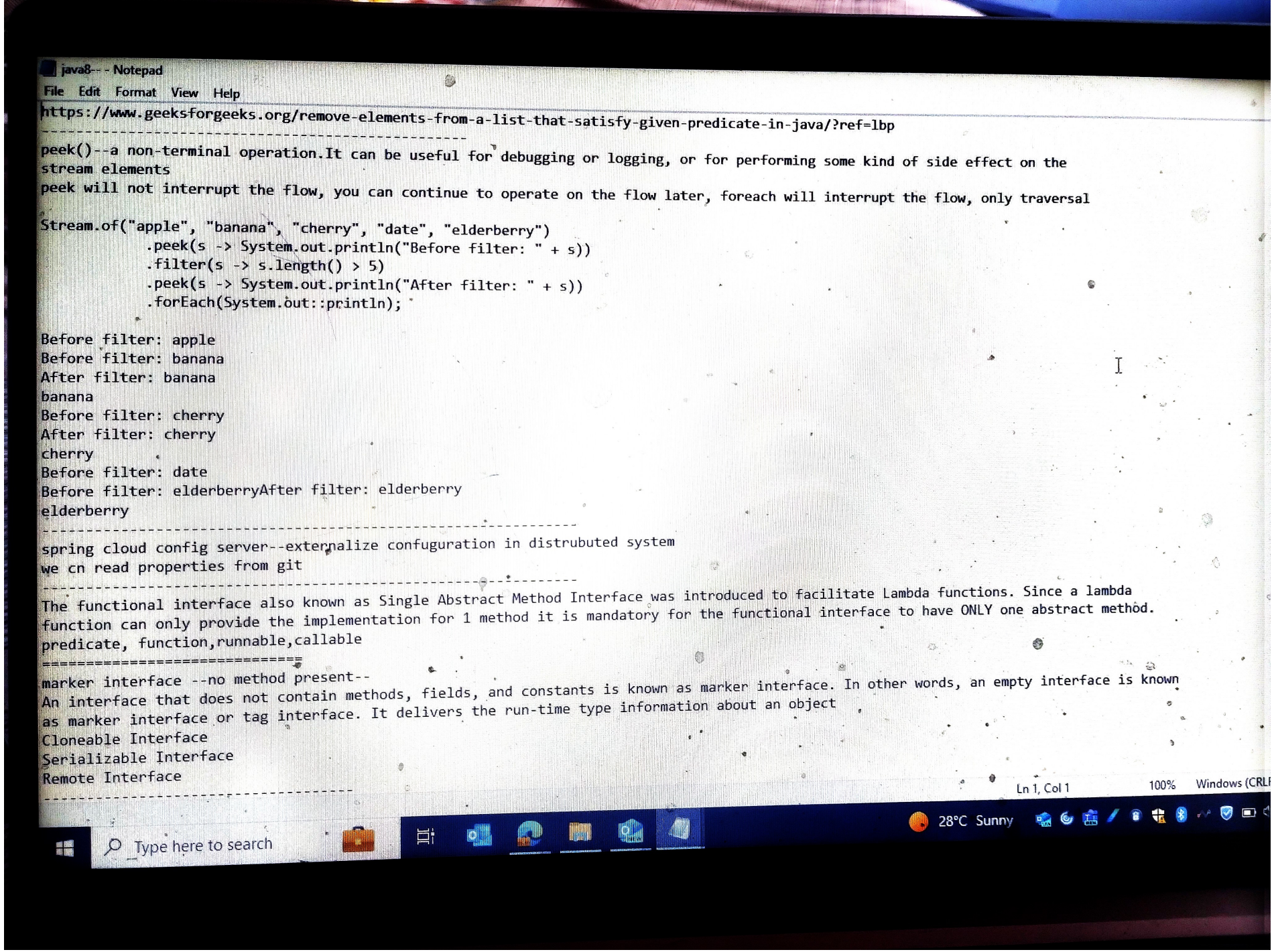
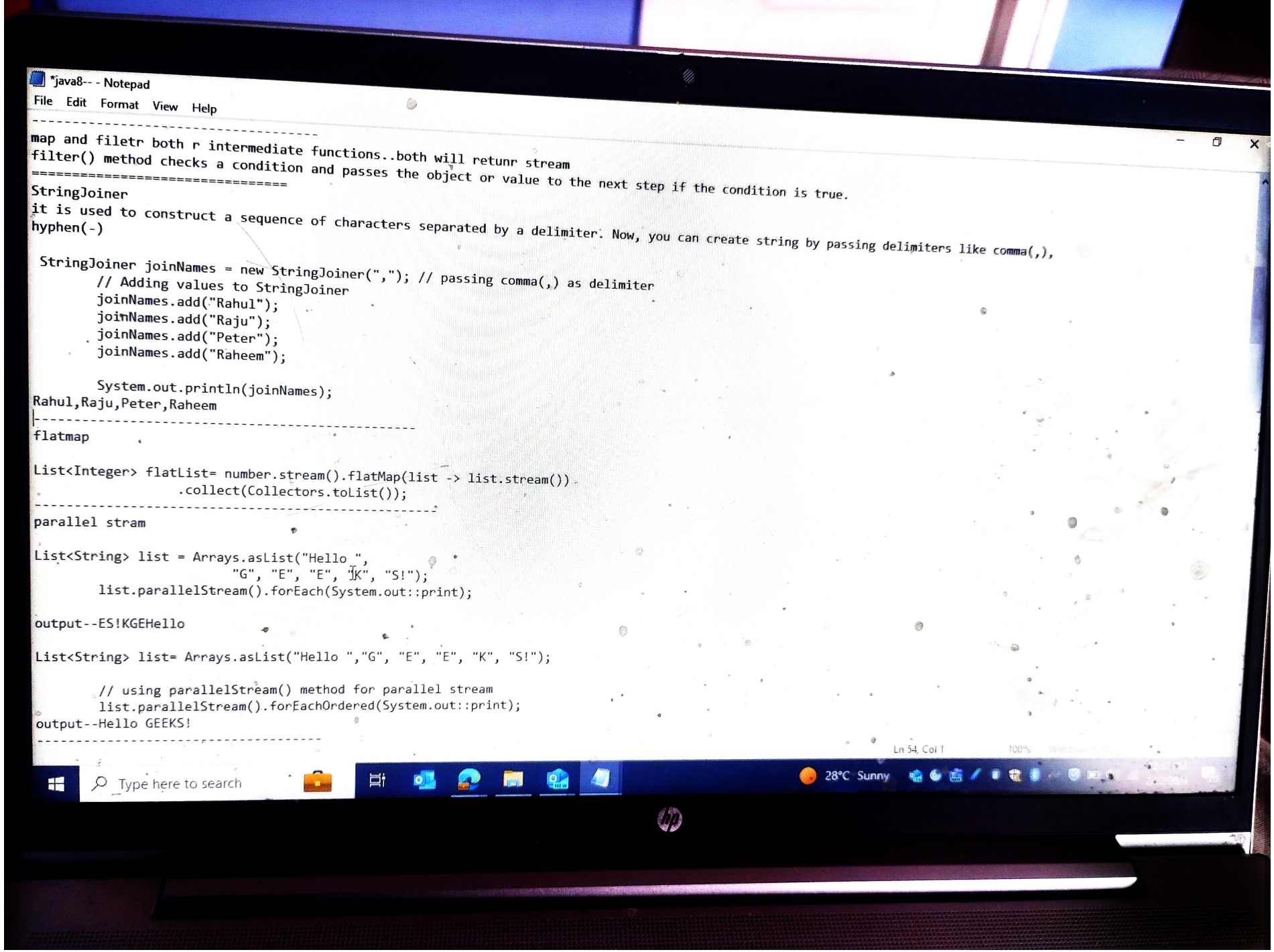




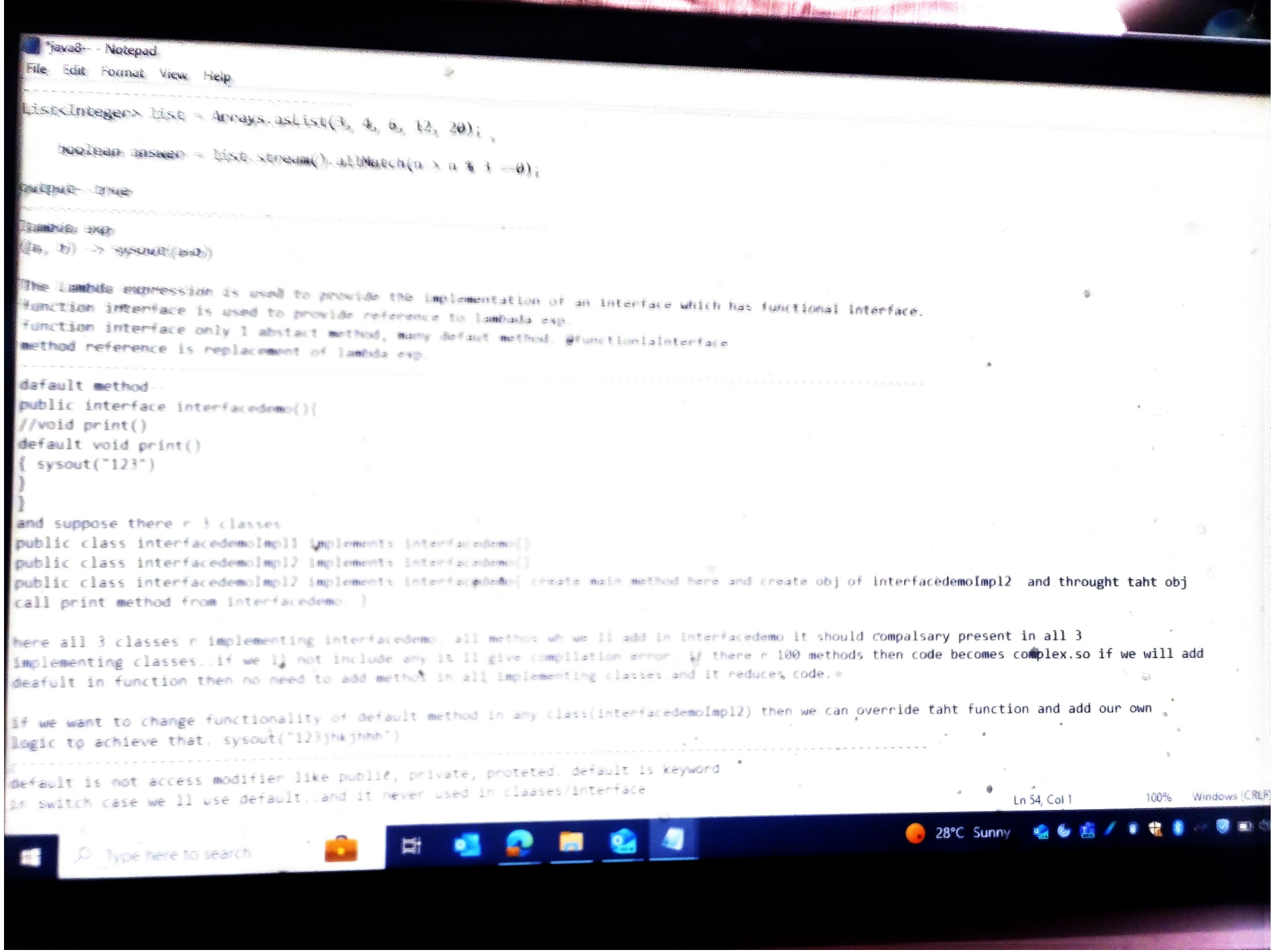
*java8-- - Notepad

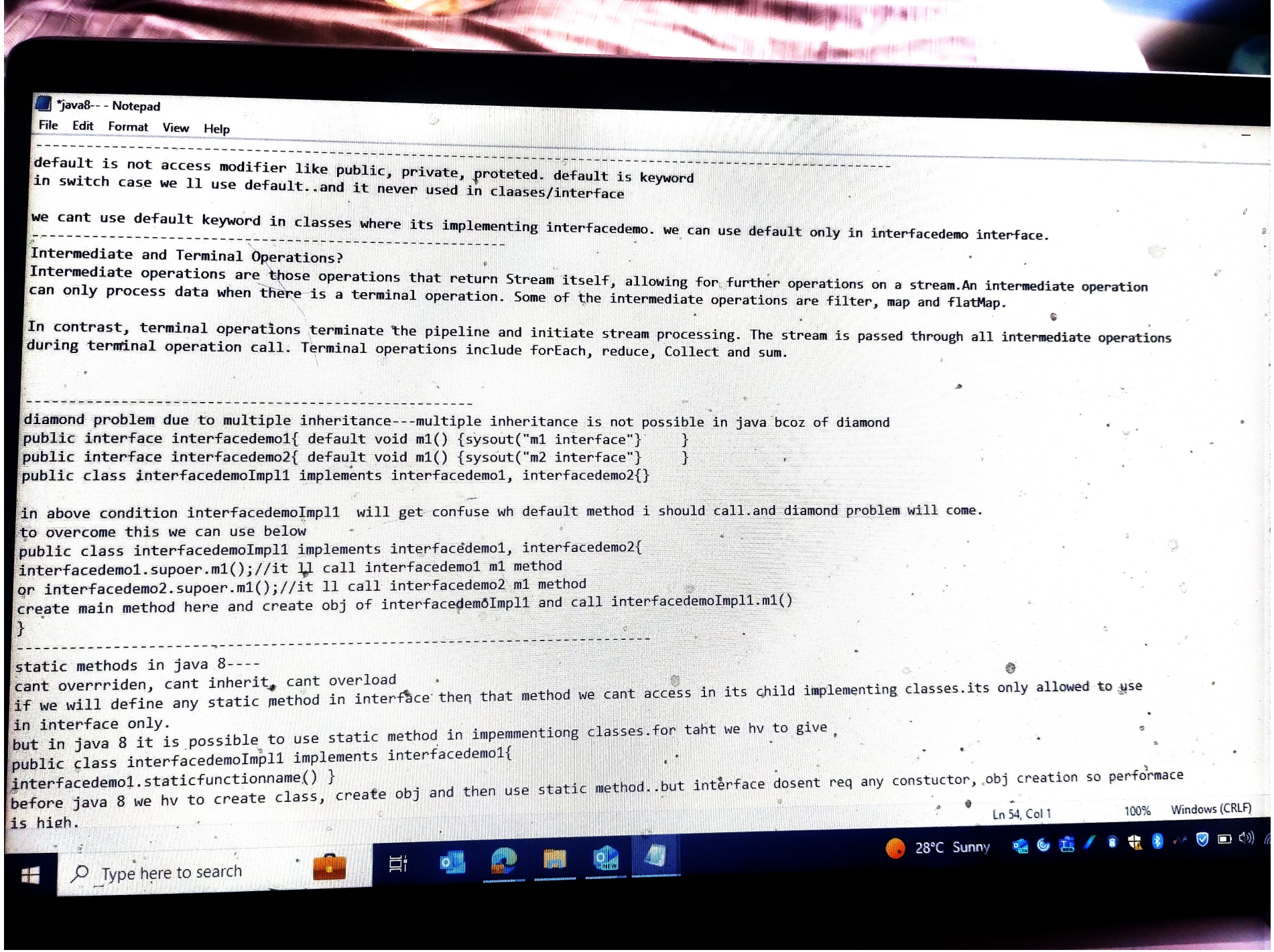
File Edit Format View Help

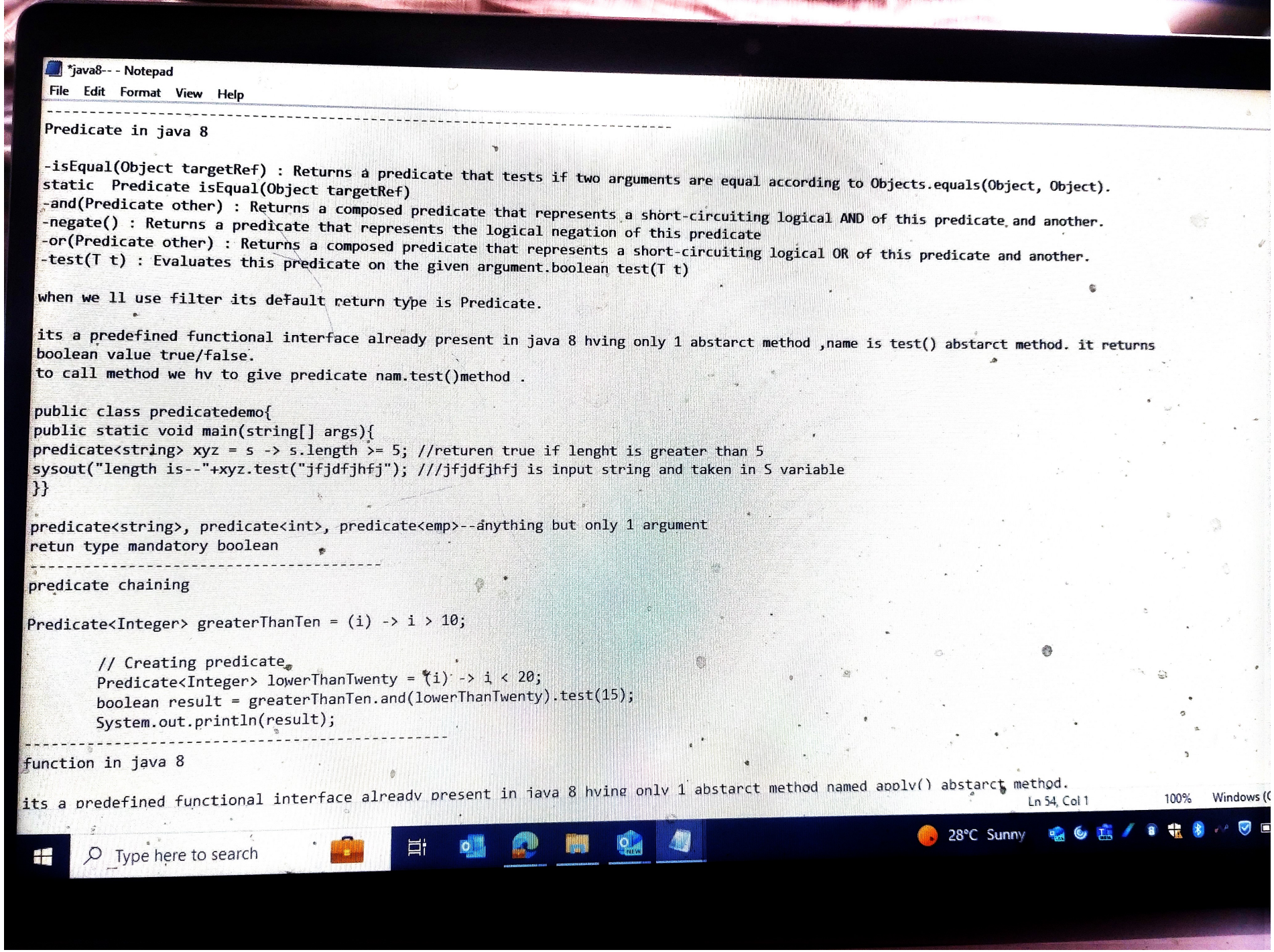
```
-----------------------------------
map and filetr both r intermediate functions..both will retunr stream
filter() method checks a condition and passes the object or value to the next step if the condition is true.
================================
StringJoiner
it is used to construct a sequence of characters separated by a delimiter. Now, you can create string by passing delimiters like comma(,),
hyphen(-)

 StringJoiner joinNames = new StringJoiner(","); // passing comma(,) as delimiter
        // Adding values to StringJoiner
        joinNames.add("Rahul");
        joinNames.add("Raju");
        joinNames.add("Peter");
        joinNames.add("Raheem");

        System.out.println(joinNames);
Rahul,Raju,Peter,Raheem
-----------------------------------------------
flatmap

List<Integer> flatList= number.stream().flatMap(list -> list.stream())
                .collect(Collectors.toList());
-----------------------------------------------------
parallel stram

List<String> list = Arrays.asList("Hello ",
                        "G", "E", "E", "K", "S!");
        list.parallelStream().forEach(System.out::print);

output--ES!KGEHello

List<String> list= Arrays.asList("Hello ","G", "E", "E", "K", "S!");

        // using parallelStream() method for parallel stream
        list.parallelStream().forEachOrdered(System.out::print);
output--Hello GEEKS!
