MANİSA CELAL BAYAR ÜNİVERSİTESİ

FACULTY OF ENGINEERING

COMPUTER ENGINEERING DEPARTMENT

2020-2021 SPRING

OPERATING SYSTEMS

BOOKS IN THE LIBRARY PROJECT

25.06.2021

200315006-170316006

Ömer BENEK - Ayseli Erem BATI

PROJECT SUBJECT

A group of students need to use the same books to perform their term project. There are 40 students in the class (they have consecutive school numbers from 1 to 40) and all need the same 6 books from the library. The students arrive at the library in random order, and they compete to talk to one of the 3 librarians (Students may try to catch one of the librarians randomly).

When a student catches one of the librarians, he/she asks for the book he/she needs. The book needed is generated randomly. The librarian should check if the book was borrowed by any other student. If not, that book is assigned to that student for a random time period in milliseconds. If the book is borrowed by someone else the student should leave, wait for a random duration and try to talk to one of the librarians once more.

Each librarian should be able to see the states of the books even if they were borrowed by another librarian.

Each student needs to read all of the 6 books to complete his/her term project.

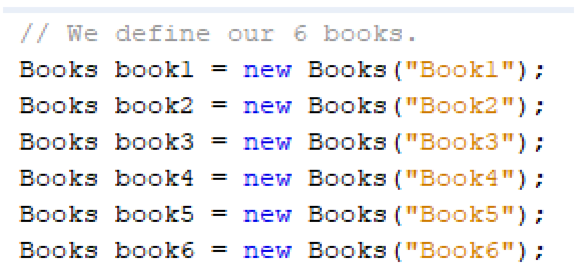
IMPLEMENTATION

While we were doing our project, we used the JAVA programming language because we were more familiar with it. First of all, we created 5 classes for our project. These are Test , Students , Library, Librarians and Books.

CLASSES

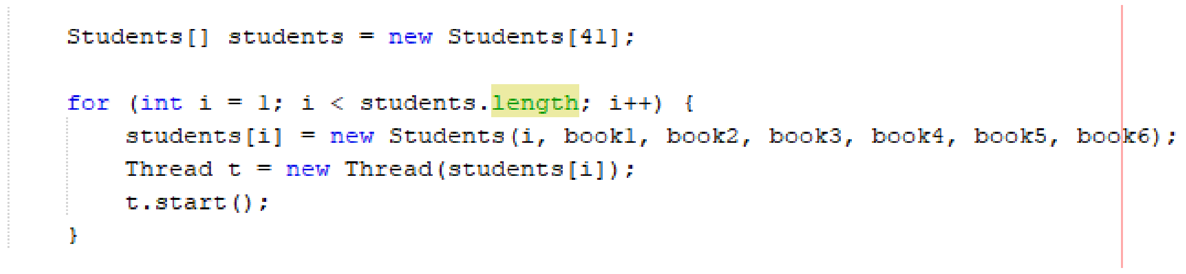
TEST:

This class is where 40 student objects and 6 book objects are created.



```java
// We define our 6 books.
Books book1 = new Books("Book1");
Books book2 = new Books("Book2");
Books book3 = new Books("Book3");
Books book4 = new Books("Book4");
Books book5 = new Books("Book5");
Books book6 = new Books("Book6");
```

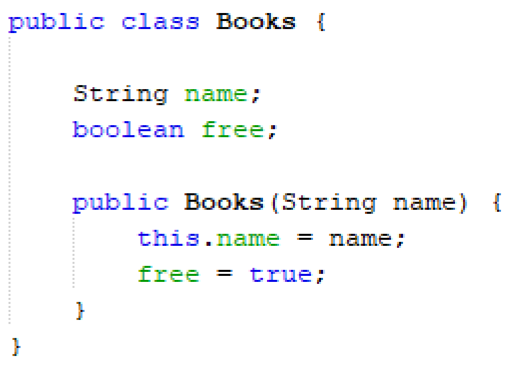
We created an array of type Students to hold 40 students. Then, we created threads for 40 Students objects with the for loop and called the start() method.



```java
Students[] students = new Students[41];

for (int i = 1; i < students.length; i++) {
    students[i] = new Students(i, book1, book2, book3, book4, book5, book6);
    Thread t = new Thread(students[i]);
    t.start();
}
```

BOOKS:

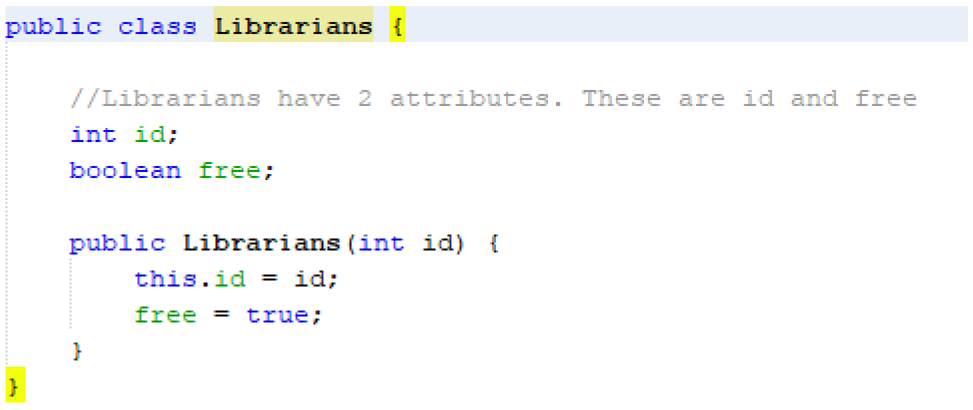
We created this class to hold our book objects. This class has attributes that hold the names of the books and whether the books are full or not.



```java
public class Books {

    String name;
    boolean free;

    public Books(String name) {
        this.name = name;
        free = true;
    }
}
```

LIBRARIANS:

This class holds whether the librarians are full or not and their ids.

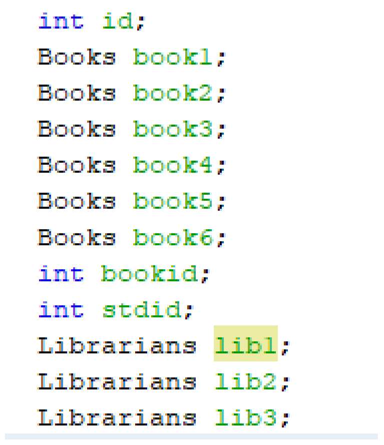


```java
public class Librarians {

    //Librarians have 2 attributes. These are id and free
    int id;
    boolean free;

    public Librarians(int id) {
        this.id = id;
        free = true;
    }
}
```

LIBRARY :

