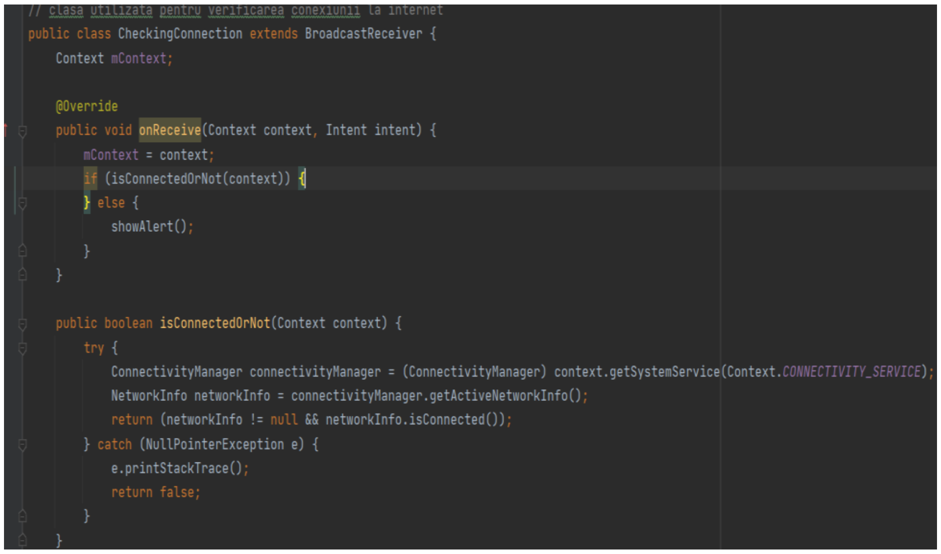


```java
public void parsareXML(XmlPullParser parser) throws XmlPullParserException, IOException {
    Thread thread = new Thread(new Runnable() {
        @Override
        public void run() {
            try {
                int event = parser.getEventType();
                final UrlParser[] linie_curenta = {null};
                String getData = "";
                int i = 0;
                while (event != XmlPullParser.END_DOCUMENT) {
                    String nume_camp;
                    switch (event) {
                        case XmlPullParser.START_DOCUMENT:
                            break;
                        case XmlPullParser.START_TAG:
                            nume_camp = parser.getName();
                            if (nume_camp.equals("Cube")) {
                                getData = parser.getAttributeValue( namespace: null,  name: "date");
                                linie_curenta[0] = new UrlParser();
                            } else if (linie_curenta[0] != null) {
                                if (nume_camp.equals("Rate")) {
                                    monede.add(parser.getAttributeValue( namespace: null,  name: "currency"));
                                    valori.add(parser.nextText());
                                    setFlags(i);
                                    i++;
                                }
                            }
                            break;
                        case XmlPullParser.END_TAG:
                            break;
```

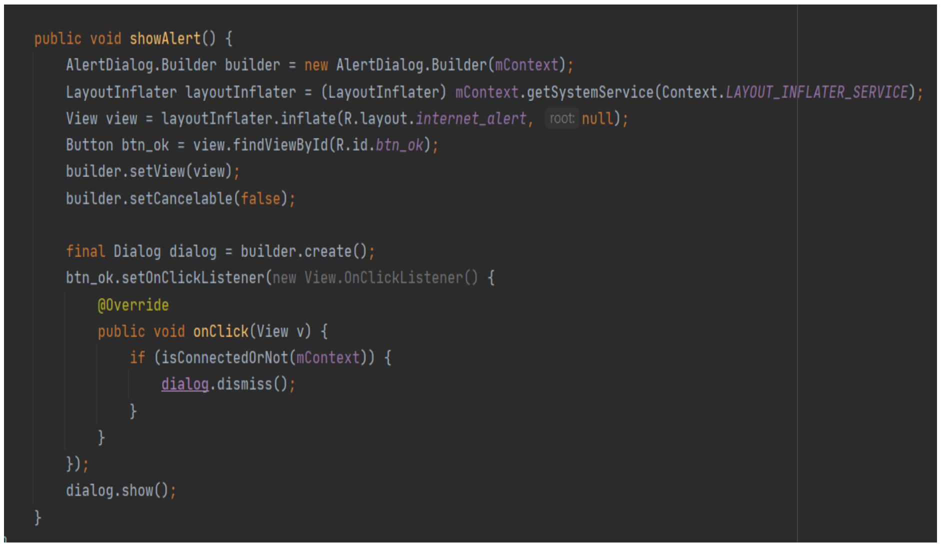


```java
                    }
                    event = parser.next();
                }

                String finalGetData = getData;
                runOnUiThread(new Runnable() {
                    @Override
                    public void run() {
                        tv_datacurenta.setText(finalGetData);
                        adapter.notifyDataSetChanged();
                    }
                });
            } catch (XmlPullParserException e) {
                e.printStackTrace();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    });
    thread.start();
}
```

- CheckingConnection -> clasa utilizata pentru verificarea conexiunii dispozitivului la internet.

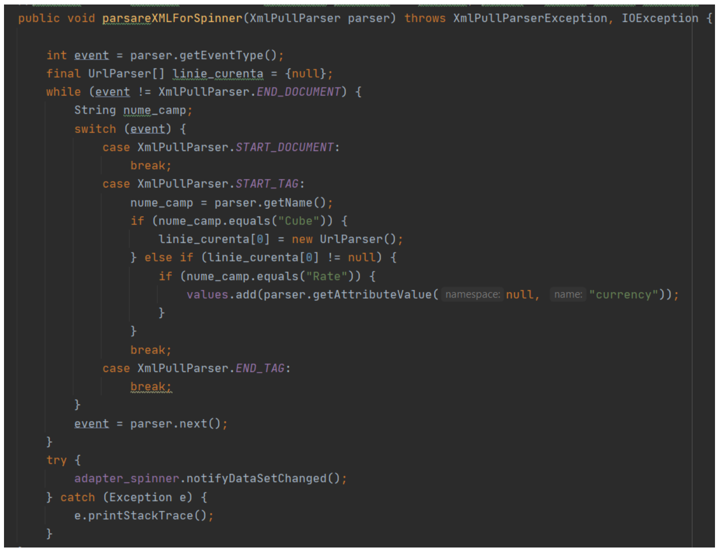


```java
// clasa utilizata pentru verificarea conexiunii la internet
public class CheckingConnection extends BroadcastReceiver {
    Context mContext;

    @Override
    public void onReceive(Context context, Intent intent) {
        mContext = context;
        if (isConnectedOrNot(context)) {
        } else {
            showAlert();
        }
    }

    public boolean isConnectedOrNot(Context context) {
        try {
            ConnectivityManager connectivityManager = (ConnectivityManager) context.getSystemService(Context.CONNECTIVITY_SERVICE);
            NetworkInfo networkInfo = connectivityManager.getActiveNetworkInfo();
            return (networkInfo != null && networkInfo.isConnected());
        } catch (NullPointerException e) {
            e.printStackTrace();
            return false;
        }
    }
```



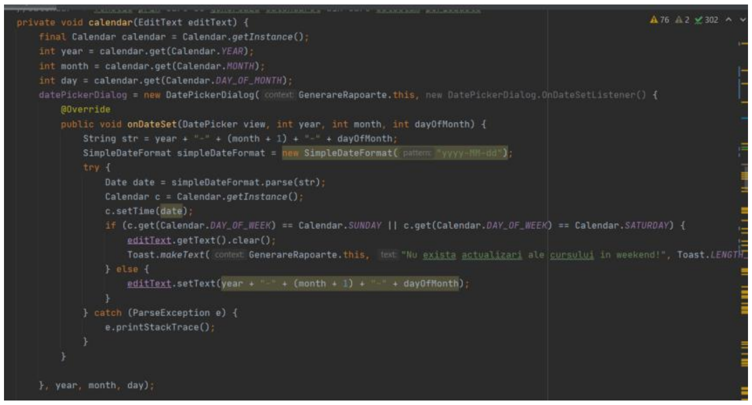
```java
public void showAlert() {
    AlertDialog.Builder builder = new AlertDialog.Builder(mContext);
    LayoutInflater layoutInflater = (LayoutInflater) mContext.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
    View view = layoutInflater.inflate(R.layout.internet_alert,  root: null);
    Button btn_ok = view.findViewById(R.id.btn_ok);
    builder.setView(view);
    builder.setCancelable(false);

    final Dialog dialog = builder.create();
    btn_ok.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            if (isConnectedOrNot(mContext)) {
                dialog.dismiss();
            }
        }
    });
    dialog.show();
}
```

- parsareXMLForSpinner(){} -> functie ce realizeaza parsarea propriu-zisa a fisierului XML , preia din acesta doar numele monedelor si le adaugata in lista ce reprezinta elementele spinner-ului.

