Control Flow Graph

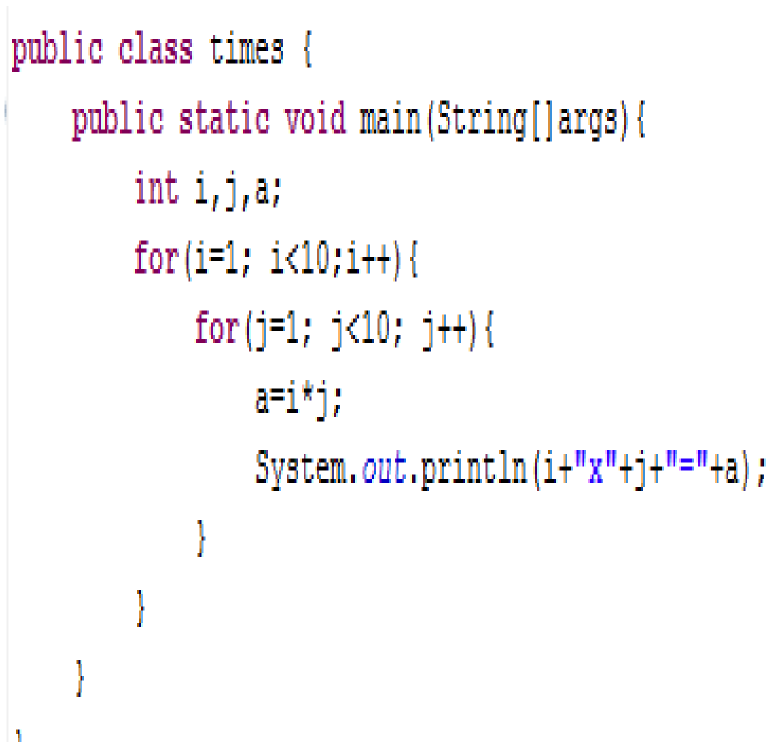
201011314 김민재 201011310 권익진
201011367 정현규 201011370 채희준

Contents

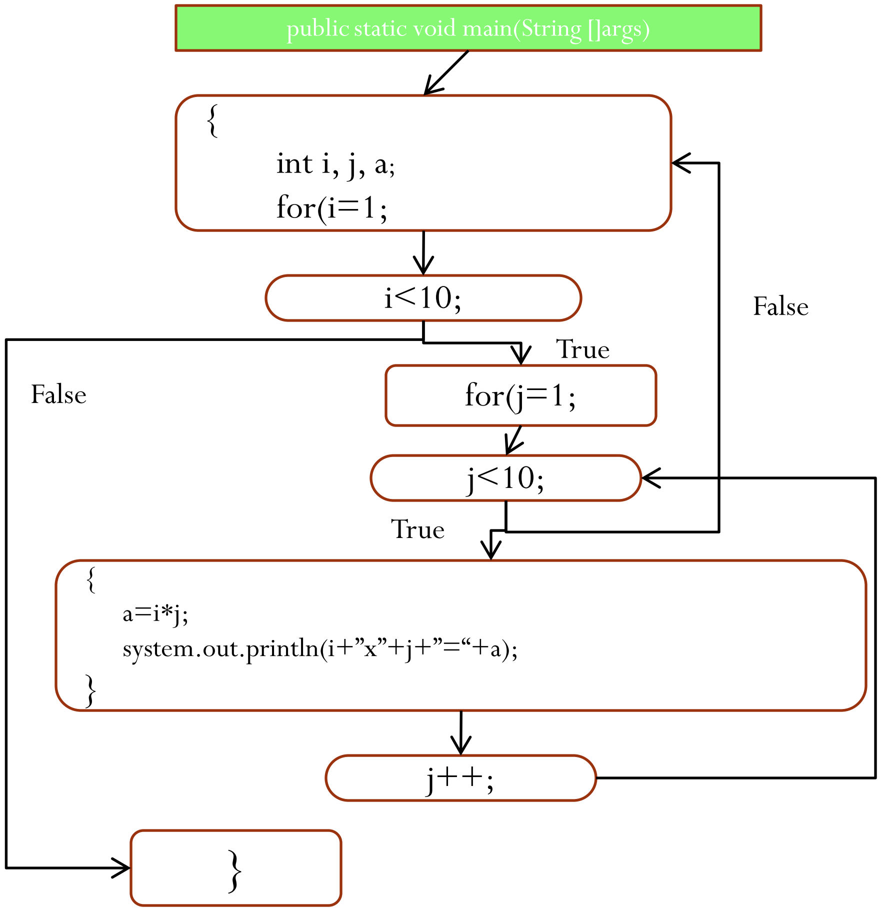
Control Flow Graph(CFG)

