

پروژه درس کامپایلر دکتر فیضی (فاز دو)

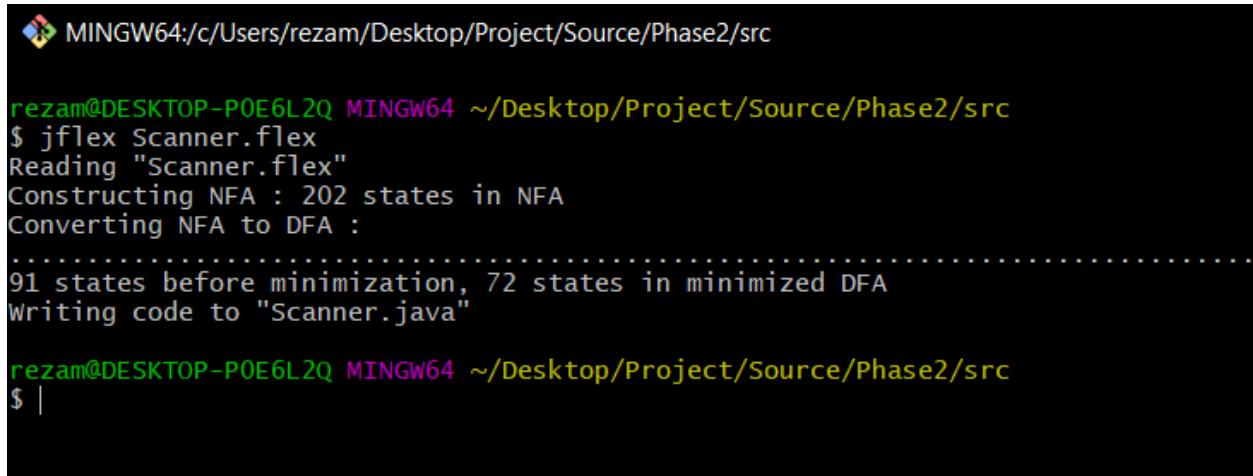
رضا محمدی - یاسان حسن زاد

1 - Build and Test	2
2 - Parser and Scanner	9

1 - Build and Test

فایل Scanner.flex همان فایل فاز یک، با اضافه شدن تغییراتی برای کار کردن با CUP و همچنین نیازمندی های جدید فاز دو است. برای تولید کد جاوای اسکنر با استفاده از این فایل، دستور زیر را اجرا می کنیم:

```
jflex Scanner.flex
```

A screenshot of a Windows command prompt window. The title bar shows the path 'MINGW64:/c/Users/rezam/Desktop/Project/Source/Phase2/src'. The prompt is 'rezam@DESKTOP-P0E6L2Q MINGW64 ~/Desktop/Project/Source/Phase2/src'. The user enters '\$ jflex Scanner.flex'. The output shows 'Reading "Scanner.flex"', 'Constructing NFA : 202 states in NFA', 'Converting NFA to DFA :', a separator line of dots, '91 states before minimization, 72 states in minimized DFA', and 'Writing code to "Scanner.java"'. The prompt returns to '\$ |'.

فایل Parser.cup که توضیحات آن در این گزارش آمده را، با دستور زیر به فایل جاوای پارسر تبدیل می کنیم:

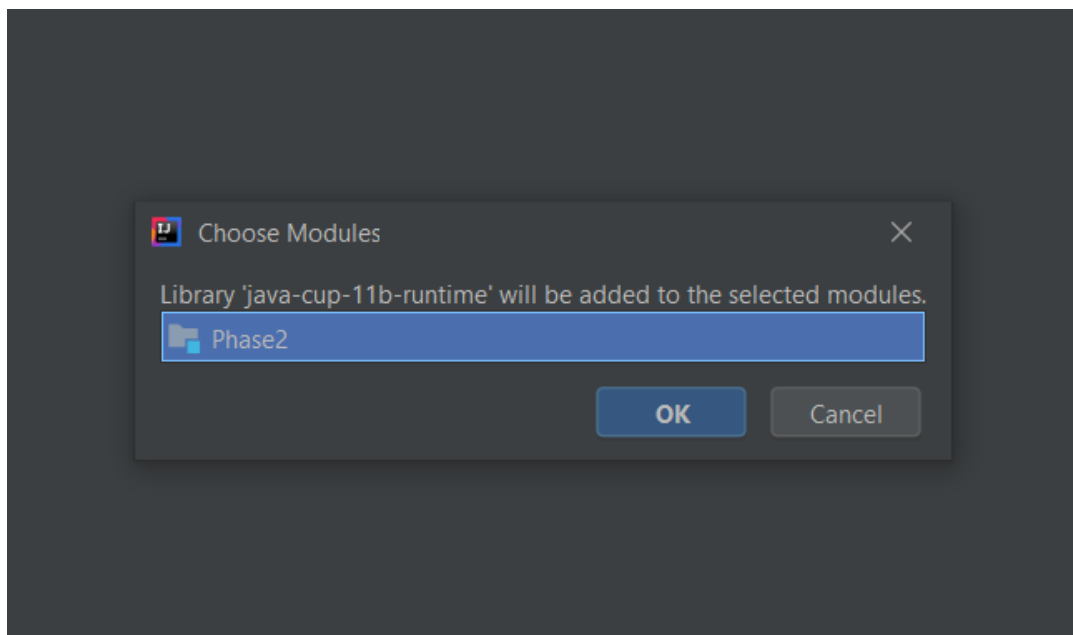
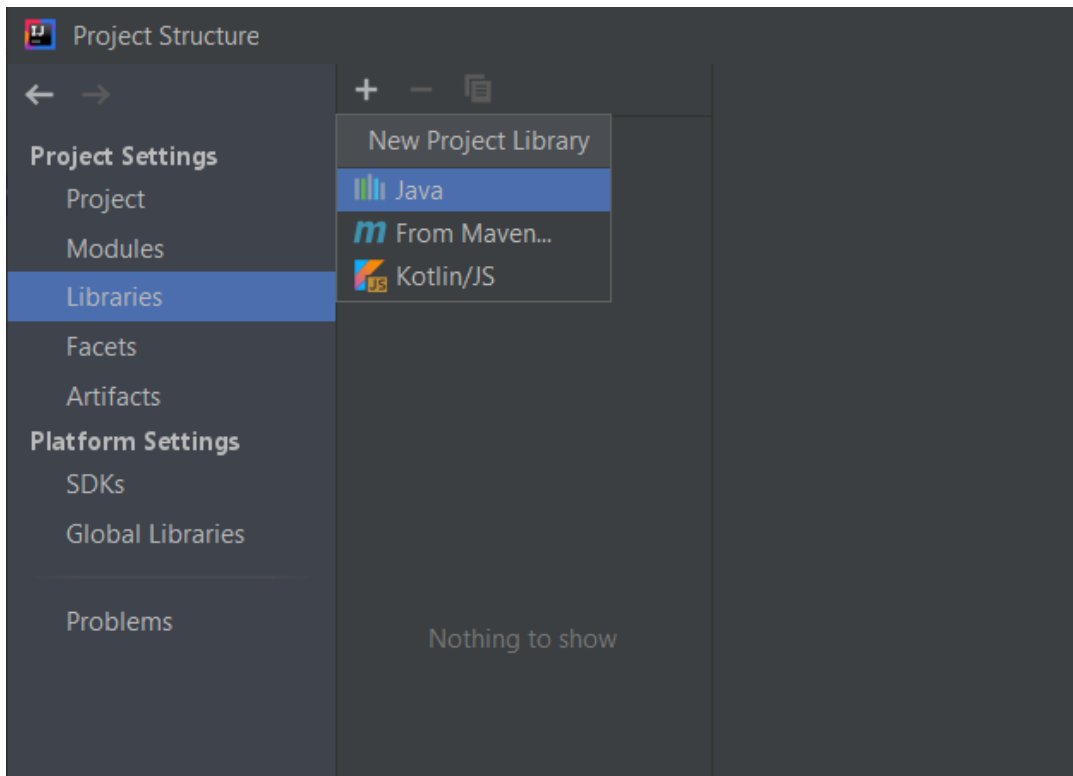
```
java -jar java-cup-11b.jar -parser "Parser" Parser.cup
```

```
MINGW64:/c:/Users/rezam/Desktop/Project/Source/Phase2/src
rezam@DESKTOP-P0E6L2Q MINGW64 ~/Desktop/Project/Source/Phase2/src
$ java -jar java-cup-11b.jar -parser "Parser" Parser.cup
----- CUP v0.11b 20160615 (GIT 4ac7450) Parser Generation Summary -----
 0 errors and 0 warnings
28 terminals, 29 non-terminals, and 61 productions declared,
producing 109 unique parse states.
 0 terminals declared but not used.
 0 non-terminals declared but not used.
 0 productions never reduced.
 0 conflicts detected (0 expected).
Code written to "Parser.java", and "sym.java".
----- (CUP v0.11b 20160615 (GIT 4ac7450))
rezam@DESKTOP-P0E6L2Q MINGW64 ~/Desktop/Project/Source/Phase2/src
$ |
```

