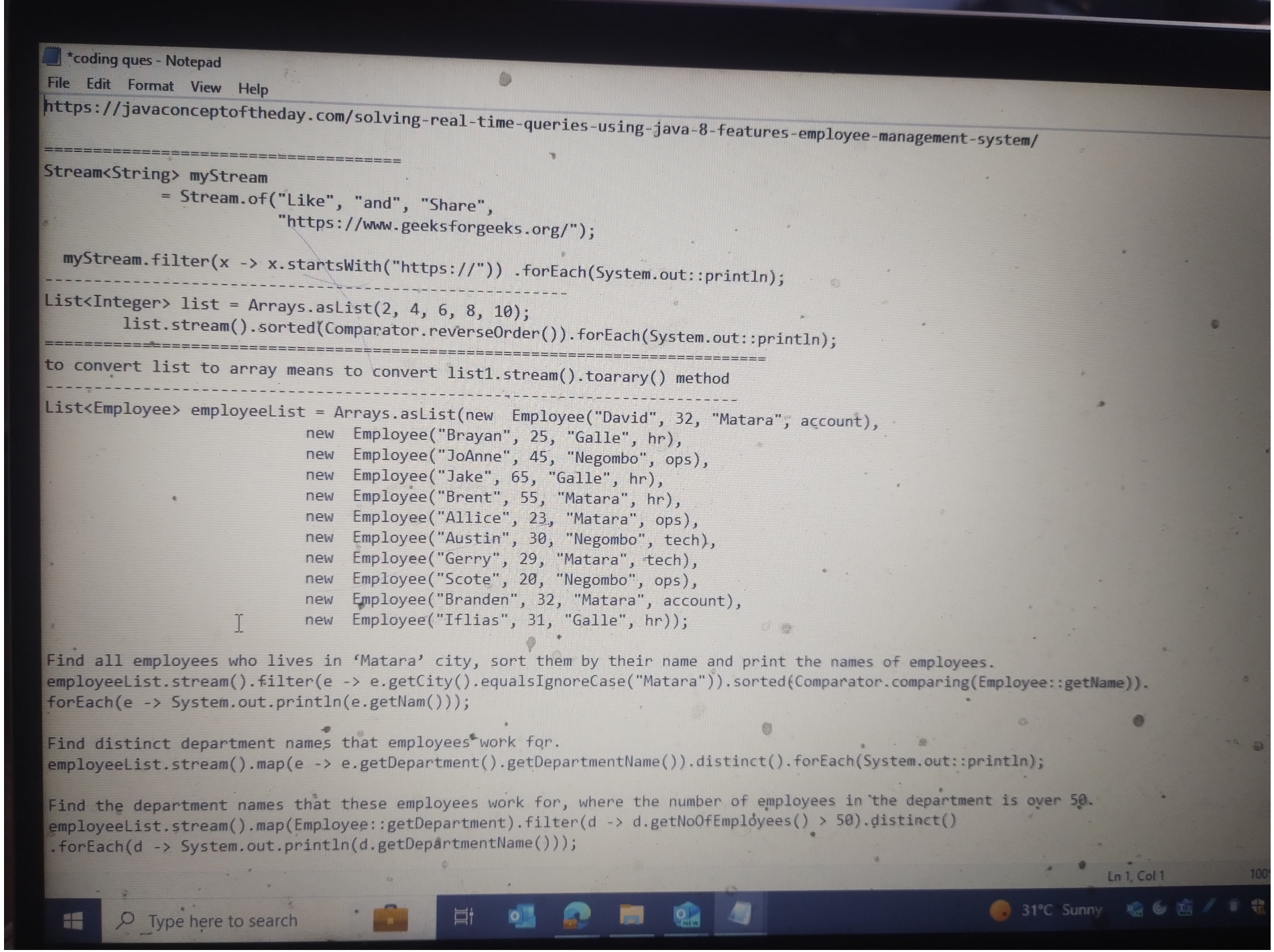


*coding ques - Notepad

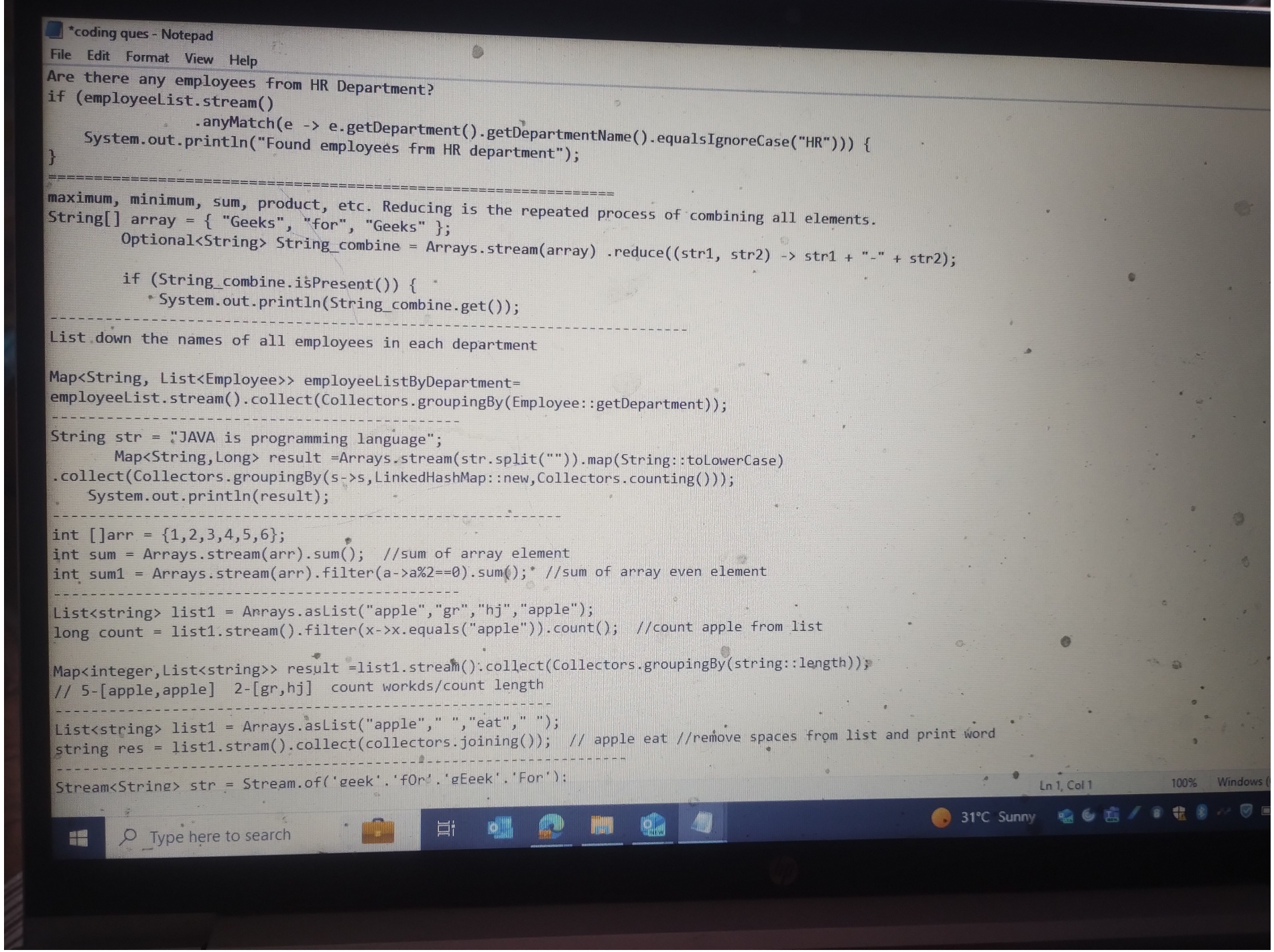
File Edit Format View Help

```
https://javaconceptoftheday.com/solving-real-time-queries-using-java-8-features-employee-management-system/

====================================
Stream<String> myStream
          = Stream.of("Like", "and", "Share",
                      "https://www.geeksforgeeks.org/");

  myStream.filter(x -> x.startsWith("https://")) .forEach(System.out::println);
-----------------------------------------------
List<Integer> list = Arrays.asList(2, 4, 6, 8, 10);
       list.stream().sorted(Comparator.reverseOrder()).forEach(System.out::println);
=================================================================
to convert list to array means to convert list1.stream().toarary() method
-----------------------------------------------------------------
List<Employee> employeeList = Arrays.asList(new  Employee("David", 32, "Matara", account),
                        new  Employee("Brayan", 25, "Galle", hr),
                        new  Employee("JoAnne", 45, "Negombo", ops),
                        new  Employee("Jake", 65, "Galle", hr),
                        new  Employee("Brent", 55, "Matara", hr),
                        new  Employee("Allice", 23, "Matara", ops),
                        new  Employee("Austin", 30, "Negombo", tech),
                        new  Employee("Gerry", 29, "Matara", tech),
                        new  Employee("Scote", 20, "Negombo", ops),
                        new  Employee("Branden", 32, "Matara", account),
                        new  Employee("Iflias", 31, "Galle", hr));

Find all employees who lives in 'Matara' city, sort them by their name and print the names of employees.
employeeList.stream().filter(e -> e.getCity().equalsIgnoreCase("Matara")).sorted(Comparator.comparing(Employee::getName)).
forEach(e -> System.out.println(e.getNam()));

Find distinct department names that employees work for.
employeeList.stream().map(e -> e.getDepartment().getDepartmentName()).distinct().forEach(System.out::println);

Find the department names that these employees work for, where the number of employees in the department is over 50.
employeeList.stream().map(Employee::getDepartment).filter(d -> d.getNoOfEmployees() > 50).distinct()
.forEach(d -> System.out.println(d.getDepartmentName()));
```

