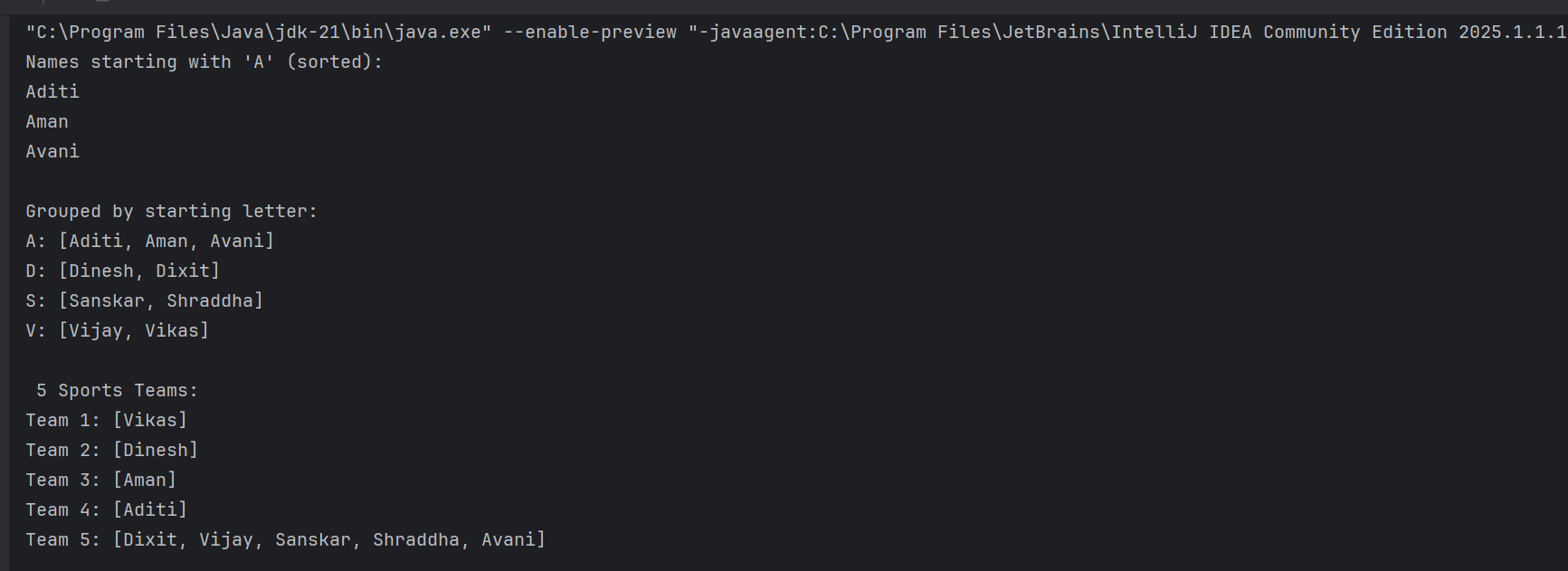
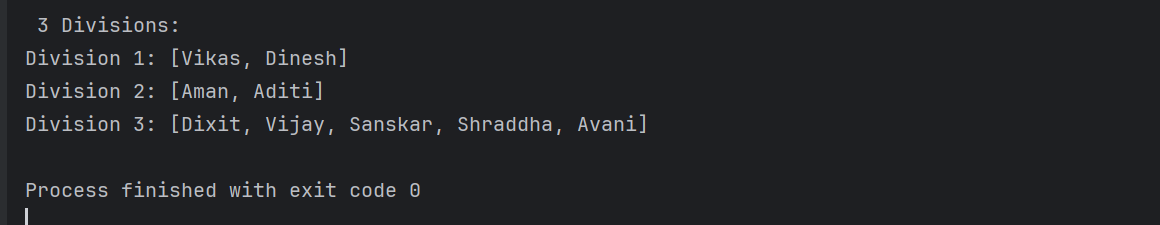
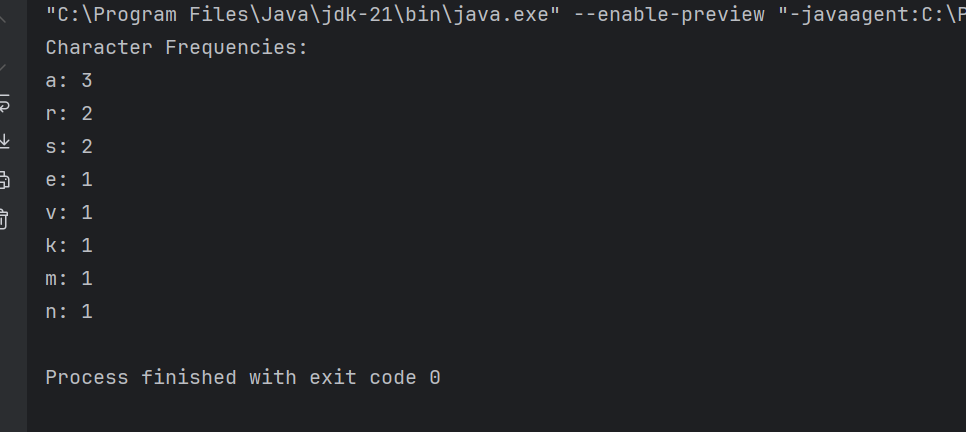
You have a list of departments, each containing a list of employees.d

