### Question 1 **Problem Statement**

You are tasked with implementing a simple Java application to manage and analyze a collection of shoe details. The application consists of three main components:

1. **ShoeDetails Class**: Represents the details of a shoe, including its ID, name, brand, arrival date, and size.
2. **ShoeUtility Class**: Provides utility methods to filter and process a list of ShoeDetails objects.
3. **UserInterface Class**: Handles user interaction, allowing users to input shoe details, and then utilizes ShoeUtility methods to display information.

**Class Descriptions:**

* **ShoeDetails Class**:
  + **Attributes**: shoeId, shoeName, brandName, arrivalDate, and size.
  + **Methods**: Getter and setter methods for each attribute, overridden hashCode(), equals(), and toString() methods.
* **ShoeUtility Class**:
  + **Methods**:
    - fetchShoesByBrand(List<ShoeDetails> shoeList, String brandName): Returns a stream of ShoeDetails objects filtered by the specified brand name.
    - fetchShoesInAscendingOrderBySize(List<ShoeDetails> shoeList): Returns a stream of ShoeDetails objects sorted in ascending order of size.
    - fetchUniqueShoes(List<ShoeDetails> shoeList): Returns a stream of unique shoe names.
* **UserInterface Class**:
  + **Functionality**:
    - Prompts the user to input the number of shoes and their details.
    - Uses ShoeUtility methods to filter by brand, sort by size, and get unique shoe names.
    - Displays the results to the user.

**Task**:

1. Implement the ShoeDetails, ShoeUtility, and UserInterface classes as described.
2. Test the application using sample input and verify that it produces the correct output.

**Sample Input and Output**

**Sample Input:**

Enter the number of Shoe details:

3

Enter the Shoe details:

S001:Air Max:Nike:2023-07-01:10

S002:Ultraboost:Adidas:2023-08-15:9

S003:Air Max:Nike:2023-08-25:11

Enter the Brand name:

Nike

**Expected Output:**

ShoeDetails [shoeId:S001, shoeName:Air Max, brandName:Nike, arrivalDate:2023-07-01, size:10]

ShoeDetails [shoeId:S003, shoeName:Air Max, brandName:Nike, arrivalDate:2023-08-25, size:11]

Shoes sorted by their size:

S002:Ultraboost:Adidas:2023-08-15:9

S001:Air Max:Nike:2023-07-01:10

S003:Air Max:Nike:2023-08-25:11

Unique shoe names are:

Air Max

Ultraboost

Boiler plate :  
  
import java.time.LocalDate;  
import java.util.Objects;  
  
public class ShoeDetails {  
 private String shoeId;  
 private String shoeName;  
 private String brandName;  
 private LocalDate arrivalDate;  
 private int size;  
  
 public ShoeDetails () {  
 }  
  
 public ShoeDetails (String shoeId, String shoeName, String brandName, LocalDate arrivalDate, int size) {  
 this.shoeId = shoeId;  
 this.shoeName = shoeName;  
 this.brandName = brandName;  
 this.arrivalDate = arrivalDate;  
 this.size = size;  
 }  
  
 public String getShoeId() {  
 return shoeId;  
 }  
  
 public void setShoeId(String shoeId) {  
 this.shoeId = shoeId;  
 }  
  
 public String getShoeName() {  
 return shoeName;  
 }  
  
 public void setShoeName(String shoeName) {  
 this.shoeName = shoeName;  
 }  
  
 public String getBrandName() {   
 return brandName;  
 }  
  
 public void setBrandName(String brandName) {  
 this.brandName = brandName;  
 }  
  
 public LocalDate getArrivalDate() {  
 return arrivalDate;  
 }  
  
 public void setArrivalDate(LocalDate arrivalDate) {  
 this.arrivalDate = arrivalDate;  
 }  
  
 public int getSize() {  
 return size;  
 }  
  
 public void setSize(int size) {  
 this.size = size;  
 }  
   
 @Override  
 public int hashCode() {  
 return Objects.hash(arrivalDate, brandName, shoeId, shoeName, size);  
 }  
  
 @Override  
 public boolean equals(Object obj) {  
 if (this == obj)  
 return true;  
 if (obj == null)  
 return false;  
 if (getClass() != obj.getClass())  
 return false;  
 ShoeDetails other = (ShoeDetails) obj;  
 return Objects.equals(arrivalDate, other.arrivalDate) && Objects.equals(brandName, other.brandName)  
 && Objects.equals(shoeId, other.shoeId) && Objects.equals(shoeName, other.shoeName)  
 && size == other.size;  
 }  
  
  
   
