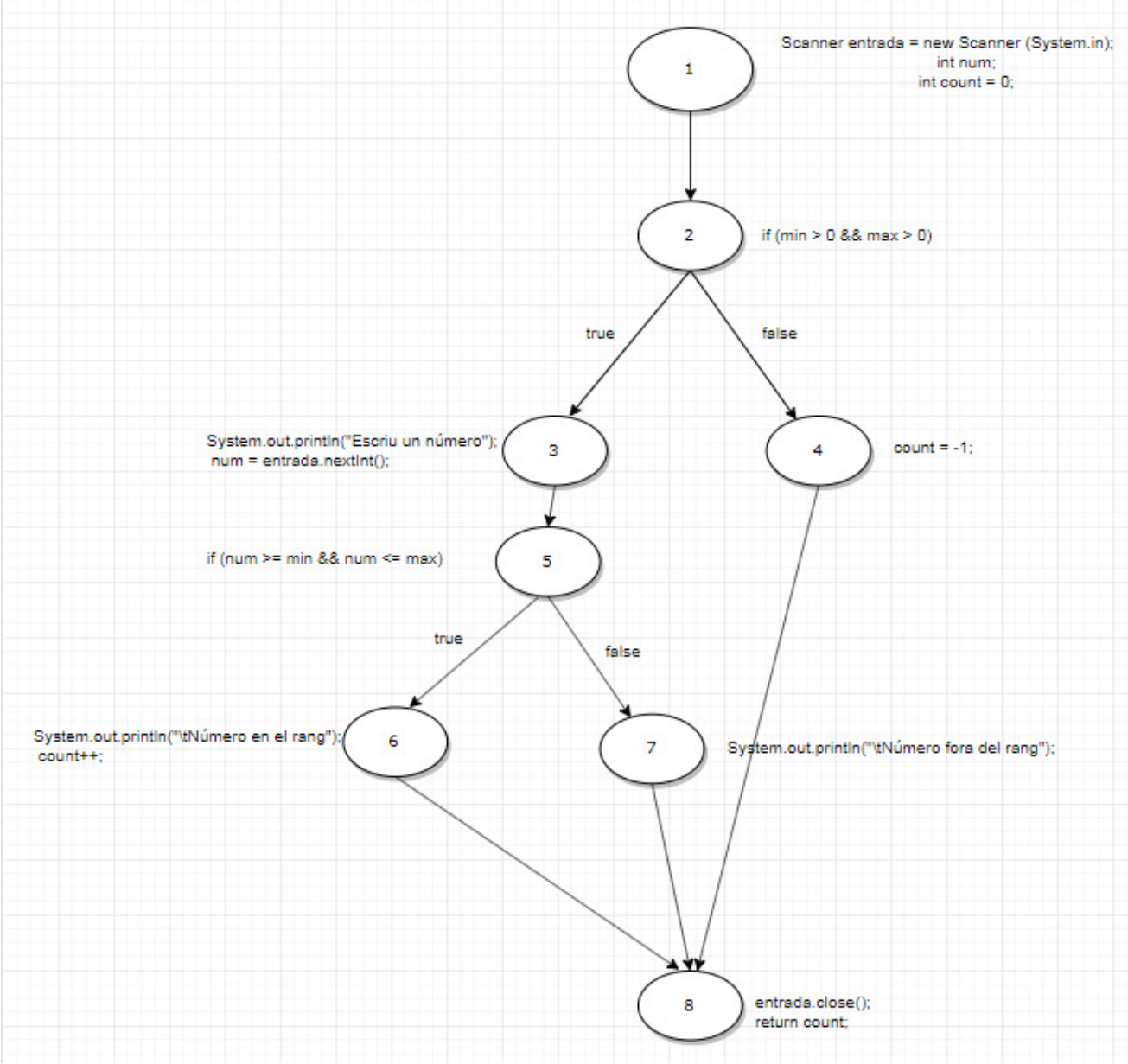
1. Reescriu el codi posant-lo seguint les *Java NamingConventions*, dibuixa el graf de flux, calcula la complexitat ciclomàtica, defineix el conjunt bàsic de camins i elabora casos de prova per cada camí pel següent mètode Java:

```
static int Comptador (int x, int y){
    Scanner entrada = new Scanner(System.in);
    int num, c=0;
    if (x>0 && y>0){
        System.out.println("Escriu un número");
        num = entrada.nextInt();
        if (num>=x && num<=y){
            System.out.println("\tNúmero en el rang");
            c++;
        }
        else
            System.out.println("\tNúmero fora del rang");
    }
    else
        c= -1;
    entrada.close();
    return c;
}
```