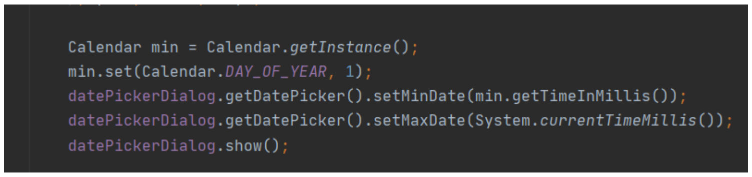


```java
public void parsareXMLForSpinner(XmlPullParser parser) throws XmlPullParserException, IOException {

    int event = parser.getEventType();
    final UrlParser[] linie_curenta = {null};
    while (event != XmlPullParser.END_DOCUMENT) {
        String nume_camp;
        switch (event) {
            case XmlPullParser.START_DOCUMENT:
                break;
            case XmlPullParser.START_TAG:
                nume_camp = parser.getName();
                if (nume_camp.equals("Cube")) {
                    linie_curenta[0] = new UrlParser();
                } else if (linie_curenta[0] != null) {
                    if (nume_camp.equals("Rate")) {
                        values.add(parser.getAttributeValue( namespace: null,  name: "currency"));
                    }
                }
                break;
            case XmlPullParser.END_TAG:
                break;
        }
        event = parser.next();
    }
    try {
        adapter_spinner.notifyDataSetChanged();
    } catch (Exception e) {
        e.printStackTrace();
    }
```

- calendar(){} -> functie care genereaza calendarul din care se aleg data de inceput si cea de sfarist.

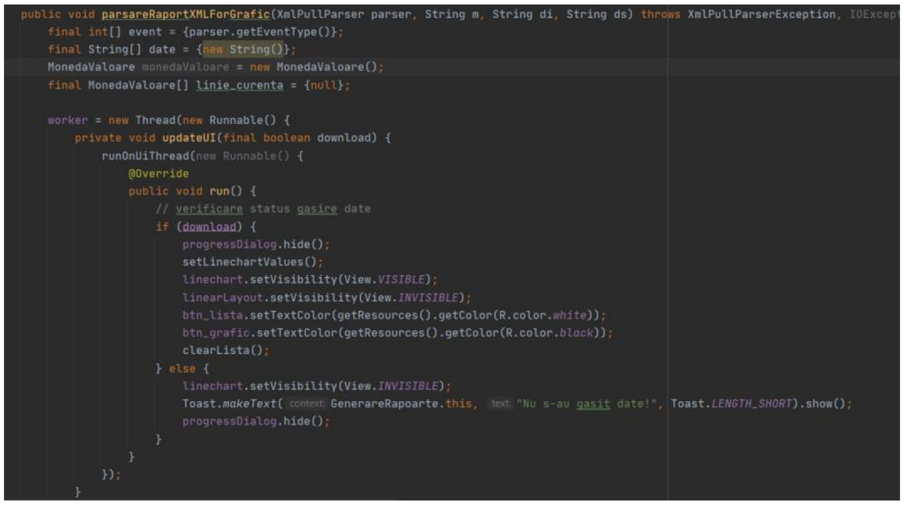


```java
private void calendar(EditText editText) {
    final Calendar calendar = Calendar.getInstance();
    int year = calendar.get(Calendar.YEAR);
    int month = calendar.get(Calendar.MONTH);
    int day = calendar.get(Calendar.DAY_OF_MONTH);
    datePickerDialog = new DatePickerDialog( context: GenerareRapoarte.this, new DatePickerDialog.OnDateSetListener() {
        @Override
        public void onDateSet(DatePicker view, int year, int month, int dayOfMonth) {
            String str = year + "-" + (month + 1) + "-" + dayOfMonth;
            SimpleDateFormat simpleDateFormat = new SimpleDateFormat( pattern: "yyyy-MM-dd");
            try {
                Date date = simpleDateFormat.parse(str);
                Calendar c = Calendar.getInstance();
                c.setTime(date);
                if (c.get(Calendar.DAY_OF_WEEK) == Calendar.SUNDAY || c.get(Calendar.DAY_OF_WEEK) == Calendar.SATURDAY) {
                    editText.getText().clear();
                    Toast.makeText( context: GenerareRapoarte.this,  text: "Nu exista actualizari ale cursului in weekend!", Toast.LENGTH_
                } else {
                    editText.setText(year + "-" + (month + 1) + "-" + dayOfMonth);
                }
            } catch (ParseException e) {
                e.printStackTrace();
            }
        }

    }, year, month, day);
```

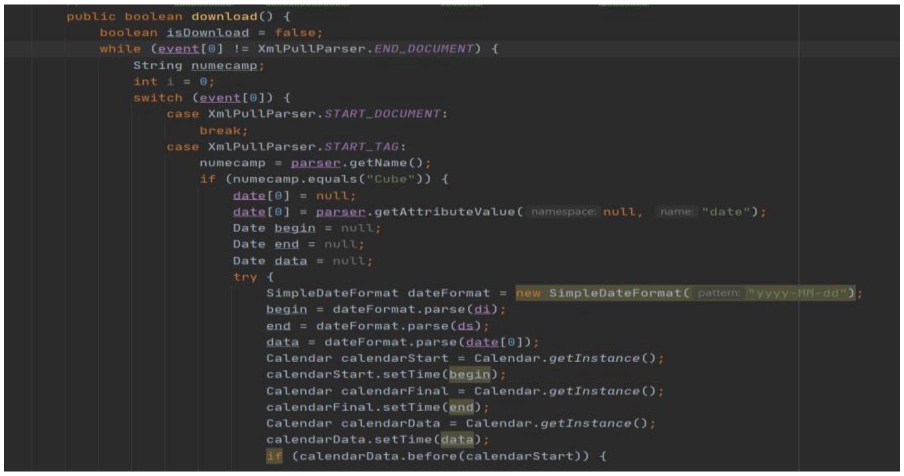


```java
    Calendar min = Calendar.getInstance();
    min.set(Calendar.DAY_OF_YEAR, 1);
    datePickerDialog.getDatePicker().setMinDate(min.getTimeInMillis());
    datePickerDialog.getDatePicker().setMaxDate(System.currentTimeMillis());
    datePickerDialog.show();
```

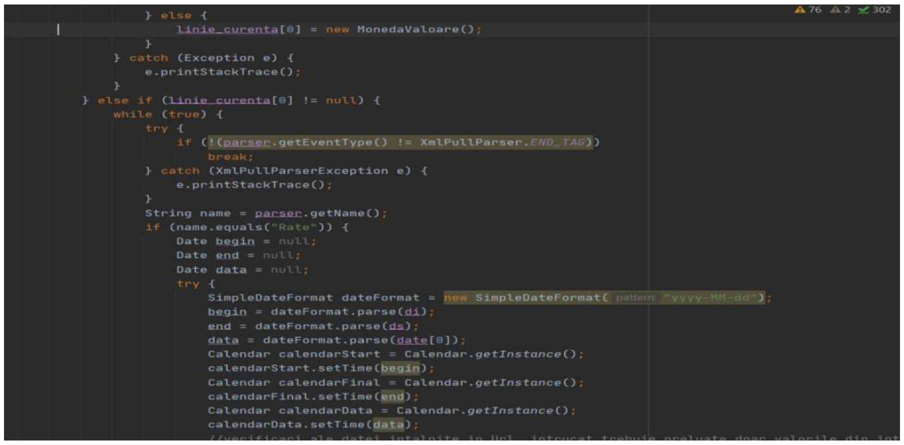
- parsareRaportXMLForGrafic(){} -> functie ce realizeaza parsarea propriu-zisa a fisierului XML, preia din acesta in functie de moneda selectata in spinner, toate valorile acesteia din intervalul ales. Valorile sunt atribuite unui vector care reprezinta valorile de pe grafic.(XML este preluat de la URL-ul :“ https://www.bnr.ro/files/xml/years/nbrfxrates2021.xml ”)