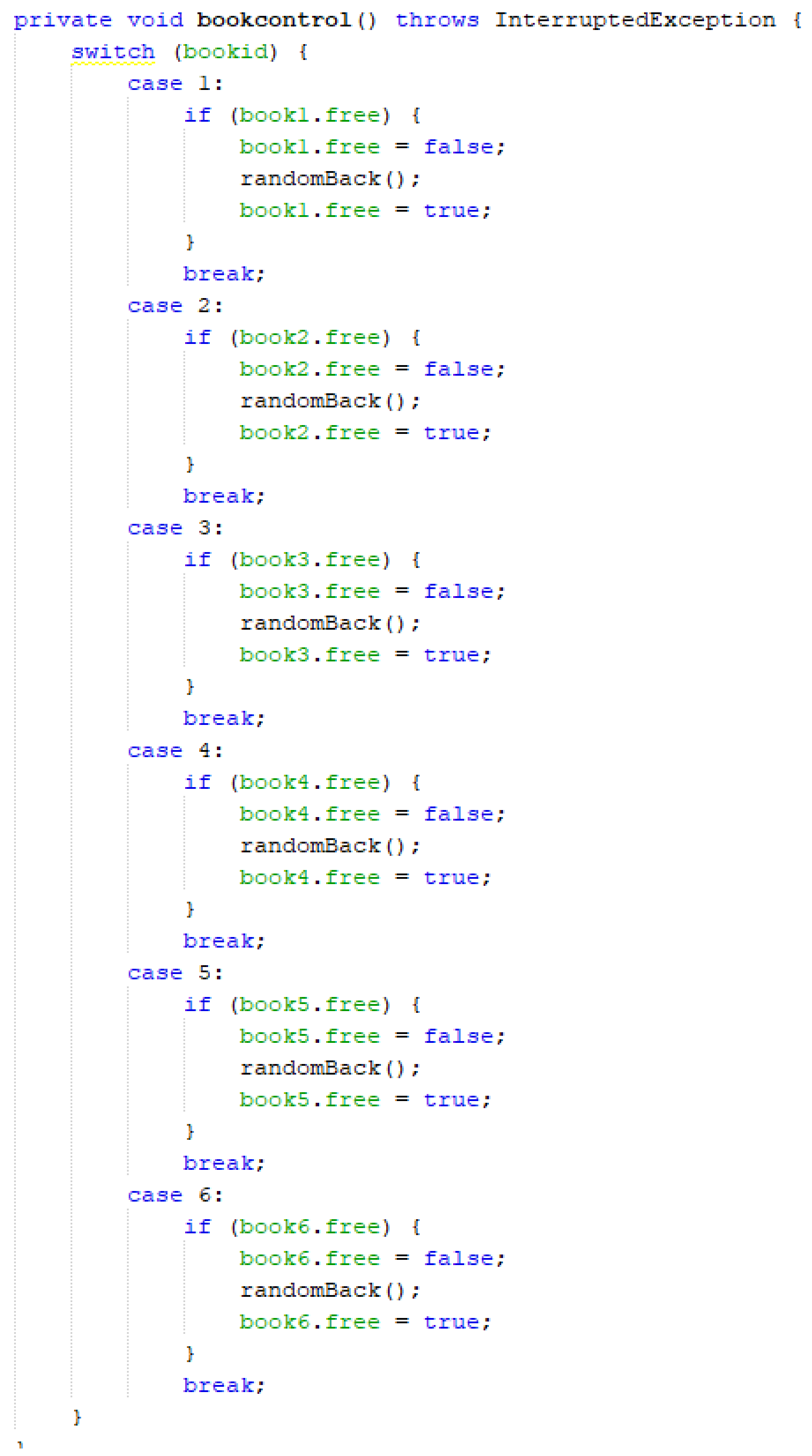
This class is implemented from Runnable Class. Because we will use this class as a thread. Therefore, we had to Override the Run() method belonging to the Runnable Class.

This class holds librarian id, 6 books, book id, student id and 3 librarians as attributes.



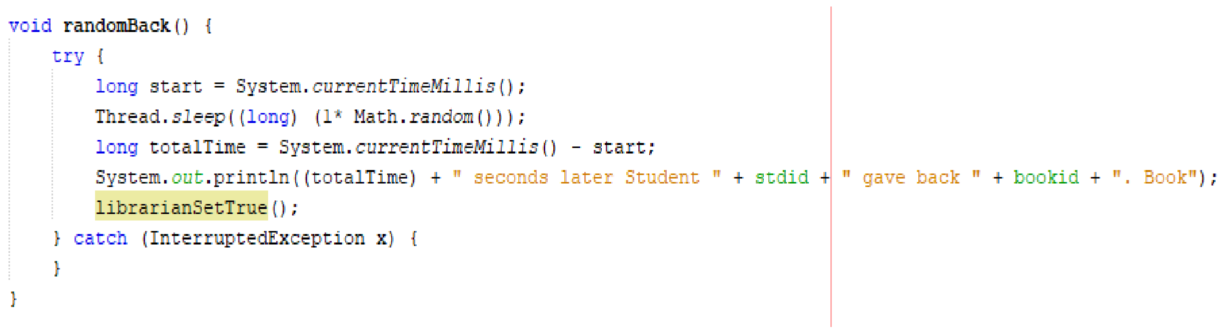
```java
  int id;
  Books book1;
  Books book2;
  Books book3;
  Books book4;
  Books book5;
  Books book6;
  int bookid;
  int stdid;
  Librarians lib1;
  Librarians lib2;
  Librarians lib3;
```

The bookControl() function takes the incoming book id, then checks whether that book is idle, and if it is, it keeps the free value of the book False for a certain period of time by using rondomBack() function. It then sets the book's Free value to True.



