Problem 1:

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

1. Combine all employees from the departments into a single list.

package Steam\_assignment;  
  
import java.util.Arrays;  
import java.util.List;  
import java.util.stream.Collectors;  
import java.util.stream.Stream;  
  
public class Employeequest {  
 public static void main(String[] args) {  
 List<String> hr = Arrays.*asList*("Aditi", "Shraddha", "Avni");  
 List<String> engineering = Arrays.*asList*("Aman", "Abhishek", "Dixit", "Manan");  
 List<String> sales = Arrays.*asList*("Sanskar", "Jhanvi", "Khusagra");  
  
 List<String> emp = Stream.*of*(hr, engineering, sales).flatMap(List::stream).collect(Collectors.*toList*());  
 emp.forEach(s -> System.*out*.println(s));  
 }  
}

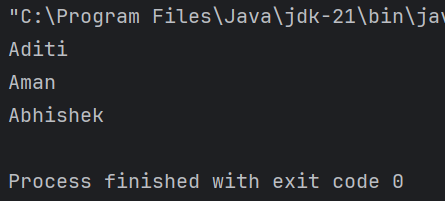
Output:



1. Identify employees whose names begin with a specified letter.

emp.stream().filter(s -> s.startsWith("A")).forEach(System.*out*::println);

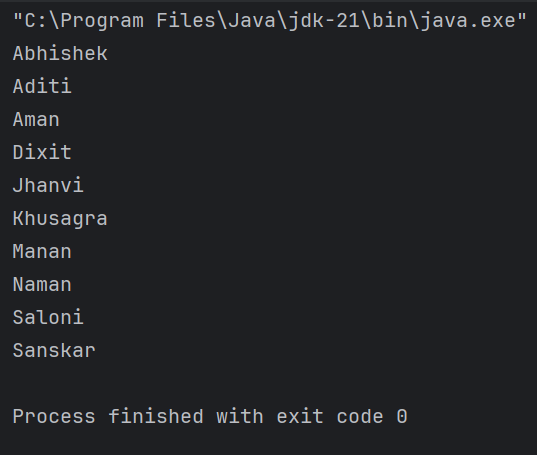
Output:



1. Arrange these employees' names in alphabetical order.

emp.stream().sorted().forEach(System.*out*::println);

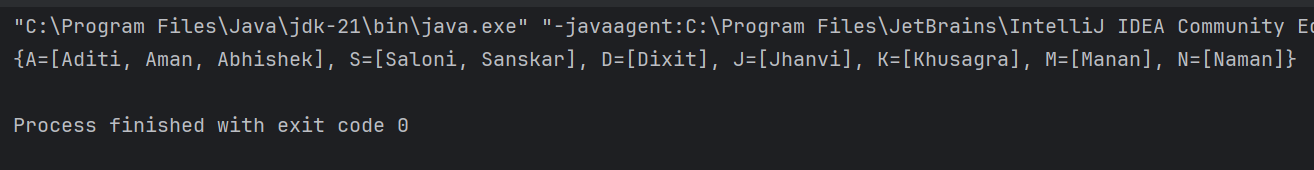
Output:



1. Gather the sorted names into a list for each starting letter.

package Steam\_assignment;  
  
import java.util.Arrays;  
import java.util.List;  
import java.util.Map;  
import java.util.TreeMap;  
import java.util.stream.Collectors;  
import java.util.stream.Stream;  
  
public class Employeequest {  
 public static void main(String[] args) {  
 List<String> hr = Arrays.*asList*("Naman", "Saloni", "Aditi");  
 List<String> engineering = Arrays.*asList*("Aman", "Abhishek", "Dixit", "Manan");  
 List<String> sales = Arrays.*asList*("Sanskar", "Jhanvi", "Khusagra");  
 List<String> emp = Stream.*of*(hr, engineering, sales).flatMap(List::stream).collect(Collectors.*toList*());  
 Map<Character, List<String>> gs = emp.stream().collect(Collectors.*groupingBy*(s -> s.charAt(0)));  
 System.*out*.println(gs);  
  
 }  
}

Output:



1. Create five Sports team with each team containing randomized employees
2. Merge the five Sports stream into three divisions

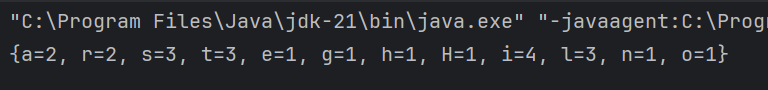
Problem 2:

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.