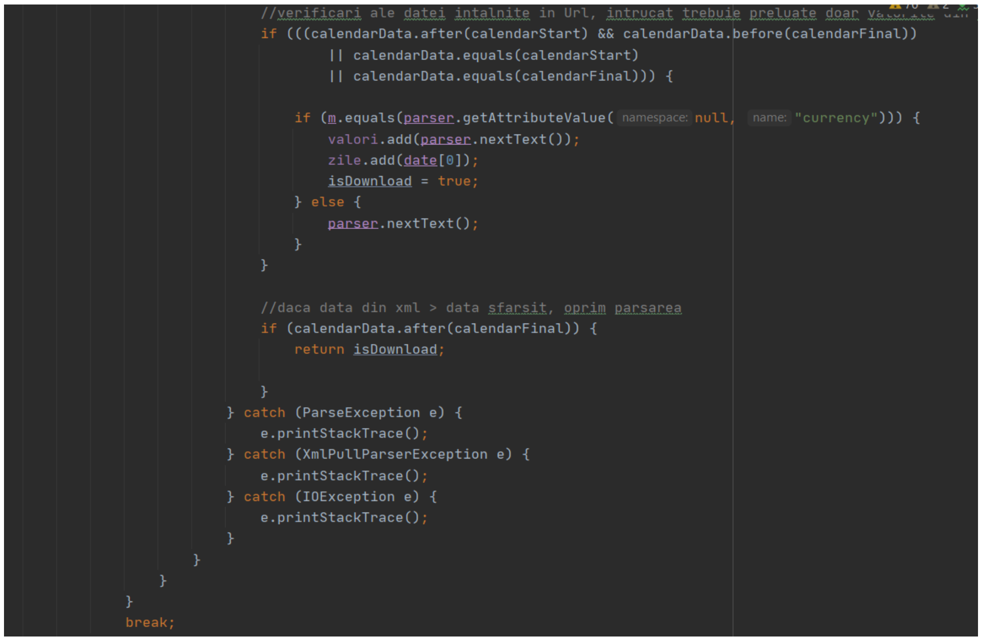
```java
public void parsareRaportXMLForGrafic(XmlPullParser parser, String m, String di, String ds) throws XmlPullParserException, IOExcept
    final int[] event = {parser.getEventType()};
    final String[] date = {new String()};
    MonedaValoare monedaValoare = new MonedaValoare();
    final MonedaValoare[] linie_curenta = {null};

    worker = new Thread(new Runnable() {
        private void updateUI(final boolean download) {
            runOnUiThread(new Runnable() {
                @Override
                public void run() {
                    // verificare status gasire date
                    if (download) {
                        progressDialog.hide();
                        setLinechartValues();
                        linechart.setVisibility(View.VISIBLE);
                        linearLayout.setVisibility(View.INVISIBLE);
                        btn_lista.setTextColor(getResources().getColor(R.color.white));
                        btn_grafic.setTextColor(getResources().getColor(R.color.black));
                        clearLista();
                    } else {
                        linechart.setVisibility(View.INVISIBLE);
                        Toast.makeText( context: GenerareRapoarte.this,  text: "Nu s-au gasit date!", Toast.LENGTH_SHORT).show();
                        progressDialog.hide();
                    }
                }
            });
        }
```



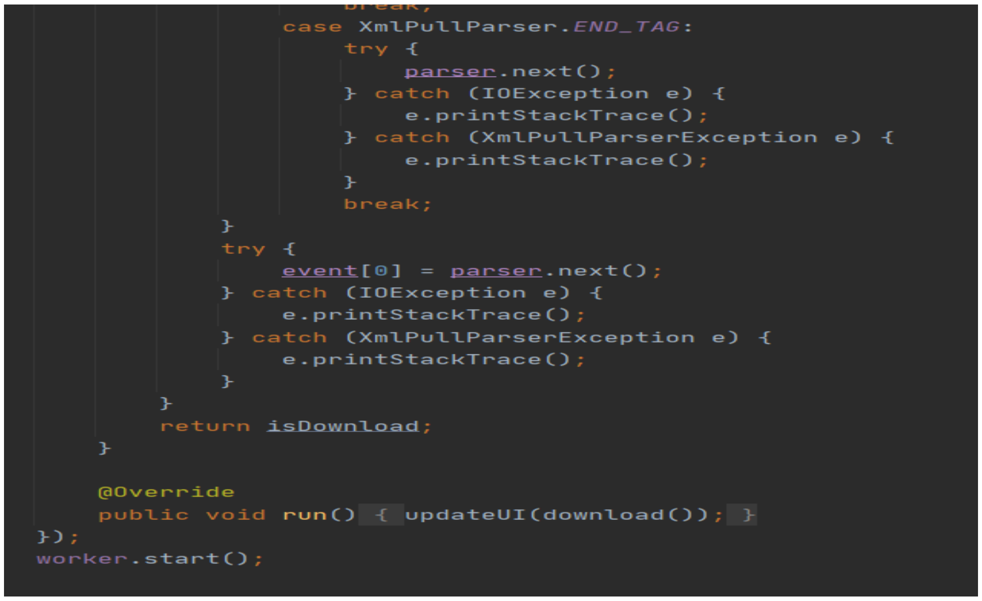
```java
        public boolean download() {
            boolean isDownload = false;
            while (event[0] != XmlPullParser.END_DOCUMENT) {
                String numecamp;
                int i = 0;
                switch (event[0]) {
                    case XmlPullParser.START_DOCUMENT:
                        break;
                    case XmlPullParser.START_TAG:
                        numecamp = parser.getName();
                        if (numecamp.equals("Cube")) {
                            date[0] = null;
                            date[0] = parser.getAttributeValue( namespace: null,  name: "date");
                            Date begin = null;
                            Date end = null;
                            Date data = null;
                            try {
                                SimpleDateFormat dateFormat = new SimpleDateFormat( pattern: "yyyy-MM-dd");
                                begin = dateFormat.parse(di);
                                end = dateFormat.parse(ds);
                                data = dateFormat.parse(date[0]);
                                Calendar calendarStart = Calendar.getInstance();
                                calendarStart.setTime(begin);
                                Calendar calendarFinal = Calendar.getInstance();
                                calendarFinal.setTime(end);
                                Calendar calendarData = Calendar.getInstance();
                                calendarData.setTime(data);
                                if (calendarData.before(calendarStart)) {
```



```java
                    } else {
                        linie_curenta[0] = new MonedaValoare();
                    }
                } catch (Exception e) {
                    e.printStackTrace();
                }
            } else if (linie_curenta[0] != null) {
                while (true) {
                    try {
                        if (!(parser.getEventType() != XmlPullParser.END_TAG))
                            break;
                    } catch (XmlPullParserException e) {
                        e.printStackTrace();
                    }
                    String name = parser.getName();
                    if (name.equals("Rate")) {
                        Date begin = null;
                        Date end = null;
                        Date data = null;
                        try {
                            SimpleDateFormat dateFormat = new SimpleDateFormat( pattern: "yyyy-MM-dd");
                            begin = dateFormat.parse(di);
                            end = dateFormat.parse(ds);
                            data = dateFormat.parse(date[0]);
                            Calendar calendarStart = Calendar.getInstance();
                            calendarStart.setTime(begin);
                            Calendar calendarFinal = Calendar.getInstance();
                            calendarFinal.setTime(end);
                            Calendar calendarData = Calendar.getInstance();
                            calendarData.setTime(data);
                            //verificari ale datei intalnite in Url , intrucat trebuie preluate doar valorile din int
```

