```
import java.util.*;

// Base class for common attributes
class CollegeEntity {
    protected String code;
    protected String name;

    public CollegeEntity(String code, String name) {
        this.code = code;
        this.name = name;
    }

    public String getCode() { return code; }
    public String getName() { return name; }
}

// Department class
class Department extends CollegeEntity {
    private String building;
    private int yearFounded;

    public Department(String deptCode, String deptName, String building, int yearFounded) {
        super(deptCode, deptName);
        this.building = building;
        this.yearFounded = yearFounded;
    }

    public String getBuilding() { return building; }
    public int getYearFounded() { return yearFounded; }

    @Override
    public String toString() {
        return String.format("Department[Code: %s, Name: %s, Building: %s, Founded: %d]",
                             code, name, building, yearFounded);
    }
}

// Teaching Staff class
class Teaching_Staff extends CollegeEntity {
    private String qualification;
    private String dateOfJoining;
    private String departmentCode;

    public Teaching_Staff(String staffId, String staffName, String qualification,
                          String dateOfJoining, String departmentCode) {
        super(staffId, staffName);
        this.qualification = qualification;
        this.dateOfJoining = dateOfJoining;
        this.departmentCode = departmentCode;
    }

    public String getQualification() { return qualification; }
    public String getDateOfJoining() { return dateOfJoining; }
    public String getDepartmentCode() { return departmentCode; }

    @Override
    public String toString() {
        return String.format("Teaching Staff[ID: %s, Name: %s, Qualification: %s, Joined: %s,
Dept: %s]",
                             code, name, qualification, dateOfJoining, departmentCode);
    }
```

```
}

// Student Details class
class Student_Details extends CollegeEntity {
    private String mobileNumber;
    private String email;
    private String departmentCode;
    private String[] teacherCodes; // Array for 4 subject teachers

    public Student_Details(String studentId, String studentName, String mobileNumber,
                           String email, String departmentCode, String[] teacherCodes) {
        super(studentId, studentName);
        this.mobileNumber = mobileNumber;
        this.email = email;
        this.departmentCode = departmentCode;
        this.teacherCodes = teacherCodes;
    }

    public String getMobileNumber() { return mobileNumber; }
    public String getEmail() { return email; }
    public String getDepartmentCode() { return departmentCode; }
    public String[] getTeacherCodes() { return teacherCodes; }

    @Override
    public String toString() {
        return String.format("Student[ID: %s, Name: %s, Mobile: %s, Email: %s, Dept: %s]",
                             code, name, mobileNumber, email, departmentCode);
    }

    public void displayWithTeachers() {
        System.out.println(this);
        System.out.print("  Subject Teachers: ");
        for (int i = 0; i < teacherCodes.length; i++) {
            System.out.print("Subject " + (i+1) + ": " + teacherCodes[i]);
            if (i < teacherCodes.length - 1) System.out.print(", ");
        }
        System.out.println();
    }
}

// Student Marks class
class Student_Marks {
    private String studentId;
    private int semesterNumber;
    private int[] marks; // Marks in 4 subjects

    public Student_Marks(String studentId, int semesterNumber, int[] marks) {
        this.studentId = studentId;
        this.semesterNumber = semesterNumber;
        this.marks = marks;
    }

    public String getStudentId() { return studentId; }
    public int getSemesterNumber() { return semesterNumber; }
    public int[] getMarks() { return marks; }

    // Method to calculate total marks
    public int calculateTotal() {
        int total = 0;
        for (int mark : marks) {
            total += mark;
        }
```

```
        return total;
    }

    // Method to calculate average marks
    public double calculateAverage() {
        return calculateTotal() / (double) marks.length;
    }

    // Method to get grade based on average
    public String getGrade() {
        double average = calculateAverage();
        if (average >= 90) return "A+";
        else if (average >= 80) return "A";
        else if (average >= 70) return "B+";
        else if (average >= 60) return "B";
        else if (average >= 50) return "C";
        else return "F";
    }

    @Override
    public String toString() {
        return String.format("Student Marks[ID: %s, Semester: %d, Marks: %s, Total: %d, Average:
%.2f, Grade: %s]",
                             studentId, semesterNumber, Arrays.toString(marks),
                             calculateTotal(), calculateAverage(), getGrade());
    }
}

// Main class to test the system
public class CollegeManagementSystem {