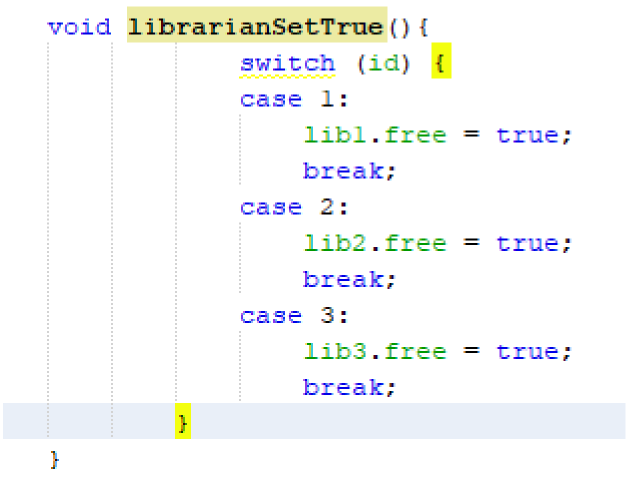
```java
private void bookcontrol() throws InterruptedException {
    switch (bookid) {
        case 1:
            if (book1.free) {
                book1.free = false;
                randomBack();
                book1.free = true;
            }
            break;
        case 2:
            if (book2.free) {
                book2.free = false;
                randomBack();
                book2.free = true;
            }
            break;
        case 3:
            if (book3.free) {
                book3.free = false;
                randomBack();
                book3.free = true;
            }
            break;
        case 4:
            if (book4.free) {
                book4.free = false;
                randomBack();
                book4.free = true;
            }
            break;
        case 5:
            if (book5.free) {
                book5.free = false;
                randomBack();
                book5.free = true;
            }
            break;
        case 6:
            if (book6.free) {
                book6.free = false;
                randomBack();
                book6.free = true;
            }
            break;
    }
}
```

The randomBack() function is a function that decides how many seconds the student reads the book and returns it. It works by generating random numbers.It prints the book that the student brought back to the screen.Finally, it calls the librarianSetTrue() function and sets the free value of the relevant librarian to True.



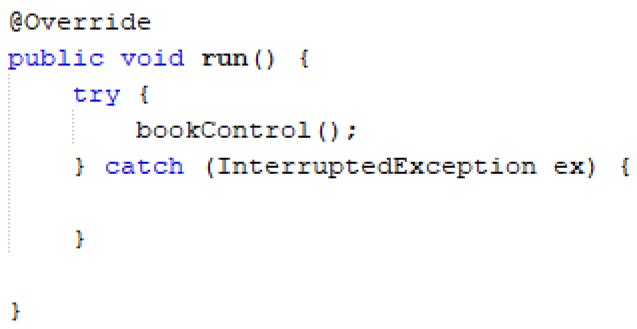
```
void randomBack() {
    try {
        long start = System.currentTimeMillis();
        Thread.sleep((long) (1* Math.random()));
        long totalTime = System.currentTimeMillis() - start;
        System.out.println((totalTime) + " seconds later Student " + stdid + " gave back " + bookid + ". Book");
        librarianSetTrue();
    } catch (InterruptedException x) {
    }
}
```

The librarySetTrue() function makes the free value of the librarian true with whichever librarian the student has completed his function



```
void librarianSetTrue(){
            switch (id) {
            case 1:
                lib1.free = true;
                break;
            case 2:
                lib2.free = true;
                break;
            case 3:
                lib3.free = true;
                break;
        }
 }
```

The run() function also calls the bookContol() method and runs the thread.

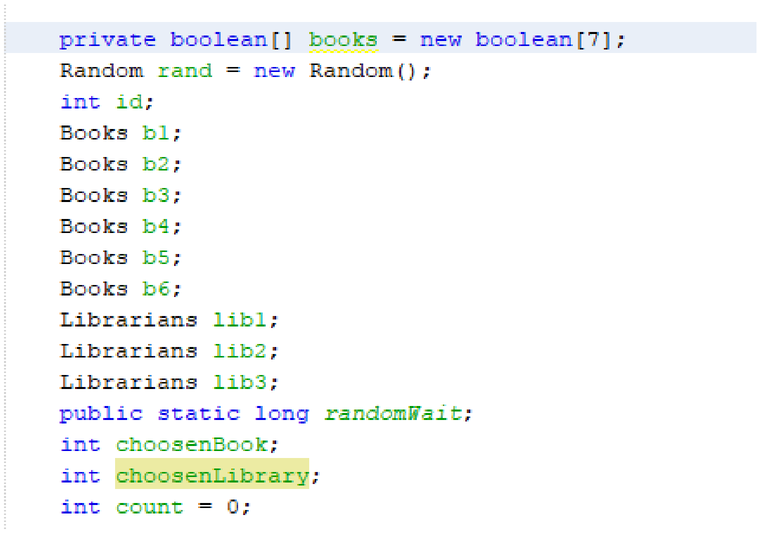


```
@Override
public void run() {
    try {
        bookControl();
    } catch (InterruptedException ex) {

    }

}
```

STUDENT:

This class is implemented from Runnable Class. Because we will use this class as a thread. Therefore, we had to Override the Run() method belonging to the Runnable Class.