Codireescrit

```
static int comptador (int min, int max) {
    Scanner entrada = new Scanner (System.in);
    int num;
    int count = 0;
    if (min > 0 && max > 0) {
        System.out.println("Escriu un número");
        num = entrada.nextInt();
        if (num >= min && num <= max) {
            System.out.println("\tNúmero en el rang");
            count++;
        } else {
            System.out.println("\tNúmero fora del rang");
        }
    } else {
        count = -1;
    }
    entrada.close();
    return count;
}
```

Graf de flux



Complexitat ciclomàtica

CC = #branques - #nodes + 2
CC = 9 – 8 + 2
CC = 3

Conjunt bàsic de camins

A – 1-2-4-8
B – 1-2-3-5-6-8
C – 1-2-3-5-7-8

Casos de prova

```java
import static org.junit.jupiter.api.Assertions.*;

import org.junit.jupiter.api.Test;

class ComptadorTest {

    @Test
    public void countZeroMinMenysDeuMaxDeuCountMenysU() {
        //Arrage (set-up)
        Comptador comptador = new Comptador(0, -10, 10);//inicialitzar a 0, min i max
        //Act
        comptador.validaPositiu(-10, 10);
        //Assert
        assertEquals(new Integer(-1), comptador.getSaldo());
    }

    @Test
    public void countZeroMinUMaxDeuEntradaSetCountVuit() {
        //Arrage (set-up)
        Comptador comptador = new Comptador(0, 1, 10);//inicialitzar a 0, min i max
        //Act
        comptador.validaRang(7);
        //Assert
        assertEquals(new Integer(1), comptador.getSaldo());
    }

    @Test
    public void countZeroMinUMaxDeuEntradaVintCountZero() {
        //Arrage (set-up)
        Comptador comptador = new Comptador(0, 1, 10);//inicialitzar a 0, min i max
        //Act
        comptador.validaRang(20);
        //Assert
        assertEquals(new Integer(0), comptador.getSaldo());
    }
}
```