Ln 1, Col 1

Type here to search

31°C Sunny



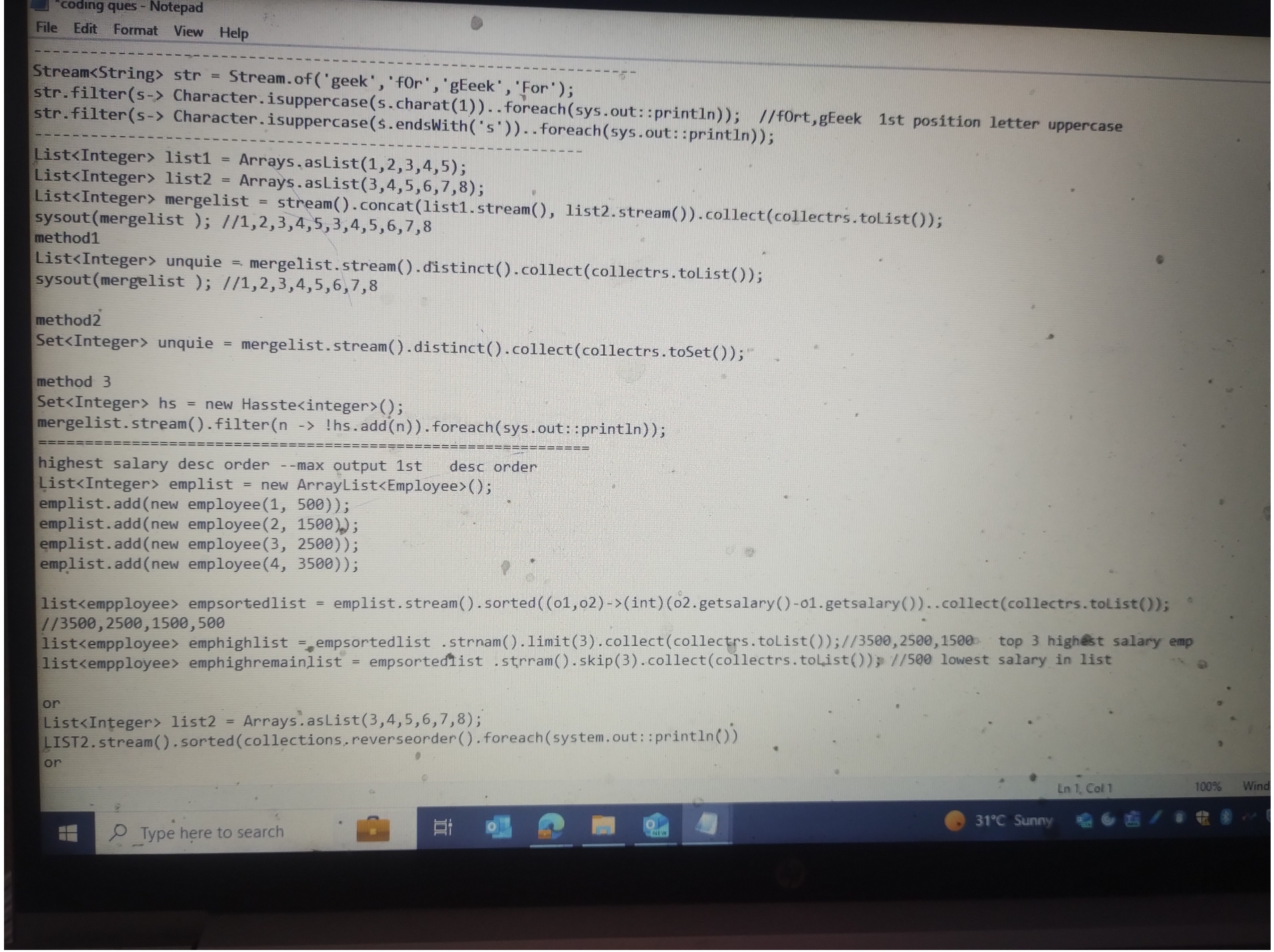
*coding ques - Notepad

File Edit Format View Help

```
Are there any employees from HR Department?
if (employeeList.stream()
                .anyMatch(e -> e.getDepartment().getDepartmentName().equalsIgnoreCase("HR"))) {
    System.out.println("Found employees frm HR department");
}
================================================================
maximum, minimum, sum, product, etc. Reducing is the repeated process of combining all elements.
String[] array = { "Geeks", "for", "Geeks" };
        Optional<String> String_combine = Arrays.stream(array) .reduce((str1, str2) -> str1 + "-" + str2);

        if (String_combine.isPresent()) {
            System.out.println(String_combine.get());
----------------------------------------------------------------------
List down the names of all employees in each department

Map<String, List<Employee>> employeeListByDepartment=
employeeList.stream().collect(Collectors.groupingBy(Employee::getDepartment));
-----------------------------------------
String str = "JAVA is programming language";
       Map<String,Long> result =Arrays.stream(str.split("")).map(String::toLowerCase)
.collect(Collectors.groupingBy(s->s,LinkedHashMap::new,Collectors.counting()));
    System.out.println(result);
-----------------------------------------------------
int []arr = {1,2,3,4,5,6};
int sum = Arrays.stream(arr).sum();  //sum of array element
int sum1 = Arrays.stream(arr).filter(a->a%2==0).sum(); //sum of array even element
-------------------------------------------
List<string> list1 = Arrays.asList("apple","gr","hj","apple");
long count = list1.stream().filter(x->x.equals("apple")).count();  //count apple from list

Map<integer,List<string>> result =list1.stream().collect(Collectors.groupingBy(string::length));
// 5-[apple,apple]  2-[gr,hj]  count workds/count length
-----------------------------------------------------
List<string> list1 = Arrays.asList("apple"," ","eat"," ");
string res = list1.stram().collect(collectors.joining());  // apple eat //remove spaces from list and print word
-----------------------------------------------------------------
Stream<String> str = Stream.of('geek','fOr','gEeek','For');
```

