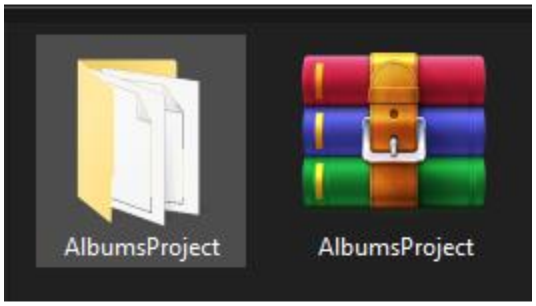
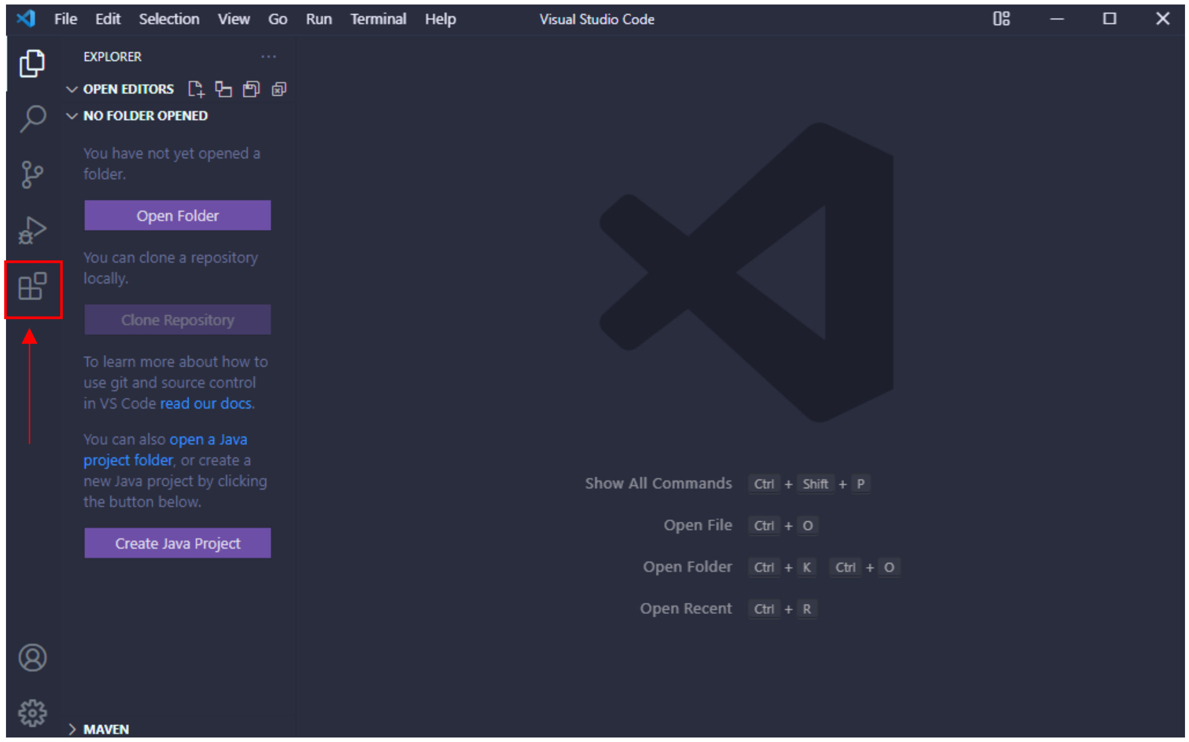
Installation of Project and extensions

First, you will need to unzip the 'AlbumsProject.rar' file, and as a result you will get the main folder of the project, 'AlbumsProject'



Second, we will proceed to install the necessary Java extensions in VS CODE.

For this, we will open our VS code and go to the extensions section



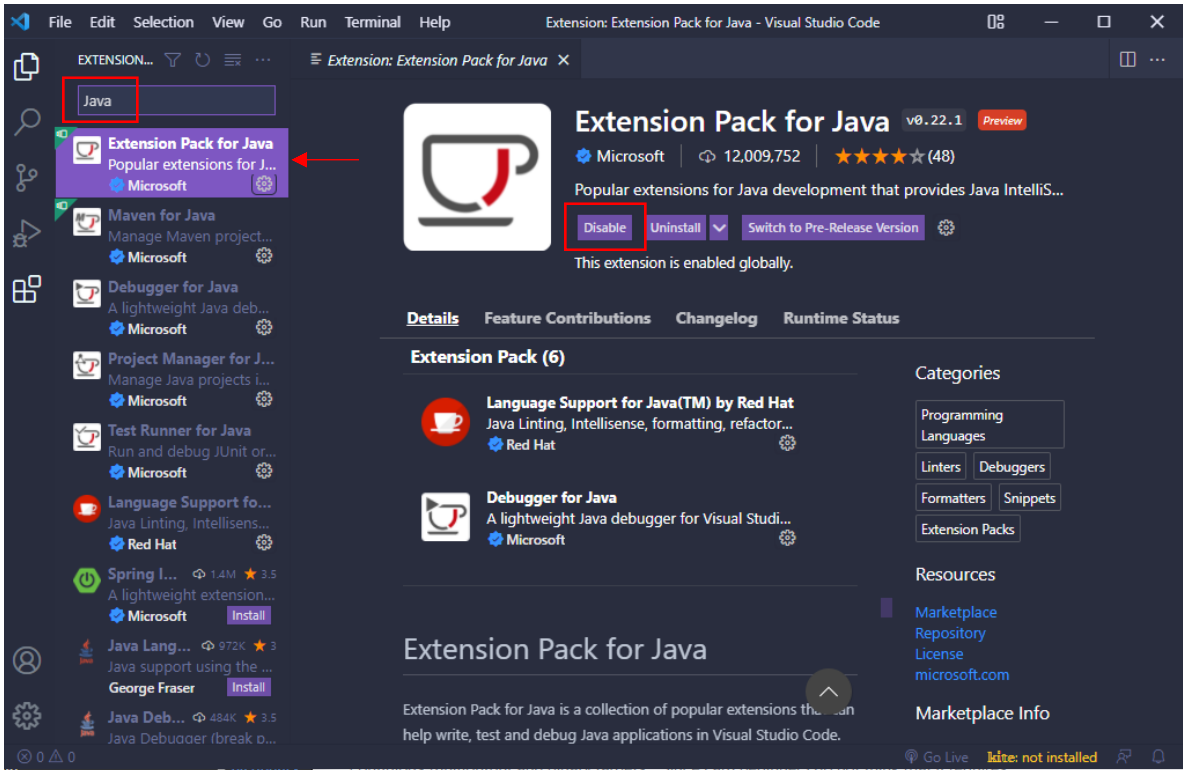
You have not yet opened a folder.

