Problem 1:

You have a list of departments, each containing a list of employees.

1. Combine all employees from the departments into a single list.
2. Identify employees whose names begin with a specified letter.
3. Arrange these employees' names in alphabetical order.
4. Gather the sorted names into a list for each starting letter.
5. Create five Sports team with each team containing randomized employees
6. Merge the five Sports stream into three divisions

**CODE:**

package StreamProblem1;  
  
import java.util.\*;  
import java.util.stream.Collectors;  
import java.util.stream.IntStream;  
import java.util.stream.Stream;  
  
public class Employees {  
 public static void main(String[] args) {  
 Map<String, List<String>> department = new HashMap<>();  
 department.put("HR", Arrays.*asList*("Aman", "Raj", "Aditi", "Dixit"));  
 department.put("IT", Arrays.*asList*("Amar", "Megha", "Abhishek", "Manav", "Vishnu"));  
 department.put("Finace", Arrays.*asList*("Reena", "Vipin"));  
 department.put("Marketing", Arrays.*asList*("Anju", "Kavish"));  
 department.put("Operations", Arrays.*asList*("Ruby", "Raghav", "Anish"));  
  
 */// Combine all employees to single list* List<String> allEmployees = new ArrayList<>(department.values().stream()  
 .flatMap(List::stream).toList());  
 */// filter names who start with specified letter* char ch = 'A';  
 List<String> filtered = allEmployees.stream()  
 .filter(name -> name.startsWith(String.*valueOf*(ch)))  
 .sorted().toList();  
 System.*out*.println("Filtered(Start with " + ch + ")" + filtered);  
  
 */// Arrange in aplhabetical order* List<String> sorted = allEmployees.stream().sorted().toList();  
 System.*out*.println("Sorted List: " + sorted);  
  
 */// gather sorted name in for each starting letters* Map<Character, List<String>> grouped = allEmployees.stream().collect(Collectors.*groupingBy*(  
 name -> name.charAt(0), TreeMap::new, Collectors.*collectingAndThen*(  
 Collectors.*toList*(), list -> {  
 list.sort(Comparator.*naturalOrder*());  
 return list;  
 }  
 )  
 ));  
  
 System.*out*.println("Grouped By First Letter (sorted):");  
 grouped.forEach((letter, name) -> System.*out*.println(letter + ": " + name));  
  
 */// Create 5 randomized teams* Collections.*shuffle*(allEmployees);  
 List<List<String>> teams = IntStream.*range*(0, 5)  
 .mapToObj(i -> new ArrayList<String>())  
 .collect(Collectors.*toList*());  
  
 IntStream.*range*(0, allEmployees.size()).forEach(i -> teams.get(i % 5).add(allEmployees.get(i)));  
 System.*out*.println("Five Randomized Teams");  
 for (int i = 0; i < teams.size(); i++) {  
 System.*out*.println("Team " + (i + 1) + ": " + teams.get(i));  
 }  
  
 */// Merge teams into 3 divisions* Map<String, List<String>> divisions = new LinkedHashMap<>();  
 divisions.put("Division A", Stream.*concat*(teams.get(0).stream(), teams.get(1).stream())  
 .collect(Collectors.*toList*()));  
 divisions.put("Division B", Stream.*concat*(teams.get(2).stream(), teams.get(3).stream())  
 .collect(Collectors.*toList*()));  
 divisions.put("Division C", new ArrayList<>(teams.get(4)));  
  
 System.*out*.println("\nThree Divisions:");  
 divisions.forEach((division, members) ->  
 System.*out*.println(division + ": " + members)  
 );  
  
 }  
}

**OUTPUT:**

**A computer screen shot of a black screen

AI-generated content may be incorrect.**

Problem 2:

Find the frequency of each character in a string using Java streams

Write a Java 8 to find the frequency of each character in each string using the stream API and collectors.

**CODE:**

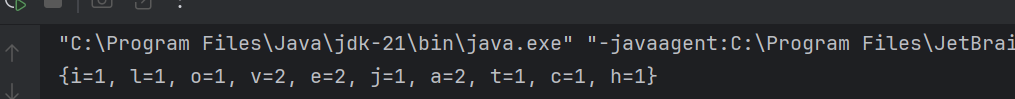
public class Main {  
 public static void main(String[] args) {

String s="ilovejavatech";

LinkedHashMap<String,Long> lhm=Arrays.*stream*(s.split("")).collect(Collectors.*groupingBy*(Function.*identity*(),LinkedHashMap::new,Collectors.*counting*()));  
System.*out*.println(lhm);

}}

**OUTPUT:**

****

Problem 3

Given a list of Student names add the correct Prefix to the names of the students using their Gender

**CODE:**

package StreamProblem1;  
  
import java.util.ArrayList;  
import java.util.List;  
import java.util.\*;  
  
public class Gender {  
 static class Student {  
 String name;  
 String gender; // "Male" or "Female"  
  
 Student(String name, String gender) {  
 this.name = name;  
 this.gender = gender;  
 }  
  
 String getName() {  
 return name;  
 }  
  
 String getGender() {  
 return gender;  
 }  
 }  
 public static void main(String[] args) {  
 List<Student> list= List.*of*(  
 new Student("Amir","Male"),  
 new Student("Aniya","Female"),  
 new Student("Aman","Male"),  
 new Student("Dixit","Male"),  
 new Student("Manya","Female")  
 );  
  
 List<String> prefixName=list.stream().map(student -> (student.getGender()  
 .equalsIgnoreCase("Male")?"Mr. ":"Ms. ")+student.getName()).toList();  
 prefixName.forEach(System.*out*::println);  
 }  
}

**OUTPUT:**

**A screen shot of a computer

AI-generated content may be incorrect.**

Problem 4

You have a list of laptops with their configurations.