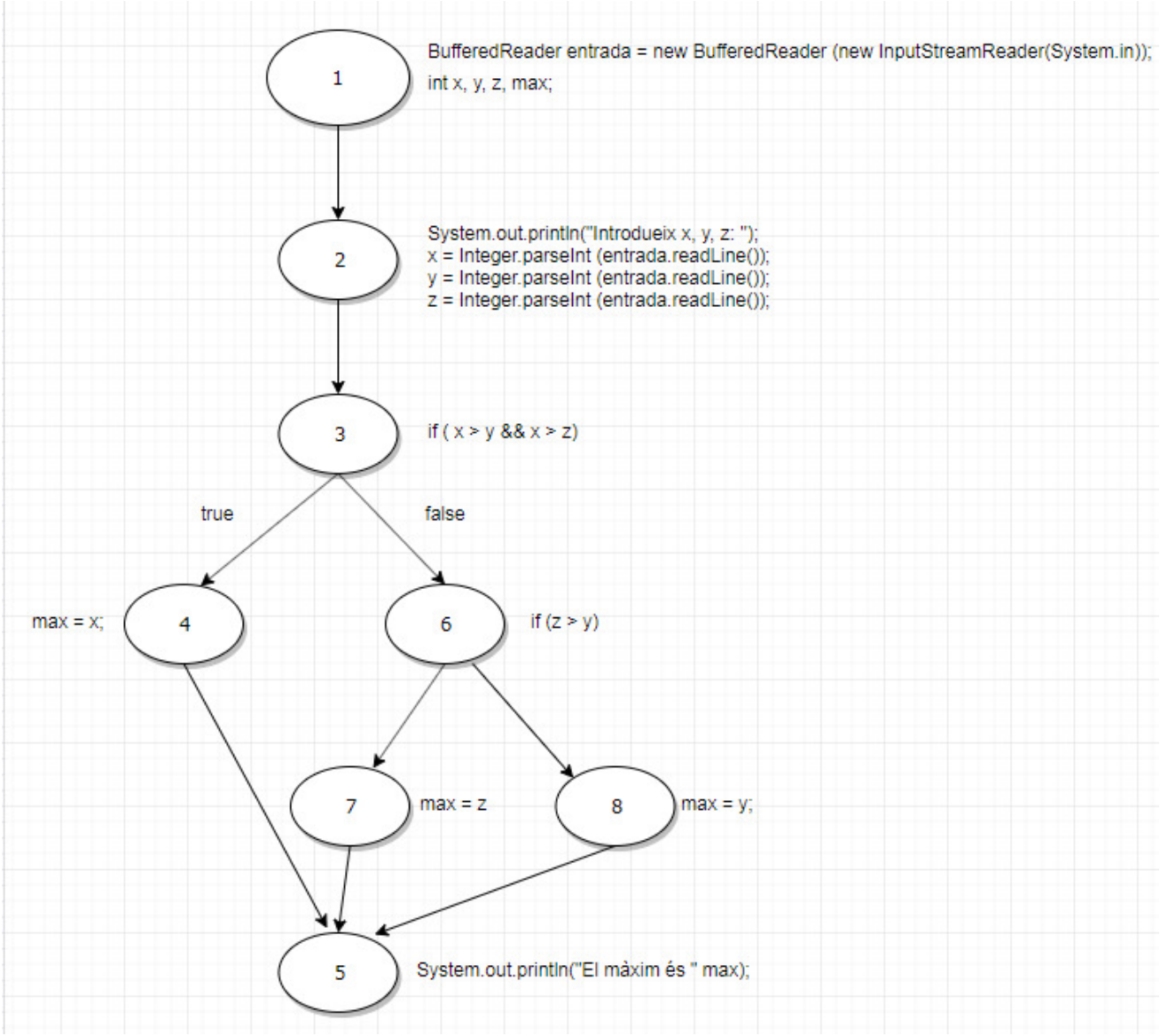
2. Ídem:

```
import java.io.*;

public class Maxim {
    public static void main (String args[]) throws IOException  {
        BufferedReader entrada = new BufferedReader (new InputStreamReader(System.in));

        int x,y,z,max;

        System.out.println("Introdueix x,y,z: ");
        x = Integer.parseInt (entrada.readLine());
        y = Integer.parseInt (entrada.readLine());
        z = Integer.parseInt (entrada.readLine());

        if (x>y && x>z)
            max = x;
        else
            if (z>y)
                max = z;
            else
                max = y;
        System.out.println ("El màxim és "+ max);
    }
}
```

Codireescrit

```
import java.io*;

public class Maxim {
    public static void public static void main(String[] args) {
        BufferedReader entrada = new BufferedReader (new InputStreamReader(System.in));

        int x;
        int y;
        int z;
        int max;

        System.out.println("Introdueix x, y, z: ");
        x = Integer.parseInt (entrada.readLine());
        y = Integer.parseInt (entrada.readLine());
        z = Integer.parseInt (entrada.readLine());

        if ( x > y && x > z) {
            max = x;
        } else if (z > y) {
            max = z;
        } else {
            max = y;
        }
        System.out.println("El màxim és " max);
    }
}
```

