Problem 1:

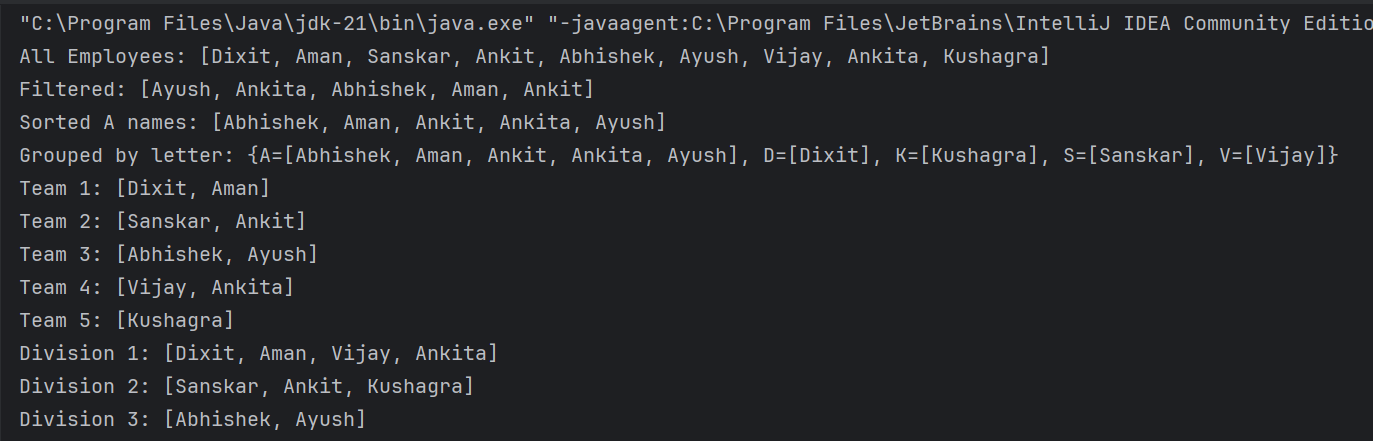
You have a list of departments, each containing a list of employees.

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 Training;  
import java.util.\*;  
import java.util.stream.Collectors;  
  
class Employee {  
 String name;  
 Employee(String name) {  
 this.name = name;  
 }  
 public String toString() {  
 return name;  
 }  
}  
  
class Department {  
 String name;  
 List<Employee> employees;  
 Department(String name, List<Employee> employees) {  
 this.name = name;  
 this.employees = employees;  
 }  
}  
public class EmployeeD {  
 public static List<Employee> getAllEmployees(List<Department> departments) {  
 return departments.stream()  
 .flatMap(d -> d.employees.stream())  
 .collect(Collectors.*toList*());  
 }  
 public static List<Employee> filterByInitial(List<Employee> employees, char initial) {  
 return employees.stream()  
 .filter(e -> e.name.toUpperCase().charAt(0) == Character.*toUpperCase*(initial))  
 .collect(Collectors.*toList*());  
 }  
 public static List<String> sortNames(List<Employee> employees) {  
 return employees.stream()  
 .map(e -> e.name)  
 .sorted()  
 .collect(Collectors.*toList*());  
 }  
 public static Map<Character, List<String>> groupByLetter(List<Employee> employees) {  
 return employees.stream()  
 .map(e -> e.name)  
 .sorted()  
 .collect(Collectors.*groupingBy*(name -> Character.*toUpperCase*(name.charAt(0)), TreeMap::new, Collectors.*toList*()));  
 }  
 public static List<List<Employee>> createTeams(List<Employee> employees, int count) {  
 Collections.*shuffle*(employees);  
 List<List<Employee>> teams = new ArrayList<>();  
 int size = employees.size(), teamSize = (int) Math.*ceil*(size / (double) count);  
 for (int i = 0; i < count; i++) {  
 int from = i \* teamSize, to = Math.*min*(from + teamSize, size);  
 if (from < to) teams.add(new ArrayList<>(employees.subList(from, to)));  
 }  
 return teams;  
 }  
 public static List<List<Employee>> mergeTeams(List<List<Employee>> teams, int divisions) {  
 List<List<Employee>> result = new ArrayList<>();  
 for (int i = 0; i < divisions; i++) result.add(new ArrayList<>());  
 for (int i = 0; i < teams.size(); i++) result.get(i % divisions).addAll(teams.get(i));  
 return result;  
 }  
  