---------------------------------------
```

Ln 54, Col 1 100%

Type here to search 28°C Sunny





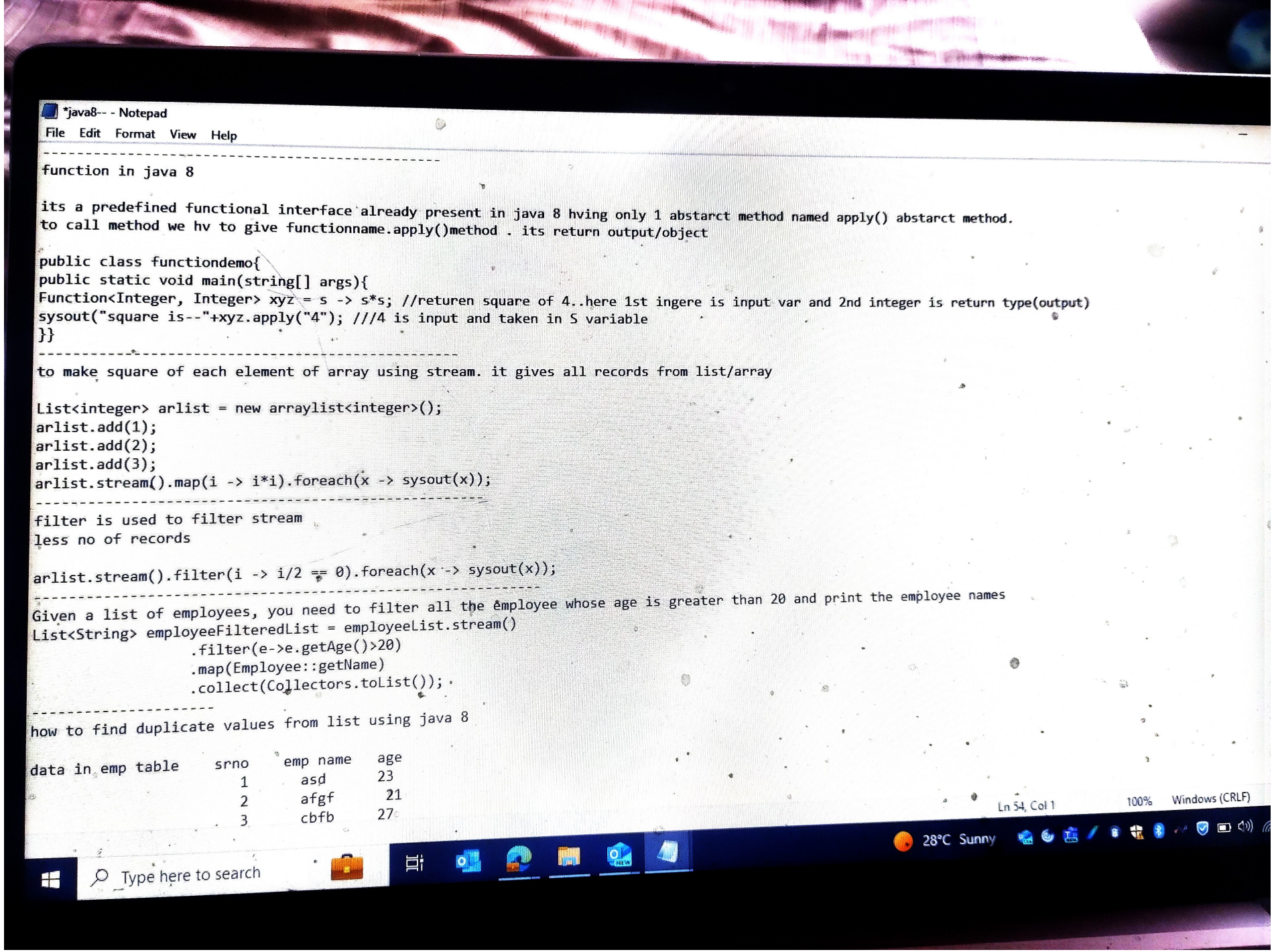


*java8-- - Notepad
File Edit Format View Help

```
--------------------------------------------------------------------------------
Predicate in java 8

-isEqual(Object targetRef) : Returns a predicate that tests if two arguments are equal according to Objects.equals(Object, Object).
static  Predicate isEqual(Object targetRef)
-and(Predicate other) : Returns a composed predicate that represents a short-circuiting logical AND of this predicate and another.