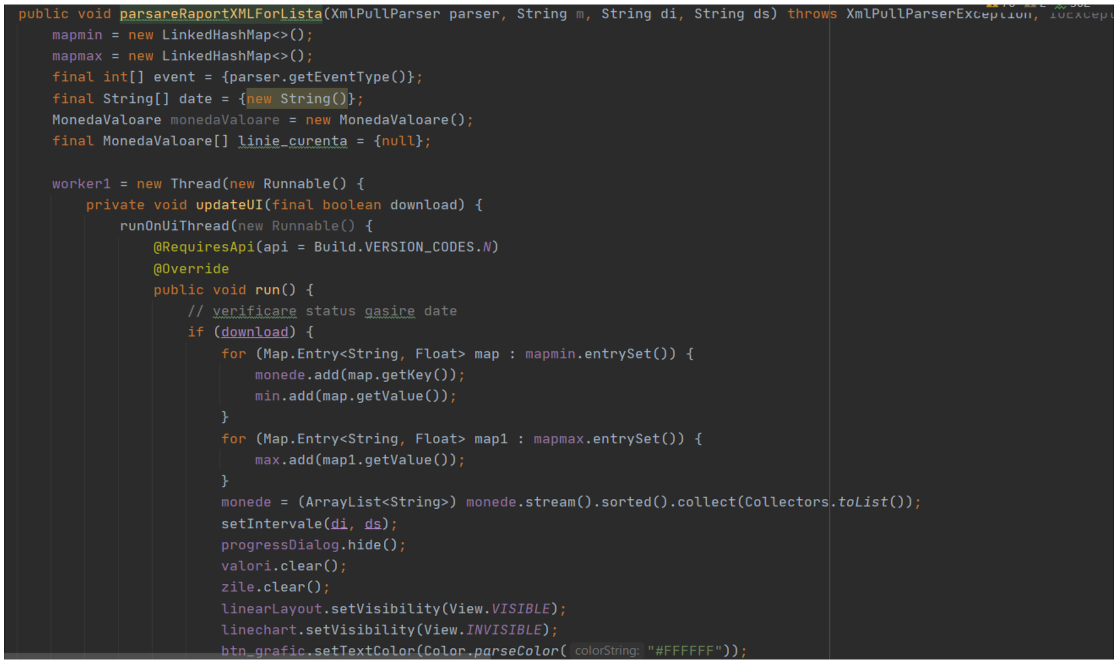


```java
                //verificari ale datei intalnite in Url, intrucat trebuie preluate doar valorile din
                if (((calendarData.after(calendarStart) && calendarData.before(calendarFinal))
                        || calendarData.equals(calendarStart)
                        || calendarData.equals(calendarFinal))) {

                    if (m.equals(parser.getAttributeValue( namespace: null,  name: "currency"))) {
                        valori.add(parser.nextText());
                        zile.add(date[0]);
                        isDownload = true;
                    } else {
                        parser.nextText();
                    }
                }

                //daca data din xml > data sfarsit, oprim parsarea
                if (calendarData.after(calendarFinal)) {
                    return isDownload;

                }
            } catch (ParseException e) {
                e.printStackTrace();
            } catch (XmlPullParserException e) {
                e.printStackTrace();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
}
break;
```

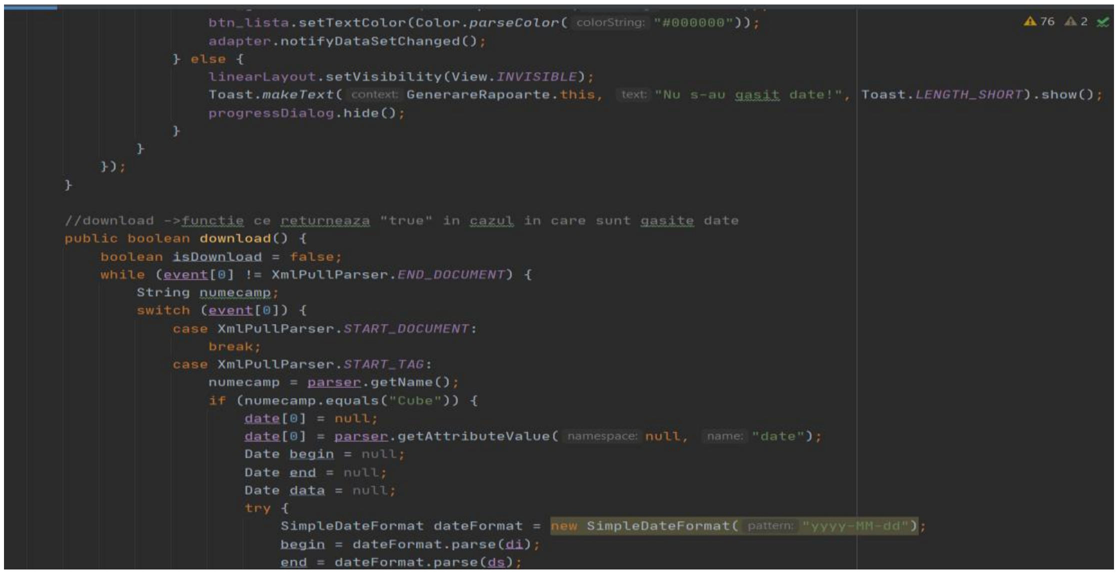


```java
                    break;
                case XmlPullParser.END_TAG:
                    try {
                        parser.next();
                    } catch (IOException e) {
                        e.printStackTrace();
                    } catch (XmlPullParserException e) {
                        e.printStackTrace();
                    }
                    break;
            }
            try {
                event[0] = parser.next();
            } catch (IOException e) {
                e.printStackTrace();
            } catch (XmlPullParserException e) {
                e.printStackTrace();
            }
        }
        return isDownload;
    }

    @Override
    public void run() { updateUI(download()); }
});
worker.start();
```

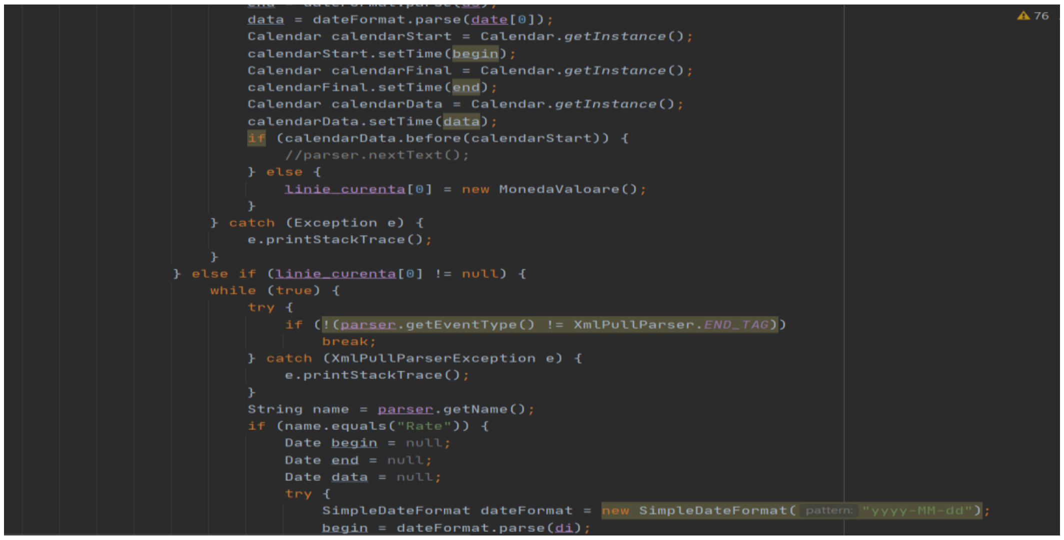
- parsareRaportXMLForLista(){} -> functie ce realizeaza parsarea propriu-zisa a fisierului XML, preia din acesta in functie de intervalul ales, valorile minime si maxime pentru fiecare moneda, valorile minime si maxime fiind decise cu ajutorul a doua LinkedHashMap-uri in care se stocheaza valorile respective. (XML este preluat de la URL-ul : " https://www.bnr.ro/files/xml/years/nbrfxrates2021.xml ")