 @Override  
 public String toString() {  
 return "ShoeDetails [shoeId:" + shoeId + ", shoeName:" + shoeName + ", brandName:" + brandName  
 + ", arrivalDate:" + arrivalDate + ", size:" + size + "]";  
 }  
}

import java.util.List;  
  
import java.util.stream.Stream;  
  
public class ShoeUtility {  
 public Stream<ShoeDetails> fetchShoesByBrand(List<ShoeDetails> shoeList, String brandName) {  
 //Fill the code here  
   
 }  
  
 public Stream<ShoeDetails> fetchShoesInAscendingOrderBySize(List<ShoeDetails> shoeList) {  
 //Fill the code here  
 }  
  
 public Stream<String> fetchUniqueShoes(List<ShoeDetails> shoeList) {  
 //Fill the code here  
   
  
 }  
}

import java.text.ParseException;  
import java.text.SimpleDateFormat;  
import java.time.LocalDate;  
import java.util.ArrayList;  
import java.util.Date;  
import java.util.List;  
import java.util.Scanner;  
import java.util.stream.Stream;  
import java.time.LocalDate;  
public class UserInterface {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.in);  
  
 List<ShoeDetails> shoedetails=new ArrayList<>();  
 System.out.println("Enter the number of Shoe details:");  
 int numberOfShoes = scanner.nextInt();  
 scanner.nextLine();  
 ShoeUtility su=new ShoeUtility();  
 SimpleDateFormat sdf=new SimpleDateFormat();  
 System.out.println("Enter the Shoe details:");  
 for (int i = 0; i < numberOfShoes; i++) {  
 String details = scanner.nextLine();  
 ShoeDetails d=new ShoeDetails();  
 String[] parts=details.split(":");  
 String shoeId=parts[0];  
 String shoeName=parts[1];  
 String brandName=parts[2];  
 LocalDate arrivalDate;  
 arrivalDate=LocalDate.parse(parts[3]);  
 int size=Integer.parseInt(parts[4]);  
 d.setShoeId(shoeId);  
 d.setShoeName(shoeName);  
 d.setBrandName(brandName);  
 d.setArrivalDate(arrivalDate);  
 d.setSize(size);  
   
 shoedetails.add(d);  
 }  
 //Fill the code here  
  
   
   
 }  
}

**Question 2**

**Problem Statement**

You are developing a Java application to manage and analyze sports matches. The application should consist of three primary components:

1. **Match Class**: Represents the details of a sports match, including its code, title, coordinator, date, and venue.
2. **MatchUtility Class**: Provides utility methods to filter and process a list of Match objects.
3. **UserInterface Class**: Manages user interaction, allowing users to input match details and utilize MatchUtility methods to display results.

**Class Descriptions:**

* **Match Class**:
  + **Attributes**: matchCode, matchTitle, coordinator, matchDate, and venue.
  + **Methods**: Getter and setter methods for each attribute, overridden equals(), hashCode(), and toString() methods.
* **MatchUtility Class**:
  + **Methods**:
    - getMatchesByCoordinator(List<Match> matchList, String coordinator): Returns a stream of Match objects filtered by the specified coordinator.
    - getMatchesInOrderByDate(List<Match> matchList): Returns a stream of Match objects sorted in ascending order of the match date.
    - getDistinctMatches(List<Match> matchList): Returns a stream of distinct match titles.
* **UserInterface Class**:
  + **Functionality**:
    - Prompts the user to input the number of matches and their details.
    - Uses MatchUtility methods to filter by coordinator, sort by date, and get distinct match titles.
    - Displays the results to the user.

**Task**:

1. Implement the Match, MatchUtility, and UserInterface classes as described.
2. Test the application using sample input to ensure it produces the correct output.

**Sample Input and Output**

**Sample Input:**

Enter the number of Matches:

3

Enter the Match details (matchCode:matchTitle:coordinator:yyyy-MM-dd:venue):

M001:Quarterfinal:John Doe:2024-09-15:Stadium A

M002:Semifinal:Jane Smith:2024-09-22:Stadium B

M003:Final:John Doe:2024-09-29:Stadium C

Enter coordinator name to filter matches:

John Doe

**Expected Output:**

Matches by coordinator:

M001 | Quarterfinal | John Doe | 2024-09-15 | Stadium A

M003 | Final | John Doe | 2024-09-29 | Stadium C

Matches in order by date:

M001 | Quarterfinal | John Doe | 2024-09-15 | Stadium A

M002 | Semifinal | Jane Smith | 2024-09-22 | Stadium B

M003 | Final | John Doe | 2024-09-29 | Stadium C

Distinct match titles:

Quarterfinal

Semifinal

Final

Boiler Plate

import java.time.LocalDate;  
import java.util.Objects;  
  
public class Match {  
 private String matchCode;  
 private String matchTitle;  
 private String coordinator;  
 private LocalDate matchDate;  
 private String venue;  
  
 public Match() {  
  
 }  
  
 public Match(String matchCode, String matchTitle, String coordinator, LocalDate matchDate, String venue) {  
 this.matchCode = matchCode;  
 this.matchTitle = matchTitle;  
 this.coordinator = coordinator;  
 this.matchDate = matchDate;  
 this.venue = venue;  
 }  
  
 public String getMatchCode() {  
 return matchCode;  
 }  
  
 public void setMatchCode(String matchCode) {  
 this.matchCode = matchCode;  
 }  
  
 public String getMatchTitle() {  
 return matchTitle;  
 }  
  
 public void setMatchTitle(String matchTitle) {  
 this.matchTitle = matchTitle;  
 }  
  
 public String getCoordinator() {  
 return coordinator;  
 }  
  
 public void setCoordinator(String coordinator) {  
 this.coordinator = coordinator;  
 }  
  
 public LocalDate getMatchDate() {  
 return matchDate;  
 }  
  
 public void setMatchDate(LocalDate matchDate) {  
 this.matchDate = matchDate;  
 }  
  
 public String getVenue() {  
 return venue;  
 }  
  
 public void setVenue(String venue) {  
 this.venue = venue;  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 Match match = (Match) o;  
 return Objects.equals(matchCode, match.matchCode) &&  
 Objects.equals(matchTitle, match.matchTitle) &&  
 Objects.equals(coordinator, match.coordinator) &&  
 Objects.equals(matchDate, match.matchDate) &&  
 Objects.equals(venue, match.venue);  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.hash(matchCode, matchTitle, coordinator, matchDate, venue);  
 }  
  
  
 @Override  
 public String toString() {  
 return matchCode+" | "+matchTitle+" | "+coordinator+" | "+matchDate+" | "+venue;  
 }  
}

import java.time.LocalDate;  
import java.util.ArrayList;  
import java.util.Comparator;  
import java.util.List;  
import java.util.stream.Stream;  
  
public class MatchUtility {  
  
 public Stream<Match> getMatchesByCoordinator(List<Match> matchList, String coordinator) {

}  
  
 public Stream<Match> getMatchesInOrderByDate(List<Match> matchList) {  
   
 }  
  
 public Stream<String> getDistinctMatches(List<Match> matchList) {  
 }  
}

import java.time.LocalDate;  
import java.util.ArrayList;  
import java.util.List;  
import java.util.Scanner;  
  
public class UserInterface {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.in);  
 MatchUtility matchUtility = new MatchUtility();  
 List<Match> matchList = new ArrayList<>();  
  
 System.out.println("Enter the number of Matches:");  
 int numberOfMatches = scanner.nextInt();  
 scanner.nextLine(); // Consume newline  
  
 System.out.println("Enter the Match details (matchCode:matchTitle:coordinator:yyyy-MM-dd:venue):");  
 for (int i = 0; i < numberOfMatches; i++) {  
 String[] matchDetails = scanner.nextLine().split(":");  
 if (matchDetails.length != 5) {  
 System.out.println("Invalid input. Please enter details in the format matchCode:matchTitle:coordinator:yyyy-MM-dd:venue");  
 i--; // Decrement counter to retry  
 continue;  
 }  
  
 String matchCode = matchDetails[0];  
 String matchTitle = matchDetails[1];  
 String coordinator = matchDetails[2];  
 LocalDate matchDate = LocalDate.parse(matchDetails[3]);  
 String venue = matchDetails[4];  
  
 Match match = new Match(matchCode, matchTitle, coordinator, matchDate, venue);  
 matchList.add(match);  
 }  
  
 //Fill the code  
 }  
}

Question 3

**Problem Statement**

You are tasked with developing a Java application to manage and query a list of training programs. The application consists of three main components:

1. **TrainingProgram Class**: Represents the details of a training program, including its name, trainer, topic, and duration.
2. **TrainingProgramUtility Class**: Provides methods to filter and retrieve training programs based on different criteria.
3. **UserInterface Class**: Manages user interaction, allowing users to input training program details and query the list using different criteria.

**Class Descriptions:**

* **TrainingProgram Class**:
  + **Attributes**: programName, trainerName, topic, and duration.
  + **Methods**: Getter and setter methods for each attribute.
* **TrainingProgramUtility Class**:
  + **Methods**:
    - retrieveProgramsByTrainer(Stream<TrainingProgram> programStream, String trainerName): Returns a list of TrainingProgram objects where the trainer's name matches the specified name (case-insensitive).
    - retrieveProgramsByTopic(Stream<TrainingProgram> programStream, String topic): Returns a list of TrainingProgram objects where the topic matches the specified topic (case-insensitive).
    - retrieveProgramsByDuration(Stream<TrainingProgram> programStream, int duration): Returns a list of TrainingProgram objects where the duration is greater than the specified duration.
* **UserInterface Class**:
  + **Functionality**:
    - Prompts the user to input the total number of training programs and their details.
    - Provides a menu for querying training programs based on trainer, topic, and duration.
    - Displays the results of these queries.

