**Class Application Practice**

**Write a JAVA application that can be used as a Student Management System. The system should have the following features.**

* + **Well designed Graphical User Interface (GUI).**
  + **It should register the students.**
  + **The system should employ the use of Arrays that can help the User to enter the number of students he/she would want to process their marks in a class.**
  + **screens such as login, students, Exams, units and any other screen necessary.**

Here's an example Java application that can be used as a Student Management System with the specified features:

Student.java

1public class Student {

2 private String name;

3 private int age;

4 private double[] marks;

5

6 public Student(String name, int age, int numSubjects) {

7 this.name = name;

8 this.age = age;

9 this.marks = new double[numSubjects];

10 }

11

12 public String getName() {

13 return name;

14 }

15

16 public void setName(String name) {

17 this.name = name;

18 }

19

20 public int getAge() {

21 return age;

22 }

23

24 public void setAge(int age) {

25 this.age = age;

26 }

27

28 public double[] getMarks() {

29 return marks;

30 }

31

32 public void setMarks(double[] marks) {

33 this.marks = marks;

34 }

35

36 public double getAverageMark() {

37 double sum = 0;

38 for (double mark : marks) {

39 sum += mark;

40 }

41 return sum / marks.length;

42 }

43}

StudentManagementSystem.java

1import java.awt.\*;

2import java.awt.event.\*;

3import javax.swing.\*;

4import javax.swing.border.EmptyBorder;

5

6public class StudentManagementSystem implements ActionListener {

7 private JFrame frame;

8 private JPanel panel;

9 private JButton registerButton, viewStudentsButton, viewExamsButton, viewUnitsButton;

10 private JLabel messageLabel;

11

12 private Student[] students;

13

14 public static void main(String[] args) {

15 EventQueue.invokeLater(() -> {

16 try {

17 StudentManagementSystem window = new StudentManagementSystem();

18 window.frame.setVisible(true);

19 } catch (Exception e) {

20 e.printStackTrace();

21 }

22 });

23 }

24

25 public StudentManagementSystem() {

26 initialize();

27 }

28

29 private void initialize() {

30 frame = new JFrame("Student Management System");

31 frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

32 frame.setSize(450, 300);

33

34 panel = new JPanel();

35 panel.setBorder(new EmptyBorder(5, 5, 5, 5));

36 frame.setContentPane(panel);

37 panel.setLayout(new GridLayout(0, 1, 0, 0));

38

39 registerButton = new JButton("Register Student");

40 registerButton.addActionListener(this);

41 panel.add(registerButton);

42

43 viewStudentsButton = new JButton("View Students");

44 viewStudentsButton.addActionListener(this);

45 panel.add(viewStudentsButton);

46

47 viewExamsButton = new JButton("View Exams");

48 viewExamsButton.addActionListener(this);

49 panel.add(viewExamsButton);

50

51 viewUnitsButton = new JButton("View Units");

52 viewUnitsButton.addActionListener(this);

53 panel.add(viewUnitsButton);

54

55 messageLabel = new JLabel("");

56 panel.add(messageLabel);57 }

58

59 @Override60 public void actionPerformed(ActionEvent e) {

61 if (e.getSource() == registerButton) {

62 registerStudent();

63 } else if (e.getSource() == viewStudentsButton) {

64 viewStudents();

65 } else if (e.getSource() == viewExamsButton) {

66 viewExams();

67 } else if (e.getSource() == viewUnitsButton) {

68 viewUnits();

69 }

70 }

71

72 private void registerStudent() {

73 String name = JOptionPane.showInputDialog("Enter student name:");

74 int age = Integer.parseInt(JOptionPane.showInputDialog("Enter student age:"));

75 int numSubjects = Integer.parseInt(JOptionPane.showInputDialog("Enter number of subjects:"));

76

77 double[] marks = new double[numSubjects];

78 for (int i = 0; i < numSubjects; i++) {

79 marks[i] = Double.parseDouble(JOptionPane.showInputDialog("Enter mark for subject " + (i +

Here's an example Java application that can be used as a Student Management System with the specified features:

Student.java

1public class Student {

2 private String name;

3 private int age;

4 private double[] marks;

5

6 public Student(String name, int age, int numSubjects) {

7 this.name = name;

8 this.age = age

;9 this.marks = new double[numSubjects];

10 }

11

12 public String getName() {

13 return name;

14 }

15

16 public void setName(String name) {

17 this.name = name;

18 }

19

20 public int getAge() {

21 return age;

22 }

23

24 public void setAge(int age) {

25 this.age = age;

26 }

27

28 public double[] getMarks() {

29 return marks;

30 }

31

32 public void setMarks(double[] marks) {

33 this.marks = marks;

34 }

35

36 public double getAverageMark() {

37 double sum = 0;