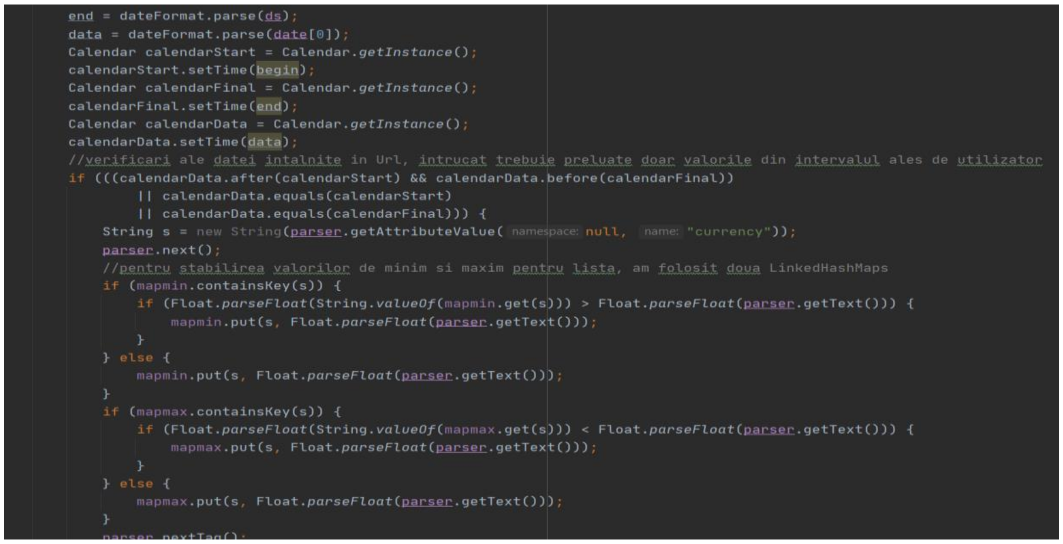
```java
public void parsareRaportXMLForLista(XmlPullParser parser, String m, String di, String ds) throws XmlPullParserException, IOExcep
    mapmin = new LinkedHashMap<>();
    mapmax = new LinkedHashMap<>();
    final int[] event = {parser.getEventType()};
    final String[] date = {new String()};
    MonedaValoare monedaValoare = new MonedaValoare();
    final MonedaValoare[] linie_curenta = {null};

    worker1 = new Thread(new Runnable() {
        private void updateUI(final boolean download) {
            runOnUiThread(new Runnable() {
                @RequiresApi(api = Build.VERSION_CODES.N)
                @Override
                public void run() {
                    // verificare status gasire date
                    if (download) {
                        for (Map.Entry<String, Float> map : mapmin.entrySet()) {
                            monede.add(map.getKey());
                            min.add(map.getValue());
                        }
                        for (Map.Entry<String, Float> map1 : mapmax.entrySet()) {
                            max.add(map1.getValue());
                        }
                        monede = (ArrayList<String>) monede.stream().sorted().collect(Collectors.toList());
                        setIntervale(di, ds);
                        progressDialog.hide();
                        valori.clear();
                        zile.clear();
                        linearLayout.setVisibility(View.VISIBLE);
                        linechart.setVisibility(View.INVISIBLE);
                        btn_grafic.setTextColor(Color.parseColor( colorString: "#FFFFFF"));
```



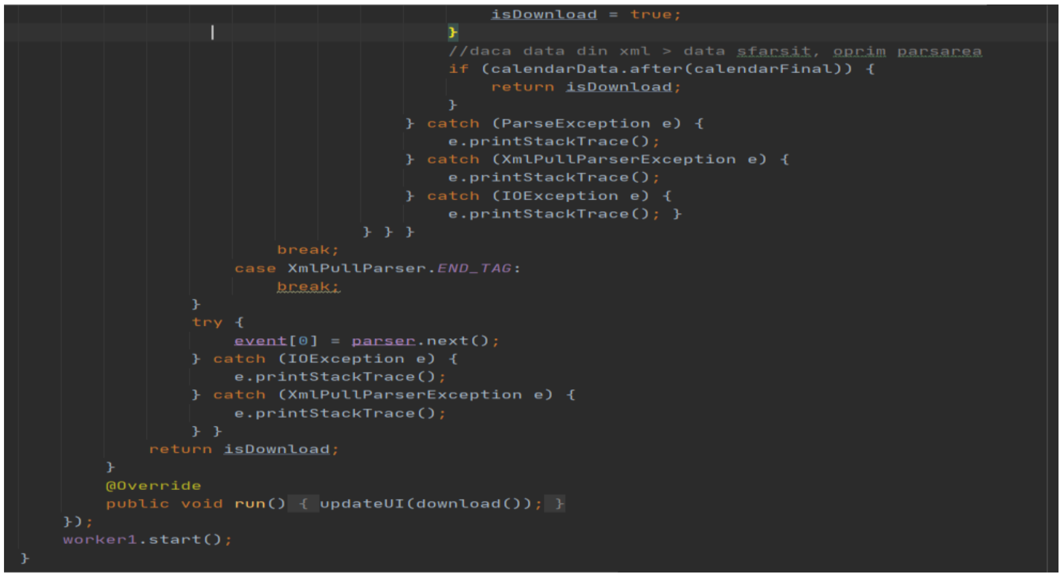
```java
                        btn_lista.setTextColor(Color.parseColor( colorString: "#000000"));
                        adapter.notifyDataSetChanged();
                    } else {
                        linearLayout.setVisibility(View.INVISIBLE);
                        Toast.makeText( context: GenerareRapoarte.this,  text: "Nu s-au gasit date!", Toast.LENGTH_SHORT).show();
                        progressDialog.hide();
                    }
                }
            });
        }

        //download ->functie ce returneaza "true" in cazul in care sunt gasite date
        public boolean download() {
            boolean isDownload = false;
            while (event[0] != XmlPullParser.END_DOCUMENT) {
                String numecamp;
                switch (event[0]) {
                    case XmlPullParser.START_DOCUMENT:
                        break;
                    case XmlPullParser.START_TAG:
                        numecamp = parser.getName();
                        if (numecamp.equals("Cube")) {
                            date[0] = null;
                            date[0] = parser.getAttributeValue( namespace: null,  name: "date");
                            Date begin = null;
                            Date end = null;
                            Date data = null;
                            try {
                                SimpleDateFormat dateFormat = new SimpleDateFormat( pattern: "yyyy-MM-dd");
                                begin = dateFormat.parse(di);
                                end = dateFormat.parse(ds);
```



```java
        data = dateFormat.parse(date[0]);
        Calendar calendarStart = Calendar.getInstance();
        calendarStart.setTime(begin);
        Calendar calendarFinal = Calendar.getInstance();
        calendarFinal.setTime(end);
        Calendar calendarData = Calendar.getInstance();
        calendarData.setTime(data);
        if (calendarData.before(calendarStart)) {
            //parser.nextText();
        } else {
            linie_curenta[0] = new MonedaValoare();
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
} else if (linie_curenta[0] != null) {
    while (true) {
        try {
            if (!(parser.getEventType() != XmlPullParser.END_TAG))
                break;
        } catch (XmlPullParserException e) {
            e.printStackTrace();
        }
        String name = parser.getName();
        if (name.equals("Rate")) {
            Date begin = null;
            Date end = null;
            Date data = null;
            try {
                SimpleDateFormat dateFormat = new SimpleDateFormat( pattern: "yyyy-MM-dd");
                begin = dateFormat.parse(di);
```