-negate() : Returns a predicate that represents the logical negation of this predicate
-or(Predicate other) : Returns a composed predicate that represents a short-circuiting logical OR of this predicate and another.
-test(T t) : Evaluates this predicate on the given argument.boolean test(T t)

when we ll use filter its default return type is Predicate.

its a predefined functional interface already present in java 8 hving only 1 abstarct method ,name is test() abstarct method. it returns
boolean value true/false.
to call method we hv to give predicate nam.test()method .

public class predicatedemo{
public static void main(string[] args){
predicate<string> xyz = s -> s.length >= 5; //returen true if lenght is greater than 5
sysout("length is--"+xyz.test("jfjdfjhfj"); ///jfjdfjhfj is input string and taken in S variable
}}

predicate<string>, predicate<int>, predicate<emp>--anything but only 1 argument
retun type mandatory boolean
----------------------------------------
predicate chaining

Predicate<Integer> greaterThanTen = (i) -> i > 10;

        // Creating predicate
        Predicate<Integer> lowerThanTwenty = (i) -> i < 20;
        boolean result = greaterThanTen.and(lowerThanTwenty).test(15);
        System.out.println(result);
------------------------------------------------
function in java 8

its a predefined functional interface already present in java 8 hving only 1 abstarct method named apply() abstarct method.
```

Ln 54, Col 1 100% Windows (C

Type here to search 28°C Sunny



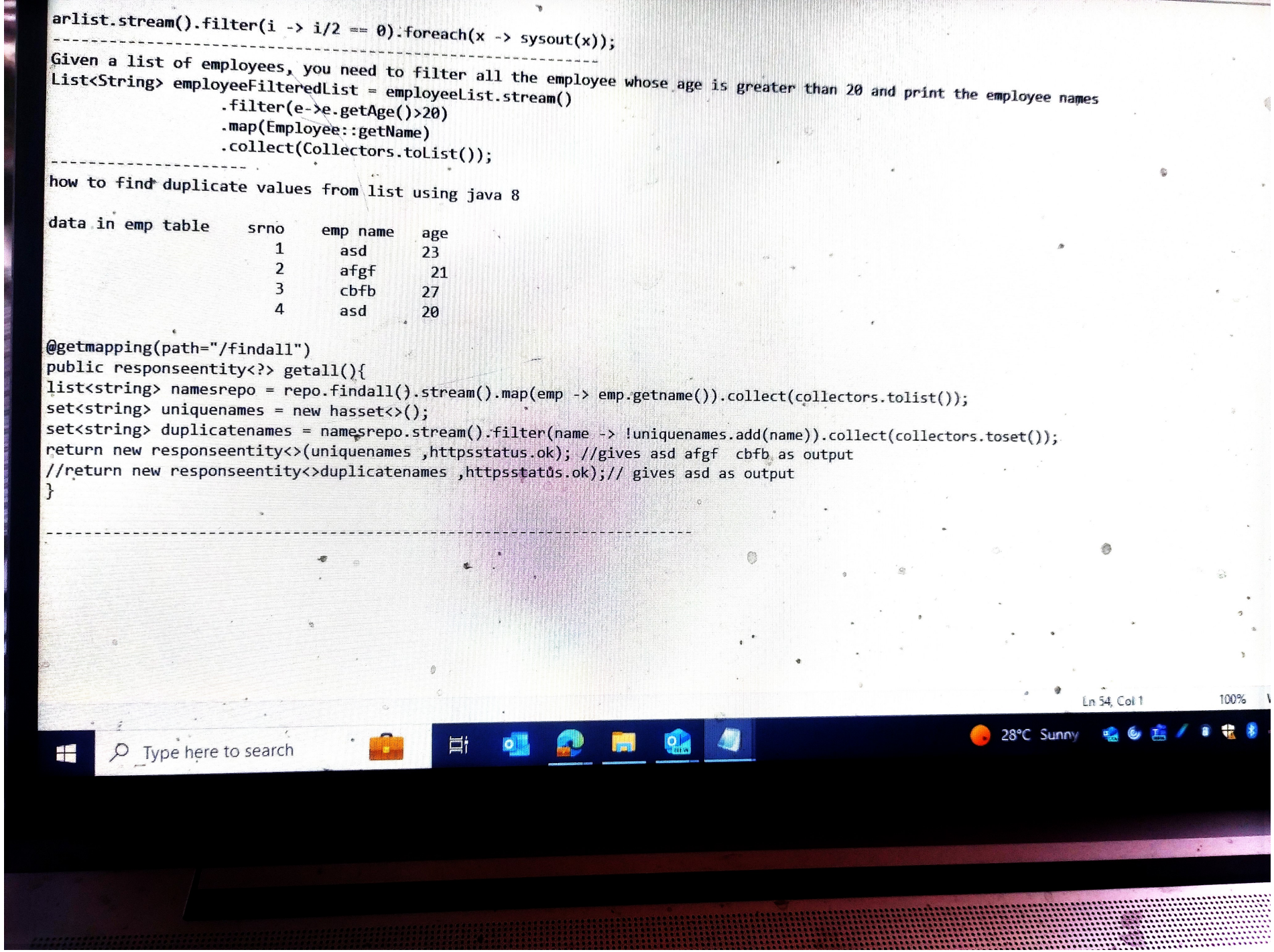
*java8-- - Notepad

File Edit Format View Help

```
-------------------------------------------------
function in java 8

its a predefined functional interface already present in java 8 hving only 1 abstarct method named apply() abstarct method.
to call method we hv to give functionname.apply()method . its return output/object

public class functiondemo{
public static void main(string[] args){
Function<Integer, Integer> xyz = s -> s*s; //returen square of 4..here 1st ingere is input var and 2nd integer is return type(output)
sysout("square is--"+xyz.apply("4"); ///4 is input and taken in S variable
}}
-------------------------------------------------
to make square of each element of array using stream. it gives all records from list/array