Ln 1, Col 1 100% Windows

Type here to search 31°C Sunny

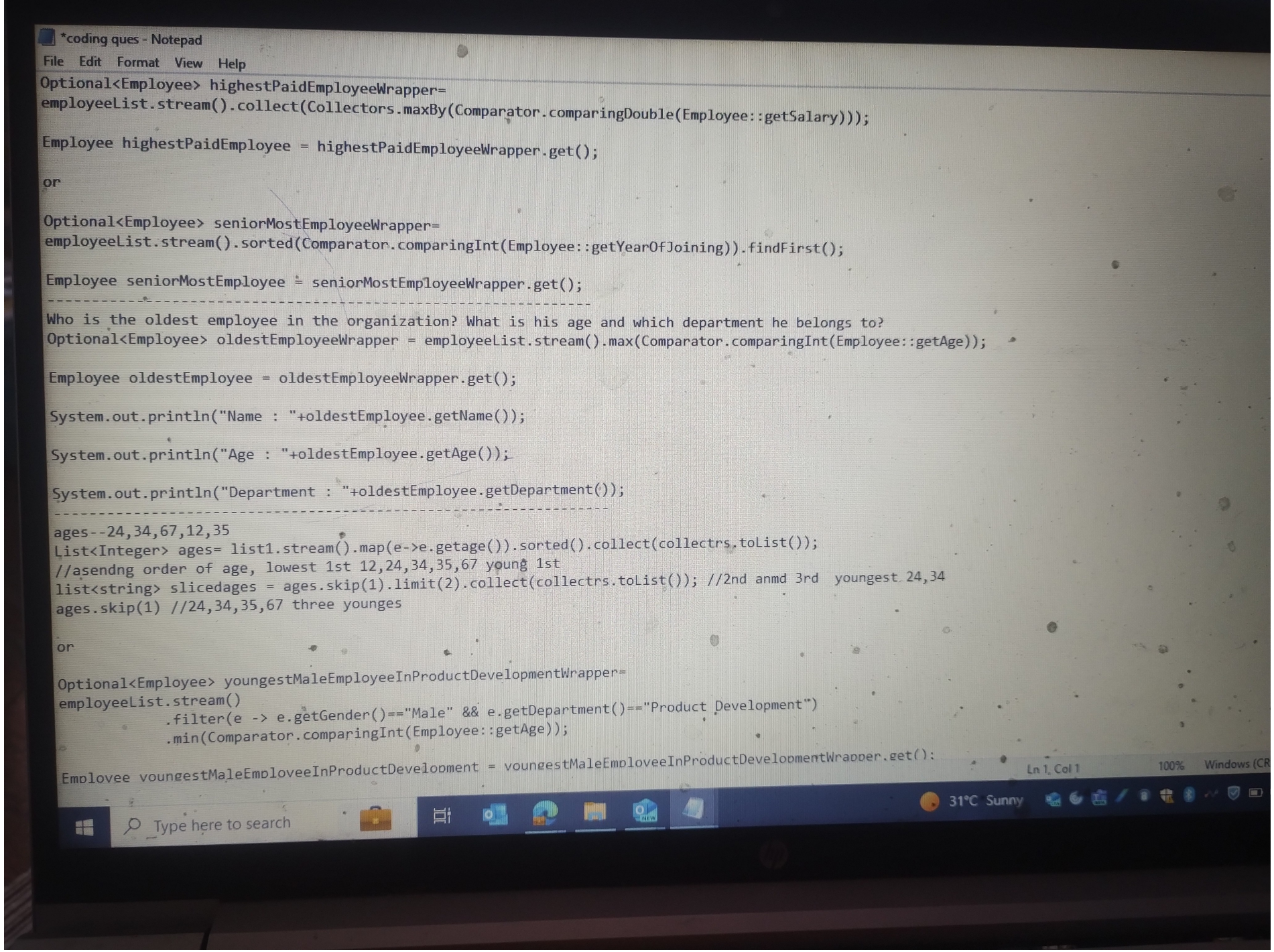


*coding ques - Notepad
File Edit Format View Help

```
-----------------------------------------------------------
Stream<String> str = Stream.of('geek','fOr','gEeek','For');
str.filter(s-> Character.isuppercase(s.charat(1))..foreach(sys.out::println));  //fOrt,gEeek  1st position letter uppercase
str.filter(s-> Character.isuppercase($.endsWith('s'))..foreach(sys.out::println));
-----------------------------------------------------------
List<Integer> list1 = Arrays.asList(1,2,3,4,5);
List<Integer> list2 = Arrays.asList(3,4,5,6,7,8);
List<Integer> mergelist = stream().concat(list1.stream(), list2.stream()).collect(collectrs.toList());
sysout(mergelist ); //1,2,3,4,5,3,4,5,6,7,8
method1
List<Integer> unquie = mergelist.stream().distinct().collect(collectrs.toList());
sysout(mergelist ); //1,2,3,4,5,6,7,8

method2
Set<Integer> unquie = mergelist.stream().distinct().collect(collectrs.toSet());

method 3
Set<Integer> hs = new Hasste<integer>();
mergelist.stream().filter(n -> !hs.add(n)).foreach(sys.out::println));
============================================================
highest salary desc order --max output 1st   desc order
List<Integer> emplist = new ArrayList<Employee>();
emplist.add(new employee(1, 500));
emplist.add(new employee(2, 1500));
emplist.add(new employee(3, 2500));
emplist.add(new employee(4, 3500));

list<empployee> empsortedlist = emplist.stream().sorted((o1,o2)->(int)(o2.getsalary()-o1.getsalary())..collect(collectrs.toList());
//3500,2500,1500,500
list<empployee> emphighlist = empsortedlist .strnam().limit(3).collect(collectrs.toList());//3500,2500,1500   top 3 highest salary emp
list<empployee> emphighremainlist = empsortedlist .strram().skip(3).collect(collectrs.toList()); //500 lowest salary in list

or
List<Integer> list2 = Arrays.asList(3,4,5,6,7,8);
LIST2.stream().sorted(collections.reverseorder().foreach(system.out::println())
or
```