Graf de flux



Complexitatciclomàtica

CC = #branques - #nodes + 2
CC = 9 – 8 + 2
CC = 3

Conjunt bàsic de camins

A – 1-2-3-4-5
B – 1-2-3-6-7-5
C – 1-2-3-6-8-5

Casos de prova

```java
import static org.junit.jupiter.api.Assertions.*;

import org.junit.jupiter.api.Test;

class MaximTest {

    @Test
    public void xEsTresYEsDosZEsUMaximTres() {
        //Arrage (set-up)
        Maxim max = new Maxim(3, 2, 1);
        //Act
        max.buscarMax();
        //Assert
        assertEquals(new Integer(3), max.getSaldo());
    }

    @Test
    public void xEsUYEsDosZEsTresMaximTres() {
        //Arrage (set-up)
        Maxim max = new Maxim(1, 2, 3);
        //Act
        max.buscarMax();
        //Assert
        assertEquals(new Integer(3), max.getSaldo());
    }

    @Test
    public void xEsUYEsDosZEsTresMaximTres() {
        //Arrage (set-up)
        Maxim max = new Maxim(1, 3, 2);
        //Act
        max.buscarMax();
        //Assert
        assertEquals(new Integer(3), max.getSaldo());
    }
}
```

3. Estudiem una aplicació bancaria, on l'usuari pot connectar-se al banc a través d'internet i realitzar una sèrie d'operacions bancaries. Un cop s'ha accedit al banc amb les degudes mesures de seguretat, es poden realitzar aquestes operacions. L'operació que s'ha de gestionar requereix la següent entrada:

– Codi del banc: pot estar en blanc o pot ser un número de 3 dígits. En aquest cas el primer d'ells ha de ser major d'1.
– Codi de sucursal: número de 4 dígits. El primer d'ells major de 0.
– Número de compte: número de 5 dígits
– Clau personal: valor alfanumèric de 5 posicions
– Ordre: pot estar en blanc o ser un dels següents valors: "talonari" o "moviments"

El programa respon de la següent manera:

- Si Ordre té el valor "Talonari", l'usuari rep un talonari de xecs.
- Si Ordre té el valor "Moviments", l'usuari rep els moviments del mes en curs.
- Si Ordre està en blanc, l'usuari rep els dos documents.
- Si ocorre algun error en la entrada de dades, el programa mostra un missatge d'error sobre la dada implicada.

Es demana definir les classes d'equivalència, casos de prova vàlids i casos de prova no vàlids que cobreixen una sola classe no vàlida.

Classes d'equivalència

Funció getOrdre (codiBanc, codiSucursal, numCompte, clauPersonal, ordre) retorna ordre

codiBanc
null
3:[[2..9]+[0..9]]
Prova vàlida: null
Prova vàlida: 200
Prova invàlida: 000
Prova invàlida: 100
Prova invàlida: 0000
Prova invàlida: -200
Prova invàlida: AAA

codiSucursal
4:[[1..9]+[0..9]]
Prova vàlida: 1000
Prova invàlida: 0000
Prova invàlida: 10000
Prova invàlida: 10
Prova invàlida: -1000
Prova invàlida: AAAA

numeroCompte
5:[0..9]
Prova vàlida: 00000
Prova invàlida: 0
Prova invàlida: 00
Prova invàlida: 000
Prova invàlida: 0000
Prova invàlida: 000000
Prova invàlida: -00000
Prova invàlida: AAAAA

clauPersonal
5:[2..9 + Aa..Zz]
Prova vàlida: 0A0A0
Prova vàlida: 000AA
Prova vàlida: AA000
Prova invàlida: AAA000
Prova invàlida: 000A
Prova invàlida: 0000
Prova invàlida: 00000
Prova invàlida: AAAAA

codiSucursal
1: [moviments – talonari]
null
Prova vàlida: moviments
Prova vàlida: talonari
Prova invàlida: 0000
Prova invàlida: moviment
Prova invàlida: AAAA

Sortida

Prova vàlida: ordre == "talonari" return xecs
Prova vàlida: ordre == "moviments" return moviments
Prova vàlida: ordre == null return xecs & moviments
Prova invàlida: ordre != ["talonari" / "moviments" / nul] return error talonari || moviments || null
Prova invàlida: ordre = 0000 return error talonari || moviments || null