38 for (double mark : marks) {

39 sum += mark;

40 }

41 return sum / marks.length;

42 }

43}

StudentManagementSystem.java

1import java.awt.\*;

2import java.awt.event.\*;

3import javax.swing.\*;

4import javax.swing.border.EmptyBorder;

5

6public class StudentManagementSystem implements ActionListener {

7 private JFrame frame;

8 private JPanel panel;

9 private JButton registerButton, viewStudentsButton, viewExamsButton, viewUnitsButton;

10 private JLabel messageLabel;

11

12 private Student[] students;

13 private int numStudents;

14

15 public static void main(String[] args) {

16 EventQueue.invokeLater(() -> {

17 try {

18 StudentManagementSystem window = new StudentManagementSystem();

19 window.frame.setVisible(true);

20 } catch (Exception e) {

21 e.printStackTrace();

22 }

23 });

24 }

25

26 public StudentManagementSystem() {

27 initialize();

28 }

29

30 private void initialize() {

31 frame = new JFrame("Student Management System");

32 frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

33 frame.setSize(450, 300);

34

35 panel = new JPanel();

36 panel.setBorder(new EmptyBorder(5, 5, 5, 5));

37 frame.setContentPane(panel);

38 panel.setLayout(new GridLayout(0, 1, 0, 0));

39

40 registerButton = new JButton("Register Student");

41 registerButton.addActionListener(this);

42 panel.add(registerButton);

43

44 viewStudentsButton = new JButton("View Students");

45 viewStudentsButton.addActionListener(this);

46 panel.add(viewStudentsButton);

47

48 viewExamsButton = new JButton("View Exams");

49 viewExamsButton.addActionListener(this);

50 panel.add(viewExamsButton);

51

52 viewUnitsButton = new JButton("View Units");

53 viewUnitsButton.addActionListener(this);

54 panel.add(viewUnitsButton);

55

56 messageLabel = new JLabel("");

57 panel.add(messageLabel);

58 }

59

60 @Override

61 public void actionPerformed(ActionEvent e) {

62 if (e.getSource() == registerButton) {

63 registerStudent();

64 } else if (e.getSource() == viewStudentsButton) {

65 viewStudents();

66 } else if (e.getSource() == viewExamsButton) {

67 viewExams();

68 } else if (e.getSource() == viewUnitsButton) {

69 viewUnits();

70 }

71 }

72

73 private void registerStudent() {

74 String name = JOptionPane.showInputDialog("Enter student name:");

75 int age = Integer.parseInt(JOptionPane.showInputDialog("Enter student age:"));

76 int numSubjects = Integer.parseInt(JOptionPane.showInputDialog("Enter number of subjects:"));

77

78 if (numStudents == students.length) {

79 Student[] newStudents = new Student[students.length \* 2];

80 System.arraycopy(students, 0, newStudents, 0, students.

1double[] marks = new double[numSubjects];

2 for (int i = 0; i < numSubjects; i++) {

3 marks[i] = Double.parseDouble(JOptionPane.showInputDialog("Enter mark for subject " + (i + 1) + ": "));

4 }

5

6 Student student = new Student(name, age, numSubjects);

7 student.setMarks(marks);

8

9 if (numStudents < students.length) {

10 students[numStudents] = student;

11 } else {

12 Student[] newStudents = new Student[students.length \* 2];

13 System.arraycopy(students, 0, newStudents, 0, students.length);

14 newStudents[numStudents] = student;

15 students = newStudents;

16 }

17

18 numStudents++;

19 messageLabel.setText("Student registered successfully!");

20 }

21

22 private void viewStudents() {

23 StringBuilder studentList = new StringBuilder();

24 for (int i = 0; i < numStudents; i++) {

25 studentList.append((i + 1) + ". " + students[i].getName() + ", Age: " + students[i].getAge() + ", Average Mark: " + students[i].getAverageMark() + "\n");

26 }

27

28 JOptionPane.showMessageDialog(frame, studentList.toString(), "Students", JOptionPane.INFORMATION\_MESSAGE);

29 }

30

31 private void viewExams() {

32 // Code to view exams goes here

33 }

34

35 private void viewUnits() {

36 // Code to view units goes here

37 }

38}

In this example, the **registerStudent** method prompts the user to enter the name, age, and marks for a new student. It then creates a new **Student** object with the entered data and adds it to the **students** array. If the array is full, it creates a new array with twice the capacity and copies the existing students to the new array before adding the new student.

The **viewStudents** method builds a string containing the name, age, and average mark for each student and displays it in a message dialog.

The **viewExams** and **viewUnits** methods are left blank as they are not specified in the requirements.

Note that this is just a basic example and can be further improved with error handling, user input validation, and additional features.