**Task**:

1. Implement the TrainingProgram, TrainingProgramUtility, and UserInterface classes as described.
2. Ensure that the TrainingProgramUtility class methods filter and retrieve data correctly.
3. Test the application using sample input to verify that it produces the correct output.

**Sample Input and Output**

**Sample Input:**

vbnet

Copy code

Enter the total number of training programs to add to the list:

3

Enter the training program details (programName, trainerName, topic, duration) separated by commas:

Java Basics, Alice Smith, Programming, 4

Advanced Java, Bob Johnson, Programming, 6

Python for Beginners, Alice Smith, Data Science, 3

Choose an option:

1. Retrieve programs by trainer

2. Retrieve programs by topic

3. Retrieve programs by duration

4. Exit

1

Enter trainer name:

Alice Smith

**Expected Output:**

csharp

Copy code

Programs by trainer Alice Smith:

Java Basics, Alice Smith, Programming, 4

Python for Beginners, Alice Smith, Data Science, 3

**Additional Sample Input and Output:**

**Sample Input:**

vbnet

Copy code

Choose an option:

2. Retrieve programs by topic

3. Retrieve programs by duration

4. Exit

2

Enter topic:

Programming

**Expected Output:**

Programs on topic Programming:

Java Basics, Alice Smith, Programming, 4

Advanced Java, Bob Johnson, Programming, 6

**Sample Input:**

Choose an option:

3. Retrieve programs by duration

4. Exit

3

Enter minimum duration:

4

**Expected Output:**

Programs with duration greater than 4:

Java Basics, Alice Smith, Programming, 4

Advanced Java, Bob Johnson, Programming, 6

Boiler Plate Code

public class TrainingProgram {  
 private String programName;  
 private String trainerName;  
 private String topic;  
 private int duration;  
  
 public TrainingProgram() {}  
  
 public TrainingProgram(String programName, String trainerName, String topic, int duration) {  
 this.programName = programName;  
 this.trainerName = trainerName;  
 this.topic = topic;  
 this.duration = duration;  
 }  
  
 public String getProgramName() {  
 return programName;  
 }  
  
 public void setProgramName(String programName) {  
 this.programName = programName;  
 }  
  
 public String getTrainerName() {  
 return trainerName;  
 }  
  
 public void setTrainerName(String trainerName) {  
 this.trainerName = trainerName;  
 }  
  
 public String getTopic() {  
 return topic;  
 }  
  
 public void setTopic(String topic) {  
 this.topic = topic;  
 }  
  
 public int getDuration() {  
 return duration;  
 }  
  
 public void setDuration(int duration) {  
 this.duration = duration;  
 }  
}

import java.util.List;  
import java.util.stream.Collectors;  
import java.util.stream.Stream;  
  
public class TrainingProgramUtility {  
  
 public List<TrainingProgram> retrieveProgramsByTrainer(Stream<TrainingProgram> programStream, String trainerName) {  
 //Fill the code here  
   
   
 }  
  
 public List<TrainingProgram> retrieveProgramsByTopic(Stream<TrainingProgram> programStream, String topic) {  
 //Fill the code here  
   
   
 }  
  
 public List<TrainingProgram> retrieveProgramsByDuration(Stream<TrainingProgram> programStream, int duration) {  
 //Fill the code here  
   
   
   
   
   
 }  
}

import java.util.ArrayList;  
import java.util.List;  
import java.util.Scanner;  
import java.util.stream.Stream;  
  
public class UserInterface {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.in);  
  
 System.out.println("Enter the total number of training programs to add to the list:");  
 int totalPrograms = Integer.parseInt(scanner.nextLine());  
  
 List<TrainingProgram> trainingProgramList = new ArrayList<>();  
  
 System.out.println("Enter the training program details (programName, trainerName, topic, duration) separated by commas:");  
 for (int i = 0; i < totalPrograms; i++) {  
 String[] details = scanner.nextLine().split(",");  
 String programName = details[0].trim();  
 String trainerName = details[1].trim();  
 String topic = details[2].trim();  
 int duration = Integer.parseInt(details[3].trim());  
  
 TrainingProgram program = new TrainingProgram(programName, trainerName, topic, duration);  
 trainingProgramList.add(program);  
 }  
  
 //Fill the code  
 }  
}