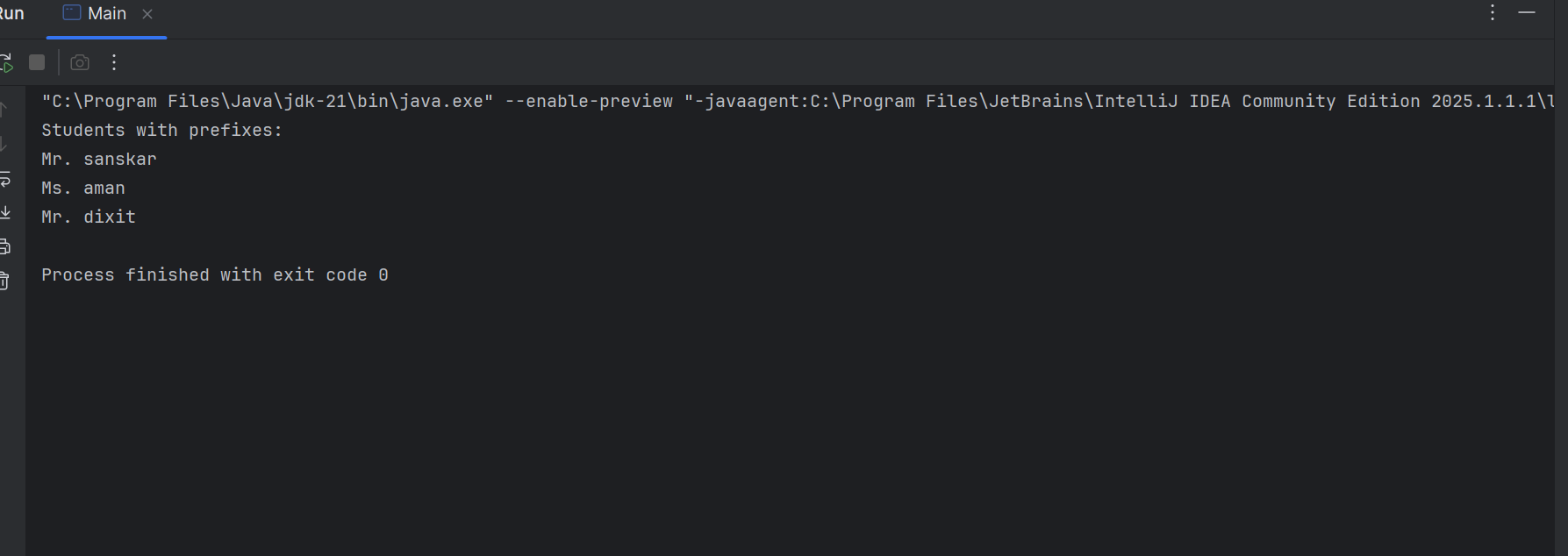
1. Combine all employees from the departments into a single list.
2. Identify employees whose names begin with a specified letter.
3. Arrange these employees' names in alphabetical order.
4. Gather the sorted names into a list for each starting letter.
5. Create five Sports team with each team containing randomized employees
6. Merge the five Sports stream into three divisions
7. import java.util.\*;  
   import java.util.stream.\*;  
     
   class Employee {  
    private String name;  
     
    public Employee(String name) {  
    this.name = name;  
    }  
     
    public String getName() { return name; }  
     
    @Override  
    public String toString() { return name; }  
   }  
     
   class Department {  
    private String name;  
    private List<Employee> employees;  
     
    public Department(String name, List<Employee> employees) {  
    this.name = name;  
    this.employees = employees;  
    }  
     
    public List<Employee> getEmployees() { return employees; }  
    public String getName() { return name; }  
     
