

علیرضا سعیدنیا - شماره دانشجویی ۴۰۰۱۰۸۳۳

پروژه شماره ۴ - تبدیل یک سی اف جی به فرم نرمال چامسکی

## توضیح کامل بخش UI :

کلاس main selection در واقع کمک به پیاده سازی تابع لیبل ها و فریم ها با استفاده از JFrame میکند . در متد مین یک بار کلاس مین سلکشن را نیو میکنیم و وارد کانستراکتور آن میشویم

```
public static void main(String[] args) {  
    MainSelection ms = new MainSelection();  
}
```

```
public MainSelection() {  
  
    // tarahi ui  
    setButtons();  
    setDefaultCloseOperation(EXIT_ON_CLOSE);  
}
```

به متد ست باتنز میرویم و هرچقدر لیبل و فریم داشتیم را همانجا اضافه میکنیم

در عکس پایین همینکار را کردم. باتن و لیبل و فریمها اینجا اضافه شده اند.

```
jButton.setText("START CFG TO CNF CONVERTOR,CLICK HERE");  
jButton.setAlignmentY(-500);  
jButton.setAlignmentX(-500);  
jButton.setSize( width: 1000, height: 200);  
JLabel label = new JLabel();  
ImageIcon imageIcon = new ImageIcon( filename: "C:\\Users\\-\\Desktop\\WholeJava\\TabdileCFGBE CNF\\Resources\\CFG.gif");  
  
jButton.addActionListener(e -> {  
    frame.setVisible(false);  
});  
jButton.setBackground(Color.PINK);  
label.setIcon(imageIcon);  
label.setText(" Ba salam, baraye estefade az epsilon az 'e' estefade konid mamnoonam");  
label.setHorizontalTextPosition(JLabel.CENTER);  
label.setVerticalTextPosition(JLabel.TOP);  
frame.add(jButton);  
frame.add(label);  
frame.setResizable(false);  
label.setFont(new Font( name: "MV Boli", Font.PLAIN, size: 29));  
label.setForeground(Color.RED);  
Border border = BorderFactory.createLineBorder(Color.orange);  
frame.setVisible(true);  
frame.setSize( width: 1000, height: 1000);  
frame.setTitle("Tabdile CFG BE CNF");  
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
frame.getContentPane().setBackground(Color.WHITE);  
label.setBorder(border);  
jButton.addActionListener( k: this);
```

در آخر هم وقتی کارمان با لیبل ها و فریم ها تمام شد دکمه ضربدر را میزنیم پس با زدن این دکمه برنامه خودش به پایان میرسد.

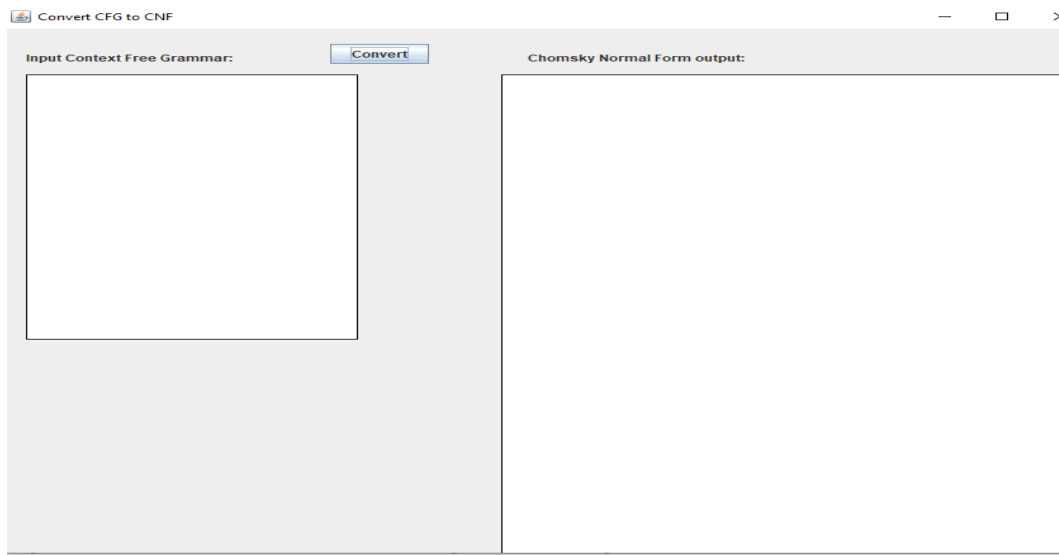
در کلاس کنسول به کمک پایپ اینپوت استریم و تکست فیلدی که در صفحه دوم داریم ، میاییم و خط به خط نوشته ای که داریم را با استفاده از یک ترد اضافه تر از ترد مین میخوانیم و به کمک بافرد ریدر به جای آنکه تعداد خط ها را مشخص کنیم با فور میاییم با هر اینتر از کاربر یک ورودی میگیریم.

```
public void run() {
    String line = null;

    while(!exit){
        try {
            while ((line = reader.readLine()) != null) {
                displayPane.append(line + "\n");
                displayPane.setCaretPosition(displayPane.getDocument().getLength());
            }

            exit = true;
        } catch (IOException ioe) {
            JOptionPane.showMessageDialog( parentComponent: null,
                message: "Error redirecting output : " + ioe.getMessage());
        }
    }
}
```

