**1. import java.util.\*;**

* This line imports the entire java.util package, which contains various utility classes like Scanner, LinkedList, ArrayList, and Arrays that are used in this program.

**2. class Student {**

* This line defines a class named Student, which serves as a blueprint for creating student objects. Each student object will store information about an individual student.

**3. int rollNo;**

* This line declares an integer variable rollNo within the Student class. This variable will store the roll number of a student.

**4. String name;**

* This line declares a string variable name within the Student class. This variable will store the name of a student.

**5. String course;**

* This line declares a string variable course within the Student class. This variable will store the course enrolled by a student.

**6. float fees=100000;**

* This line declares a float variable fees within the Student class, initializing it to 100000. This variable will store the fees of a student.

**7. public Student(int rollNo, String name, String course, float fees) {**

* This line defines a constructor for the Student class. A constructor is a special method that is automatically called when a new object of the class is created. This constructor takes the roll number, name, course, and fees as input and initializes the corresponding instance variables of the Student object.

**8. this.rollNo = rollNo;**

* Inside the constructor, this line assigns the value of the rollNo parameter to the rollNo instance variable of the object being created.

**9. this.name = name;**

* This line assigns the value of the name parameter to the name instance variable of the object being created.

**10. this.course = course;**

* This line assigns the value of the course parameter to the course instance variable of the object being created.

**11. this.fees = this.fees - fees;**

* This line calculates the remaining fees by subtracting the provided fees from the initial fees of 100000 and assigns the result to the fees instance variable of the object being created. Note: This calculation appears to be incorrect as it always sets the fees to 0. It should likely be this.fees = fees; to directly assign the provided fees.

**12. }**

* This line closes the Student constructor.

**13. class CollegeManagementSystem {**

* This line defines a class named CollegeManagementSystem, which contains methods to manage student information.

**14. public static LinkedList<Student> students = new LinkedList<>();**

* This line declares a static LinkedList named students, which will store Student objects. A LinkedList is a data structure that allows efficient insertion and deletion of elements.

**15. static ArrayList<Integer> roll = new ArrayList<>();**

* This line declares a static ArrayList named roll, which will store integers representing roll numbers. An ArrayList is a resizable array that allows dynamic addition and removal of elements.

**16. public static int[] fees = {10000, 20000, 30000};**

* This line declares a static array named fees containing three integers representing different fee amounts.

**17. public static String[] courses = {"CSE", "ECE", "MECH", "AI&DS"};**

* This line declares a static array named courses containing four strings representing different course names.

**18. static ArrayList<String> coursess = new ArrayList<>(Arrays.asList(courses));**

* This line creates a static ArrayList named coursess by converting the courses array into a list using Arrays.asList() and then passing it to the ArrayList constructor.

**19. public static String[] subjects = {"Maths", "Physics", "Chemistry", "English", "Computer Science"};**

* This line declares a static array named subjects containing five strings representing different subject names.

**20. public static int[][] studentMarks = new int[100][5];**

* This line declares a static two-dimensional array named studentMarks to store marks of up to 100 students in 5 subjects.

**21. public static int numStudents = 0;**

* This line declares a static integer variable numStudents initialized to 0 to track the current number of students in the system.

**22. static Scanner sc = new Scanner(System.in);**

* This line creates a static Scanner object named sc to read input from the user through the console.

**23. public void addStudent(int rollNo, String name, String course, float fees) {**

* This line defines a method named addStudent that takes the roll number, name, course, and fees as input and adds a new student to the system.

**24. Student student = new Student(rollNo, name, course, fees);**

* Inside the addStudent method, this line creates a new Student object using the provided information and assigns it to the variable student.

**25. students.add(student);**

* This line adds the newly created Student object to the students linked list.

**26. studentMarks[numStudents] = new int[5];**

* This line creates a new array of integers of size 5 within the studentMarks array at the index numStudents to store marks for the new student.

**27. numStudents++;**

* This line increments the numStudents counter to reflect the addition of a new student.

**28. }**

* This line closes the addStudent method.

**29. public void displayStudents() {**

* This line defines a method named displayStudents that displays the information of all registered students.

**30. System.out.println("Roll No\tName\tCourse\tFees");**

* Inside the displayStudents method, this line prints the header for the student information table.

**31. for (Student student : students) {**

* This line starts a for-each loop that iterates through each Student object in the students linked list.

**32. System.out.println(student.rollNo + "\t" + student.name + "\t" + student.course + "\t" + student.fees);**

* Inside the loop, this line prints the roll number, name, course, and fees of the current student in a tabular format.

**33. }**

* This line closes the for-each loop.

**34. }**

* This line closes the displayStudents method.