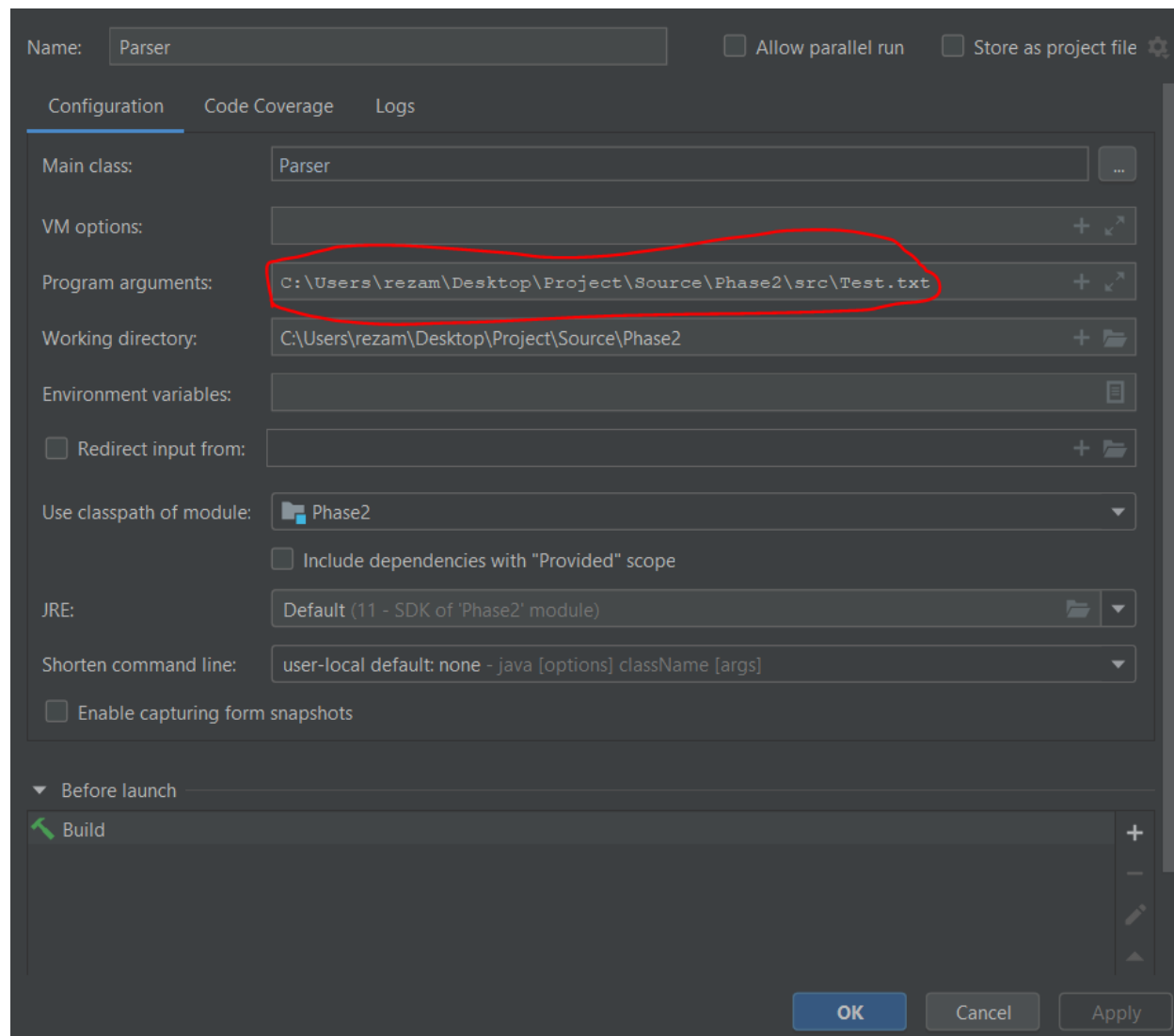
فایل java-cup-11b.jar همان کتابخانه CUP است که باید در پوشه فعلی وجود داشته باشد.