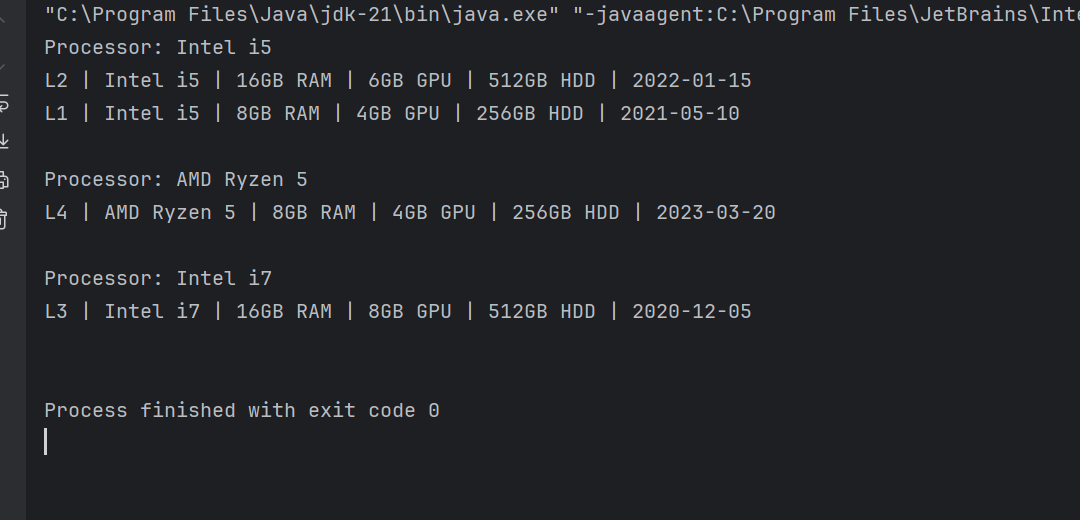
1. Write a function to find all laptops that have at least the specified RAM capacity and graphics card capacity.
2. Group these laptops by their processor model.
3. Sort the laptops within each group by memory, hard disk size, and date of manufacturing.

**Code:**

package StreamProblem1;  
  
import java.util.Collections;  
import java.util.Comparator;  
import java.util.List;  
import java.util.Map;  
import java.time.LocalDate;  
import java.util.stream.Collectors;  
  
 class Laptop {  
 private String model;  
 private String processor;  
 private int ram; // in GB  
 private int graphicsCard; // in GB  
 private int hardDisk; // in GB  
 private LocalDate manufacturingDate;  
  
 public Laptop(String model, String processor, int ram, int graphicsCard, int hardDisk, LocalDate manufacturingDate) {  
 this.model = model;  
 this.processor = processor;  
 this.ram = ram;  
 this.graphicsCard = graphicsCard;  
 this.hardDisk = hardDisk;  
 this.manufacturingDate = manufacturingDate;  
 }  
  
 // Getters  
 public String getModel() { return model; }  
 public String getProcessor() { return processor; }  
 public int getRam() { return ram; }  
 public int getGraphicsCard() { return graphicsCard; }  
 public int getHardDisk() { return hardDisk; }  
 public LocalDate getManufacturingDate() { return manufacturingDate; }  
  
 @Override  
 public String toString() {  
 return model + " | " + processor + " | " + ram + "GB RAM | " + graphicsCard + "GB GPU | " +  
 hardDisk + "GB HDD | " + manufacturingDate;  
 }  
}  
  
public class Laptops {  
 static Map<String,List<Laptop>> find(List<Laptop> laptops,int minRam,int minGpu){  
 return laptops.stream().filter(laptop -> laptop.getRam()>=minRam  
 &&laptop.getGraphicsCard()>=minGpu).collect(Collectors.*groupingBy*(  
 Laptop::getProcessor, Collectors.*collectingAndThen*(Collectors.*toList*(),list->{  
 list.sort(  
 Comparator.*comparingInt*(Laptop::getRam).reversed()  
 .thenComparingInt(Laptop::getHardDisk).reversed()  
 .thenComparing(Laptop::getManufacturingDate).reversed()  
 );  
 return list;  
 })  
 ));  
 }  
 public static void main(String[] args) {  
 List<Laptop> laptops = List.*of*(  
 new Laptop("L1", "Intel i5", 8, 4, 256, LocalDate.*of*(2021, 5, 10)),  
 new Laptop("L2", "Intel i5", 16, 6, 512, LocalDate.*of*(2022, 1, 15)),  
 new Laptop("L3", "Intel i7", 16, 8, 512, LocalDate.*of*(2020, 12, 5)),  
 new Laptop("L4", "AMD Ryzen 5", 8, 4, 256, LocalDate.*of*(2023, 3, 20)),  
 new Laptop("L5", "Intel i5", 8, 3, 512, LocalDate.*of*(2021, 8, 30))  
 );  
 int minRam = 8;  
 int minGpu = 4;  
  
 Map<String, List<Laptop>> grouped = *find*(laptops, minRam, minGpu);  
  
 grouped.forEach((processor, laptopList) -> {  
 System.*out*.println("Processor: " + processor);  
 laptopList.forEach(System.*out*::println);  
 System.*out*.println();  
 });  
 }  
}

\

**OUTPUT:**

****