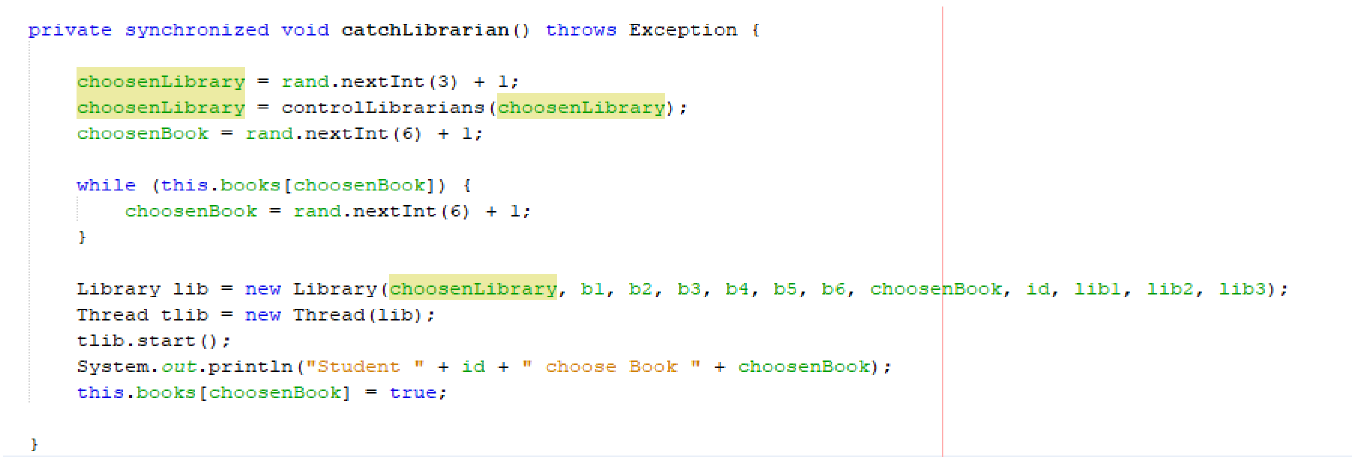
This class holds librarian id, 6 books, book id, student id,choosenBook, choosenLibrary, count, random wait and 3 librarians as attributes. We also have an array with 6 boolean values. This array makes the indexes true as the student reads the books.



```java
private boolean[] books = new boolean[7];
Random rand = new Random();
int id;
Books b1;
Books b2;
Books b3;
Books b4;
Books b5;
Books b6;
Librarians lib1;
Librarians lib2;
Librarians lib3;
public static long randomWait;
int choosenBook;
int choosenLibrary;
int count = 0;
```

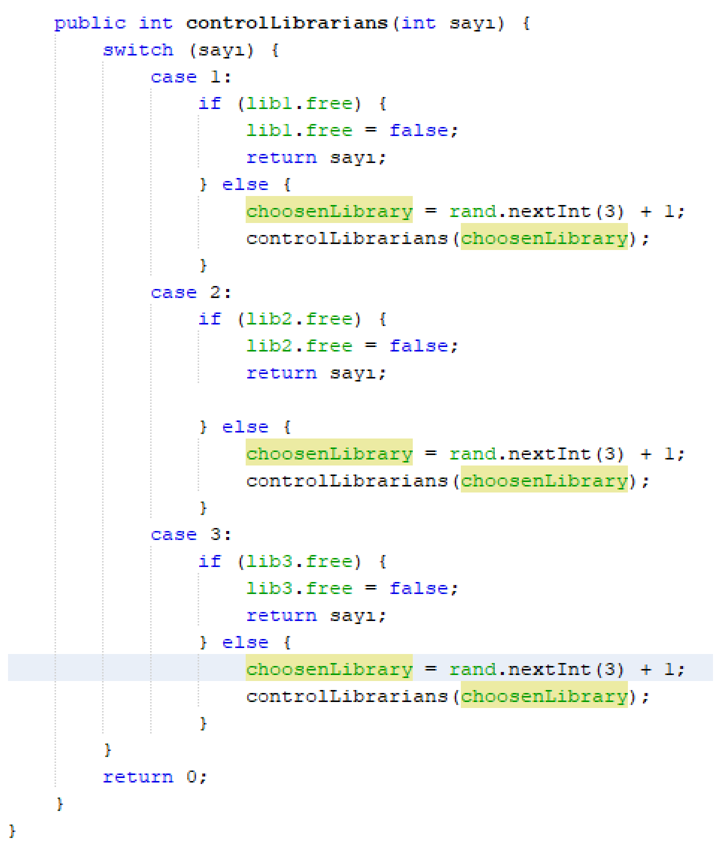
The catchLibrarian() function generates a random number between one and three, which is used to select the librarian.It checks if the librarian is idle by sending this number to the controlLibrarians() function.And then chooses a book between 1 and 6 randomly. In the loop, it checks from the array whether the student has read that book. If he has read it, he randomly chooses a number again.