فایل های ایجاد شده شامل Scanner.java، Parser.java و sym.java می باشد.

کتابخانه CUP 0.11b را مطابق تصاویر به پروژه IntelliJ IDEA اضافه می کنیم:

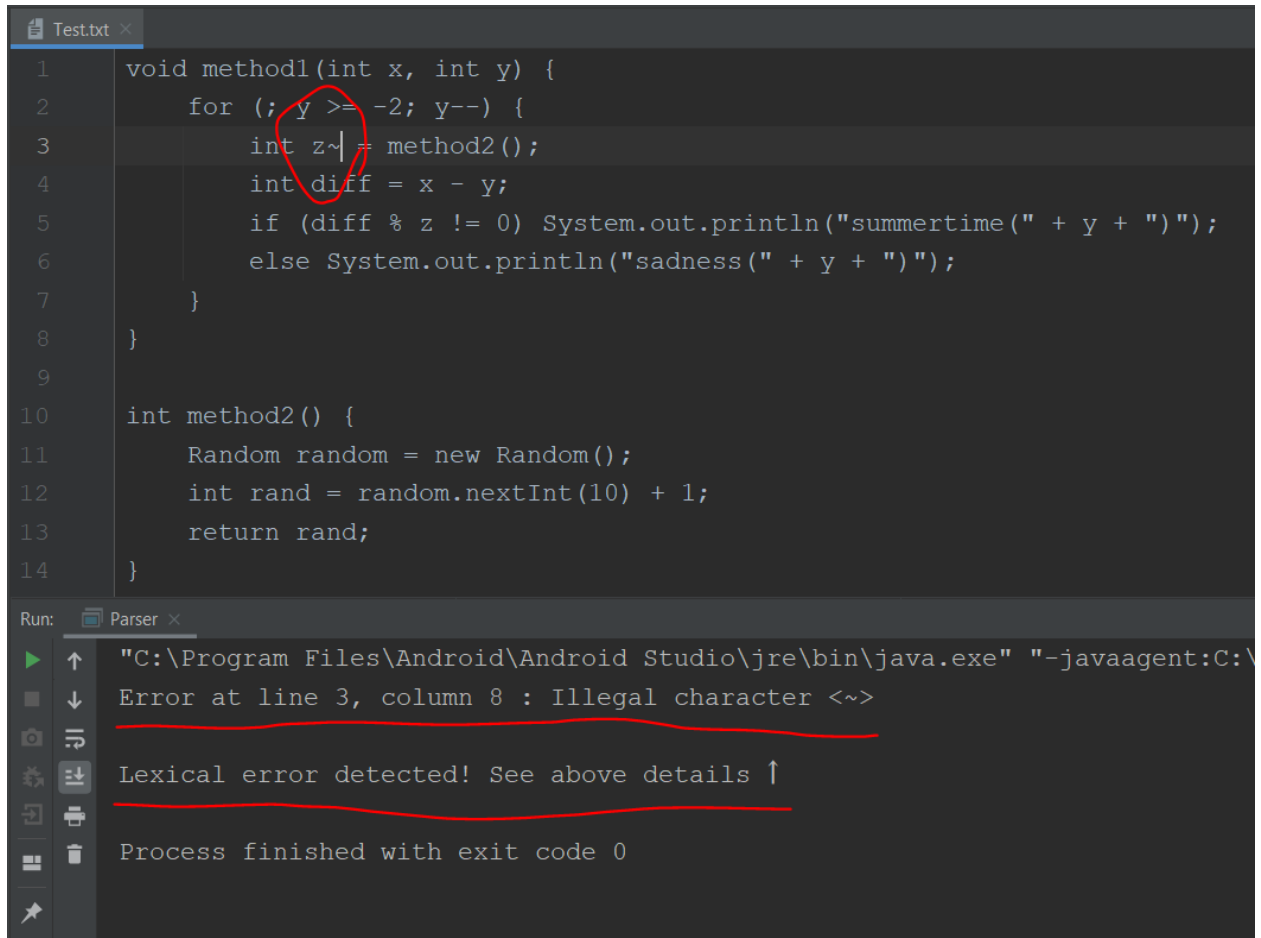


فایل Test.txt در پوشه src پروژه وجود دارد. باید برای تست، آدرس این فایل را به عنوان آرگومان تابع main که در Parser.java وجود دارد بدهیم. برای این کار تنظیمات build و اجرای این تابع در IntelliJ IDEA را به شکل زیر تغییر می دهیم:



سپس با اجرای تابع main کار اسکن و پارس محتویات فایل Test.txt آغاز می شود.