You can clone a repository locally.

To learn more about how to use git and source control in VS Code read our docs.

You can also open a Java project folder, or create a new Java project by clicking the button below.

Once inside, we will search for 'Java' and click on the first package called 'Extension Pack for Java'


File Edit Selection View Go Run Terminal Help
Extension: Extension Pack for Java - Visual Studio Code
EXTENSION...
Java
Extension: Extension Pack for Java
Disable
Uninstall
Switch to Pre-Release Version
Details
Feature Contributions
Changelog
Runtime Status
Install
Go Live
kite: not installed

Extension Pack for Java v0.22.1 Preview

Microsoft | 12,009,752 | (48)

Popular extensions for Java development that provides Java IntelliS...

This extension is enabled globally.

Extension Pack (6)

Language Support for Java(TM) by Red Hat
Java Linting, Intellisense, formatting, refactor...
Red Hat

Debugger for Java
A lightweight Java debugger for Visual Studi...
Microsoft

Extension Pack for Java

Extension Pack for Java is a collection of popular extensions th... can help write, test and debug Java applications in Visual Studio Code.

Categories: Programming Languages, Linters, Debuggers, Formatters, Snippets, Extension Packs

Resources: Marketplace, Repository, License, microsoft.com

Marketplace Info

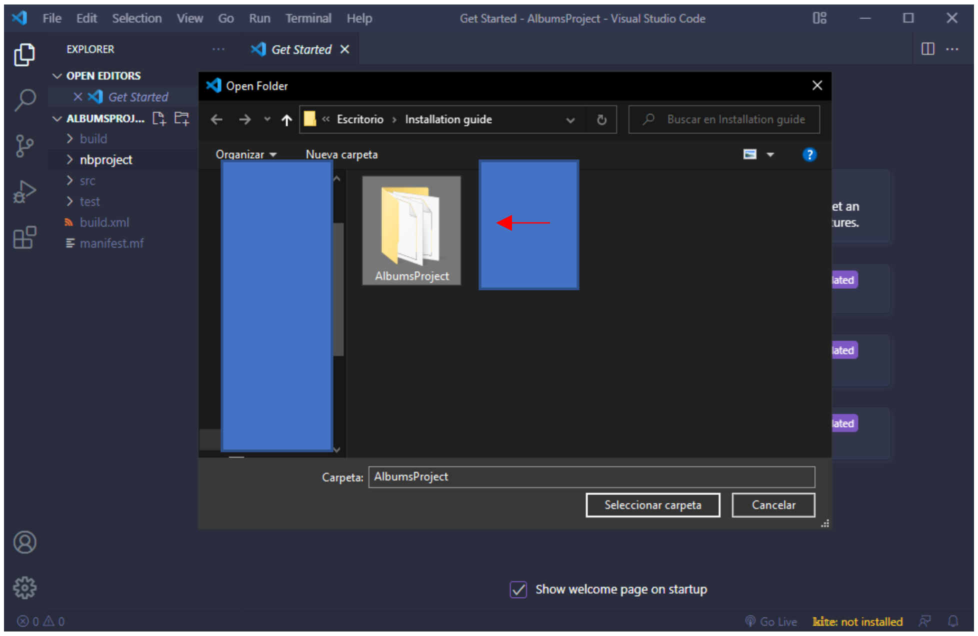
Extension Pack for Java	Popular extensions for J...	Microsoft
Maven for Java	Manage Maven project...	Microsoft
Debugger for Java	A lightweight Java deb...	Microsoft
Project Manager for J...	Manage Java projects i...	Microsoft
Test Runner for Java	Run and debug JUnit or...	Microsoft
Language Support fo...	Java Linting, Intellisens...	Red Hat
Spring I... 1.4M ★ 3.5	A lightweight extension...	Microsoft
Java Lang... 972K ★ 3	Java support using the ...	George Fraser
Java Deb... 484K ★ 3.5	Java Debugger (break p...	

In the button that appears to me as 'Disable', it will probably show 'Install'. We click on install, and we wait for VS code to take care of installing the extensions.

Deploying the project

We will close the extension tab, and return to the files section.

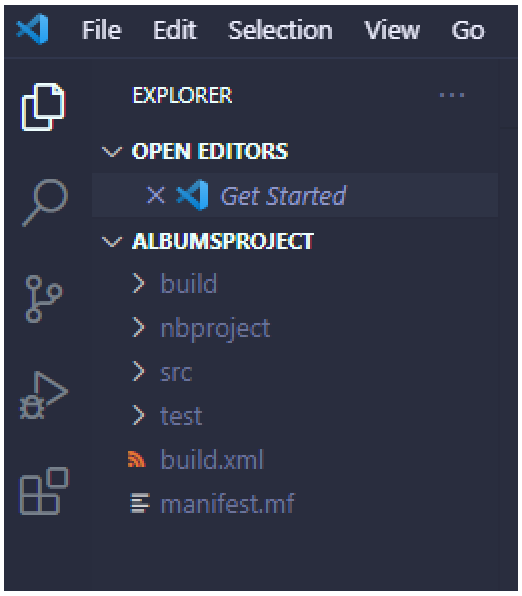
Now we will go to the 'Files' menu in the upper left, and we will click on 'Open folder':



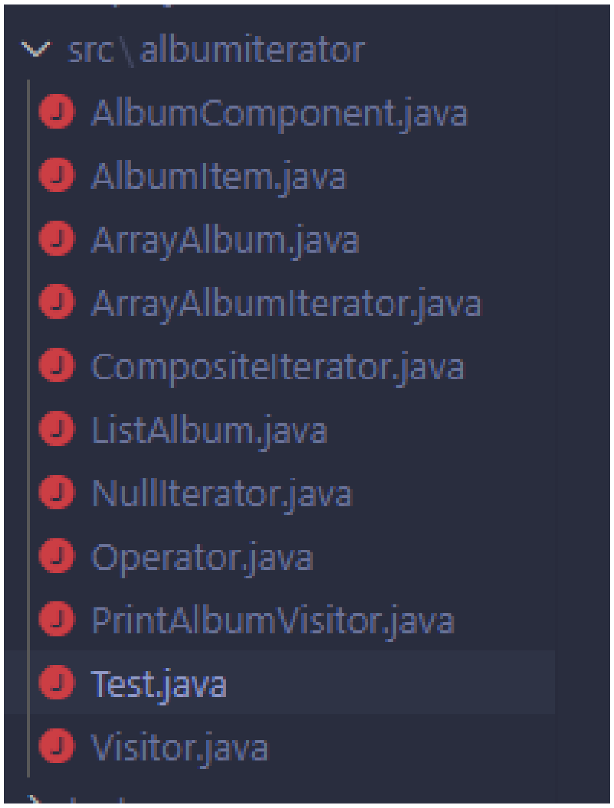
We select the previously unzipped folder ('AlbumProject') and click on 'Select folder'

It is possible that later we will get a pop-up box asking us if we trust the author of the folder, we click yes.

