Course Enrollment System

Part A — The `Course` class

Instance fields
- `courseCode` (String) — e.g. `"CS301"`
- `courseName` (String) — e.g. `"Data Structures"`
- `credits` (int)
- `maxSeats` (int)
- `enrolledStudents` — a list of the students currently enrolled

Static fields
- `collegeName` — shared across every course
- `totalCoursesCreated` — increases every time a new `Course` object is created, and never decreases even if a course is later removed from a list
- `MAX_COURSES` — a constant capping the department to 50 courses

Constructors
- One constructor should accept just `courseCode` and `courseName`, and assign sensible defaults for `credits` (e.g. 3) and `maxSeats` (e.g. 60).
- Another constructor should accept `courseCode`, `courseName`, `credits`, and `maxSeats` explicitly.

Methods
- `addStudent(Student s)` — enrolls a student, but only if the course isn't already full; print a message either way.
- `addStudent(Student s, boolean allowOverEnroll)` — a second version that, when `allowOverEnroll` is `true`, enrolls the student even if the course is technically full (e.g. instructor override).
- `removeStudent(Student s)` — drops a student from the course.
- `display()` — prints the course's basic details.
- `display(boolean showEnrolledStudents)` — also lists every enrolled student's name when `true`.

Static methods
- One that reports whether a course is full, given its current enrollment count and `maxSeats`.
- One that prints the college name and the total number of courses ever created.

Equality
- Two `Course` objects should be considered equal if they have the same `courseCode`, even if they are different objects in memory. Override the method(s) needed for this to work correctly with `ArrayList` methods like `contains()` and `indexOf()`.

Part B — The `Student` class

Instance fields
- `rollNumber` (int) — unique per student
- `name` (String)
- `department` (String)

- `enrolledCourses` — a list of the courses this student has enrolled in

Static fields
- `totalStudentsRegistered` — increases with every new `Student`
- A private static counter used to auto-generate each student's `rollNumber`. No two students should ever get the same roll number, and nothing outside the class should be able to set it directly.

Constructors
- One constructor should accept just `name`, defaulting `department` to `"Not Assigned"`.
- Another should accept both `name` and `department`.

Methods
- `enroll(Course c)` — adds the course to this student's list AND updates the course's own enrolled-student list (think carefully about which object should call which method so the two lists stay consistent with each other).
- `drop(Course c)` — the reverse of `enroll`.
- `calculateTotalCredits()` — sums the credits of every enrolled course.
- `calculateTotalCredits(String excludeCourseCode)` — a second version that sums the credits of every enrolled course except the one matching the given course code.
- `display()` / `display(boolean showCourses)` — similar pattern to `Course.display()`.

Equality
- Two `Student` objects should be considered equal if they have the same `rollNumber`. Override the method(s) needed for `contains()` and `indexOf()` to work correctly.

Part C — The `EnrollmentManager` class (with `main`)

Maintain two lists — `ArrayList<Course> courses` and `ArrayList<Student> students` — and demonstrate the following,

1. Create courses using both overloaded constructors. Create at least 4 courses, and set `maxSeats` low for at least one of them (e.g. 2) so you can demonstrate a "course full" scenario later.

2. Register students using both overloaded constructors. Create at least 6 students.

3. Enroll students in courses, using `Student.enroll()`. Enroll enough students into your low-capacity course to fill it, then attempt one more enrollment and show that it's correctly rejected

4. Display all courses along with he list of students enrolled in each one (`display(true)`), and all students with their enrolled courses.

5. Search using two independent search methods — one that finds a `Course` by `courseCode`, and one that finds a `Student` by `rollNumber`.