در صورتی که خطای لغوی (lexical) داشته باشیم اروری مشابه زیر چاپ می شود:

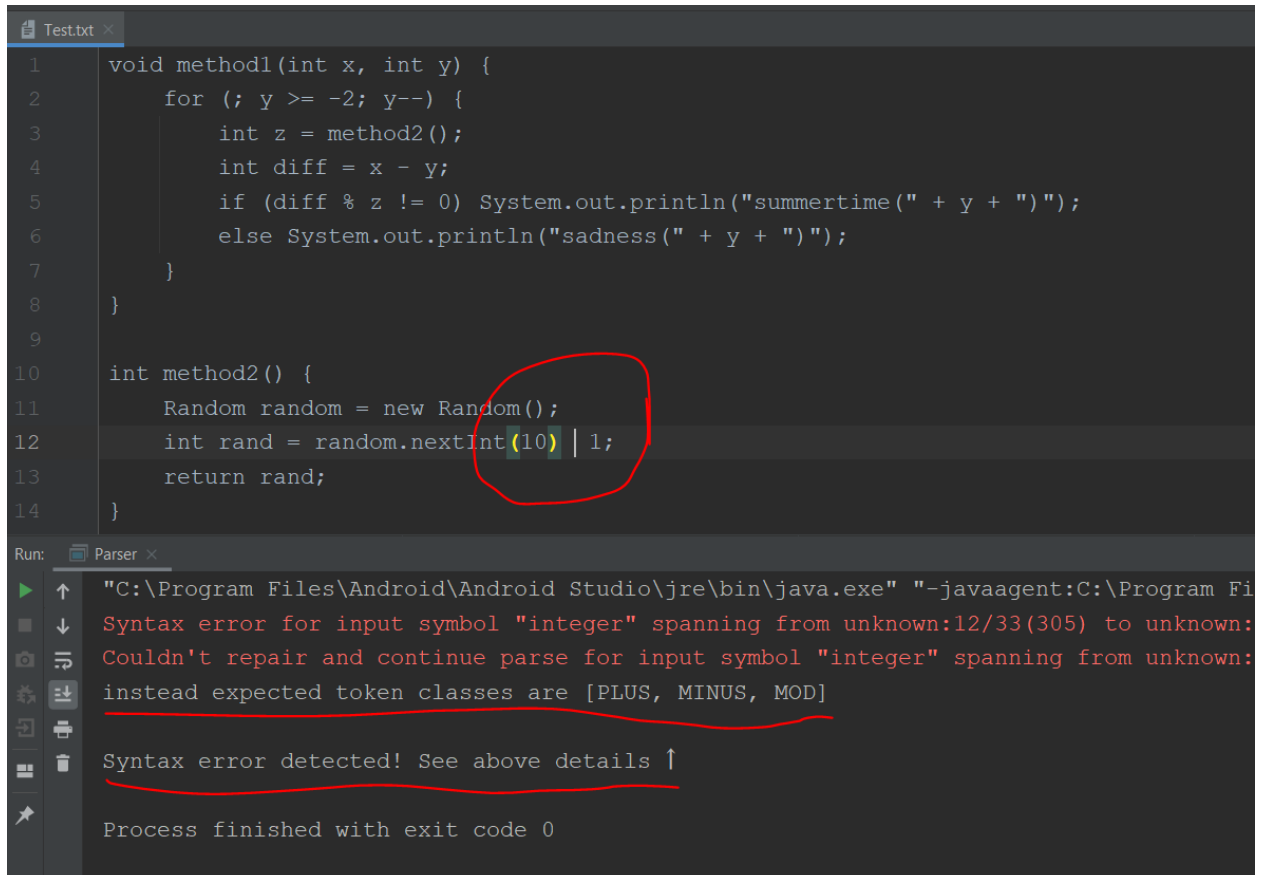


```
1 void method1(int x, int y) {
2     for (; y >= -2; y--) {
3         int z~ = method2();
4         int diff = x - y;
5         if (diff % z != 0) System.out.println("summertime(" + y + ")");
6         else System.out.println("sadness(" + y + ")");
7     }
8 }
9
10 int method2() {
11     Random random = new Random();
12     int rand = random.nextInt(10) + 1;
13     return rand;
14 }
```