And we will be able to see the project folder next to the program.

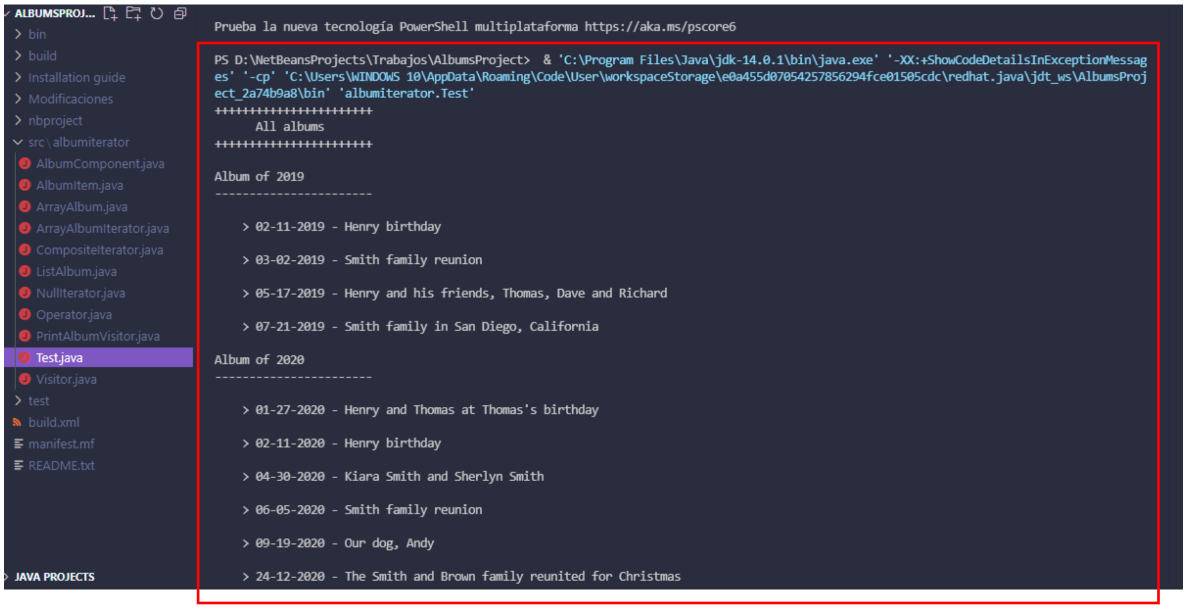


To run the program, we will deploy the 'src' folder



Later, we will RIGHT click on the 'Test.java' class (it is the class that contains our Main method) and then click on 'Run java'

In this way, our program will be executed in the terminal of VS code:

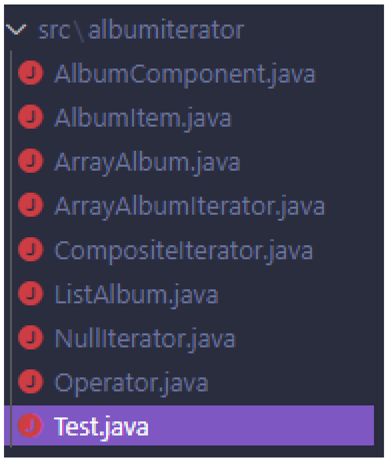


```
Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/pscore6

PS D:\NetBeansProjects\Trabajos\AlbumsProject>  & 'C:\Program Files\Java\jdk-14.0.1\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessag
es' '-cp' 'C:\Users\WINDOWS 10\AppData\Roaming\Code\User\workspaceStorage\e0a455d07054257856294fce01505cdc\redhat.java\jdt_ws\AlbumsProj
ect_2a74b9a8\bin' 'albumiterator.Test'
+++++++++++++++++++++++
      All albums
+++++++++++++++++++++++

Album of 2019
-----------------------

    > 02-11-2019 - Henry birthday

    > 03-02-2019 - Smith family reunion

    > 05-17-2019 - Henry and his friends, Thomas, Dave and Richard

    > 07-21-2019 - Smith family in San Diego, California

Album of 2020
-----------------------

    > 01-27-2020 - Henry and Thomas at Thomas's birthday

    > 02-11-2020 - Henry birthday

    > 04-30-2020 - Kiara Smith and Sherlyn Smith

    > 06-05-2020 - Smith family reunion

    > 09-19-2020 - Our dog, Andy

    > 24-12-2020 - The Smith and Brown family reunited for Christmas
```

Explanation of the program

Java classes:

- Composite objects:
    - AlbumComponent
    - ListAlbum
    - ArrayAlbum
- Leaf objects:
    - AlbumItem



```
src\albumiterator
  AlbumComponent.java
  AlbumItem.java
  ArrayAlbum.java
  ArrayAlbumIterator.java
  CompositeIterator.java
  ListAlbum.java
  NullIterator.java
  Operator.java
  Test.java
```