    @Override  
    public String toString() {  
    return name + ": " + employees;  
    }  
   }  
     
   public class Main {  
    public static void main(String[] args) {  
    // Step 1: Create departments and employees  
    List<Department> departments = Arrays.*asList*(  
    new Department("HR", Arrays.*asList*(  
    new Employee("Avani"),  
    new Employee("Aditi"),  
    new Employee("Shraddha")  
    )),  
    new Department("Technology", Arrays.*asList*(  
    new Employee("Sanskar"),  
    new Employee("Dixit"),  
    new Employee("Aman")  
    )),  
    new Department("Management", Arrays.*asList*(  
    new Employee("Vikas"),  
    new Employee("Dinesh"),  
    new Employee("Vijay")  
    ))  
    );  
     
    // Step 2: Combine all employees  
    List<Employee> allEmployees = departments.stream()  
    .flatMap(dept -> dept.getEmployees().stream())  
    .collect(Collectors.*toList*());  
     
    // Step 3: Filter employees by name starting with 'A'  
    char startLetter = 'A';  
    List<String> namesStartingWithA = allEmployees.stream()  
    .map(Employee::getName)  
    .filter(name -> name.startsWith(String.*valueOf*(startLetter)))  
    .sorted()  
    .collect(Collectors.*toList*());  
     
    System.*out*.println("Names starting with '" + startLetter + "' (sorted):");  
    namesStartingWithA.forEach(System.*out*::println);  
     
    // Step 4: Group all names by starting letter (sorted)  
    Map<Character, List<String>> groupedByFirstLetter = allEmployees.stream()  
    .map(Employee::getName)  
    .sorted()  
    .collect(Collectors.*groupingBy*(name -> name.charAt(0), TreeMap::new, Collectors.*toList*()));  
     
    System.*out*.println("\nGrouped by starting letter:");  
    groupedByFirstLetter.forEach((ch, list) -> {  
    System.*out*.println(ch + ": " + list);  
    });  
     
    // Step 5: Create 5 random sports teams  
    Collections.*shuffle*(allEmployees);  
    int teamCount = 5;  
    int size = allEmployees.size();  
    int baseSize = size / teamCount;  
    List<List<Employee>> teams = new ArrayList<>();  
     
    for (int i = 0; i < teamCount; i++) {  
    int start = i \* baseSize;  
    int end = (i == teamCount - 1) ? size : start + baseSize;  
    teams.add(allEmployees.subList(start, end));  
    }  
     
    System.*out*.println("\n 5 Sports Teams:");  
    for (int i = 0; i < teams.size(); i++) {  
    System.*out*.println("Team " + (i + 1) + ": " + teams.get(i));  
    }  
     
    // Step 6: Merge 5 teams into 3 divisions  
    List<List<Employee>> divisions = new ArrayList<>();  
    divisions.add(Stream.*concat*(teams.get(0).stream(), teams.get(1).stream()).collect(Collectors.*toList*()));  
    divisions.add(Stream.*concat*(teams.get(2).stream(), teams.get(3).stream()).collect(Collectors.*toList*()));  
    divisions.add(teams.get(4));  
     
    System.*out*.println("\n 3 Divisions:");  
    for (int i = 0; i < divisions.size(); i++) {  
    System.*out*.println("Division " + (i + 1) + ": " + divisions.get(i));  
    }  
    }  
   }

  
  
  
  
  
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.

import java.util.\*;  
import java.util.function.Function;  
import java.util.stream.Collectors;  
  
public class Main{  
 public static void main(String[] args) {  
 String input = "sanskarverma";  
  
 Map<Character, Long> frequency = input.chars() // IntStream of chars  
 .mapToObj(c -> (char) c)  
 .collect(Collectors.*groupingBy*(Function.*identity*(), Collectors.*counting*()));  
  