- A control flow graph (CFG) in computer science is a representation, using graph notation, of all path s that might be traversed through a program durin g its execution.
- The CFG is essential to many compiler optimizati ons and static analysis tools.
- static analysis only and not compiling or running a ny of the code which needs to be analyzed.
- The methods to do it are explained in details to g et a better understanding of how the evaluation of code can lead to a CFG.

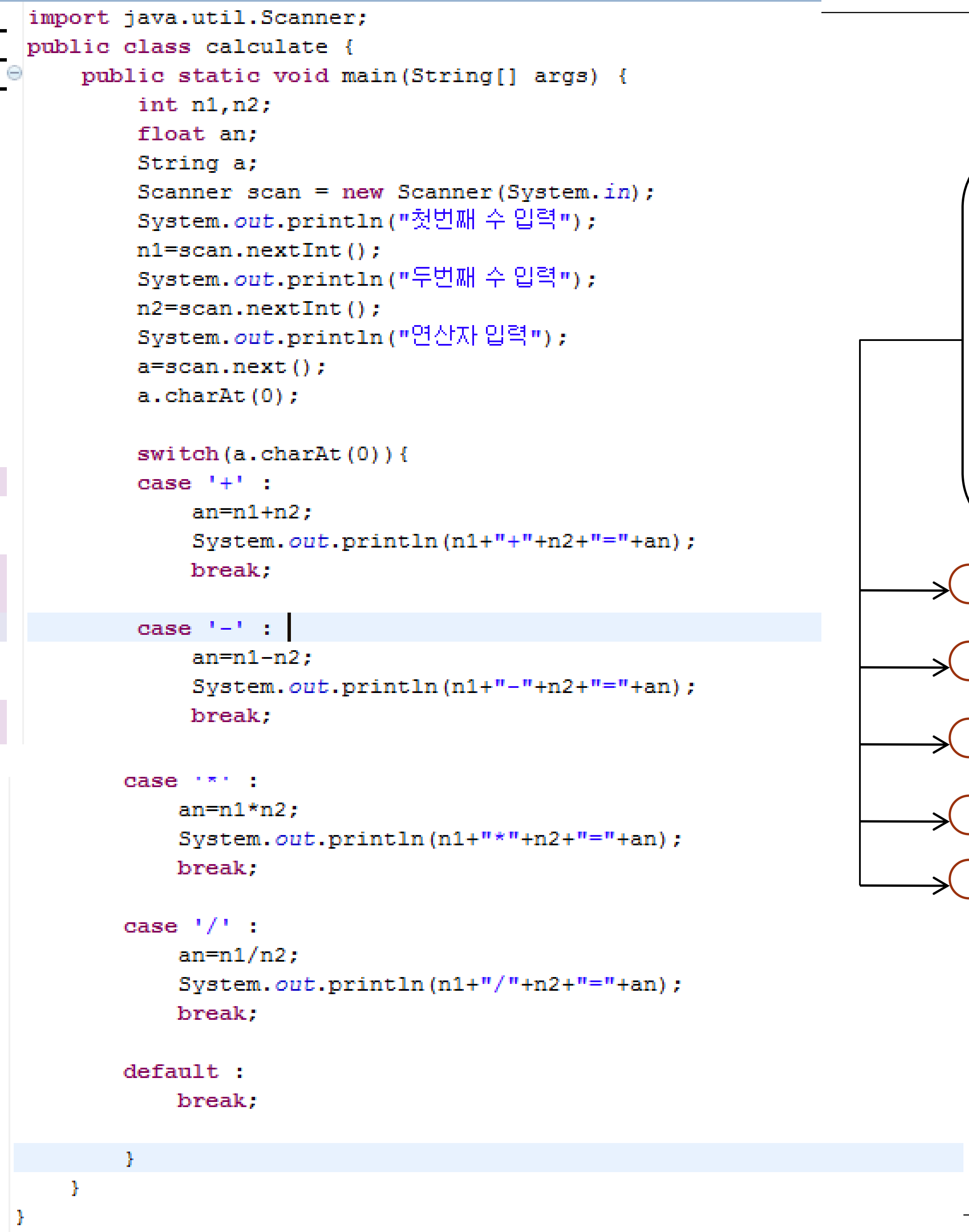
CFG of some codes



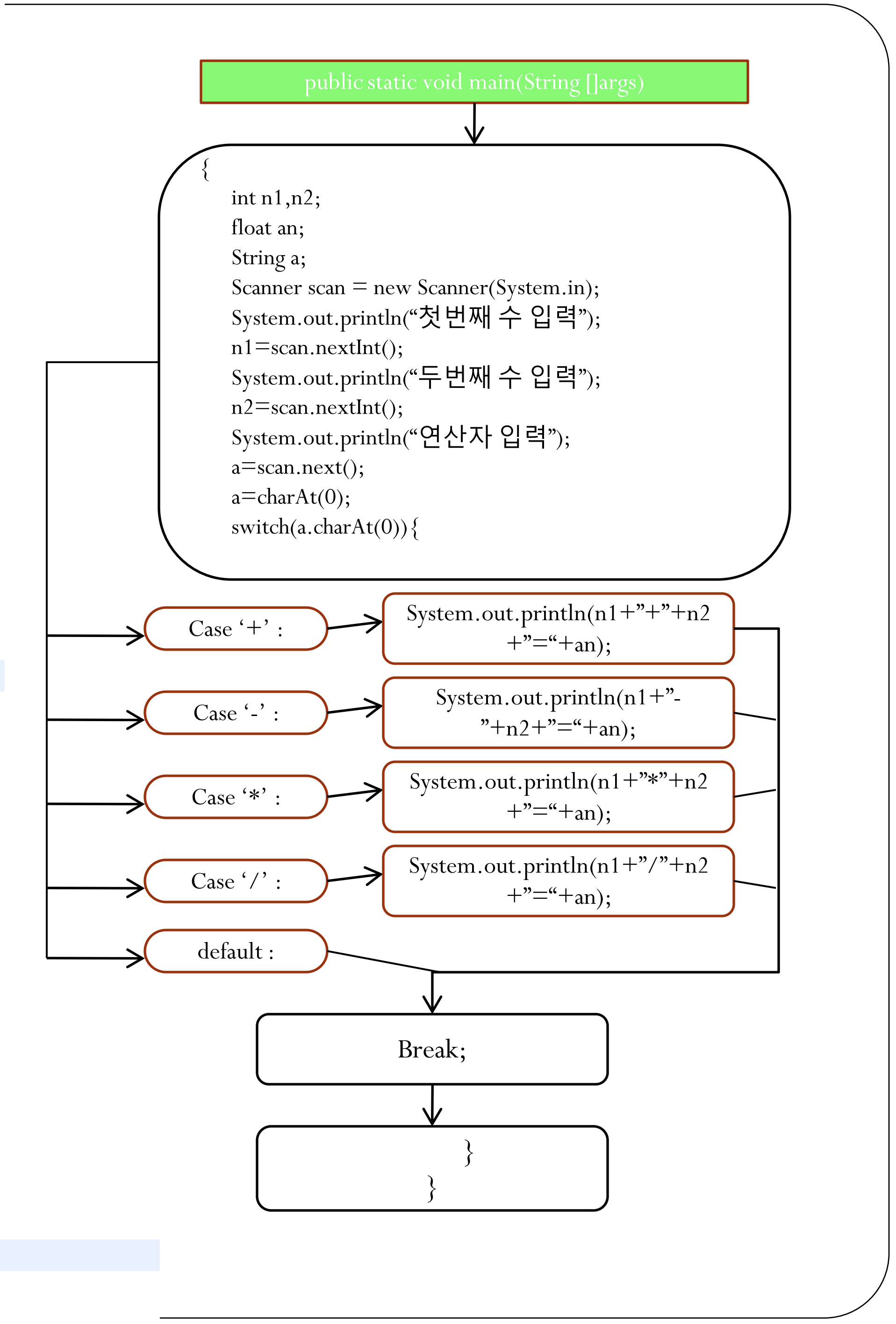
```
public class times {
    public static void main(String[]args){
        int i,j,a;
        for(i=1; i<10;i++){
            for(j=1; j<10; j++){
                a=i*j;
                System.out.println(i+"x"+j+"="+a);
            }
        }
    }
}
```