در کلاس second label بخش لیبل های صفحه دوم کارش انجام شده است



در کانستراکتور این تابع باتن کانونر ت عکس بالا اینپوت لیبل و اوتپوت لیبل ست شده اند ( با استفاده از متد ستر )

```
public SecondLabel(){
    super( title: "Convert CFG to CNF");
    setSize( width: 875, height: 725);
    setVisible(true);
    setLayout(SpringLayout);

    // program will launch at center
    setLocationRelativeTo(null);

    cnf = new CNF();

    setDefaultCloseOperation(DISPOSE_ON_CLOSE);
    setInputLabel();
    setInputTextArea();
    setCNFOutputTextArea();
    setSubmitButton();
    setOutputLabel();
}
```

```
public void setSubmitButton() {
    submit = new JButton( text: "Convert");
    setVisible(true);
    submit.addActionListener( e: this);
    add(submit);

    springLayout.putConstraint(SpringLayout.WEST, submit, pad: 240, SpringLayout.WEST, inputLabel);
    springLayout.putConstraint(SpringLayout.NORTH, submit, pad: 20, SpringLayout.NORTH, c2: this);
}
```

در متد ست سامیشن باتن میاییم و یک باتن کانونر ت را میسازیم همانطور که در عکس بالا مشخص است. بعد از آن اکشن لیسنر که در واقع باتن استاتیک هست را به عنوان اکشن لیسنر این باتن اد و اضافه میکنیم. سپس بقیه کارها از جمله سائز و موقعیتش را تنظیم میکنیم.

```

public void setInputTextArea() {
    inputTextArea = new JTextArea( rows: 15, columns: 20);
    inputTextArea.setFont(inputTextArea.getFont().deriveFont(17f));
    inputTextArea.setBorder(thinBorder);
    inputTextArea.setVisible(true);
    add(inputTextArea);

    springLayout.putConstraint(SpringLayout.WEST, inputTextArea, pad: 20, SpringLayout.WEST, c2: this);
    springLayout.putConstraint(SpringLayout.NORTH, inputTextArea, pad: 30, SpringLayout.NORTH, inputLabel);
}

public void setCNFOutputTextArea() {
    outputCNFTextArea = new JTextArea( rows: 40, columns: 40);
    outputCNFTextArea.setBorder(thinBorder);
    outputCNFTextArea.setEditable(false);
    outputCNFTextArea.setVisible(true);
    add(outputCNFTextArea);
    cnf.setOutputTextArea(outputCNFTextArea);

    springLayout.putConstraint(SpringLayout.WEST, outputCNFTextArea, pad: 375, SpringLayout.WEST, inputTextArea);
    springLayout.putConstraint(SpringLayout.NORTH, outputCNFTextArea, pad: 60, SpringLayout.NORTH, c2: this);
}

```

**در متد اوتپوت تکست اریا و اینپوت تکست اریا هم میاییم و همینکار را انجام میدهیم و محل قرار گیری متن و متن خروجی را به صورت یک تکست فیلد مشخص میکنیم.**

```

public void setOutputLabel() {
    outputLabel = new JLabel();
    outputLabel.setText("Chomsky Normal Form output: ");
    add(outputLabel);

    springLayout.putConstraint(SpringLayout.WEST, outputLabel, pad: 400, SpringLayout.EAST, c2: this);
    springLayout.putConstraint(SpringLayout.NORTH, outputLabel, pad: 30, SpringLayout.NORTH, c2: this);
}

public void setInputLabel() {
    inputLabel = new JLabel();
    inputLabel.setText("Input Context Free Grammar:");
    add(inputLabel);

    springLayout.putConstraint(SpringLayout.WEST, inputLabel, pad: 20, SpringLayout.WEST, c2: this);
    springLayout.putConstraint(SpringLayout.NORTH, inputLabel, pad: 30, SpringLayout.NORTH, c2: this);
}

```

**یک تکست فیلد میزنیم که همان متن Chomsky normal form output و input context free grammer را در بر بگیرد.**

## توضیحات کامل الگوریتم :

این الگوریتم طبق الگوریتم خود چامسکی انجام شده است و مرحله به مرحله برای انجام هر کار متد زده شده است!

