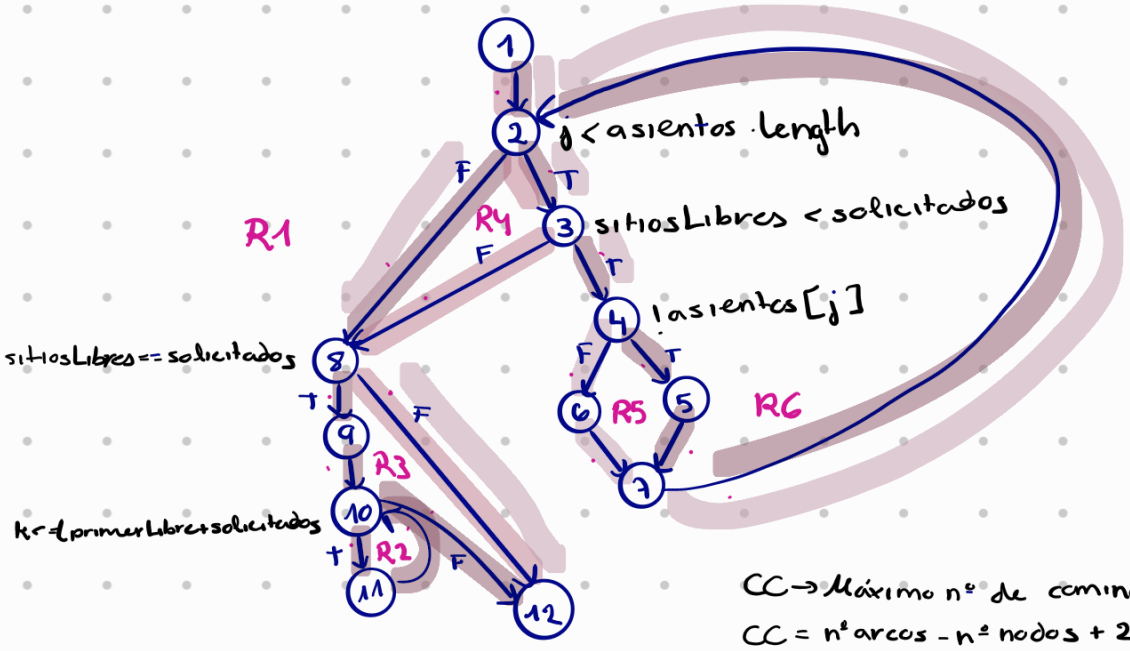
Ejercicio 1

```
1. public boolean reservaButacas(boolean[] asientos, int solicitados) {
2.  boolean reserva = false; int j=0;
3.  int sitiosLibres =0;  int primerLibre;
4.  while ( (j< asientos.length) && (sitiosLibres < solicitados) ) {
5.      if (!asientos[j]) {
6.          sitiosLibres++;
7.      } else {
8.          sitiosLibres=0;
9.      }
10.     j++;
11.  }
12.  if (sitiosLibres == solicitados) {
13.      primerLibre = (j-solicitados);
14.      reserva = true;
15.      for (int k=primerLibre; k<=(primerLibre+solicitados); k++) {
16.          asientos[k] = true;
17.      }
18.  }
19.  return reserva;
20.}
```



CC → Máximo nº de caminos independientes

CC = nº arcos - nº nodos + 2 → 16 - 12 + 2 = 6

CC = nº regiones cerradas → R6

C1 → 1-2-3-4-5-7-2-8-9-10-11-10-12

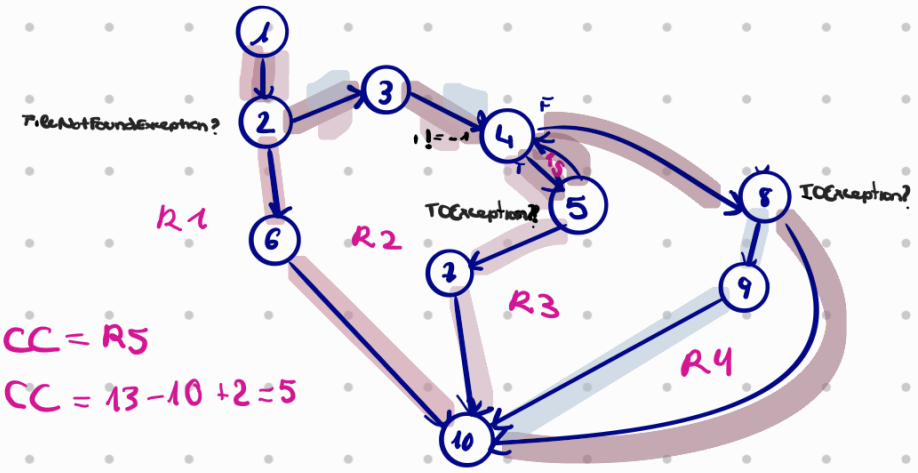
C2 → 1-2-3-8-12

C3 → 1-2-3-4-6-7-2-8-12

	Entradas		Salidas	
Camino	asientos[ ]	solicitados	asientos[ ]	reserva(boolean)
C1	[False]	1	[True]	true
C2	[False]	-1	[False]	false
C3	[True]	1	[True]	false