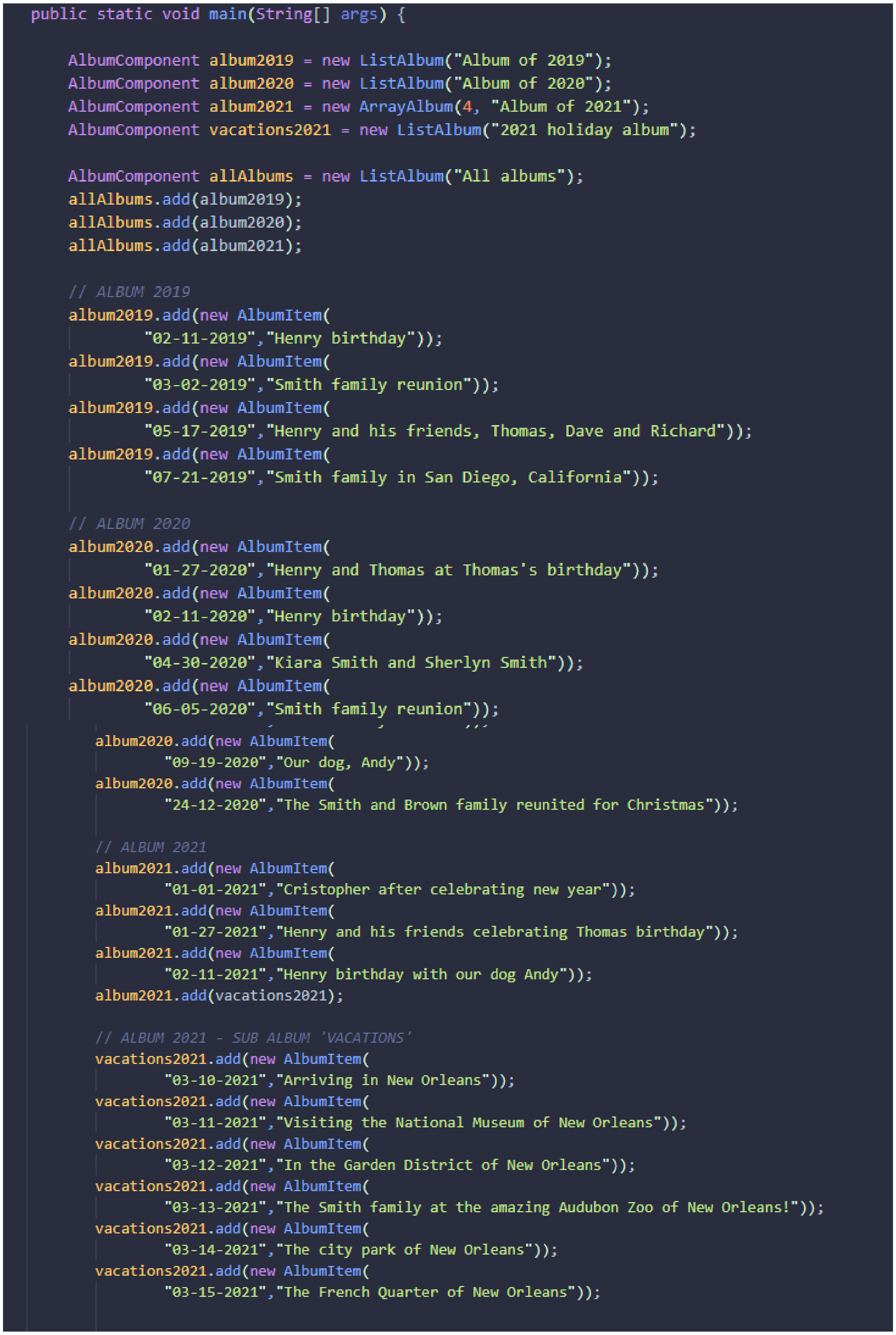
For the development of this program, we will have albums divided by year (2019,2020,2021), which will be composite objects, whether they are list-type or arrangement-type albums.

2019 and 2021 are list-type albums, which contain 'AlbumItem' type leaf objects that would be the photographs, with their corresponding date and description.

On the other hand, the 2020 album is an arrangement-type album, since it contains 3 photographs and a sub-album called 'vacations of 2021', which contains within it more leaf objects type 'AlbumItem' with their corresponding dates and descriptions.

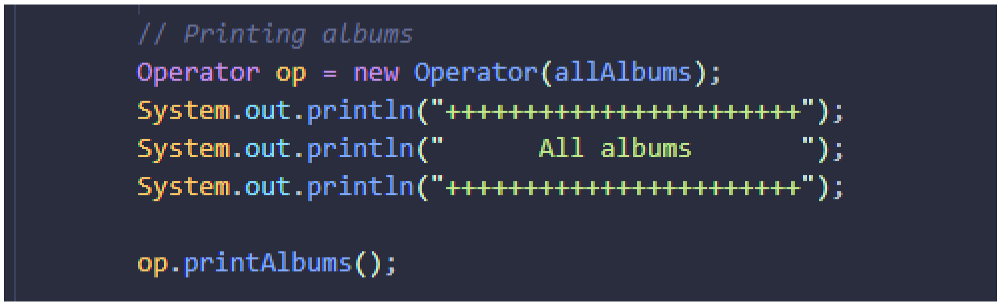
In addition, an album that contains all the albums to be able to go through them later.

Code in the next page.



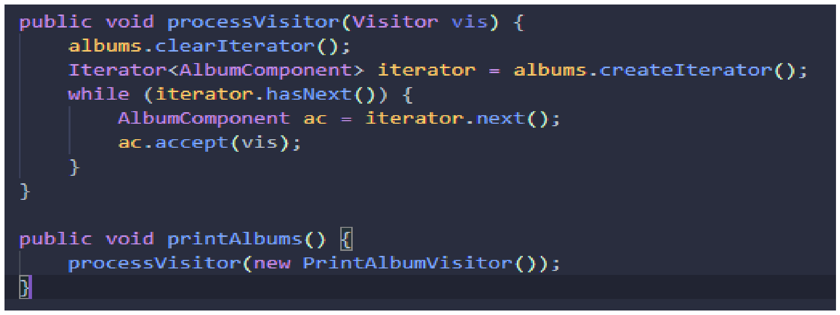
```java
public static void main(String[] args) {

    AlbumComponent album2019 = new ListAlbum("Album of 2019");
    AlbumComponent album2020 = new ListAlbum("Album of 2020");
    AlbumComponent album2021 = new ArrayAlbum(4, "Album of 2021");
    AlbumComponent vacations2021 = new ListAlbum("2021 holiday album");

    AlbumComponent allAlbums = new ListAlbum("All albums");
    allAlbums.add(album2019);
    allAlbums.add(album2020);
    allAlbums.add(album2021);

    // ALBUM 2019
    album2019.add(new AlbumItem(
            "02-11-2019","Henry birthday"));
    album2019.add(new AlbumItem(
            "03-02-2019","Smith family reunion"));
    album2019.add(new AlbumItem(
            "05-17-2019","Henry and his friends, Thomas, Dave and Richard"));
    album2019.add(new AlbumItem(
            "07-21-2019","Smith family in San Diego, California"));

    // ALBUM 2020
    album2020.add(new AlbumItem(
            "01-27-2020","Henry and Thomas at Thomas's birthday"));
    album2020.add(new AlbumItem(
            "02-11-2020","Henry birthday"));
    album2020.add(new AlbumItem(
            "04-30-2020","Kiara Smith and Sherlyn Smith"));
    album2020.add(new AlbumItem(
            "06-05-2020","Smith family reunion"));
    album2020.add(new AlbumItem(
            "09-19-2020","Our dog, Andy"));
    album2020.add(new AlbumItem(
            "24-12-2020","The Smith and Brown family reunited for Christmas"));

    // ALBUM 2021
    album2021.add(new AlbumItem(
            "01-01-2021","Cristopher after celebrating new year"));
    album2021.add(new AlbumItem(
            "01-27-2021","Henry and his friends celebrating Thomas birthday"));
    album2021.add(new AlbumItem(
            "02-11-2021","Henry birthday with our dog Andy"));
    album2021.add(vacations2021);

    // ALBUM 2021 - SUB ALBUM 'VACATIONS'
    vacations2021.add(new AlbumItem(
            "03-10-2021","Arriving in New Orleans"));
    vacations2021.add(new AlbumItem(
            "03-11-2021","Visiting the National Museum of New Orleans"));
    vacations2021.add(new AlbumItem(
            "03-12-2021","In the Garden District of New Orleans"));
    vacations2021.add(new AlbumItem(
            "03-13-2021","The Smith family at the amazing Audubon Zoo of New Orleans!"));
    vacations2021.add(new AlbumItem(
            "03-14-2021","The city park of New Orleans"));
    vacations2021.add(new AlbumItem(
            "03-15-2021","The French Quarter of New Orleans"));
```