**35. public void displayCourses() {**

* This line defines a method named displayCourses that displays the list of available courses.

**36. System.out.println("Courses offered:");**

* Inside the displayCourses method, this line prints a message indicating the display of available courses.

**37. for (String course : courses) {**

* This line starts a for-each loop that iterates through each string in the courses array.

**38. System.out.println(course);**

* Inside the loop, this line prints the current course name.

**39. }**

* This line closes the for-each loop.

**40. }**

* This line closes the displayCourses method.

**41. public void displayFees() {**

* This line defines a method named displayFees that displays the fees of a student based on their roll number and allows payment.

**42. System.out.println("Enter roll number of student");**

* Inside the displayFees method, this line prompts the user to enter a roll number.

**43. int rollNo = sc.nextInt();**

* This line reads the roll number entered by the user using the Scanner object sc and stores it in the variable rollNo.

**44. Student cstudent = null;**

* This line declares a Student variable cstudent and initializes it to null. This variable will store the student object corresponding to the entered roll number.

**45. for (Student a : students) {**

* This line starts a for-each loop that iterates through each Student object in the students linked list.

**46. if (a.rollNo == rollNo)**

* Inside the loop, this line checks if the roll number of the current student (a.rollNo) matches the entered roll number (rollNo).

**47. cstudent = a;**

* If the roll numbers match, this line assigns the current student object (a) to cstudent, indicating that the corresponding student has been found.

**48. }**

* This line closes the for-each loop.

**49. if (cstudent == null)**

* This line checks if cstudent is still null, which means that no student was found with the entered roll number.

**50. System.out.println("No such student registered");**

* If no student was found, this line prints a message indicating that no student is registered with the entered roll number.

**51. else {**

* This line starts an else block that executes if a student with the entered roll number was found.

**52. System.out.println("Fees to be paid = " + cstudent.fees);**

* Inside the else block, this line prints the fees of the found student.

**53. System.out.println("Enter the amount you want to pay");**

* This line prompts the user to enter the amount they want to pay.

**54. float amount = sc.nextFloat();**

* This line reads the amount entered by the user using the Scanner object sc and stores it in the variable amount.

**55. if (amount > cstudent.fees) {**

* This line checks if the entered amount is greater than the student's fees.

**56. System.out.println("Excess amount paid\nRefund = " + (amount - cstudent.fees));**

* If the amount is greater, this line prints messages indicating the excess amount and the refund amount.

**57. cstudent.fees = 0;**

* This line sets the student's fees to 0 since the excess amount has been paid.

**58. } else {**

* This line starts an else block that executes if the entered amount is less than or equal to the student's fees.

**59. cstudent.fees -= amount;**

* Inside the else block, this line subtracts the paid amount from the student's fees, updating the remaining fees.

**60. System.out.println("Amount paid successfully\nRemaining fees = " + cstudent.fees);**

* This line prints messages indicating successful payment and the remaining fees.

**61. }**

* This line closes the inner else block.

**62. }**

* This line closes the outer else block.

**63. }**

* This line closes the displayFees method.

**64. public void enterMarks(int rollNo, int marks[]) {**

* This line defines a method named enterMarks that takes a roll number and an array of marks as input and stores the marks for the corresponding student.

**65. for (int i = 0; i < 5; i++) {**

* Inside the enterMarks method, this line starts a for loop that iterates 5 times, representing the 5 subjects.

**66. studentMarks[rollNo - 1][i] = marks[i];**

* Inside the loop, this line assigns the mark for the current subject from the marks array to the corresponding position in the studentMarks array. The rollNo - 1 is used to adjust the index since array indices start from 0.

**67. }**

* This line closes the for loop.

**68. }**

* This line closes the enterMarks method.

**69. public void displayMarks(int rollNo) {**

* This line defines a method named displayMarks that takes a roll number as input and displays the marks of the corresponding student.

**70. System.out.print("Roll No ");**

* Inside the displayMarks method, this line prints the label "Roll No".

**71. int total = 0;**

* This line declares an integer variable total and initializes it to 0 to store the total marks.

**72. System.out.println(rollNo);**

* This line prints the roll number of the student.

**73. for (int i = 0; i < 5; i++) {**

* This line starts a for loop that iterates 5 times, representing the 5 subjects.

**74. total += studentMarks[rollNo - 1][i];**

* Inside the loop, this line adds the mark for the current subject from the studentMarks array to the total.

**75. System.out.println(subjects[i] + " " + studentMarks[rollNo - 1][i]);**

* This line prints the name of the current subject and the corresponding mark.

**76. }**

* This line closes the for loop.

**77. System.out.println();**

* This line prints a blank line.

**78. System.out.println("Total: " + total);**

* This line prints the calculated total marks.