public static void main(String []args)
{
int i, j, a;
for(i=1;
i<10;
False
True
for(j=1;
False
j<10;
True
{
a=i*j;
system.out.println(i+"x"+j+"="+a);
}
j++;
}

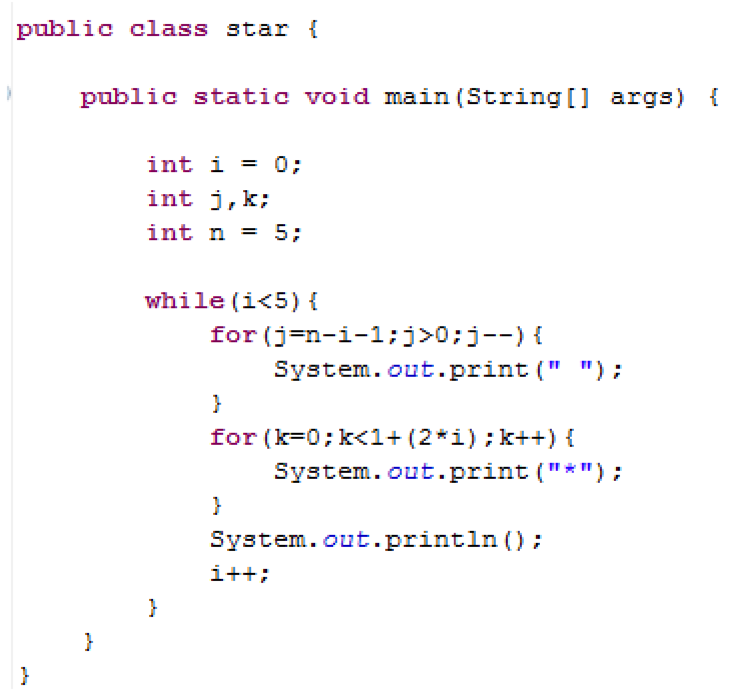


```java
import java.util.Scanner;
public class calculate {
    public static void main(String[] args) {
        int n1,n2;
        float an;
        String a;
        Scanner scan = new Scanner(System.in);
        System.out.println("첫번째 수 입력");
        n1=scan.nextInt();
        System.out.println("두번째 수 입력");
        n2=scan.nextInt();
        System.out.println("연산자 입력");
        a=scan.next();
        a.charAt(0);

        switch(a.charAt(0)){
        case '+' :
            an=n1+n2;
            System.out.println(n1+"+"+n2+"="+an);
            break;

        case '-' :
            an=n1-n2;
            System.out.println(n1+"-"+n2+"="+an);
            break;

        case '*' :
            an=n1*n2;
            System.out.println(n1+"*"+n2+"="+an);
            break;

        case '/' :
            an=n1/n2;
            System.out.println(n1+"/"+n2+"="+an);
            break;

        default :
            break;

        }
    }
}
```