 public static void main(String[] args) {  
 List<Department> departments = Arrays.*asList*(  
 new Department("HR", Arrays.*asList*(new Employee("Dixit"), new Employee("Ayush"), new Employee("Ankita"))),  
 new Department("IT", Arrays.*asList*(new Employee("Sanskar"), new Employee("Kushagra"), new Employee("Abhishek"))),  
 new Department("Finance", Arrays.*asList*(new Employee("Aman"), new Employee("Ankit"), new Employee("Vijay"))));  
 List<Employee> all = *getAllEmployees*(departments);  
 List<Employee> filtered = *filterByInitial*(all, 'A');  
 List<String> sorted = *sortNames*(filtered);  
 Map<Character, List<String>> grouped = *groupByLetter*(all);  
 List<List<Employee>> teams = *createTeams*(all, 5);  
 List<List<Employee>> divisions = *mergeTeams*(teams, 3);  
 System.*out*.println("All Employees: " + all);  
 System.*out*.println("Filtered: " + filtered);  
 System.*out*.println("Sorted A names: " + sorted);  
 System.*out*.println("Grouped by letter: " + grouped);  
 for (int i = 0; i < teams.size(); i++)  
 System.*out*.println("Team " + (i + 1) + ": " + teams.get(i));  
 for (int i = 0; i < divisions.size(); i++)  
 System.*out*.println("Division " + (i + 1) + ": " + divisions.get(i));  
 }  
}

***Output –***

******

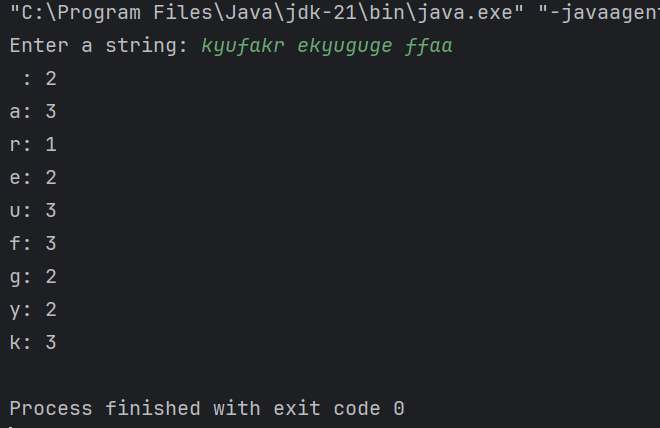
Problem 2:

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

***Code –***

package Day2;  
import java.util.\*;  
import java.util.stream.Collectors;  
  
public class Freqmap {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
 System.*out*.print("Enter a string: ");  
 String input = scanner.nextLine();  
 Map<String, Long> frequencyMap = Arrays.*stream*(input.split(""))  
 .collect(Collectors.*groupingBy*(c -> c, Collectors.*counting*()));  
 frequencyMap.forEach((key, value) -> System.*out*.println(key + ": " + value));  
 }  
}