    private List<Department> departments;
    private List<Teaching_Staff> teachingStaff;
    private List<Student_Details> students;
    private List<Student_Marks> studentMarks;

    public CollegeManagementSystem() {
        departments = new ArrayList<>();
        teachingStaff = new ArrayList<>();
        students = new ArrayList<>();
        studentMarks = new ArrayList<>();
        initializeData();
    }

    private void initializeData() {
        // Initialize Departments
        departments.add(new Department("CS", "Computer Science", "Tech Building", 1990));
        departments.add(new Department("ME", "Mechanical Engineering", "Engineering Block",
1985));
        departments.add(new Department("EE", "Electrical Engineering", "Power Block", 1988));

        // Initialize Teaching Staff
        teachingStaff.add(new Teaching_Staff("T001", "Dr. Smith", "Ph.D", "2010-08-15", "CS"));
        teachingStaff.add(new Teaching_Staff("T002", "Prof. Johnson", "M.Tech", "2015-06-20",
"CS"));
        teachingStaff.add(new Teaching_Staff("T003", "Dr. Brown", "Ph.D", "2008-03-10", "ME"));
        teachingStaff.add(new Teaching_Staff("T004", "Prof. Davis", "M.E", "2012-11-05", "EE"));
        teachingStaff.add(new Teaching_Staff("T005", "Dr. Wilson", "Ph.D", "2005-09-12", "CS"));

        // Initialize Students with 4 subject teachers each
        students.add(new Student_Details("S001", "Alice Johnson", "9876543210",
                                         "alice@college.edu", "CS",
                                         new String[]{"T001", "T002", "T005", "T001"}));
```

```
        students.add(new Student_Details("S002", "Bob Smith", "9876543211",
                                         "bob@college.edu", "CS",
                                         new String[]{"T002", "T001", "T005", "T002"}));
        students.add(new Student_Details("S003", "Carol Davis", "9876543212",
                                         "carol@college.edu", "ME",
                                         new String[]{"T003", "T003", "T001", "T004"}));
        students.add(new Student_Details("S004", "David Wilson", "9876543213",
                                         "david@college.edu", "EE",
                                         new String[]{"T004", "T002", "T003", "T004"}));
        students.add(new Student_Details("S005", "Eva Brown", "9876543214",
                                         "eva@college.edu", "CS",
                                         new String[]{"T001", "T005", "T002", "T005"}));

        // Initialize Student Marks for Semester 1
        studentMarks.add(new Student_Marks("S001", 1, new int[]{85, 92, 78, 88}));
        studentMarks.add(new Student_Marks("S002", 1, new int[]{76, 82, 90, 79}));
        studentMarks.add(new Student_Marks("S003", 1, new int[]{92, 88, 85, 90}));
        studentMarks.add(new Student_Marks("S004", 1, new int[]{65, 72, 68, 74}));
        studentMarks.add(new Student_Marks("S005", 1, new int[]{88, 95, 82, 91}));
    }

    public void displayAllDepartments() {
        System.out.println("=== ALL DEPARTMENTS ===");
        for (Department dept : departments) {
            System.out.println(dept);
        }
        System.out.println();
    }

    public void displayAllTeachingStaff() {
        System.out.println("=== ALL TEACHING STAFF ===");
        for (Teaching_Staff staff : teachingStaff) {
            System.out.println(staff);
        }
        System.out.println();
    }

    public void displayAllStudents() {
        System.out.println("=== ALL STUDENTS ===");
        for (Student_Details student : students) {
            student.displayWithTeachers();
        }
        System.out.println();
    }

    public void displayStudentMarksAndResults() {
        System.out.println("=== STUDENT MARKS AND RESULTS - SEMESTER 1 ===");
        System.out.printf("%-10s %-15s %-8s %-25s %-8s %-8s %-5s%n",
                          "Student ID", "Name", "Dept", "Marks (4 Subjects)", "Total", "Average",
"Grade");
        System.out.println("=".repeat(90));

        for (Student_Marks marks : studentMarks) {
            Student_Details student = findStudentById(marks.getStudentId());
            if (student != null) {
                System.out.printf("%-10s %-15s %-8s %-25s %-8d %-8.2f %-5s%n",
                                  marks.getStudentId(),
                                  student.getName(),
                                  student.getDepartmentCode(),
                                  Arrays.toString(marks.getMarks()),
                                  marks.calculateTotal(),
                                  marks.calculateAverage(),
```

```
                                marks.getGrade());
            }
        }
        System.out.println();
    }

    public void displayDepartmentWiseResults() {
        System.out.println("=== DEPARTMENT-WISE RESULTS ===");

        Map<String, List<Student_Marks>> deptMarks = new HashMap<>();

        // Group marks by department