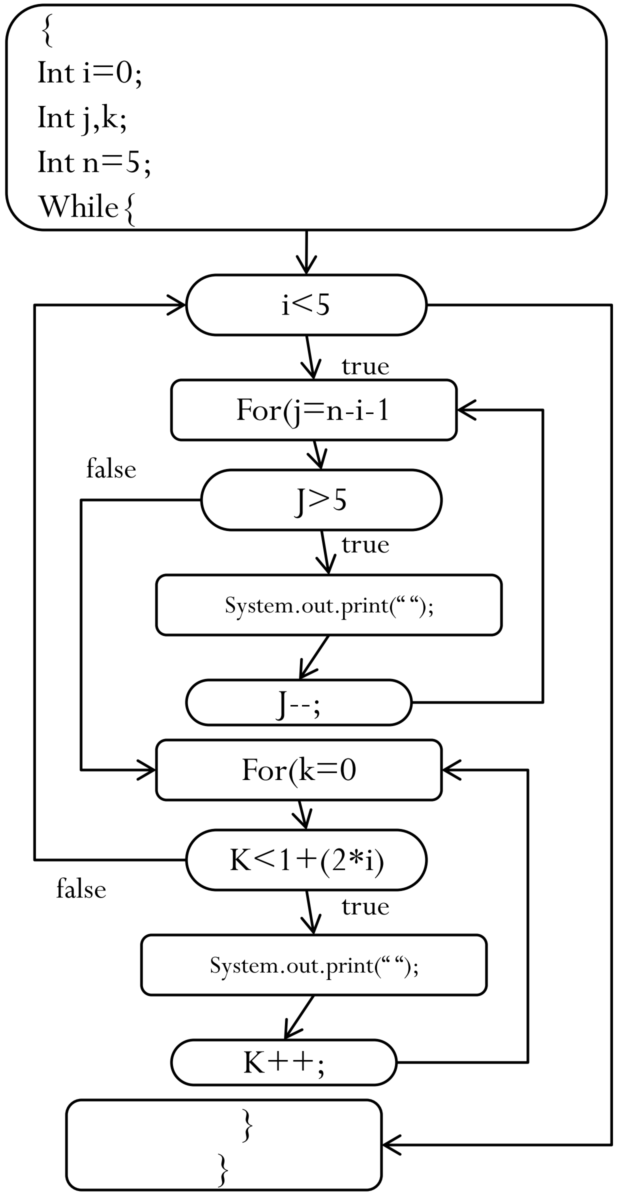


public static void main(String []args)
{
int n1,n2;
float an;
String a;
Scanner scan = new Scanner(System.in);
System.out.println(“첫번째 수 입력”);
n1=scan.nextInt();
System.out.println(“두번째 수 입력”);
n2=scan.nextInt();
System.out.println(“연산자 입력”);
a=scan.next();
a=charAt(0);
switch(a.charAt(0)){
Case ‘+’ :
System.out.println(n1+”+”+n2 +”=“+an);
Case ‘-’ :
System.out.println(n1+”- ”+n2+”=“+an);
Case ‘*’ :
System.out.println(n1+”*”+n2 +”=“+an);
Case ‘/’ :
System.out.println(n1+”/”+n2 +”=“+an);
default :
Break;
}
}

public static void main(String []args)



```
public class star {

    public static void main(String[] args) {

        int i = 0;
        int j,k;
        int n = 5;

        while(i<5){
            for(j=n-i-1;j>0;j--){
                System.out.print(" ");
            }
            for(k=0;k<1+(2*i);k++){
                System.out.print("*");
            }
            System.out.println();
            i++;
        }
    }
}
```



Statement of purpose

- Reachability or different coverage strategies are some of the most common objectives.
- 간단하게 소스코드를 그래프로 그려주는 목적을 가지고 있다.
- 소스코드를 분석하여 간단한 구조로 표현하며 설계한다.
- 설계된 구조를 토대로 그래프로 표현하였다.

Thank you
for Listening