Ejercicio 2

```
1.  public int contarCaracteres(String nombreFichero) throws FicheroException {
2.     int contador = 0;
3.     FileReader fichero = null;                                  1
4.     try {
5.        fichero = new FileReader(nombreFichero);                 2
6.        int i=0;  3   4
7.        while (i != -1) { // comprobación de fin de fichero
8.           i = fichero.read();
9.           contador++;                                           5
10.       }
11.    6  catch (FileNotFoundException e1) {
12.          throw new FicheroException(nombreFichero +
13.                                     " (No existe el archivo o el directorio)");
14.    7  catch (IOException e2) {
15.          throw new FicheroException(nombreFichero +
16.                                     " (Error al leer el archivo)");
17.    }
18.    if (fichero != null) {          Refactorización
19.       try {
20.          fichero.close();                                      8
21.       } catch (IOException e) {
22.    9     throw new FicheroException(nombreFichero +
23.                                     " (Error al cerrar el archivo)");
24.       }
25.    }
26.    return contador;                                            10
27. }
```



CC = R5

CC = 13 − 10 + 2 = 5

C1 → 1-2-3-4-5-4-8-10

C2 → 1-2-6-11

C3 → 1-2-3-4-5-7-10

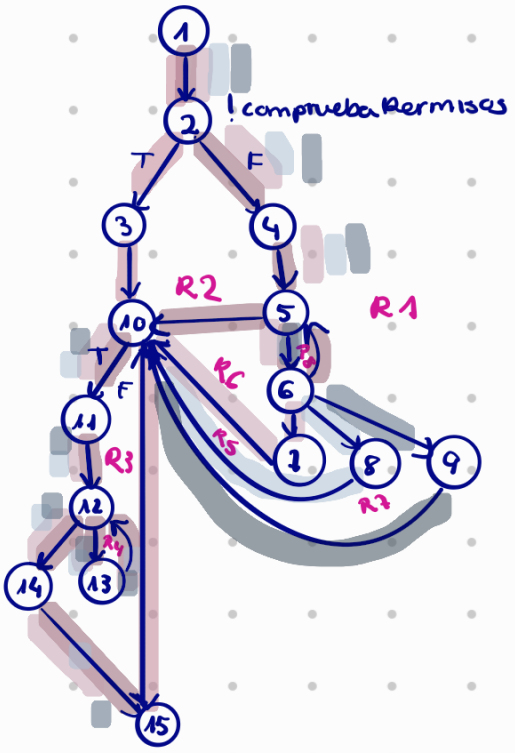
C4 → 1-2-3-4-5-8-9-10

Suponemos que String fich = "src/test/resorces/fich.txt";

	Entradas			Salidas
Camino	nombreFichero	{read(), ..., read()}	close()	contador (int)
C1	fich	{a,b,c,-1}	No se lanza excepción	3
C2	fich.txt	—	—	FileNotFoundException con mensaje "fich.txt (No existe el archivo o el directorio)"
C3	fich	{a,IOException}	—	IOException con mensaje "Error al leer el archivo"
C4	fich	{a,b,c,-1}	IOException	IOException con mensaje "Error al cerrar el archivo"

Ejercicio 3

```
1. public void reservaLibros(String login, String password,
2.                           String socio, String [] isbns) throws Exception {
3.   ArrayList<String> errores = new ArrayList<String>();
4.     //El método compruebaPermisos() devuelve cierto si la persona que hace
5.     //la reserva es el bibliotecario y falso en caso contrario
6.   if(!compruebaPermisos(login, password, Usuario.BIBLIOTECARIO)) {
7.     errores.add("ERROR de permisos");
8.   } else {
9.     FactoriaBOs fd = FactoriaBOs.getInstance();
10.    //El método getOperacionBO() devuelve un objeto de tipo IOperacionBO
11.    //a partir del cual podemos hacer efectiva la reserva
12.    IOperacionBO io = fd.getOperacionBO();
13.    try {
14.      for(String isbn: isbns) {
15.        try {
16.          //El método reserva() registra la reserva de un libro (para un socio)
17.          //dados el identificador del socio e isbn del libro a reservar
18.          io.reserva(socio, isbn);
19.        } catch (IsbnInvalidoException iie) {
20.          errores.add("ISBN invalido" + ":" +  isbn);
21.        }
22.      }
23.    } catch (SocioInvalidoException sie) {
24.      errores.add("SOCIO invalido");
25.    } catch (JDBCException je) {
26.      errores.add("CONEXION invalida");
27.    }
28.  }
29.  if (errores.size() > 0) {
30.    String mensajeError = "";
31.    for(String error: errores) {
32.      mensajeError += error + "; ";
33.    }
34.    throw new ReservaException(mensajeError);
35.  }
36. }
```



CC = R8

CC = 21 − 15 + 2 = 8

C1 1-2-3-10-15

C2 1-2-4-5-6-5-10-11-12-13-12-14-15

C3 1-2-4-5-6-7-10-11-12-13-12-14-15

C4 1-2-4-5-6-8-10-11-12-13-12-14-15

C5 1-2-4-5-6-9-10-11-12-13-12-14-15

	Entradas					Salida
Camino	login	pasword	socio	isbns[]	{reserva{}}	ReservaException() o void()
C1	"xxxx"	"xxxx"	"Paula"	{"22222"}	—	ReservaException "ERROR de permisos"
C2	"ppss"	"ppss"	"Paula"	{"22222"}	{No Exception}	—
C3	"ppss"	"ppss"	"Paula"	{"88888"}	{IsbnExcep}	ReservaException "ISBN invalido: 88888"
C4	"ppss"	"ppss"	"Pablo"	{"22222"}	{SocioExcep}	ReservaException "SOCIO invalido"
C5	"ppss"	"ppss"	"Paula"	{"22222"}	{JDBCExcep}	ReservaException "CONEXION invalida"

Login correcto = "ppss"
password correcto = "ppss"
socio correcto = "Paula"
isbns en la base de datos = {"22222", "33333"}