- درج متغیر شروع جدید
- حذف اپسیلون
- حذف تک متغیر
- حذف بیش از طول ۳ تا متغیر در rhs
- جایگذاری ترمینال ها با متغیرها

وارد کلاس cnf میشویم. الگوریتم اصلی ما در کلاس cnf انجام میشود.

وارد کاسنتراکتور آن میشویم. ابتدا در تکست اریای ما هیچی نیست پس باید هرچیزی که قرار است در بخش ران برنامه نمایش داده شود را وارد outputtextarea کنیم پس از دستور Console.redirectOutput استفاده میکنیم. برای هر فرم نرمال چامسکی میتوانیم از متغیر شروع جدید استفاده کنیم پس با متد مربوطه اینکار را انجام میدهیم. هر استرینگ را خط به خط به یک مپ تبدیل میکنیم که شامل کلید متغیر و ولیوی رایت هند ساید است.

```
private Map<String, List<String>> mapVariableProduction = new LinkedHashMap<>();
```

چون رایت هند ساید برای ما اهمیت دارد که کجا میرود پس از یک لیست استفاده میکنیم.

```

public void convertCFGtoCNF() {

    outputTextArea.setText("");

    Console.redirectOutput(outputTextArea);
    insertNewStartSymbol();
    convertStringtoMap();
    eliminateEpsilon();
    removeDuplicateKeyValue();
    eliminateSingleVariable();
    onlyTwoTerminalandOneVariable();
    eliminateThreeTerminal();

    Console.redirectOutput(displayPane: null);

}

```

در متد های بعدی میاییم و طبق الگوی چامسکی اپسیلون را حذف میکنیم ، تک متغیر را حذف میکنیم ، بیش از ۳ تایی قانون رایت هند شاید را حذف میکنیم و متغیر جدید تولید میکنیم.

```

private void eliminateSingleVariable() {

    System.out.println("Remove Single Variable in Every Production ... ");

    for (int i = 0; i < lineCount; i++) {
        removeSingleVariable();
    }

    printMap();

}

```


در متد eliminatesingle variable میاییم و تعداد خطهایی که کاربر وارد کرده است را

می‌شمریم و در هر خط دنبال تک متغیر می‌گردیم و وارد متد `removeSingleVariable` می‌شویم.

```
private void removeSingleVariable() {  
    Iterator itr4 = mapVariableProduction.entrySet().iterator();  
    String key = null;  
  
    while (itr4.hasNext()) {  
        Map.Entry entry = (Map.Entry) itr4.next();  
        Set set = mapVariableProduction.keySet();  
        ArrayList<String> keySet = new ArrayList<>(set);  
        ArrayList<String> productionList = (ArrayList<String>) entry.getValue();  
  
        for (int i = 0; i < productionList.size(); i++) {  
            String temp = productionList.get(i);  
  
            for (int j = 0; j < temp.length(); j++) {  
                for (int k = 0; k < keySet.size(); k++) {  
                    if (keySet.get(k).equals(temp)) {  
                        key = entry.getKey().toString();  
                        List<String> productionValue = mapVariableProduction.get(temp);  
                        productionList.remove(temp);  
  
                        for (int l = 0; l < productionValue.size(); l++) {  
                            mapVariableProduction.get(key).add(productionValue.get(l));  
                        }  
                    }  
                }  
            }  
        }  
    }  
}
```