Run: Parser ×

↑ "C:\Program Files\Android\Android Studio\jre\bin\java.exe" "-javaagent:C:\...
↓ Error at line 3, column 8 : Illegal character <~>
⌕ Lexical error detected! See above details ↑
🖨 Process finished with exit code 0

در صورتی که خطای نحوی (syntax error) داشته باشیم اروری مشابه زیر چاپ می شود:



```
Test.txt x
1 void method1(int x, int y) {
2     for (; y >= -2; y--) {
3         int z = method2();
4         int diff = x - y;
5         if (diff % z != 0) System.out.println("summertime(" + y + ")");
6         else System.out.println("sadness(" + y + ")");
7     }
8 }
9
10 int method2() {
11     Random random = new Random();
12     int rand = random.nextInt(10) | 1;
13     return rand;
14 }
```

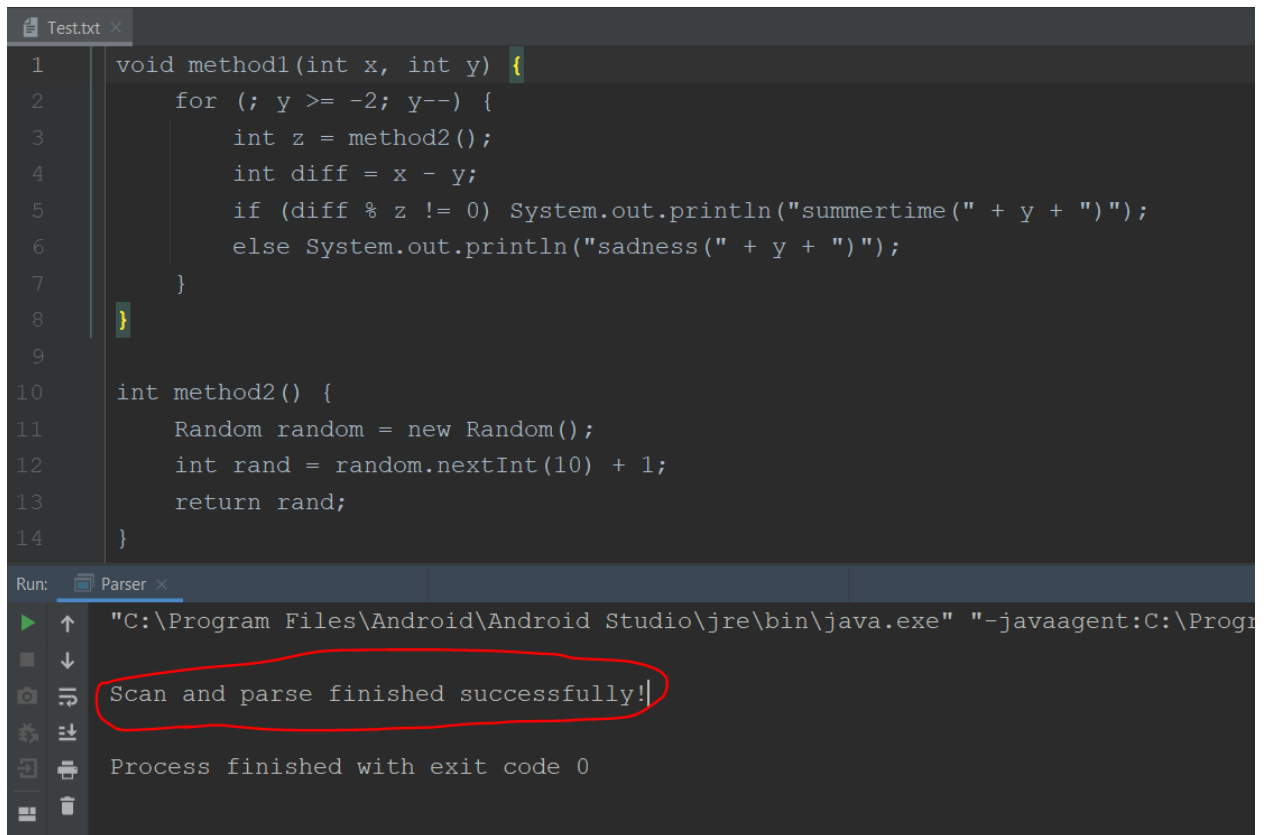
Run: Parser x

"C:\Program Files\Android\Android Studio\jre\bin\java.exe" "-javaagent:C:\Program Fi
Syntax error for input symbol "integer" spanning from unknown:12/33(305) to unknown:
Couldn't repair and continue parse for input symbol "integer" spanning from unknown:
instead expected token classes are [PLUS, MINUS, MOD]

Syntax error detected! See above details ↑

Process finished with exit code 0

در غیر این صورت اسکن و پارس با موفقیت و بدون ارور انجام شده و پیغام زیر چاپ می شود:



The screenshot displays an IDE window titled 'Test.txt' containing the following Java code:

```
1 void method1(int x, int y) {  
2     for (; y >= -2; y--) {  
3         int z = method2();  
4         int diff = x - y;  
5         if (diff % z != 0) System.out.println("summertime(" + y + ")");  
6         else System.out.println("sadness(" + y + ")");  
7     }  
8 }  
9  
10 int method2() {  
11     Random random = new Random();  
12     int rand = random.nextInt(10) + 1;  
13     return rand;  
14 }
```

Below the code editor, the 'Run' tab is active, showing the command: `"C:\Program Files\Android\Android Studio\jre\bin\java.exe" "-javaagent:C:\Progr`. The output console displays the message `Scan and parse finished successfully!`, which is circled in red, followed by `Process finished with exit code 0`.