Ln 1, Col 1 100% Wind

Type here to search 31°C Sunny



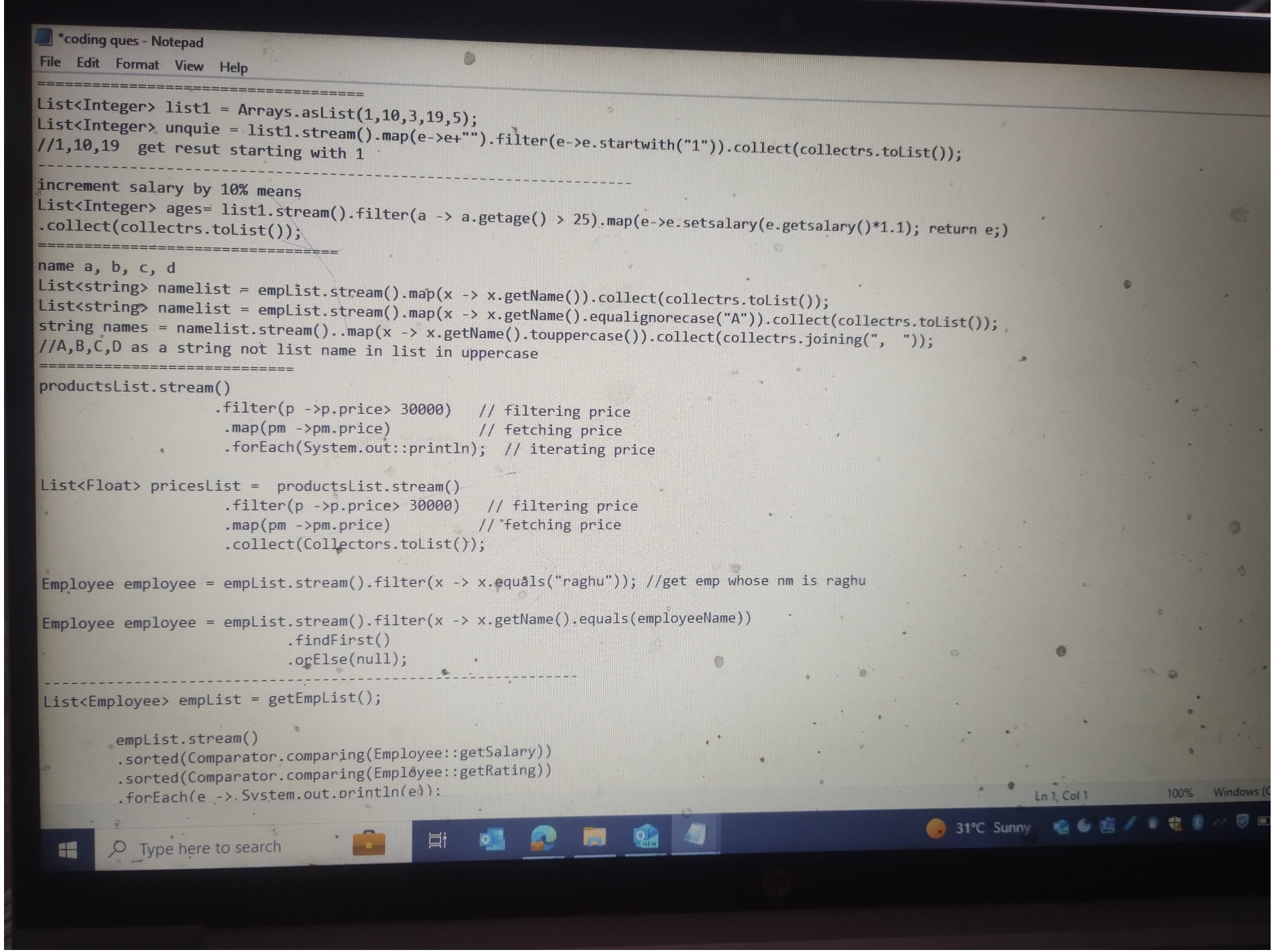
*coding ques - Notepad

File Edit Format View Help

```
Optional<Employee> highestPaidEmployeeWrapper=
employeeList.stream().collect(Collectors.maxBy(Comparator.comparingDouble(Employee::getSalary)));

Employee highestPaidEmployee = highestPaidEmployeeWrapper.get();

or

Optional<Employee> seniorMostEmployeeWrapper=
employeeList.stream().sorted(Comparator.comparingInt(Employee::getYearOfJoining)).findFirst();

Employee seniorMostEmployee = seniorMostEmployeeWrapper.get();
-----------------------------------------------------------
Who is the oldest employee in the organization? What is his age and which department he belongs to?
Optional<Employee> oldestEmployeeWrapper = employeeList.stream().max(Comparator.comparingInt(Employee::getAge));

Employee oldestEmployee = oldestEmployeeWrapper.get();

System.out.println("Name : "+oldestEmployee.getName());

System.out.println("Age : "+oldestEmployee.getAge());

System.out.println("Department : "+oldestEmployee.getDepartment());
-----------------------------------------------------------------
ages--24,34,67,12,35
List<Integer> ages= list1.stream().map(e->e.getage()).sorted().collect(collectrs.toList());
//asendng order of age, lowest 1st 12,24,34,35,67 young 1st
list<string> slicedages = ages.skip(1).limit(2).collect(collectrs.toList()); //2nd anmd 3rd  youngest 24,34
ages.skip(1) //24,34,35,67 three younges

or

Optional<Employee> youngestMaleEmployeeInProductDevelopmentWrapper=
employeeList.stream()
            .filter(e -> e.getGender()=="Male" && e.getDepartment()=="Product Development")
            .min(Comparator.comparingInt(Employee::getAge));

Employee youngestMaleEmployeeInProductDevelopment = youngestMaleEmployeeInProductDevelopmentWrapper.get();
```