در این متد باید یک پیمایشگر تولید کنیم که می‌تواند در صفحه قبل گفتیم تولید کردیم که سمت چپ و راستش متغیر و رایت هند سایید بود را پیمایش کند و دنبال یک تک متغیر بگردد پس با دستور `entryset.iterator` این کار را انجام می‌دهیم. متد `entryset` یک ست را ریترن می‌کند که با هر تغییری که بعد از ایتريت کردن بدهیم این مجموعه یا ست تغییر می‌کند یعنی یک `collection view` را ریترن می‌کند که باعث می‌شود روی مپ ایتريت کنیم. تا زمانی که ایتريتور مقدار بعدیش نال نیست یعنی هنوز چیزی برای پیمایش دارد می‌آید و یک اری لیست از کلید ها که متغیرهای ما است تولید می‌کند و همچنین با دستور `entry.getValue` می‌آید rhs ها را می‌گیرد. و به اندازه طول اری لیست رایت هند سایید می‌آید و فور می‌زند.

```
Map.Entry entry = (Map.Entry) itr4.next();  
Set set = mapVariableProduction.keySet();  
ArrayList<String> keySet = new ArrayList<>(set);  
ArrayList<String> productionList = (ArrayList<String>) entry.getValue();
```



```
for (int i = 0; i < productionList.size(); i++) {  
    String temp = productionList.get(i);
```

هر چیزی که داخل این لیست هست را میریزد داخل استرینگ تمپ (ممکن است | علامت یا باشد) حال دوباره روی طول تمپ لوپ میزنم چون کلا تمپ تک حرف است پس باید بیاید و ببیند آیا از متغیرهای سمت چپ که داخل اری لیست کی ست هستند

چیزی هست که در سمت راست باشد؟ اگر بود آن را از رایت هند شاید ها حذف کن !

```
key = entry.getKey().toString();  
List<String> productionValue = mapVariableProduction.get(temp);  
productionList.remove(temp);
```

که در عکس بالا میبینیم از رایت هند شاید ها حذف کردیم.

```
private void eliminateEpsilon() {  
  
    System.out.println("Remove Epsilon....");  
  
    for (int i = 0; i < lineCount; i++) {  
        removeEpsilon();  
    }  
  
    printMap();  
}
```



وارد متد حذف اپسیلون میشویم ، هر خطی که کاربر وارد کرده را به تعداد آن فور اجرا میشود.

```
private void removeEpsilon() {  
  
    Iterator itr = mapVariableProduction.entrySet().iterator();  
    Iterator itr2 = mapVariableProduction.entrySet().iterator();  
  
    while (itr.hasNext()) {  
        Map.Entry entry = (Map.Entry) itr.next();  
        ArrayList<String> productionRow = (ArrayList<String>) entry.getValue();  
  
        if (productionRow.contains("e")) {  
            if (productionRow.size() > 1) {  
                productionRow.remove(0, "e");  
                epsilonFound = entry.getKey().toString();  
            } else {  
  
                // remove if less than 1  
                epsilonFound = entry.getKey().toString();  
                mapVariableProduction.remove(epsilonFound);  
            }  
        }  
    }  
}
```

در اینجا به دوتا ایتريتور نیاز داریم که اولی برای پیدا کردن اپسیلون در رایت هند سایید است که اگر طول رایت هند سایید فقط شامل اپسیلون بود ان اپسیلون حذف میشود در غیر این صورت هم اپسیلون را حذف میکند و میرود دنبال آنکه متغیری که اپسیلون را تولید کرده است ببیند در سمت چپ قوانین وجود دارد یا خیر ( با ایتريتور شماره ۲ )

```

while (itr2.hasNext()) {

    Map.Entry entry = (Map.Entry) itr2.next();
    ArrayList<String> productionList = (ArrayList<String>) entry.getValue();

    for (int i = 0; i < productionList.size(); i++) {
        String temp = productionList.get(i);

        for (int j = 0; j < temp.length(); j++) {
            if (epselonFound.equals(Character.toString(productionList.get(i).charAt(j)))) {

                if (temp.length() == 2) {

                    // remove specific character in string
                    temp = temp.replace(epselonFound, replacement: "");

                    if (!mapVariableProduction.get(entry.getKey().toString()).contains(temp)) {
                        mapVariableProduction.get(entry.getKey().toString()).add(temp);
                    }

                } else if (temp.length() == 3) {

                    String deletedTemp = new StringBuilder(temp).deleteCharAt(j).toString();

                    if (!mapVariableProduction.get(entry.getKey().toString()).contains(deletedTemp)) {
                        mapVariableProduction.get(entry.getKey().toString()).add(deletedTemp);
                    }

                } else if (temp.length() == 4) {

                    String deletedTemp = new StringBuilder(temp).deleteCharAt(j).toString();

                    if (!mapVariableProduction.get(entry.getKey().toString()).contains(deletedTemp)) {
                        mapVariableProduction.get(entry.getKey().toString()).add(deletedTemp);
                    }

                } else {

                    if (!mapVariableProduction.get(entry.getKey().toString()).contains("e")) {
                        mapVariableProduction.get(entry.getKey().toString()).add("e");
                    }

                }

            }

        }

    }

}