List<integer> arlist = new arraylist<integer>();
arlist.add(1);
arlist.add(2);
arlist.add(3);
arlist.stream().map(i -> i*i).foreach(x -> sysout(x));
-------------------------------------------------
filter is used to filter stream
less no of records

arlist.stream().filter(i -> i/2 == 0).foreach(x -> sysout(x));
-------------------------------------------------------------
Given a list of employees, you need to filter all the employee whose age is greater than 20 and print the employee names
List<String> employeeFilteredList = employeeList.stream()
                .filter(e->e.getAge()>20)
                .map(Employee::getName)
                .collect(Collectors.toList());
---------------------
how to find duplicate values from list using java 8

data in emp table    srno    emp name    age
                       1       asd       23
                       2       afgf       21
                       3       cbfb      27
```

Ln 54, Col 1 100% Windows (CRLF)

Type here to search 28°C Sunny



```
arlist.stream().filter(i -> i/2 == 0).foreach(x -> sysout(x));
-----------------------------------------------------------
Given a list of employees, you need to filter all the employee whose age is greater than 20 and print the employee names
List<String> employeeFilteredList = employeeList.stream()
                    .filter(e->e.getAge()>20)
                    .map(Employee::getName)
                    .collect(Collectors.toList());
---------------------
how to find duplicate values from list using java 8

data in emp table     srno     emp name    age
                        1        asd       23
                        2        afgf       21
                        3        cbfb      27
                        4        asd       20

@getmapping(path="/findall")
public responseentity<?> getall(){
list<string> namesrepo = repo.findall().stream().map(emp -> emp.getname()).collect(collectors.tolist());
set<string> uniquenames = new hasset<>();
set<string> duplicatenames = namesrepo.stream().filter(name -> !uniquenames.add(name)).collect(collectors.toset());
return new responseentity<>(uniquenames ,httpsstatus.ok); //gives asd afgf  cbfb as output
//return new responseentity<>duplicatenames ,httpsstatus.ok);// gives asd as output
}

-------------------------------------------------------------------------
```