Ln 1, Col 1 100% Windows (CR

Type here to search 31°C Sunny



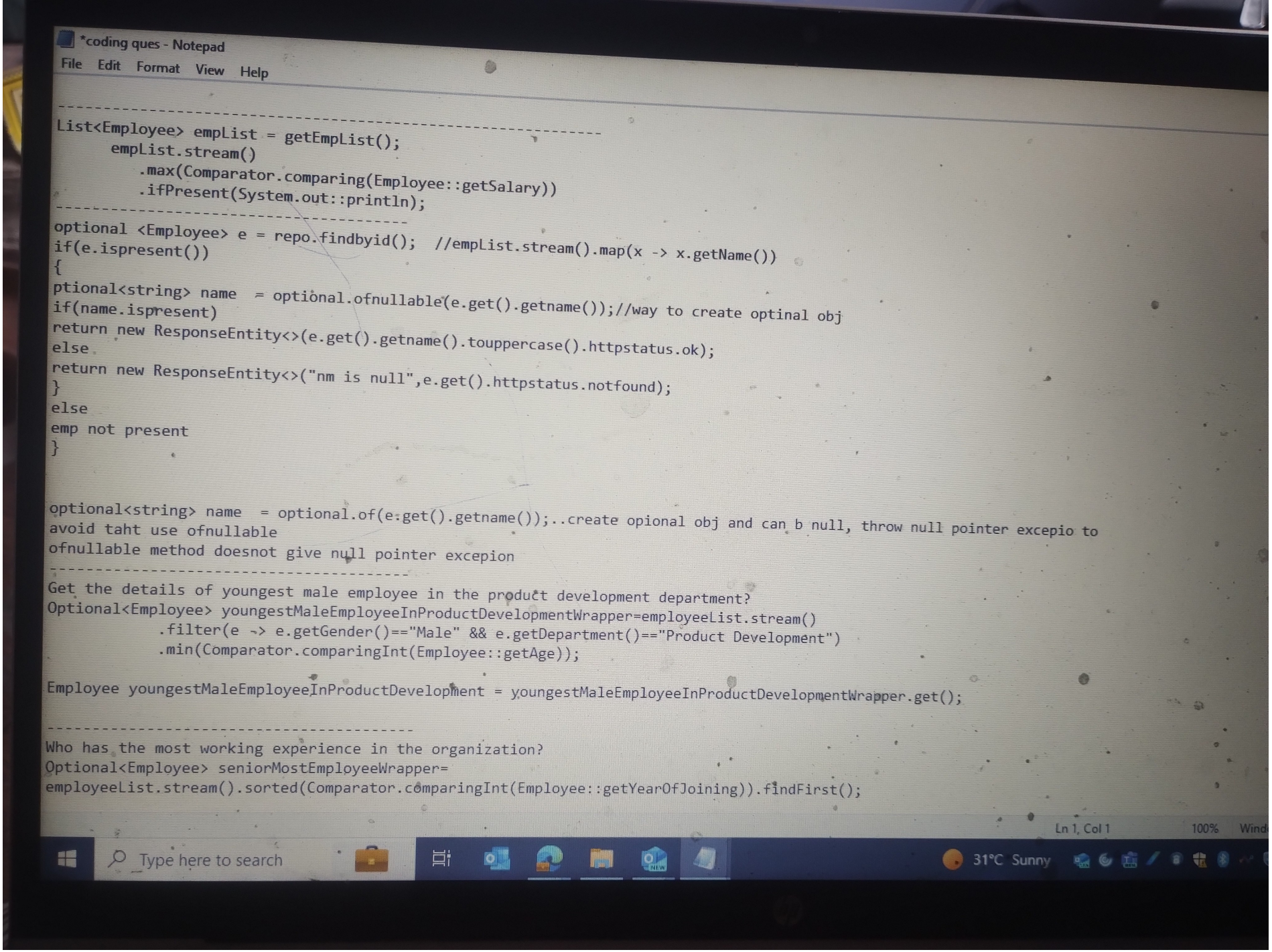
*coding ques - Notepad

File Edit Format View Help

```
==================================
List<Integer> list1 = Arrays.asList(1,10,3,19,5);
List<Integer> unquie = list1.stream().map(e->e+"").filter(e->e.startwith("1")).collect(collectrs.toList());
//1,10,19  get resut starting with 1
-----------------------------------------------------------------
increment salary by 10% means
List<Integer> ages= list1.stream().filter(a -> a.getage() > 25).map(e->e.setsalary(e.getsalary()*1.1); return e;)
.collect(collectrs.toList());
================================
name a, b, c, d
List<string> namelist = empList.stream().map(x -> x.getName()).collect(collectrs.toList());
List<string> namelist = empList.stream().map(x -> x.getName().equalignorecase("A")).collect(collectrs.toList());
string names = namelist.stream()..map(x -> x.getName().touppercase()).collect(collectrs.joining(",  "));
//A,B,C,D as a string not list name in list in uppercase
============================
productsList.stream()
                  .filter(p ->p.price> 30000)   // filtering price
                   .map(pm ->pm.price)          // fetching price
                   .forEach(System.out::println);  // iterating price

List<Float> pricesList =  productsList.stream()
                  .filter(p ->p.price> 30000)   // filtering price
                  .map(pm ->pm.price)          // fetching price
                  .collect(Collectors.toList());

Employee employee = empList.stream().filter(x -> x.equals("raghu")); //get emp whose nm is raghu

Employee employee = empList.stream().filter(x -> x.getName().equals(employeeName))
                          .findFirst()
                          .orElse(null);
-----------------------------------------------------------------
List<Employee> empList = getEmpList();

       empList.stream()