```

حال تا زمانی که ایتريتور دوم مقدار بعدی دارد و نال نیست میاید و ایتريت می کند و سپس رایت هند سایید را دونه دونه با یک حلقه فور میگیرد و داخل استرینگ تمپ میریزد ( چون رایت هند سایید شامل استرینگ ها بود )، حال دنبال کرکتری میگردد که سمت چپ بوده و خودش به عنوان یک متغیر اپسیلون را تولید کرده است. اگر این استرینگ تمپ دو حرفی بود میاید و با "" جای گذاری اش میکند. در غیر این صورت تعداد آن کرکتری که خودش اپسیلون تولید کرده بیشتر بوده است و باید در هر مرحله تعداد بیشتری از آنها را حذف

کنیم با اپسیلون پس به یک استرینگ بیلدر نیاز داریم که هر دفعه تغییر میکند پس طول بزرگتر از ۲ یعنی ۳ و ۴ این اتفاق میفتد و باید به عنوان قانون جدید به mapVariableProduction اضافه کنیم! مانند کاری که در الگوریتم نرمال انجام میدادیم مثلا دیده بودیم BBB و بی میرفت به اپسیلون ، حالا باید داشته باشیم BB-B BBB یعنی بدون تغییر ، یک تغییر و دو تغییر! این برای زمانی است که طول temp برابر ۳ باشد ، میتواند بیشتر هم باشد.

```
private void convertStringtoMap() {  
  
    String[] splitedEnterInput = splitEnter(input);  
  
    for (int i = 0; i < splitedEnterInput.length; i++) {  
  
        String[] tempString = splitedEnterInput[i].split(regex: "->|\\\\|");  
        String variable = tempString[0].trim();  
  
        String[] production = Arrays.copyOfRange(tempString, from: 1, tempString.length);  
        List<String> productionList = new ArrayList<>();  
  
        // trim the empty space  
        for (int k = 0; k < production.length; k++) {  
            production[k] = production[k].trim();  
        }  
  
        // import array into ArrayList  
        for (int j = 0; j < production.length; j++) {  
            productionList.add(production[j]);  
        }  
  
        //insert element into map  
        mapVariableProduction.put(variable, productionList);  
    }  
}
```

در متد convertStringToMap میاییم و استرینگ سمت راست را به مپ تبدیل میکنیم پس نباید علامت یا بین آنها باشد بنابراین با دستور

```
String[] tempString = splitedEnterInput[i].split("->|\\\\|");  
String variable = tempString[0].trim();
```

میاید و هر جا که یا وجود داشت را میبریم. دستور `copyOfRange` به ما کمک میکند تا

از یک جای مشخصی چیزی که میخواهیم را ببریم

```
String[] production = Arrays.copyOfRange(tempString, from: 1, tempString.length);  
List<String> productionList = new ArrayList<>();
```

پس از قبل یا را میبریم

```
for (int k = 0; k < production.length; k++) {  
    production[k] = production[k].trim();  
}  
  
// import array into ArrayList  
for (int j = 0; j < production.length; j++) {  
    productionList.add(production[j]);  
}
```

اینکار را به طول رایت هند سایدمان انجام میدهم

```
//insert element into map  
mapVariableProduction.put(variable, productionList);
```

سپس داخل ولیوی مپمان میریزیم که الان در واقع یا ها از بین رفت.

```
private void insertNewStartSymbol() {  
  
    String newStart = "S0";  
    ArrayList<String> newProduction = new ArrayList<>();  
    newProduction.add("S");  
  
    mapVariableProduction.put(newStart, newProduction);  
}
```

در متد `insertNewStartSymbol` میاییم و متغیر `S0` را جایگذاری میکنیم به عنوان متغیر شروع جدید.

```

private void removeDuplicateKeyValue() {

    System.out.println("Remove Duplicate Key Value ... ");

    Iterator itr3 = mapVariableProduction.entrySet().iterator();

    while (itr3.hasNext()) {
        Map.Entry entry = (Map.Entry) itr3.next();
        ArrayList<String> productionRow = (ArrayList<String>) entry.getValue();

        for (int i = 0; i < productionRow.size(); i++) {
            if (productionRow.get(i).contains(entry.getKey().toString())) {
                productionRow.remove(entry.getKey().toString());
            }
        }
    }

    printMap();
}

```

در متد `removeDuplicateKeyValue` می‌ایم و لیوی تکراری اگر در رایت هند ساید دیدیم را حذف می‌کنیم. باز هم از ایت‌یتور استفاده می‌کنیم مانند متدهای قبلی و یک فور می‌زنیم و اگر ایت‌یتور و لیویی که می‌گیرد با متغیر داخل رایت هند ساید یکی بود می‌ایم و ان را حذف می‌کنیم.

