به نام خدا

نام: فاطمه زهرا بخشنده **استاد: دکتر سعید پارسا**

شماره دانشجویی: 98522157 **درس اصول طراحی کامپایلر**

گزارش تمرین 2:

سوال پنجم:

کد این تمرین در refactoring.py قرار دارد. گرامر های جاوا در پوشه grammars اضافه شدند و فایل های lexer و مابقی فایل های ساخته شده از گرامر نیز در پوشه gen موجود است.

ابتدا با استفاده از argparser آدرس فایل ورودی و آدرس فایل خروجی که کد ریفکتور شده را در آن می‌نویسیم را ورودی می گیریم. برای این مقادیر، مقدار دیفالت نیز قرار داده شده است، و help نیز ایجاد شده است.

```
arg_parser = argparse.ArgumentParser(description='Java Refactor If Else Statements')
arg_parser.add_argument(
    '-n', '--file',
    help='input java file path',
    default=r"Game.java"
)
arg_parser.add_argument(
    '-out', '--out',
    help='output java file path',
    default=r"GameRefactored.java"
)
```

```
PS D:\uni\compiler\assignments\HW2\HW2_98522157\Q5> python refactoring.py -h
usage: refactoring.py [-h] [-n FILE] [-out OUT]

Java Refactor If Else Statements

optional arguments:
  -h, --help            show this help message and exit
  -n FILE, --file FILE  input java file path
  -out OUT, --out OUT   output java file path
```

یک کلاس به نام FindStatements ایجاد می‌کنیم که از کلاس JavaParserLabeledListener ارث بری می‌کند. در کلاس ایجاد شده یک لیست به نام statements داریم که وظیفه نگه داری اطلاعات inline if else statement ها را دارد. تابع enterExpression20 را در این کلاس Override می‌کنیم. کانتکست آن را Expression20Context در نظر می‌گیریم و هرگاه که وارد یک inline if else statement شدیم اطلاعات آن را در لیست statements اضافه می‌کنیم.

تابع refactor را ایجاد می کنیم که با ورودی گرفتن مسیر فایل جاوا و ایجاد Listener، Lexer، Parser و انجام عملیات Walk اطلاعات inline if else statement ها را از کد استخراج می‌کند.

سپس متن فایل ورودی را استخراج کرده و هر inline if else statement استخراج شده توسط listener را در متن به if statement تغییر می دهیم. روش این تغییر را به نحوه ای نوشتم که در همه حالات درست انجام می شود. در نهایت کد اصلاح شده در فایل خروجی می ریزیم.

- تابع refactor را با آرگیومنت های دریافت شده فراخوانی می کنیم. کد refactor شده را در آدرس فایل مقصد ذخیره می کنیم. همچنین محتوای کد ریفکتور شده را در خروجی چاپ می کنیم. در فایل ورودی Game.java سه inline if else statement وجود دارد که در کد زیر قابل مشاهده است. و همه آن ها در GameRefactored.java به نحوه درستی به if statement تبدیل شده اند.

```
private void move0()
{
    // line comment
    Move = this.y < 7?String.format("x:%d y:%d ==> {0}, {1}", this.x + this.y):move();
    System.out.printf(Move);
}

/* block comment line 1
block comment line 2
block comment line 3
*/
// line comment
private void move1()
{
    X = this.x < 3?x++:String.format("x:%d y:%d ==> {0}, {1}", this.x + this.y);
    System.out.printf(X);
}

// line comment
/* block comment */
private void move2()
{
    Y = this.x > 1?this.flag[this.y][this.x]:x--;
}
```

عکس از نحوه ورودی دادن به برنامه:

```
PS D:\uni\compiler\assignments\HW2\HW2_98522157\Q5> python refactoring.py -n Game.java -out GameRefactored.java
import java.util.Random;
 import java.util.ArrayList;

 //first line comment
 //second line comment
 //third line comment
 //fourth line comment
 public abstract class Piece
 {
     protected String name;
```

آدرس فایل ورودی آن فایل Game.java است که در همین zip موجود است. آدرس فایل خروجی آن هم فایل GameRefactored.java است که در همین پوشه ایجاد شده است. و می بینیم که کد ریفکتور شده در آن قرار دارد.

خروجی کامل (محتوای فایل GameRefactored.java):

```java
import java.util.Random;
import java.util.ArrayList;

//first line comment
//second line comment
//third line comment
//fourth line comment
public abstract class Piece
{
    protected String name;
    protected int x;
    protected int y;
    protected boolean[][] flag = new boolean[8][4];

    protected Piece(String name, int x, int y)
    {
        this.name = name;
        this.x = x;
        this.y = y;
    }

    protected abstract void move();
}

/* block comment line 1
block comment line 2
block comment line 3
```

```
*/

public class Castle extends Piece
{
    protected Castle(String name, int x, int y)
    {
        super(name, x, y);
        this.flag[this.y][this.x] = true;
    }

    /* block comment */
    @Override
    protected void move()
    {
        switch ((int) (Math.random() * 3))
        {
            case 0: move0(); break;
            case 1: move1(); break;
            case 2: move2(); break;
        }
    }

    private void move0()
    {
        // line comment
        if (this.y < 7)
        {
            Move = String.format("x:%d y:%d ==> {0}, {1}", this.x + this.y);
        }
        else
        {
            Move = move();
        }
        System.out.printf(Move);
    }

    /* block comment line 1
    block comment line 2
    block comment line 3
    */
    // line comment
    private void move1()
    {
        if (this.x < 3)
        {
```

```
            X = x++;
        }
        else
        {
            X = String.format("x:%d y:%d ==> {0}, {1}", this.x + this.y);
        }
        System.out.printf(X);
    }

    // line comment
    /* block comment */
    private void move2()
    {
        if (this.x > 1)
        {
            Y = this.flag[this.y][this.x];
        }
        else
        {
            Y = x--;
        }
    }
}
```