       .sorted(Comparator.comparing(Employee::getSalary))
       .sorted(Comparator.comparing(Employee::getRating))
       .forEach(e -> System.out.println(e));
```

Ln 1, Col 1 100% Windows (C

Type here to search 31°C Sunny



*coding ques - Notepad
File Edit Format View Help

```
-----------------------------------------------------------
List<Employee> empList = getEmpList();
      empList.stream()
         .max(Comparator.comparing(Employee::getSalary))
         .ifPresent(System.out::println);
-------------------------------------
optional <Employee> e = repo.findbyid();  //empList.stream().map(x -> x.getName())
if(e.ispresent())
{
ptional<string> name  = optional.ofnullable(e.get().getname());//way to create optinal obj
if(name.ispresent)
return new ResponseEntity<>(e.get().getname().touppercase().httpstatus.ok);
else
return new ResponseEntity<>("nm is null",e.get().httpstatus.notfound);
}
else
emp not present
}


optional<string> name  = optional.of(e.get().getname());..create opional obj and can b null, throw null pointer excepio to
avoid taht use ofnullable
ofnullable method doesnot give null pointer excepion
---------------------------------------
Get the details of youngest male employee in the product development department?
Optional<Employee> youngestMaleEmployeeInProductDevelopmentWrapper=employeeList.stream()
           .filter(e -> e.getGender()=="Male" && e.getDepartment()=="Product Development")
           .min(Comparator.comparingInt(Employee::getAge));

Employee youngestMaleEmployeeInProductDevelopment = youngestMaleEmployeeInProductDevelopmentWrapper.get();

-----------------------------------------
Who has the most working experience in the organization?
Optional<Employee> seniorMostEmployeeWrapper=
employeeList.stream().sorted(Comparator.comparingInt(Employee::getYearOfJoining)).findFirst();
```

Ln 1, Col 1 100% Wind

Type here to search 31°C Sunny