**متد** `checkDuplicateInProductionList`

می‌ایم و همین اتفاق را چک می‌کنیم که به صورت بولین ترو یا فالس برای ما برمی‌گرداند.

```

private Boolean checkDuplicateInProductionList(Map<String, List<String>> map, String key) {

    Boolean notFound = true;

    Iterator itr = map.entrySet().iterator();
    outerloop:

    while (itr.hasNext()) {

        Map.Entry entry = (Map.Entry) itr.next();
        ArrayList<String> productionList = (ArrayList<String>) entry.getValue();

        for (int i = 0; i < productionList.size(); i++) {
            if (productionList.size() < 2) {

                if (productionList.get(i).equals(key)) {
                    notFound = false;
                    break outerloop;
                } else {
                    notFound = true;
                }
            }
        }
    }

    return notFound;
}

```

```

private void printMap() {

    Iterator it = mapVariableProduction.entrySet().iterator();
    while (it.hasNext()) {
        Map.Entry pair = (Map.Entry) it.next();
        System.out.println(pair.getKey() + " -> " + pair.getValue());
    }

    System.out.println(" ");
}

```

**در متد پرینت مپ میاییم و می که ساختیم را پرینت میکنیم که باز هم با ایتريتور این کار انجام میشود.**

onlyTwoTerminalandOneVariable در متد

به دنبال این هستیم سمت راست را یک متغیره و یک ترمیناله کنیم.

یک ایتريتور ديگر ميسازيم

```
Iterator itr5 = mapVariableProduction.entrySet().iterator();
String key = null;
int asciiBegin = 71; //G

Map<String, List<String>> tempList = new LinkedHashMap<>();
```

قرار است متغیرهای زیاد حذف شوند و جایشگان متغیرهای جدید جایگذاری شوند

پس از یک حرف انگلیسی جدید باید استفاده بکنیم که میتوان از حرف G شروع کرد و ادامه داد .

```
for (int i = 0; i < productionList.size(); i++) {
    String temp = productionList.get(i);
```

به اندازه سايز و اندازه اری لیستی که برای رایت هند ساید ها درست کردیم فور میزنیم و هر مقدار را که دیگر میدانیم شامل یا نیست را داخل تمپ میریزیم.

```
for (int j = 0; j < temp.length(); j++) {

    if (temp.length() == 3) {
```

حال باید ببینیم تمپ چه اندازه ای است ، حداقل اندازه اش میتواند برابر ۳ باشد چون اگر دوتا بود که به هدفمان رسیده بودیم.

```
String newProduction = temp.substring(1, 3); // SA

if (checkDuplicateInProductionList(templist, newProduction) && checkDuplicateInProductionList(mapVariableProduction, newProduction)) {
    found = true;
} else {
    found = false;
}

if (found) {
    ArrayList<String> newVariable = new ArrayList<>();
    newVariable.add(newProduction);
    key = Character.toString((char) asciiBegin);

    templist.put(key, newVariable);
    asciiBegin++;
}
}
```

**در این بخش باید دوتا از حروف را بگیریم و کاری کنیم که asciiBegin به آن برود .**

**این کار را در عکس بالا انجام داده ایم.**

```
else if (temp.length() == 2) { // if only two substring

    for (int k = 0; k < keySet.size(); k++) {

        if (!keySet.get(k).equals(Character.toString(productionList.get(i).charAt(j)))) { // if substring not equals to keySet
            found = false;
        } else {
            found = true;
            break;
        }
    }

    if (!found) {
        String newProduction = Character.toString(productionList.get(i).charAt(j));

        if (checkDuplicateInProductionList(templist, newProduction) && checkDuplicateInProductionList(mapVariableProduction, newProduction)) {

            ArrayList<String> newVariable = new ArrayList<>();
            newVariable.add(newProduction);
            key = Character.toString((char) asciiBegin);

            templist.put(key, newVariable);

            asciiBegin++;
        }
    }
}
}
```