Through the operator class (class to which we will send an album with all the albums), we are going to print all the albums



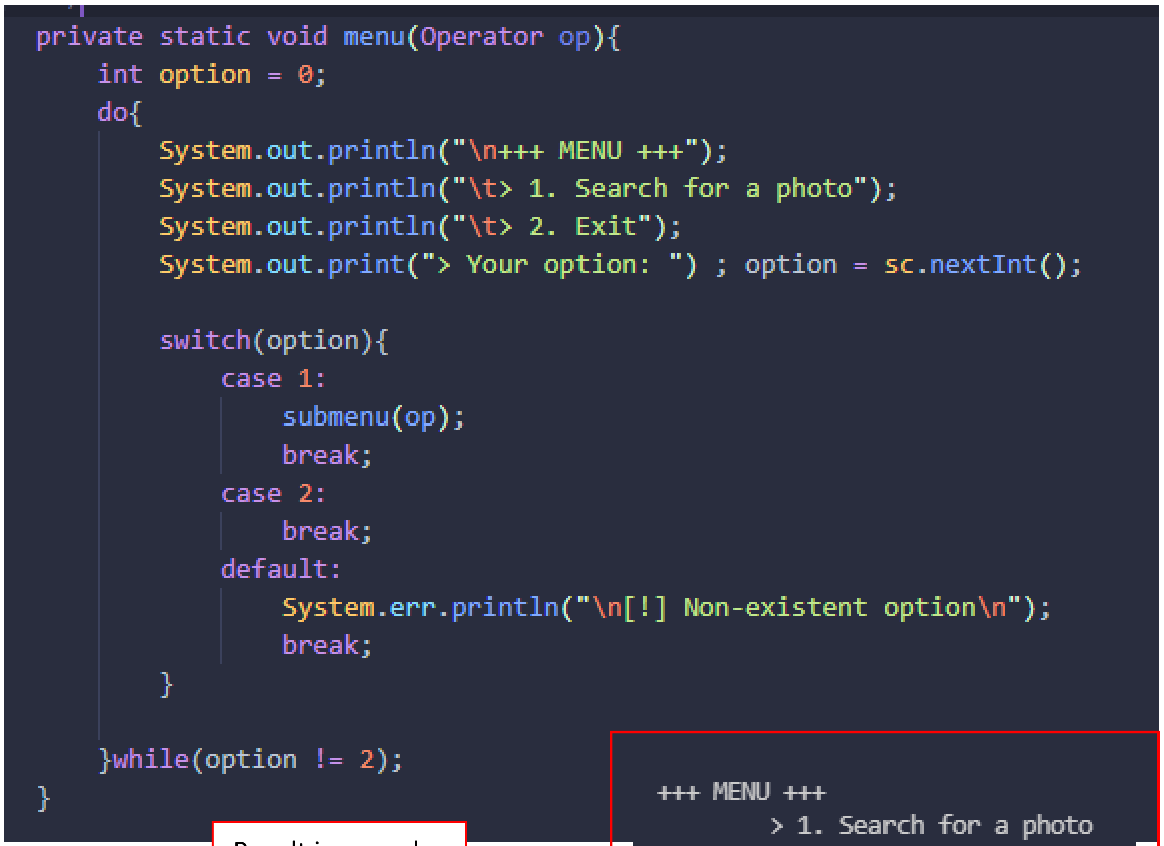
```
// Printing albums
Operator op = new Operator(allAlbums);
System.out.println("++++++++++++++++++++++");
System.out.println("      All albums      ");
System.out.println("++++++++++++++++++++++");

op.printAlbums();
```

Method 'printAlbums()' details, now thats use a processVisitor() method to print the álbum components with a Visitor



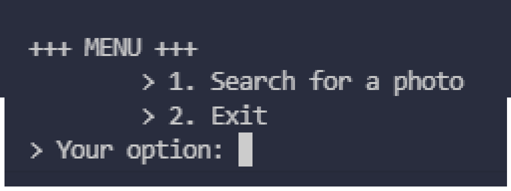
```
public void processVisitor(Visitor vis) {
    albums.clearIterator();
    Iterator<AlbumComponent> iterator = albums.createIterator();
    while (iterator.hasNext()) {
        AlbumComponent ac = iterator.next();
        ac.accept(vis);
    }
}

public void printAlbums() {
    processVisitor(new PrintAlbumVisitor());
}
```

And later, we send the user to a menu to search for photos or exit the program.