**79. }**

* This line closes the displayMarks method.

**80. public void calculatePercentage(int rollNo) {**

* This line defines a method named calculatePercentage that takes a roll number as input and calculates the percentage of marks obtained by the corresponding student.

**81. int total = 0;**

* Inside the calculatePercentage method, this line declares an integer variable total and initializes it to 0 to store the total marks.

**82. for (int i = 0; i < 5; i++) {**

* This line starts a for loop that iterates 5 times, representing the 5 subjects.

**83. total += studentMarks[rollNo - 1][i];**

* Inside the loop, this line adds the mark for the current subject from the studentMarks array to the total.

**84. }**

* This line closes the for loop.

**85. float percentage = (float) total / 500 \* 100;**

* This line calculates the percentage of marks by dividing the total by 500 (assuming each subject has a maximum mark of 100) and multiplying by 100. The (float) casting is used to ensure that the division is performed as a floating-point operation.

**86. System.out.println("Percentage: " + percentage + "%");**

* This line prints the calculated percentage.

**87. }**

* This line closes the calculatePercentage method.

**88. public static void main(String[] args) {**

* This line defines the main method, which is the entry point of the Java program.

**89. CollegeManagementSystem cms = new CollegeManagementSystem();**

* Inside the main method, this line creates an object of the CollegeManagementSystem class named cms.

**90. int choice;**

* This line declares an integer variable choice to store the user's choice from the menu.

**91. do {**

* This line starts a do-while loop that will continue to execute as long as the user does not choose to exit.

**92. System.out.println("\nCollege Management System\n1. Add Student\n2. Display Students\n3. Display Courses\n4. Display Fees\n5. Enter Marks\n6. Display Marks\n7. Calculate Percentage\n8. Exit");**

* Inside the loop, this line prints the menu of options for the user.

**93. choice = sc.nextInt();**

* This line reads the user's choice using the Scanner object sc and stores it in the choice variable.

**94. switch (choice) {**

* This line starts a switch statement that will execute different code blocks based on the user's choice.

**95. case 1:**

* This case handles the option to add a new student.

**96. System.out.print("Enter roll number: ");**

* This line prompts the user to enter a roll number.

**97. int rollNo = sc.nextInt();**

* This line reads the entered roll number and stores it in the variable rollNo.

**98. while (roll.contains(rollNo) == true) {**

* This line starts a while loop that continues as long as the entered roll number already exists in the roll array.

**99. System.out.println("Roll no already exist\nEnter different Roll Number");**

* Inside the while loop, this line prints a message indicating that the entered roll number already exists and prompts the user to enter a different one.

**100. rollNo = sc.nextInt();**

* This line reads the new roll number entered by the user.

**101. }**

* This line closes the while loop.

**102. roll.add(rollNo);**

* This line adds the entered roll number to the roll array.

**103. System.out.print("Enter name: ");**

* This line prompts the user to enter the student's name.

**104. sc.nextLine();**

* This line consumes the newline character from the previous input to prevent issues with reading the next line.

**105. String name = sc.nextLine();**

* This line reads the entered name and stores it in the variable name.

**106. System.out.print("Enter course: ");**

* This line prompts the user to enter the student's course.

**107. String Ccourse;**

* This line declares a string variable Ccourse to store the entered course.

**108. int chc;**

* This line declares an integer variable chc to store the user's choice for course and fees.

**109. do {**

* This line starts a do-while loop to repeatedly prompt the user for a course choice until a valid choice is entered.

**110. System.out.println("\nChoose one between this ");**

* This line prints a message asking the user to choose a course from the available options.

**111. for (int i = 0; i < coursess.size(); i++) {**

* This line starts a for loop that iterates through the coursess array.

**112. System.out.println((i + 1) + " for " + coursess.get(i));**

* Inside the loop, this line prints the course options with their corresponding numbers.

**113. }**

* This line closes the for loop.

**114. System.out.println();**

* This line prints a blank line.

**115. chc = sc.nextInt();**

* This line reads the user's choice for course and stores it in the chc variable.

**116. chc -= 1;**

* This line subtracts 1 from the chc variable to adjust for zero-based indexing.

**117. } while (!(chc >= 0 && chc <= 3));**

* This line continues the do-while loop until the user enters a valid choice between 0 and 3.

**118. Ccourse = coursess.get(chc);**

* This line assigns the chosen course from the coursess array to the Ccourse variable.

**119. System.out.print("Enter fees: ");**

* This line prompts the user to enter the student's fees.

**120. chc = 0;**

* This line resets the chc variable to 0 for use in fee selection.

**121. do {**

* This line starts a do-while loop to repeatedly prompt the user for a fee choice until a valid choice is entered.