**اگر طول برابر دو باشد هم میاییم اینکار را میکنیم ولی باید مواظب باشیم که اگر اینکار را مشابه طول ۳ انجام دهیم به مشکل میخوریم و سمت راست یک تک متغیر تولید میشود که شرط قبلی ما را نقض میکند ! پس باید فقط یک بخش آن را تبدیل به متغیر کنیم تا جفت متغیر شود. اگر ساب استرینگ دوتایی ما با کی ست ما برابری نکرد مقدار بولین**



فاوندی که تعریف کردیم را فالس میگذاریم ، در غیر این صورت ترو میگذاریم ، حال اگر فالس بود میاییم و متغیر جدید که با `asciiBegin` تولید کردیم را به عنوان متغیر قرار میدهیم.

```
else if (temp.length() == 4) {

    String newProduction1 = temp.substring(0, 2); // SA
    String newProduction2 = temp.substring(2, 4); // SA

    if (checkDuplicateInProductionList(templist, newProduction1) && checkDuplicateInProductionList(mapVariableProduction, newProduction1)) {
        found1 = true;
    } else {
        found1 = false;
    }

    if (checkDuplicateInProductionList(templist, newProduction2) && checkDuplicateInProductionList(mapVariableProduction, newProduction2)) {
        found2 = true;
    } else {
        found2 = false;
    }

    if (found1) {

        ArrayList<String> newVariable = new ArrayList<>();
        newVariable.add(newProduction1);
        key = Character.toString((char) asciiBegin);

        templist.put(key, newVariable);
        asciiBegin++;
    }

    if (found2) {

        ArrayList<String> newVariable = new ArrayList<>();
        newVariable.add(newProduction2);
        key = Character.toString((char) asciiBegin);

        templist.put(key, newVariable);
        asciiBegin++;
    }
}
```

برای اندازه ۴ هم به همین صورت عمل میکنیم ولی دوبار باید این کار انجام شود یعنی یک بخش سه تایی به یک متغیر برود و یک بار اسکی بگین زیاد شود ، بعد از آن طبق کاری که خودمان هم در الگوریتم چامسکی انجام میدادیم ، آن سه تا به سمت یک متغیر دیگر برود.

بعد از آن طبق عکسهای اولی که فرستادم باید به سراغ `removethreeterminals` برویم.

```
private void removeThreeTerminal() {

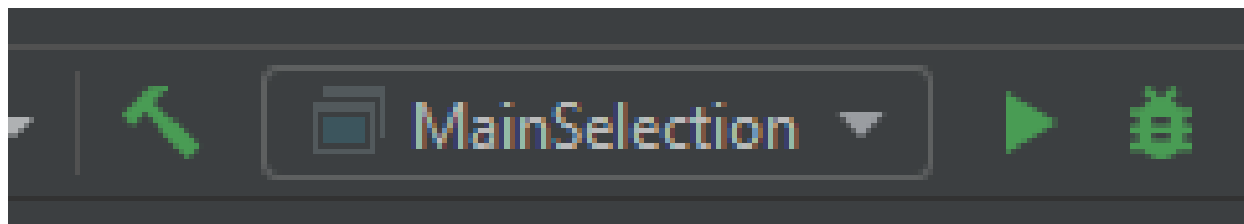
    Iterator itr = mapVariableProduction.entrySet().iterator();
    ArrayList<String> keyList = new ArrayList<>();
    Iterator itr2 = mapVariableProduction.entrySet().iterator();

    // obtain key that use to eliminate two terminal and above
    while (itr.hasNext()) {
        Map.Entry entry = (Map.Entry) itr.next();
        ArrayList<String> productionRow = (ArrayList<String>) entry.getValue();

        if (productionRow.size() < 2) {
            keyList.add(entry.getKey().toString());
        }
    }
}
```

در این متد هم باید بیاییم متغیر هایی که رایت هند سایدشان به سه تا متغیر دیگر یا ترمینال دیگر می‌رود را حذف کنیم.

## نحوه اجرا و ران گرفتن کد :



## در انتلیجی ران را بزنید

[START CFG TO CNF CONVERTOR,CLICK HERE](#)

*Ba salam, baraye estefade az epsilon az 'e' estefade konid mamnoonam*



**این صفحه باز میشود ، روی باتن صورتی  
بزنید ، قبل از آن توجه کنید که برای**

استفاده از اپسیلون از e استفاده  
کنید. این صفحه زیر باز میشود !

Convert CFG to CNF

Input Context Free Grammar:

Chomsky Normal Form output:

Input Context Free Grammar: Conve

هر خطی که می‌خواهید قانون بگذارید را  
بگذارید و اینتر بزنید و بعد از آن دکمه  
کانورت را بزنید. برای استفاده از  
اپسیلون از  $\epsilon$  استفاده کنید.