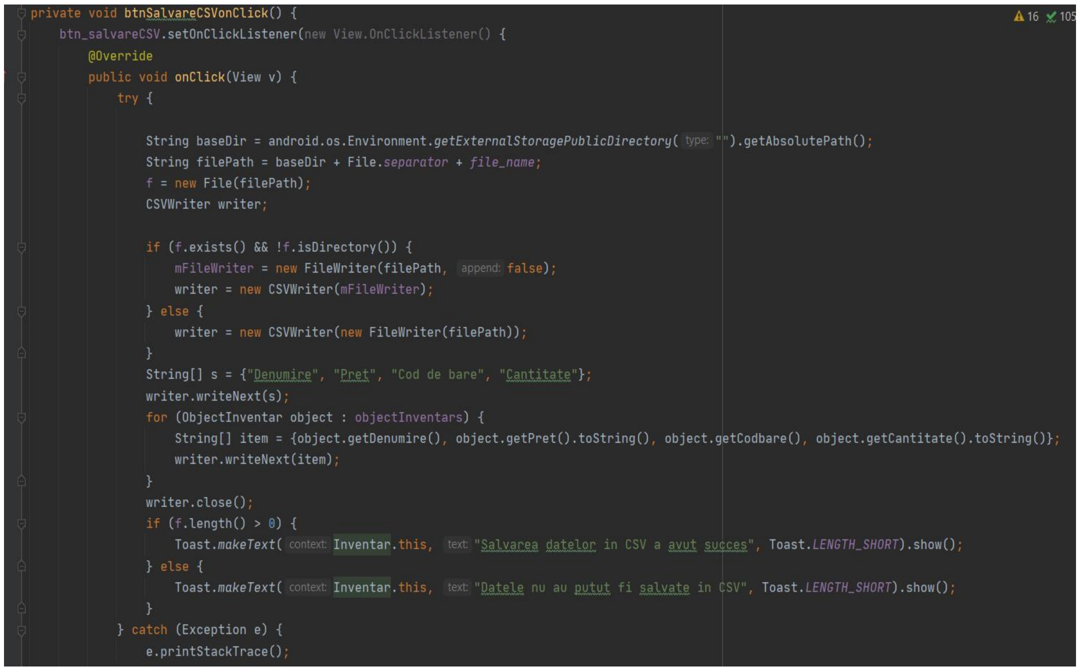


```java
end = dateFormat.parse(ds);
data = dateFormat.parse(date[0]);
Calendar calendarStart = Calendar.getInstance();
calendarStart.setTime(begin);
Calendar calendarFinal = Calendar.getInstance();
calendarFinal.setTime(end);
Calendar calendarData = Calendar.getInstance();
calendarData.setTime(data);
//verificari ale datei intalnite in Url, intrucat trebuie preluate doar valorile din intervalul ales de utilizator
if (((calendarData.after(calendarStart) && calendarData.before(calendarFinal))
        || calendarData.equals(calendarStart)
        || calendarData.equals(calendarFinal))) {
    String s = new String(parser.getAttributeValue( namespace: null,  name: "currency"));
    parser.next();
    //pentru stabilirea valorilor de minim si maxim pentru lista, am folosit doua LinkedHashMaps
    if (mapmin.containsKey(s)) {
        if (Float.parseFloat(String.valueOf(mapmin.get(s))) > Float.parseFloat(parser.getText())) {
            mapmin.put(s, Float.parseFloat(parser.getText()));
        }
    } else {
        mapmin.put(s, Float.parseFloat(parser.getText()));
    }
    if (mapmax.containsKey(s)) {
        if (Float.parseFloat(String.valueOf(mapmax.get(s))) < Float.parseFloat(parser.getText())) {
            mapmax.put(s, Float.parseFloat(parser.getText()));
        }
    } else {
        mapmax.put(s, Float.parseFloat(parser.getText()));
    }
    parser.nextTag();
```



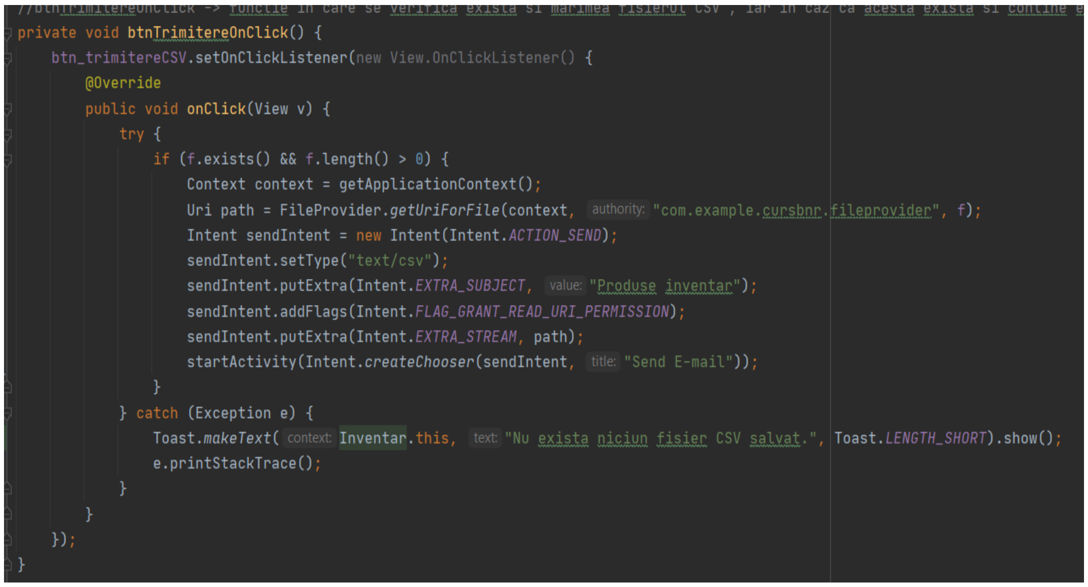
```java
                                            isDownload = true;
                                        }
                                        //daca data din xml > data sfarsit, oprim parsarea
                                        if (calendarData.after(calendarFinal)) {
                                            return isDownload;
                                        }
                                    } catch (ParseException e) {
                                        e.printStackTrace();
                                    } catch (XmlPullParserException e) {
                                        e.printStackTrace();
                                    } catch (IOException e) {
                                        e.printStackTrace(); }
                                } } }
                        break;
                    case XmlPullParser.END_TAG:
                        break;
                }
                try {
                    event[0] = parser.next();
                } catch (IOException e) {
                    e.printStackTrace();
                } catch (XmlPullParserException e) {
                    e.printStackTrace();
                } }
            return isDownload;
        }
        @Override
        public void run() { updateUI(download()); }
    });
    worker1.start();
}
```

- btnSalvareCSVonClick(){} ->functie pe onClick care salveaza toate datele din lista din Inventar intr-un fisier de tip CSV , in memoria telefonului.