2 - Parser and Scanner

تفاوت عمده فایل Scanner.flex با فاز یک این است که هنگام تشخیص یک توکن، متد symbol را فراخوانی کرده و خروجی آن را return می کنیم. متد های symbol در بالای همین فایل تعریف شده اند که در داخل آنها از ComplexSymbolFactory که یک کلاس مربوط به CUP است، برای ایجاد symbol استفاده می شود. این کار برای همکاری اسکنر و پارسر است. همچنین در صورتی که هیچ یک از regex ها مچ نشوند، متد error را فراخوانی می کنیم. این متد هم در بالای فایل تعریف شده و در داخل آن جزئیات ارور را چاپ کرده و یک Exception پرتاب می کنیم تا پارسر متوقف شود.

در فایل Parser.cup و در بلاک parser code متد main را می نویسیم. وقتی کد جاوای پارسر تولید شود این متد main دقیقاً به همین شکل در آن وجود خواهد داشت. در این متد یک نمونه از Scanner ایجاد می کنیم و فایلی که آدرس آن را به عنوان آرگومان تابع main دریافت کردیم را برایش ارسال می کنیم. یک نمونه از کلاس Parser هم ایجاد می کنیم و پارامترهای مورد نیاز را می دهیم. کلاس های دیگری که در این تابع استفاده شده اند کلاس های کتابخانه CUP هستند. سپس تابع parse را روی Parser صدا می زنیم و اگر به خط بعدی رفت یعنی خطایی نبوده و پیغام موفقیت آمیز بودن را چاپ می کنیم. اما اگر وارد catch شد یعنی یک Exception پرتاب شده. در این حالت اگر ارور

سینتکسی باشد خود CUP اطلاعات مربوط به ارور را نمایش می دهد. ما فقط یک پیغام ساده چاپ می کنیم که بگوییم ارور سینتکسی تشخیص داده شد. اگر ارور لغوی تشخیص داده شده باز هم خود Scanner پیغام خطا را چاپ کرده (متد error در Scanner.flex)، ما فقط یک پیغام چاپ می کنیم که بگوییم خطای لغوی شناسایی شده. برای تشخیص اینکه خطا لغوی بوده یا نحوی، کافست message مربوط به Exception را چک کنیم. چون در متد error در اسکنر، پیغام "lexical" را برای Exception در نظر گرفته بودیم. اگر همچنین پیغامی وجود نداشت، یعنی Exception مربوط به CUP و خطای نحوی است.

در ادامه فایل CUP، ترمینال ها، نان-ترمینال ها، اولویت ها و گرامر را مشخص کرده ایم.