# Convert

## Chomsky Normal Form output:

Remove Epselon....

$S_0 \rightarrow [S, e]$

$S \rightarrow [asb, SS, ]$

Remove Duplicate Key Value ...

$S_0 \rightarrow [S, e]$

$S \rightarrow [asb, SS, ]$

Remove Single Variable in Every Production ...

$S_0 \rightarrow [e, asb, SS, ]$

$S \rightarrow [asb, SS, ]$

Assign new variable for two non-terminal or one terminal ...

$S_0 \rightarrow [e, asb, SS, ]$

$S \rightarrow [asb, SS, ]$

$G \rightarrow [sb]$

Replace two terminal variable with new variable ...

$S_0 \rightarrow [e, aG, SS, ]$

$S \rightarrow [aG, SS, ]$

$G \rightarrow [sb]$

# فرم چامسکی شده در تکست فیلد راست نمایش داده میشود.

حداقل دو مثال از ورودی و خروجی

برنامه :

مثال ۱:

Input Context Free Grammar:	Convert	Chomsky Normal Form output:
<div>S-&gt;aSb bSa SS e</div>		<div>Remove Epselon.... S0 -&gt; [S, e] S -&gt; [aSb, bSa, SS, ab, ba, ]  Remove Duplicate Key Value ... S0 -&gt; [S, e] S -&gt; [aSb, bSa, SS, ab, ba, ]  Remove Single Variable in Every Production ... S0 -&gt; [e, aSb, bSa, SS, ab, ba, ] S -&gt; [aSb, bSa, SS, ab, ba, ]  Assign new variable for two non-terminal or one terminal ... S0 -&gt; [e, aSb, bSa, SS, ab, ba, ] S -&gt; [aSb, bSa, SS, ab, ba, ] G -&gt; [Sb] H -&gt; [Sa] I -&gt; [a] J -&gt; [b]  Replace two terminal variable with new variable ... S0 -&gt; [e, aG, bH, SS, IJ, ] S -&gt; [aG, bH, SS, IJ, ] G -&gt; [SJ] H -&gt; [SI] I -&gt; [a] J -&gt; [b]</div>

## مثال 2:

Input Context Free Grammar:	Convert	Chomsky Normal Form output:
<pre>S-&gt;AB A-&gt;AabB e B-&gt;b</pre>		<pre>Remove Epsilon.... S0 -&gt; [S] S -&gt; [AB, B] A -&gt; [AabB, abB] B -&gt; [b]  Remove Duplicate Key Value ... S0 -&gt; [S] S -&gt; [AB, B] A -&gt; [AabB, abB] B -&gt; [b]  Remove Single Variable in Every Production ... S0 -&gt; [AB, b] S -&gt; [AB, b] A -&gt; [AabB, abB] B -&gt; [b]  Assign new variable for two non-terminal or one terminal ... S0 -&gt; [AB, b] S -&gt; [AB, b] A -&gt; [AabB, abB] B -&gt; [b] G -&gt; [Aa] H -&gt; [bB]  Replace two terminal variable with new variable ... S0 -&gt; [AB, b] S -&gt; [AB, b] A -&gt; [GH, aH] B -&gt; [b] G -&gt; [Aa] H -&gt; [BB]</pre>



## مثال ۳:

Input Context Free Grammar:	Convert	Chomsky Normal Form output:
<div>S-&gt;bSb A A-&gt;aA ε</div>		<div>Remove Epsilon.... S0 -&gt; [S] S -&gt; [bSb, A, bb] A -&gt; [aA, a] G -&gt; [a]  Remove Duplicate Key Value ... S0 -&gt; [S] S -&gt; [bSb, A, bb] A -&gt; [aA, a] G -&gt; [a]  Remove Single Variable in Every Production ... S0 -&gt; [bSb, bb, aA, a] S -&gt; [bSb, bb, aA, a] A -&gt; [aA, a] G -&gt; [a]  Assign new variable for two non-terminal or one terminal ... S0 -&gt; [bSb, bb, aA, a] S -&gt; [bSb, bb, aA, a] A -&gt; [aA, a] G -&gt; [Sb] H -&gt; [b]  Replace two terminal variable with new variable ... S0 -&gt; [HG, HH, aA, a] S -&gt; [HG, HH, aA, a] A -&gt; [aA, a] G -&gt; [SH] H -&gt; [b]</div>