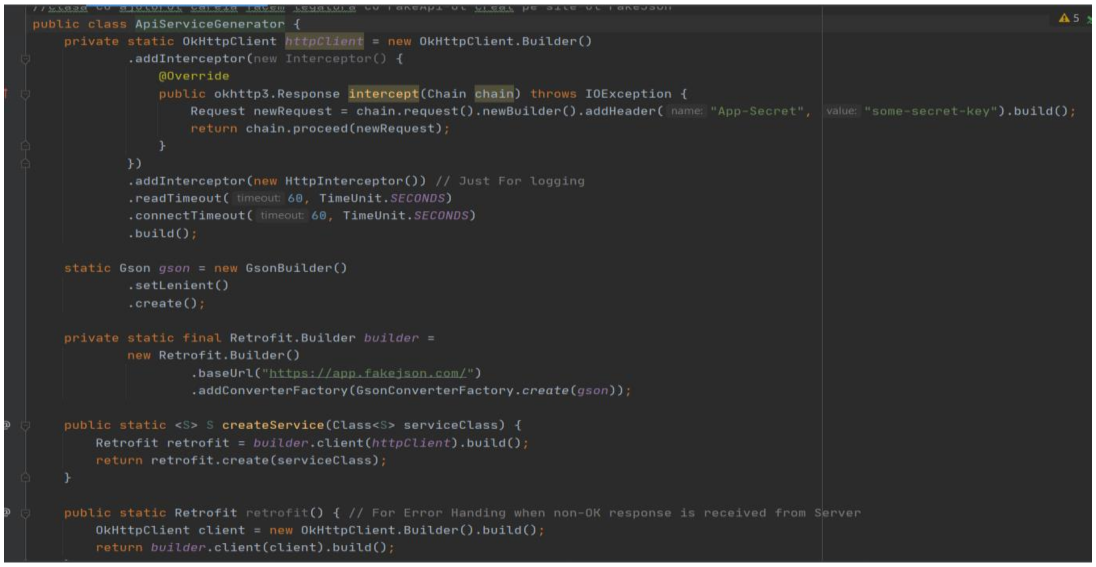
```
private void btnSalvareCSVonClick() {
    btn_salvareCSV.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            try {

                String baseDir = android.os.Environment.getExternalStoragePublicDirectory( type: "").getAbsolutePath();
                String filePath = baseDir + File.separator + file_name;
                f = new File(filePath);
                CSVWriter writer;

                if (f.exists() && !f.isDirectory()) {
                    mFileWriter = new FileWriter(filePath,  append: false);
                    writer = new CSVWriter(mFileWriter);
                } else {
                    writer = new CSVWriter(new FileWriter(filePath));
                }
                String[] s = {"Denumire", "Pret", "Cod de bare", "Cantitate"};
                writer.writeNext(s);
                for (ObjectInventar object : objectInventars) {
                    String[] item = {object.getDenumire(), object.getPret().toString(), object.getCodbare(), object.getCantitate().toString()};
                    writer.writeNext(item);
                }
                writer.close();
                if (f.length() > 0) {
                    Toast.makeText( context: Inventar.this,  text: "Salvarea datelor in CSV a avut succes", Toast.LENGTH_SHORT).show();
                } else {
                    Toast.makeText( context: Inventar.this,  text: "Datele nu au putut fi salvate in CSV", Toast.LENGTH_SHORT).show();
                }
            } catch (Exception e) {
                e.printStackTrace();
```

- btnTrimitereOnClick(){} -> functie pe onClick care verifica intai existenta fisierului de tip CSV, apoi daca acesta exista si contine date , afiseaza un popup cu variante prin care se poate trimite fisierul.



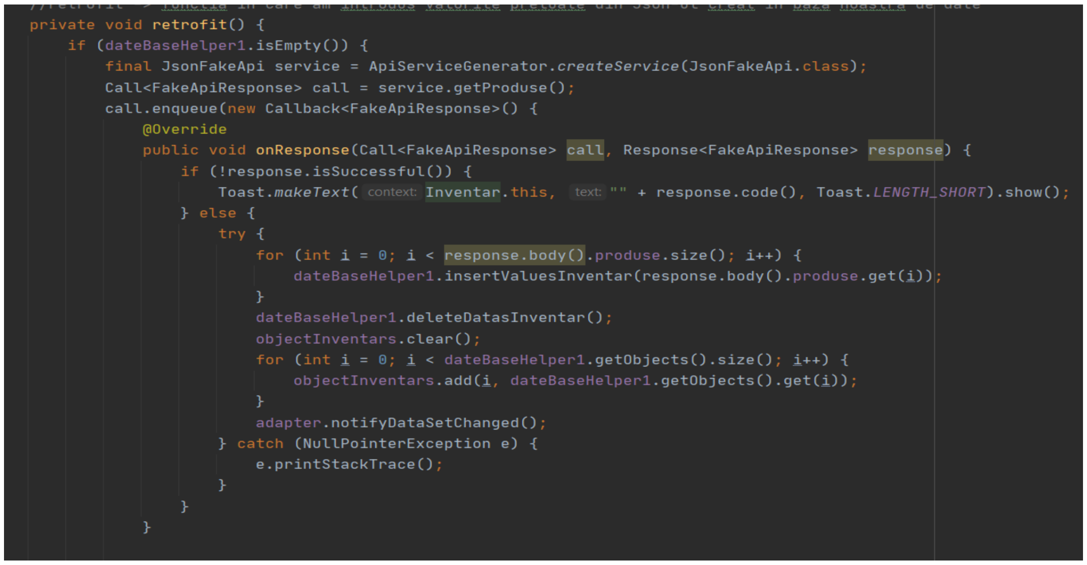
```
//btnTrimitereOnClick -> functie in care se verifica exista si marimea fisierul CSV , iar in caz ca acesta exista si contine e
private void btnTrimitereOnClick() {
    btn_trimitereCSV.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            try {
                if (f.exists() && f.length() > 0) {
                    Context context = getApplicationContext();
                    Uri path = FileProvider.getUriForFile(context,  authority: "com.example.cursbnr.fileprovider", f);
                    Intent sendIntent = new Intent(Intent.ACTION_SEND);
                    sendIntent.setType("text/csv");
                    sendIntent.putExtra(Intent.EXTRA_SUBJECT,  value: "Produse inventar");
                    sendIntent.addFlags(Intent.FLAG_GRANT_READ_URI_PERMISSION);
                    sendIntent.putExtra(Intent.EXTRA_STREAM, path);
                    startActivity(Intent.createChooser(sendIntent,  title: "Send E-mail"));
                }
            } catch (Exception e) {
                Toast.makeText( context: Inventar.this,  text: "Nu exista niciun fisier CSV salvat.", Toast.LENGTH_SHORT).show();
                e.printStackTrace();
            }
        }
    });
}
```

- ApiServiceGenerator() -> clasa folosita pentru generarea unui retrofit catre URL-ul: ” https://app.fakejson.com/q/lxLELIAa?token=Hs8D_qCghmZCie47XKkg9w ” ,care contine fisierul Json care cuprinde datele necesare popularii listei din Inventar.

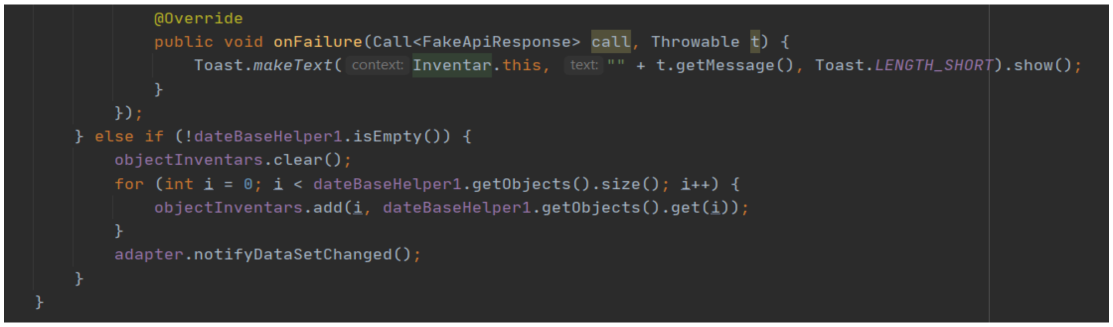


```java
public class ApiServiceGenerator {
    private static OkHttpClient httpClient = new OkHttpClient.Builder()
            .addInterceptor(new Interceptor() {
                @Override
                public okhttp3.Response intercept(Chain chain) throws IOException {
                    Request newRequest = chain.request().newBuilder().addHeader( name: "App-Secret",  value: "some-secret-key").build();
                    return chain.proceed(newRequest);
                }
            })
            .addInterceptor(new HttpInterceptor()) // Just For logging
            .readTimeout( timeout: 60, TimeUnit.SECONDS)
            .connectTimeout( timeout: 60, TimeUnit.SECONDS)
            .build();

    static Gson gson = new GsonBuilder()
            .setLenient()
            .create();

    private static final Retrofit.Builder builder =
            new Retrofit.Builder()
                    .baseUrl("https://app.fakejson.com/")
                    .addConverterFactory(GsonConverterFactory.create(gson));

    public static <S> S createService(Class<S> serviceClass) {
        Retrofit retrofit = builder.client(httpClient).build();
        return retrofit.create(serviceClass);
    }

    public static Retrofit retrofit() { // For Error Handing when non-OK response is received from Server
        OkHttpClient client = new OkHttpClient.Builder().build();
        return builder.client(client).build();
```

- retrofit() -> functie care verifica intai daca baza de date contine date din fisierul Json, daca nu, face apel catre acesta cu ajutorul clasei “ApiServiceGenerator”,clasei de tip obiect “FakeApiResponse” si a interfetei “JsonFakeApi”(care contine comanda de GET din fisier), iar daca apelul este realizat cu succes , raspunsul acestuia este inserat in baza de date de unde va fi adaugat in lista afisata in activitatea Inventar.