**122. System.out.println("\nChoose one between this ");**

* This line prints a message asking the user to choose a fee option from the available options.

**123. for (int i = 0; i < fees.length; i++) {**

* This line starts a for loop that iterates through the fees array.

**124. System.out.println((i + 1) + " for " + fees[i]);**

* Inside the loop, this line prints the fee options with their corresponding numbers.

**125. }**

* This line closes the for loop.

**126. System.out.println();**

* This line prints a blank line.

**127. chc = sc.nextInt();**

* This line reads the user's choice for fee and stores it in the chc variable.

**128. chc -= 1;**

* This line subtracts 1 from the chc variable to adjust for zero-based indexing.

**129. } while (!(chc >= 0 && chc <= 2));**

* This line continues the do-while loop until the user enters a valid choice between 0 and 2.

**130. float fee = fees[chc];**

* This line assigns the chosen fee from the fees array to the fee variable.

**131. cms.addStudent(rollNo, name, Ccourse, fee);**

* This line calls the addStudent method of the cms object to add the new student with the entered information.

**132. break;**

* This line exits the case 1 block.

**133. case 2:**

* This case handles the option to display all students.

**134. cms.displayStudents();**

* This line calls the displayStudents method of the cms object to display the information of all registered students.

**135. break;**

* This line exits the case 2 block.

**136. case 3:**

* This case handles the option to display all courses.

**137. cms.displayCourses();**

* This line calls the displayCourses method of the cms object to display the list of available courses.

**138. break;**

* This line exits the case 3 block.

**139. case 4:**

* This case handles the option to display the fees of a student and allow payment.

**140. cms.displayFees();**

* This line calls the displayFees method of the cms object to display and process the fees of a student based on their roll number.

**141. break;**

* This line exits the case 4 block.

**142. case 5:**

* This case handles the option to enter marks for a student.

**143. System.out.print("Enter roll number: ");**

* This line prompts the user to enter a roll number.

**144. int rollNoToEnterMarks = sc.nextInt();**

* This line reads the entered roll number and stores it in the variable rollNoToEnterMarks.

**145. System.out.println("Enter marks for 5 subjects:\n");**

* This line prints a message asking the user to enter marks for 5 subjects.

**146. int marks[] = new int[5];**

* This line declares an integer array named marks of size 5 to store the entered marks.

**147. for (int i = 0; i < 5; i++) {**

* This line starts a for loop that iterates 5 times, representing the 5 subjects.

**148. System.out.println("Enter marks for " + subjects[i]);**

* Inside the loop, this line prompts the user to enter the mark for the current subject.

**149. marks[i] = sc.nextInt();**

* This line reads the entered mark and stores it in the marks array.

**150. System.out.println();**

* This line prints a blank line.

**151. }**

* This line closes the for loop.

**152. cms.enterMarks(rollNoToEnterMarks, marks);**

* This line calls the enterMarks method of the cms object to store the entered marks for the student with the entered roll number.

**153. break;**

* This line exits the case 5 block.

**154. case 6:**

* This case handles the option to display marks of a student.

**155. System.out.print("Enter roll number: ");**

* This line prompts the user to enter a roll number.

**156. int rollNoToDisplayMarks = sc.nextInt();**

* This line reads the entered roll number and stores it in the variable rollNoToDisplayMarks.

**157. cms.displayMarks(rollNoToDisplayMarks);**

* This line calls the displayMarks method of the cms object to display the marks of the student with the entered roll number.

**158. break;**

* This line exits the case 6 block.

**159. case 7:**

* This case handles the option to calculate the percentage of marks of a student.

**160. System.out.print("Enter roll number: ");**

* This line prompts the user to enter a roll number.

**161. int rollNoToCalculatePercentage = sc.nextInt();**

* This line reads the entered roll number and stores it in the variable rollNoToCalculatePercentage.

**162. cms.calculatePercentage(rollNoToCalculatePercentage);**

* This line calls the calculatePercentage method of the cms object to calculate and display the percentage of marks obtained by the student with the entered roll number.

**163. break;**

* This line exits the case 7 block.

**164. case 8:**

* This case handles the option to exit the program.

**165. System.out.println("Exiting...");**

* This line prints a message indicating that the program is exiting.

**166. break;**

* This line exits the case 8 block.

**167. default:**

* This case handles invalid choices.

**168. System.out.println("Invalid choice!");**

* This line prints a message indicating an invalid choice.

**169. }**

* This line closes the switch statement.

**170. } while (choice != 8);**

* This line continues the do-while loop until the user chooses to exit (choice 8).

**171. }**

* This line closes the main method.

**172. }**

* This line closes the CollegeManagementSystem class.