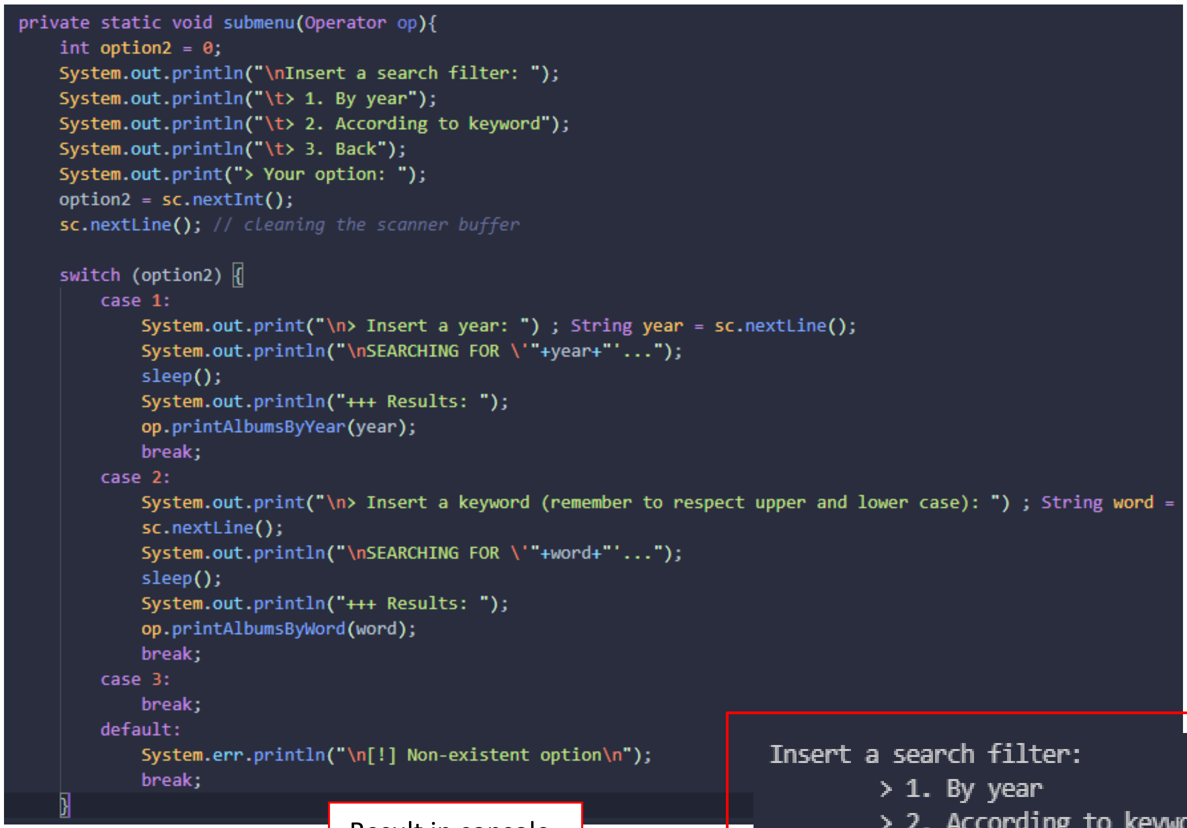
```
private static void menu(Operator op){
    int option = 0;
    do{
        System.out.println("\n+++ MENU +++");
        System.out.println("\t> 1. Search for a photo");
        System.out.println("\t> 2. Exit");
        System.out.print("> Your option: ") ; option = sc.nextInt();

        switch(option){
            case 1:
                submenu(op);
                break;
            case 2:
                break;
            default:
                System.err.println("\n[!] Non-existent option\n");
                break;
        }

    }while(option != 2);
}
```

Result in console



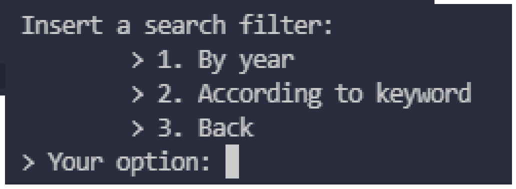
```
+++ MENU +++
        > 1. Search for a photo
        > 2. Exit
> Your option: 
```

If the user decides to search for a photograph, they are sent to a submenu asking them to enter a filter (search by year, or search by description).



```java
private static void submenu(Operator op){
    int option2 = 0;
    System.out.println("\nInsert a search filter: ");
    System.out.println("\t> 1. By year");
    System.out.println("\t> 2. According to keyword");
    System.out.println("\t> 3. Back");
    System.out.print("> Your option: ");
    option2 = sc.nextInt();
    sc.nextLine(); // cleaning the scanner buffer

    switch (option2) {
        case 1:
            System.out.print("\n> Insert a year: ") ; String year = sc.nextLine();
            System.out.println("\nSEARCHING FOR \'"+year+"'...");
            sleep();
            System.out.println("+++ Results: ");
            op.printAlbumsByYear(year);
            break;
        case 2:
            System.out.print("\n> Insert a keyword (remember to respect upper and lower case): ") ; String word =
            sc.nextLine();
            System.out.println("\nSEARCHING FOR \'"+word+"'...");
            sleep();
            System.out.println("+++ Results: ");
            op.printAlbumsByWord(word);
            break;
        case 3:
            break;
        default:
            System.err.println("\n[!] Non-existent option\n");
            break;
    }
```

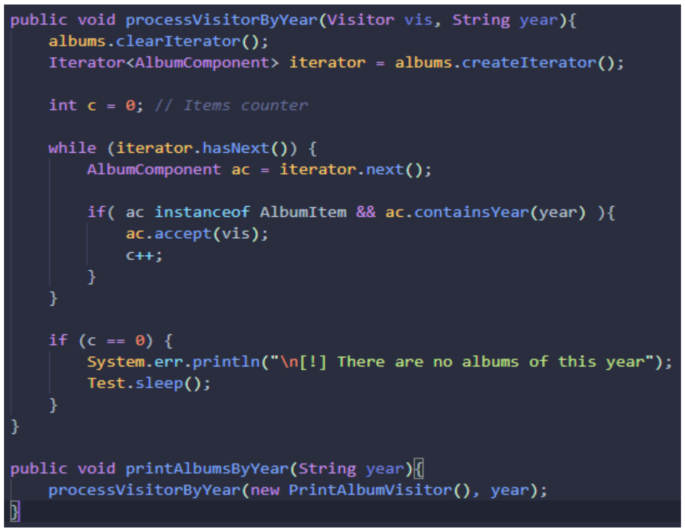
Result in console



```
Insert a search filter:
        > 1. By year
        > 2. According to keyword
        > 3. Back
> Your option: 
```

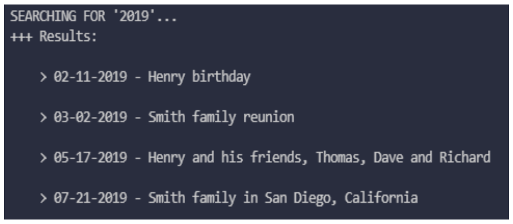
Following the instructions we can see that:

> If the user decides to search by year: he is asked to enter a year, which is read by a Scanner, and this variable is sent to the 'printAlbumByYear()' method of the Operator object.