***Output –***

******

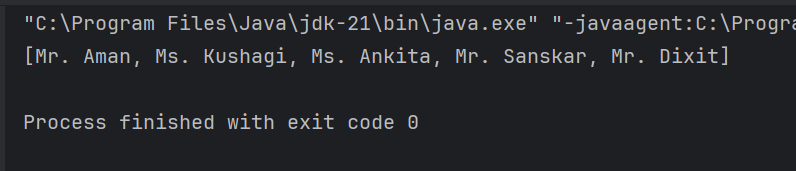
Problem 3

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

***Code –***

package Day2;  
import java.util.\*;  
import java.util.stream.\*;  
class Student {  
 String name;  
 String gender;  
 public Student(String name, String gender) {  
 this.name = name;  
 this.gender = gender;  
 }  
 public String getName() { return name; }  
 public String getGender() { return gender; }  
}  
  
public class Pref {  
 public static void main(String[] args) {  
 List<Student> students = Arrays.*asList*(  
 new Student("Aman", "Male"),  
 new Student("Kushagi", "Female"),  
 new Student("Ankita", "Female"),  
 new Student("Sanskar", "Male"),  
 new Student("Dixit", "Male"));  
 List prefixedNames = students.stream()  
 .map(s -> (s.getGender().equalsIgnoreCase("Male") ? "Mr. " : "Ms. ") + s.getName())  
 .collect(Collectors.*toList*());  
 System.*out*.println(prefixedNames);  
 }  
}

***Output –***

******

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 Training;  
import java.time.LocalDate;  
import java.util.\*;  
import java.util.stream.Collectors;  
  
class Laptop {  
 String brand;  
 String processor;  
 int ram;  
 int graphicsCard;  
 int hardDisk;  
 LocalDate manufacturingDate;  
 public Laptop(String brand, String processor, int ram, int graphicsCard, int hardDisk, LocalDate manufacturingDate) {  
 this.brand = brand;  
 this.processor = processor;  
 this.ram = ram;  
 this.graphicsCard = graphicsCard;  
 this.hardDisk = hardDisk;  
 this.manufacturingDate = manufacturingDate;  
 }  
 @Override  
 public String toString() {  
 return brand + " | " + processor + " | " + ram + "GB RAM | " + graphicsCard + "GB GPU | " +  
 hardDisk + "GB HDD | " + manufacturingDate;  
 }  
}  
  
public class LaptopS {  
 public static List<Laptop> filterLaptops(List<Laptop> laptops, int minRam, int minGraphicsCard) {  
 return laptops.stream()  
 .filter(l -> l.ram >= minRam && l.graphicsCard >= minGraphicsCard)  
 .collect(Collectors.*toList*());  
 }  
 public static Map<String, List<Laptop>> groupAndSortLaptops(List<Laptop> laptops) {  
 return laptops.stream()  
 .collect(Collectors.*groupingBy*(  
 l -> l.processor,  
 Collectors.*collectingAndThen*(  
 Collectors.*toList*(),  
 list -> list.stream()  
 .sorted(Comparator.*comparingInt*((Laptop l) -> l.ram).reversed()  
 .thenComparingInt(l -> l.hardDisk)  
 .thenComparing(l -> l.manufacturingDate))  
 .collect(Collectors.*toList*())  
 )  
 ));  
 }  
  
 public static void main(String[] args) {  
 List<Laptop> laptops = Arrays.*asList*(  
 new Laptop("Dell", "i5", 8, 2, 512, LocalDate.*of*(2022, 5, 1)),  
 new Laptop("HP", "i7", 16, 4, 1024, LocalDate.*of*(2023, 3, 15)),  
 new Laptop("Lenovo", "i5", 8, 1, 256, LocalDate.*of*(2021, 8, 10)),  
 new Laptop("Asus", "i7", 32, 6, 2048, LocalDate.*of*(2024, 1, 5)),  
 new Laptop("Acer", "i5", 16, 4, 512, LocalDate.*of*(2022, 11, 25)));  
 int minRam = 8;  
 int minGPU = 2;  
 List<Laptop> filtered = *filterLaptops*(laptops, minRam, minGPU);  
 Map<String, List<Laptop>> groupedAndSorted = *groupAndSortLaptops*(filtered);  
 groupedAndSorted.forEach((processor, list) -> {  
 System.*out*.println("Processor: " + processor);  
 list.forEach(System.*out*::println);  
 System.*out*.println();  
 });  
 }  
}

***Output –***