        for (Student_Marks marks : studentMarks) {
            Student_Details student = findStudentById(marks.getStudentId());
            if (student != null) {
                String deptCode = student.getDepartmentCode();
                deptMarks.computeIfAbsent(deptCode, k -> new ArrayList<>()).add(marks);
            }
        }

        // Display results by department
        for (Map.Entry<String, List<Student_Marks>> entry : deptMarks.entrySet()) {
            String deptCode = entry.getKey();
            List<Student_Marks> marksList = entry.getValue();

            Department dept = findDepartmentByCode(deptCode);
            System.out.println("\n" + dept.getName() + " Department:");
            System.out.printf("%-10s %-15s %-8s %-8s%n", "Student ID", "Name", "Total",
"Average");
            System.out.println("-".repeat(45));

            double deptTotal = 0;
            for (Student_Marks marks : marksList) {
                Student_Details student = findStudentById(marks.getStudentId());
                System.out.printf("%-10s %-15s %-8d %-8.2f%n",
                                  marks.getStudentId(),
                                  student.getName(),
                                  marks.calculateTotal(),
                                  marks.calculateAverage());
                deptTotal += marks.calculateAverage();
            }

            double deptAverage = deptTotal / marksList.size();
            System.out.printf("Department Average: %.2f%n", deptAverage);
        }
        System.out.println();
    }

    public void displayTopPerformers() {
        System.out.println("=== TOP PERFORMERS ===");

        // Sort students by average marks in descending order
        List<Student_Marks> sortedMarks = new ArrayList<>(studentMarks);
        sortedMarks.sort((m1, m2) -> Double.compare(m2.calculateAverage(),
m1.calculateAverage()));

        System.out.printf("%-5s %-10s %-15s %-8s %-8s%n",
                          "Rank", "Student ID", "Name", "Average", "Grade");
        System.out.println("=".repeat(50));

        for (int i = 0; i < Math.min(3, sortedMarks.size()); i++) {
            Student_Marks marks = sortedMarks.get(i);
```

```
        Student_Details student = findStudentById(marks.getStudentId());
        System.out.printf("%-5d %-10s %-15s %-8.2f %-8s%n",
                          i + 1,
                          marks.getStudentId(),
                          student.getName(),
                          marks.calculateAverage(),
                          marks.getGrade());
    }
    System.out.println();
}

// Utility methods to find entities by code
private Student_Details findStudentById(String studentId) {
    for (Student_Details student : students) {
        if (student.getCode().equals(studentId)) {
            return student;
        }
    }
    return null;
}

private Department findDepartmentByCode(String deptCode) {
    for (Department dept : departments) {
        if (dept.getCode().equals(deptCode)) {
            return dept;
        }
    }
    return null;
}

private Teaching_Staff findTeacherById(String teacherId) {
    for (Teaching_Staff teacher : teachingStaff) {
        if (teacher.getCode().equals(teacherId)) {
            return teacher;
        }
    }
    return null;
}

public static void main(String[] args) {
    CollegeManagementSystem college = new CollegeManagementSystem();

    // Display all information
    college.displayAllDepartments();
    college.displayAllTeachingStaff();
    college.displayAllStudents();
    college.displayStudentMarksAndResults();
    college.displayDepartmentWiseResults();
    college.displayTopPerformers();

    // Additional detailed information
    System.out.println("=== DETAILED STUDENT REPORT ===");
    for (Student_Marks marks : college.studentMarks) {
        Student_Details student = college.findStudentById(marks.getStudentId());
        Department dept = college.findDepartmentByCode(student.getDepartmentCode());

        System.out.println("\n" + student);
        System.out.println("  Department: " + dept.getName());
        System.out.println("  Semester: " + marks.getSemesterNumber());
        System.out.println("  Marks: " + Arrays.toString(marks.getMarks()));
        System.out.println("  Total: " + marks.calculateTotal());
        System.out.println("  Average: " + String.format("%.2f", marks.calculateAverage()));
```

```
            System.out.println("  Grade: " + marks.getGrade());

            System.out.println("  Subject Teachers:");
            String[] teacherCodes = student.getTeacherCodes();
            for (int i = 0; i < teacherCodes.length; i++) {
                Teaching_Staff teacher = college.findTeacherById(teacherCodes[i]);
                if (teacher != null) {
                    System.out.println("    Subject " + (i+1) + ": " + teacher.getName() +
                                       " (" + teacher.getQualification() + ")");
                }
            }
        }
    }
}
```