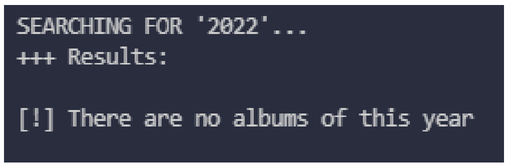
```java
public void processVisitorByYear(Visitor vis, String year){
    albums.clearIterator();
    Iterator<AlbumComponent> iterator = albums.createIterator();

    int c = 0; // Items counter

    while (iterator.hasNext()) {
        AlbumComponent ac = iterator.next();

        if( ac instanceof AlbumItem && ac.containsYear(year) ){
            ac.accept(vis);
            c++;
        }
    }

    if (c == 0) {
        System.err.println("\n[!] There are no albums of this year");
        Test.sleep();
    }
}

public void printAlbumsByYear(String year){
    processVisitorByYear(new PrintAlbumVisitor(), year);
}
```

If there are results, it is shown:



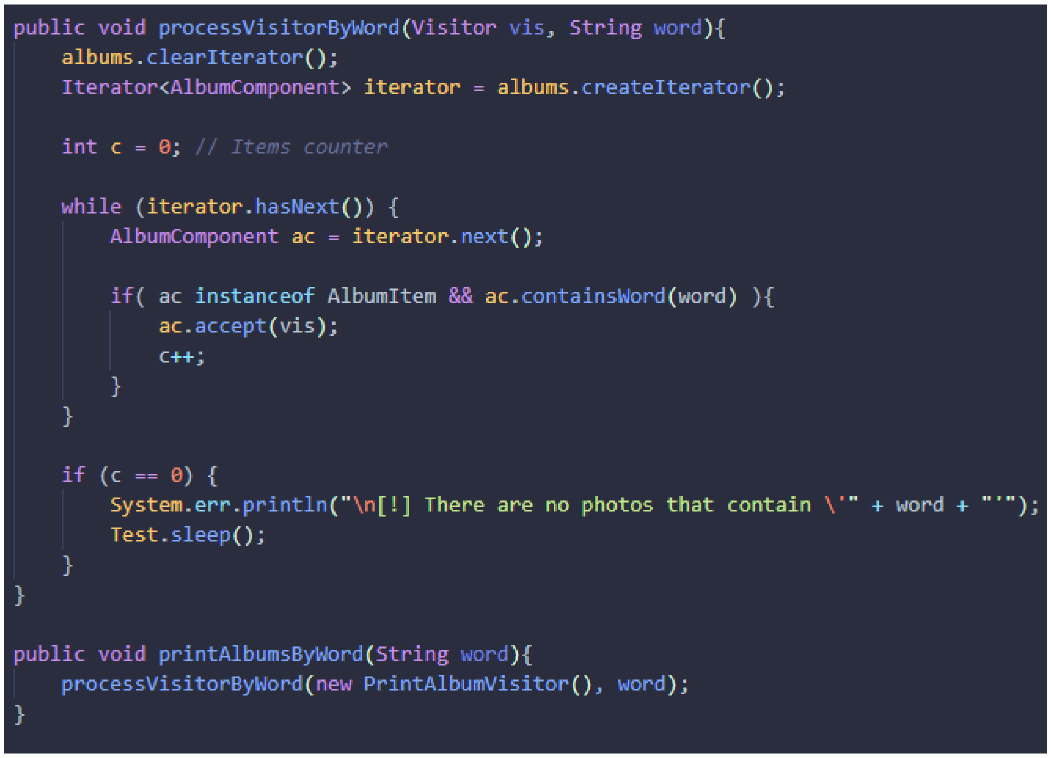
```
SEARCHING FOR '2019'...
+++ Results:

   > 02-11-2019 - Henry birthday

   > 03-02-2019 - Smith family reunion

   > 05-17-2019 - Henry and his friends, Thomas, Dave and Richard

   > 07-21-2019 - Smith family in San Diego, California
```

If there are no results we will receive:



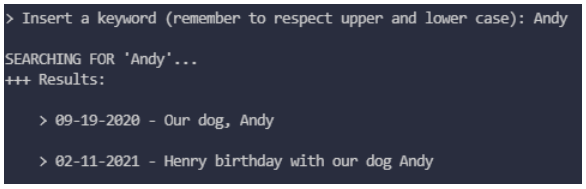
```
SEARCHING FOR '2022'...
+++ Results:

[!] There are no albums of this year
```

> If the user decides to search by keyword: the program will request a keyword and save it using a Scanner, then it will call the 'printAlbumsByWord()' method where it will search the descriptions of all the photographs to see which ones contain this entered keyword.



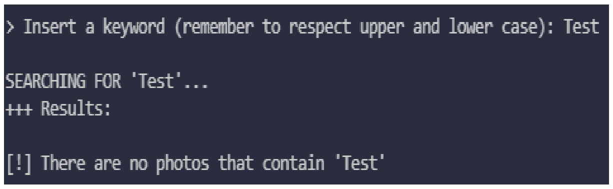
```java
public void processVisitorByWord(Visitor vis, String word){
    albums.clearIterator();
    Iterator<AlbumComponent> iterator = albums.createIterator();

    int c = 0; // Items counter

    while (iterator.hasNext()) {
        AlbumComponent ac = iterator.next();

        if( ac instanceof AlbumItem && ac.containsWord(word) ){
            ac.accept(vis);
            c++;
        }
    }

    if (c == 0) {
        System.err.println("\n[!] There are no photos that contain \'" + word + "'");
        Test.sleep();
    }
}

public void printAlbumsByWord(String word){
    processVisitorByWord(new PrintAlbumVisitor(), word);
}
```

If there are results, it is shown:



```
> Insert a keyword (remember to respect upper and lower case): Andy

SEARCHING FOR 'Andy'...
+++ Results:

    > 09-19-2020 - Our dog, Andy

    > 02-11-2021 - Henry birthday with our dog Andy
```

If there are no results we will receive:

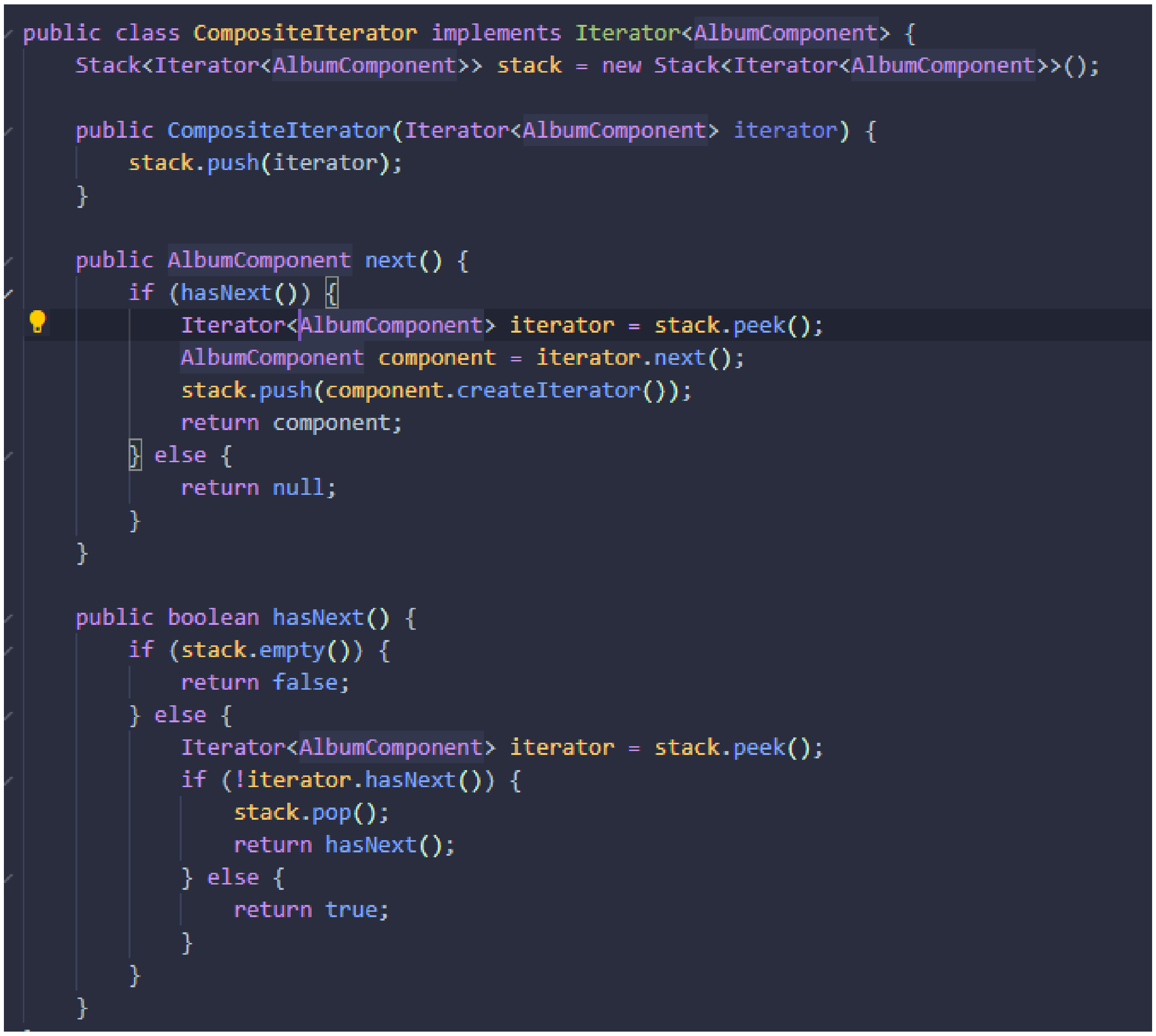


```
> Insert a keyword (remember to respect upper and lower case): Test

SEARCHING FOR 'Test'...
+++ Results:

[!] There are no photos that contain 'Test'
```

In this way the program allows you to search for photographs.

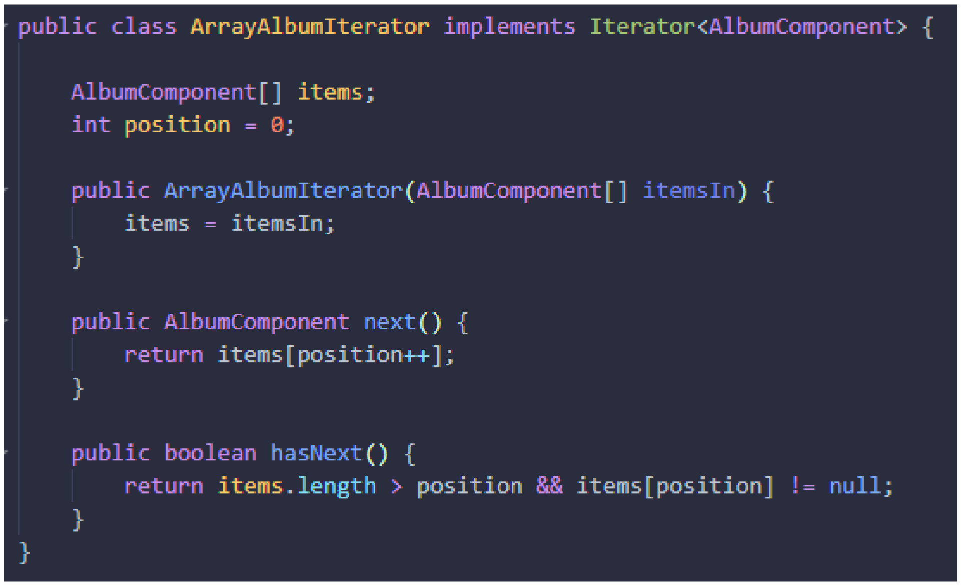
Now, we will review the 'CompositeIterator' class by which we can iterate through all existing lists.

Code in the next page.



```java
public class CompositeIterator implements Iterator<AlbumComponent> {
    Stack<Iterator<AlbumComponent>> stack = new Stack<Iterator<AlbumComponent>>();

    public CompositeIterator(Iterator<AlbumComponent> iterator) {
        stack.push(iterator);
    }

    public AlbumComponent next() {
        if (hasNext()) {
            Iterator<AlbumComponent> iterator = stack.peek();
            AlbumComponent component = iterator.next();
            stack.push(component.createIterator());
            return component;
        } else {
            return null;
        }
    }

    public boolean hasNext() {
        if (stack.empty()) {
            return false;
        } else {
            Iterator<AlbumComponent> iterator = stack.peek();
            if (!iterator.hasNext()) {
                stack.pop();
                return hasNext();
            } else {
                return true;
            }
        }
    }
```

Which in turn has a special iterator for array-type albums.



```java
public class ArrayAlbumIterator implements Iterator<AlbumComponent> {

    AlbumComponent[] items;
    int position = 0;

    public ArrayAlbumIterator(AlbumComponent[] itemsIn) {
        items = itemsIn;
    }

    public AlbumComponent next() {
        return items[position++];
    }

    public boolean hasNext() {
        return items.length > position && items[position] != null;
    }
}
```

Since they implement the well-known Java 'Iterator' interface, we already know how they work.

FINISH.