After passing these stages, we created an object from the library class and put it into a thread object. Finally, we ran the thread and made the index of the selected book true from the array and arranged it as read.



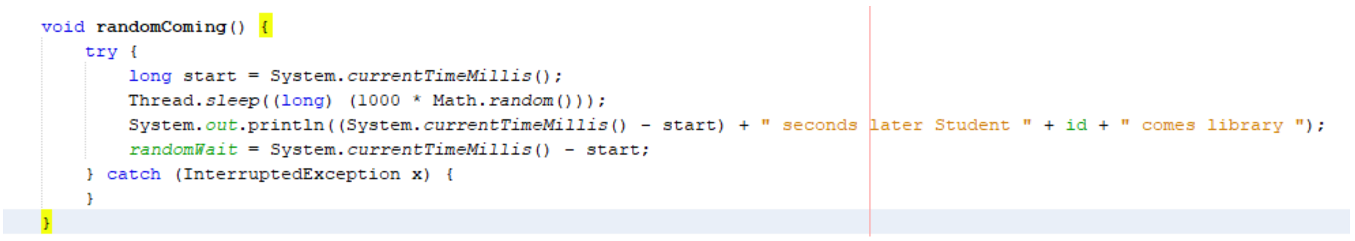
```java
private synchronized void catchLibrarian() throws Exception {

    choosenLibrary = rand.nextInt(3) + 1;
    choosenLibrary = controlLibrarians(choosenLibrary);
    choosenBook = rand.nextInt(6) + 1;

    while (this.books[choosenBook]) {
        choosenBook = rand.nextInt(6) + 1;
    }

    Library lib = new Library(choosenLibrary, b1, b2, b3, b4, b5, b6, choosenBook, id, lib1, lib2, lib3);
    Thread tlib = new Thread(lib);
    tlib.start();
    System.out.println("Student " + id + " choose Book " + choosenBook);
    this.books[choosenBook] = true;

}
```

The controlLibrarian() function checks the availability value of the librarians we have created before by inserting the random number it receives from the catchLibrarian() function into the switch-case structure, and sets it to false if true. If false, it calls itself recursively, generating a random number between 1 and 3. With this event, it generates a random number until it comes across an empty librarian.



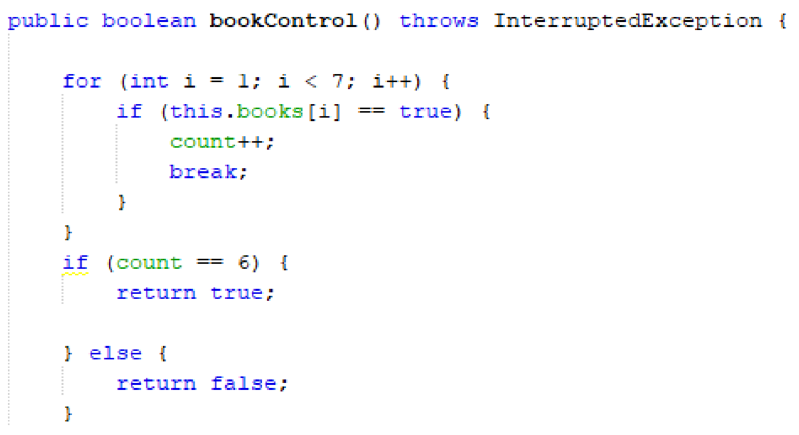
```java
    public int controlLibrarians(int sayı) {
        switch (sayı) {
            case 1:
                if (lib1.free) {
                    lib1.free = false;
                    return sayı;
                } else {
                    choosenLibrary = rand.nextInt(3) + 1;
                    controlLibrarians(choosenLibrary);
                }
            case 2:
                if (lib2.free) {
                    lib2.free = false;
                    return sayı;

                } else {
                    choosenLibrary = rand.nextInt(3) + 1;
                    controlLibrarians(choosenLibrary);
                }
            case 3:
                if (lib3.free) {
                    lib3.free = false;
                    return sayı;
                } else {
                    choosenLibrary = rand.nextInt(3) + 1;
                    controlLibrarians(choosenLibrary);
                }
        }
        return 0;
    }
}
```

