CS23304 -JAVA PROGRAMMING LABORATORY
Files and Serialisation

Date: 19.8.26 **max.marks:20**

1. In reference to the spot question dated 22/7/26, Course Enrollment System make the following modifications:

Part A — File Formats (Design Your Own, Within These Constraints)

courses.txt

- One line per course.
- Each line must contain, at minimum: courseCode, courseName, credits, maxSeats.
- Fields on a line must be separated by a fixed delimiter of your choice (e.g. comma or pipe) — document your choice at the top of your source file.

students.txt

- One line per student.
- Each line must contain, at minimum: rollNumber, name, department.

enrollments.txt

- One line per enrollment pair: rollNumber and the courseCode they are enrolled in.

Part B — Required Methods

- Implement these on EnrollmentManager, or as a separate helper class (e.g. FileManager) — your choice.
- **1.** saveCourses(String filename) — writes every Course currently held by the manager to the given file, one line per course, in the chosen format.
- **2.** loadCourses(String filename) — reads the file back and reconstructs the ArrayList<Course>. Any line that does not match the expected format must be skipped and reported (e.g. printed as a warning), not allowed to crash the program.
- **3.** saveStudents(String filename) and loadStudents(String filename) — similar to course apply to students.
- **4.** saveEnrollments(String filename) — writes the rollNumber–courseCode pair for every currently active enrollment.
- **5.** loadEnrollments(String filename) — called only after courses and students have already been loaded. For each pair in the file, find the matching Course and Student objects already in memory and re-establish the enrollment by calling the appropriate method on them. This must restore genuine object references and update both sides' lists consistently not simply storing the text pair is not sufficient.

Part C — main()

1. create at least 4 courses (with at least one low-capacity course, e.g. maxSeats = 2), register at least 6 students, and enroll students including filling the low-capacity course and showing one rejected (or overridden) over-enrollment attempt.

2. Save all three files

3. Create a second, completely empty set of course and student lists and repopulate them entirely using your load methods, in the correct order (courses and students before enrollments).

4. After reading the data back from the files, print out every course (showing its list of enrolled students) and every student (showing their list of enrolled courses) — using **only the objects you just read from the files**, not the original ones from before you saved. Compare this output to what you had before saving: it should match exactly. If a student was enrolled in a course before saving, that same enrollment must still show up correctly on both sides (the course's roster *and* the student's course list) after reading the files back in — this is how you check that your loadEnrollments() method properly reconnected the students and courses to each other, and didn't just read the text without rebuilding the actual links between them..

5. Rename or delete one of the three files, then re-run the corresponding load(read) step, and show that your program reports the problem clearly and continues running rather than crashing.

2. In reference to the inheritance question dated 29/7/26, Course Enrollment System make the following modifications:

Part A — File Formats

customer.ser / customer.dat

Part B — Preparing the Hierarchy

- Add a registrationDate field (type java.util.Date) to Customer, set once in its constructor.(Date already implements Serializable)
- PreferredCustomer object should be serialized. (where Serializable actually needs to be declared in the hierarchy?)
- Add a private static final long serialVersionUID to each class explicitly — do not rely on the compiler-generated default.
- Add one field to PreferredCustomer that should NOT be persisted (e.g. a sessionToken relevant only to the current run) and mark it transient.

Part C — Demonstration in main()

- Build a Person[] or ArrayList<Person> with at least 2 Customer objects and 2 PreferredCustomer objects, each with a registrationDate, and with the PreferredCustomer objects having varying purchaseAmt values.
- Serialize the whole collection to one file, then deserialize it back into a fresh collection, and print every recovered object's details
- Print the transient field's value immediately before saving and immediately after loading, side by side, showing it was not preserved.