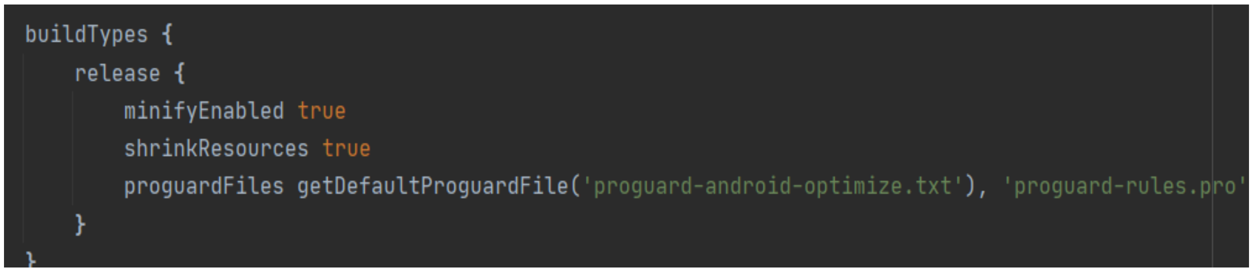


```java
private void retrofit() {
    if (dateBaseHelper1.isEmpty()) {
        final JsonFakeApi service = ApiServiceGenerator.createService(JsonFakeApi.class);
        Call<FakeApiResponse> call = service.getProduse();
        call.enqueue(new Callback<FakeApiResponse>() {
            @Override
            public void onResponse(Call<FakeApiResponse> call, Response<FakeApiResponse> response) {
                if (!response.isSuccessful()) {
                    Toast.makeText( context: Inventar.this,  text: "" + response.code(), Toast.LENGTH_SHORT).show();
                } else {
                    try {
                        for (int i = 0; i < response.body().produse.size(); i++) {
                            dateBaseHelper1.insertValuesInventar(response.body().produse.get(i));
                        }
                        dateBaseHelper1.deleteDatasInventar();
                        objectInventars.clear();
                        for (int i = 0; i < dateBaseHelper1.getObjects().size(); i++) {
                            objectInventars.add(i, dateBaseHelper1.getObjects().get(i));
                        }
                        adapter.notifyDataSetChanged();
                    } catch (NullPointerException e) {
                        e.printStackTrace();
                    }
                }
            }
```

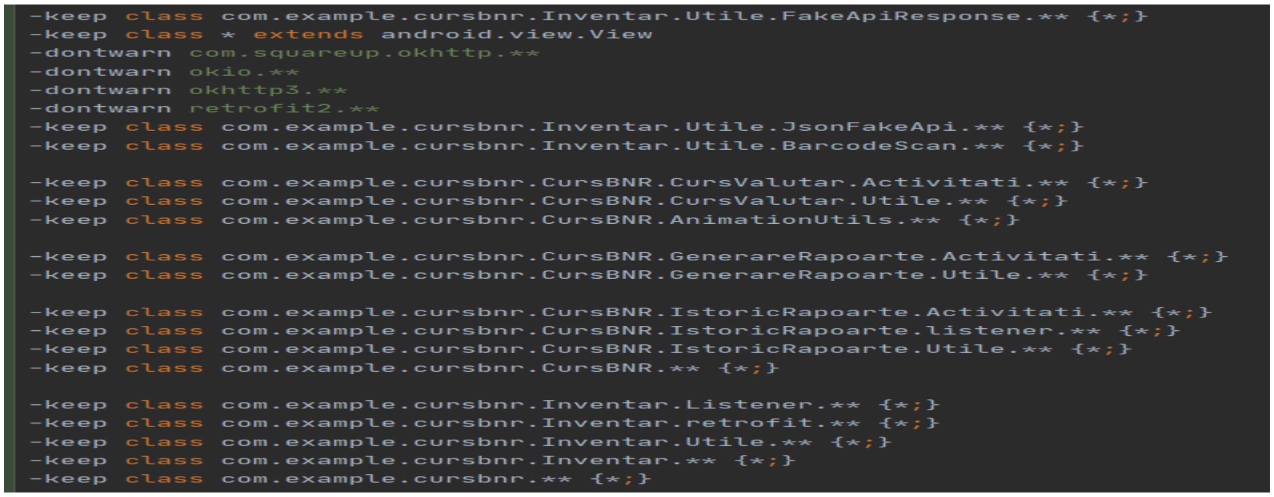


```java
                @Override
                public void onFailure(Call<FakeApiResponse> call, Throwable t) {
                    Toast.makeText( context: Inventar.this,  text: "" + t.getMessage(), Toast.LENGTH_SHORT).show();
                }
            });
        } else if (!dateBaseHelper1.isEmpty()) {
            objectInventars.clear();
            for (int i = 0; i < dateBaseHelper1.getObjects().size(); i++) {
                objectInventars.add(i, dateBaseHelper1.getObjects().get(i));
            }
            adapter.notifyDataSetChanged();
        }
    }
```

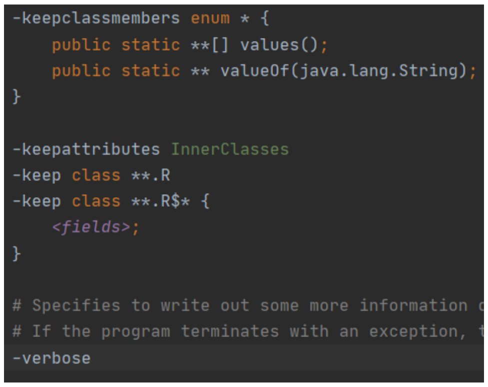
- Am activat Proguard pentru aplicatie pentru obfuscarea codului si reducerea memoriei totale a APK-ului.



```groovy
buildTypes {
    release {
        minifyEnabled true
        shrinkResources true
        proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'
    }
}
```



```
-keep class com.example.cursbnr.Inventar.Utile.FakeApiResponse.** {*;}
-keep class * extends android.view.View
-dontwarn com.squareup.okhttp.**
-dontwarn okio.**
-dontwarn okhttp3.**
-dontwarn retrofit2.**
-keep class com.example.cursbnr.Inventar.Utile.JsonFakeApi.** {*;}
-keep class com.example.cursbnr.Inventar.Utile.BarcodeScan.** {*;}

-keep class com.example.cursbnr.CursBNR.CursValutar.Activitati.** {*;}
-keep class com.example.cursbnr.CursBNR.CursValutar.Utile.** {*;}
-keep class com.example.cursbnr.CursBNR.AnimationUtils.** {*;}

-keep class com.example.cursbnr.CursBNR.GenerareRapoarte.Activitati.** {*;}
-keep class com.example.cursbnr.CursBNR.GenerareRapoarte.Utile.** {*;}

-keep class com.example.cursbnr.CursBNR.IstoricRapoarte.Activitati.** {*;}
-keep class com.example.cursbnr.CursBNR.IstoricRapoarte.listener.** {*;}
-keep class com.example.cursbnr.CursBNR.IstoricRapoarte.Utile.** {*;}
-keep class com.example.cursbnr.CursBNR.** {*;}

-keep class com.example.cursbnr.Inventar.Listener.** {*;}
-keep class com.example.cursbnr.Inventar.retrofit.** {*;}
-keep class com.example.cursbnr.Inventar.Utile.** {*;}
-keep class com.example.cursbnr.Inventar.** {*;}
-keep class com.example.cursbnr.** {*;}
```



```
-keepclassmembers enum * {
    public static **[] values();
    public static ** valueOf(java.lang.String);
}

-keepattributes InnerClasses
-keep class **.R
-keep class **.R$* {
    <fields>;
}

# Specifies to write out some more information d
# If the program terminates with an exception, t
-verbose
```