The randomComing() function provides a delay to the threads by generating random numbers, which means students arrive at different times.



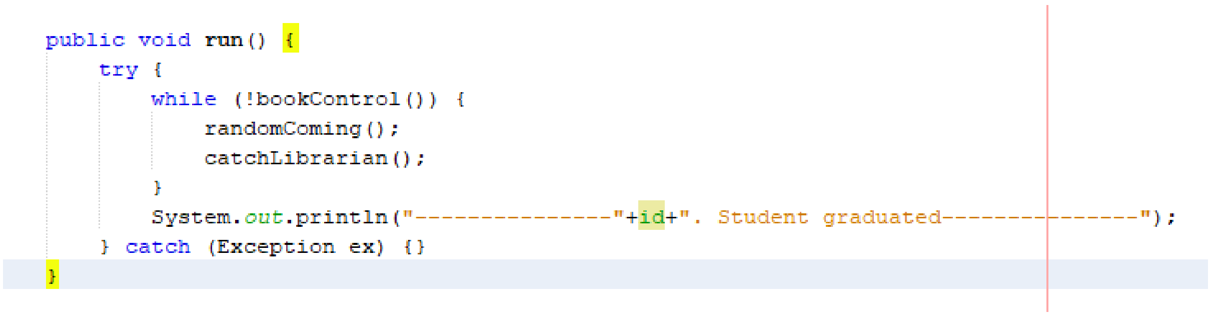
```
void randomComing() {
    try {
        long start = System.currentTimeMillis();
        Thread.sleep((long) (1000 * Math.random()));
        System.out.println((System.currentTimeMillis() - start) + " seconds later Student " + id + " comes library ");
        randomWait = System.currentTimeMillis() - start;
    } catch (InterruptedException x) {
    }
}
```

The bookControl() function scans all the elements of the books array and keeps the true values in count. Returns true if our count is 6. Otherwise it returns false.



```
public boolean bookControl() throws InterruptedException {

    for (int i = 1; i < 7; i++) {
        if (this.books[i] == true) {
            count++;
            break;
        }
    }
    if (count == 6) {
        return true;

    } else {
        return false;
    }
```

The run() function works as long as the value from the bookControl() method is false. So when the student has read all the books, bookControl() returns true and the student reads all the books and graduates.The randomComing() and catchLibrarians() functions run in this loop.



```
public void run() {
    try {
        while (!bookControl()) {
            randomComing();
            catchLibrarian();
        }
        System.out.println("--------------"+id+". Student graduated-------------");
    } catch (Exception ex) {}
}
```

As you can see in the output, students come to the library at random intervals and read the books from the librarian. When they finish six books, they give back the book and graduate.

```
913 seconds later Student 9 comes library
Librarian 3 welcomes to Student 9
Student 9 choose Book 6
---------------9. Student graduated---------------
913 seconds later Student 9 gave back 6. Book
395 seconds later Student 1 comes library
Librarian 3 welcomes to Student 1
Student 1 choose Book 4
---------------1. Student graduated---------------
395 seconds later Student 1 gave back 4. Book
995 seconds later Student 14 comes library
Librarian 1 welcomes to Student 14
Student 14 choose Book 3
---------------14. Student graduated---------------
995 seconds later Student 14 gave back 3. Book
717 seconds later Student 20 comes library
Librarian 1 welcomes to Student 20
Student 20 choose Book 3
---------------20. Student graduated---------------
717 seconds later Student 20 gave back 3. Book
629 seconds later Student 37 comes library
```