 System.*out*.println("Character Frequencies:");  
 frequency.forEach((ch, count) -> System.*out*.println(ch + ": " + count));  
 }  
  
}

  
  
  
  
Given a list of Student names add the correct Prefix to the names of the students using their Gender   
  
import java.util.Arrays;  
import java.util.List;  
import java.util.\*;  
import java.util.stream.Collectors;  
  
class Student {  
 private String name;  
 private String gender; // "Male" or "Female"  
  
 public Student(String name, String gender) {  
 this.name = name;  
 this.gender = gender;  
 }  
  
 public String getGender() { return gender; }  
 public String getName() { return name; }  
  
 @Override  
 public String toString() {  
 String prefix = gender.equalsIgnoreCase("Male") ? "Mr." : "Ms.";  
 return prefix + " " + name;  
 }  
}  
  
  
public class Main{  
 public void main(String[] args) {  
 List<Student> students = Arrays.*asList*(  
 new Student("sanskar","Male"),  
 new Student("aman", "Female"),  
 new Student("dixit", "Male")  
 );  
  
 List<String> prefixedNames = students.stream()  
 .map(Student::toString)  
 .collect(Collectors.*toList*());  
  
 System.*out*.println("Students with prefixes:");  
 prefixedNames.forEach(System.*out*::println);  
 }  
}

  
  
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.

import java.time.LocalDate;  
import java.util.Arrays;  
import java.util.Comparator;  
import java.util.List;  
import java.util.Map;  
import java.util.stream.Collectors;  
  
class Laptop {  
 private String model;  
 private String processor;  
 private int ramGB;  
 private int graphicsGB;  
 private int hddGB;  
 private LocalDate manufactureDate;  
  
 public Laptop(String model, String processor, int ramGB, int graphicsGB, int hddGB, LocalDate manufactureDate) {  
 this.model = model;  
 this.processor = processor;  
 this.ramGB = ramGB;  
 this.graphicsGB = graphicsGB;  
 this.hddGB = hddGB;  
 this.manufactureDate = manufactureDate;  
 }  
  
 public String getProcessor() { return processor; }  
 public int getRamGB() { return ramGB; }  
 public int getGraphicsGB() { return graphicsGB; }  
 public int getHddGB() { return hddGB; }  
 public LocalDate getManufactureDate() { return manufactureDate; }  
  
 @Override  
 public String toString() {  
 return model + " | RAM: " + ramGB + "GB | Graphics: " + graphicsGB + "GB | HDD: " + hddGB + "GB | Date: " + manufactureDate;  
 }  
}  
  
  
public class Main {  
 public static void main(String[] args) {  
 List<Laptop> laptops = Arrays.*asList*(  
 new Laptop("Dell X1", "i5", 8, 2, 512, LocalDate.*of*(2023, 5, 12)),  
 new Laptop("HP Pro", "i7", 16, 4, 1024, LocalDate.*of*(2022, 3, 8)),  
 new Laptop("Acer Nitro", "i5", 16, 6, 512, LocalDate.*of*(2021, 11, 25)),  
 new Laptop("Lenovo IdeaPad", "i3", 4, 1, 256, LocalDate.*of*(2024, 1, 15)),  
 new Laptop("MSI Stealth", "i7", 32, 8, 1024, LocalDate.*of*(2023, 10, 1))  
 );  
  
 int minRam = 8;  
 int minGraphics = 2;  
  
 laptops.stream().filter(laptop -> laptop.getRamGB() >= minRam && laptop.getGraphicsGB()>=2).forEach(System.*out*::println);  
 System.*out*.println();  
 System.*out*.println();  
 Map<String, List<Laptop>> mp = laptops.stream()  
 .collect(Collectors.*groupingBy*(laptop -> laptop.getProcessor()));  
 System.*out*.println(mp);  
  
 System.*out*.println();  
 System.*out*.println();  
 mp.forEach((i,j)->{  
 System.*out*.println(i+":");  
 j.stream().sorted(Comparator.*comparingInt*((Laptop l)-> l.getRamGB()).thenComparingInt(l->l.getHddGB()).thenComparing(l->l.getManufactureDate())).forEach(System.*out*::println);  
 System.*out*.println();  
 });  
 }  
}