package Steam\_assignment;  
  
import java.util.Arrays;  
import java.util.stream.Collectors;  
  
public class question2 {  
 public static void main(String[] args) {  
 String s = "Hellothisisatrialstring";  
 System.*out*.println(Arrays.*stream*(s.split("")).  
 collect(Collectors.*groupingBy*(c-> c, Collectors.*counting*())));  
  
 }  
}

Output:

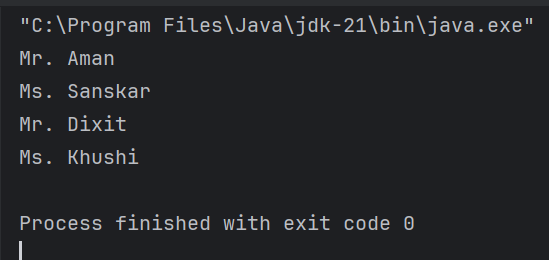


Problem 3

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

package Steam\_assignment;  
  
import java.util.ArrayList;  
import java.util.Arrays;  
import java.util.List;  
import java.util.stream.Collectors;  
  
class student{  
 String name;  
 String gender;  
  
 student(String name, String gender){  
 this.name = name;  
 this.gender = gender;  
 }  
  
}  
  
public class question4 {  
 public static void main(String[] args) {  
 List<student> l1 = Arrays.*asList*(new student("Aman", "Male"),  
 new student("Sanskar", "Female"),  
 new student("Dixit", "Male"),  
 new student("Khushi", "Female"));  
  
 l1.stream().map(student -> {  
 String p = student.gender.equals("Male") ? "Mr. " : "Ms. ";  
 return p + student.name;  
 }).collect(Collectors.*toList*()).forEach(System.*out*::println);  
 }  
}

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.

package Steam\_assignment;  
  
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 {  
 String model;  
 String processorModel;  
 int ram;  
 int graphicsCard;  
 int memory;  
 int hardDisk;  
 LocalDate manufactureDate;  
  
 public Laptop(String model, String processorModel, int ram, int graphicsCard, int memory, int hardDisk, LocalDate manufactureDate) {  
 this.model = model;  
 this.processorModel = processorModel;  
 this.ram = ram;  
 this.graphicsCard = graphicsCard;  
 this.memory = memory;  
 this.hardDisk = hardDisk;  
 this.manufactureDate = manufactureDate;  
 }  
  
 @Override  
 public String toString() {  
 return model + " | " + processorModel + " | RAM: " + ram + "GB | GPU: " + graphicsCard + "GB | Memory: " + memory + "GB | HDD: " + hardDisk + "GB | Date: " + manufactureDate;  
 }  
}  
  
  
public class problem4 {  
 public static void main(String[] args) {  
 List<Laptop> laptops = Arrays.*asList*(  
 new Laptop("HP Pavilion", "Intel i5", 16, 4, 512, 1000, LocalDate.*of*(2022, 5, 1)),  
 new Laptop("Dell XPS", "Intel i7", 32, 6, 1024, 2000, LocalDate.*of*(2023, 3, 15)),  
 new Laptop("Acer Nitro", "AMD Ryzen 7", 16, 4, 512, 1000, LocalDate.*of*(2021, 9, 10)),  
 new Laptop("Lenovo Legion", "AMD Ryzen 7", 32, 8, 1024, 2000, LocalDate.*of*(2023, 1, 20)),  
 new Laptop("Asus ZenBook", "Intel i5", 8, 2, 256, 512, LocalDate.*of*(2020, 2, 5))  
 );  
 System.*out*.println("All laptops with at least 16GB RAM and 6GB VRAM");  
 laptops.stream().  
 filter(lap -> lap.ram >= 16 && lap.graphicsCard>=6)  
 .collect(Collectors.*toList*()).forEach(System.*out*::println);  
 System.*out*.println();  
 System.*out*.println("Laptops Grouped by Processor Type: ");  
 Map<String, List<Laptop>> mp = laptops.stream().  
 collect(Collectors.*groupingBy*(lap -> lap.processorModel));  
 mp.forEach((key, value) -> {  
 System.*out*.println(key + ":");  
 value.forEach(System.*out*::println);  
 System.*out*.println();  
 });  
  
 System.*out*.println("Printing with sorted according to Memory, HDD, DOM:");  
 mp.forEach((key, value) ->{  
 System.*out*.println(key + ":");  
 value.stream().sorted(Comparator.*comparingInt*((Laptop l)->l.memory).  
 thenComparingInt(l->l.hardDisk)  
 .thenComparing(l -> l.manufactureDate))  
 .forEach(System.*out*::println);  
 System.*out*.println();  
 });  
 }  
}

Output:

