```java
import java.io.Serializable;
import java.time.LocalDate;
import java.time.format.*;
import java.util.*;

public class Student implements Serializable
{
    private String sNum;
    private String name;
    private String password;
    private String phone;
    private String address;
    private LocalDate dateOfBirth;
    private ArrayList<CourseEnrolment> currentEnrolments;
    private ArrayList<Course> completedCourses;
    private ArrayList<Course> exemptCourses;
    static DateTimeFormatter dateFormat = DateTimeFormatter
            .ofPattern("d/M/yyyy");
    static DateTimeFormatter timeFormat = DateTimeFormatter.ofPattern("h:mm a");

    public Student(String sNum, String name, String password, String phone,
            String address, LocalDate dateOfBirth,
            ArrayList<Course> exemptCourses)
    {
        this.sNum = sNum;
        this.name = name;
        this.password = password;
        this.phone = phone;
        this.address = address;
        this.dateOfBirth = dateOfBirth;
        this.currentEnrolments = new ArrayList<CourseEnrolment>();
        this.completedCourses = new ArrayList<Course>();
        this.exemptCourses = exemptCourses;
    }

    public CourseEnrolment enrol(CourseOffering offering)
            throws OverEnrolmentException, AlreadyEnrolledException,
            IncompletePrerequisitesException
    {
        // Check if the student is already enrolled in 4 courses for this period
        if(currentEnrolments.size() >= 4)
        {
            throw new OverEnrolmentException("You are already enrolled in 4 "
                    + "courses");
        }

        // Check if the student is already enrolled in this course
        if(alreadyEnrolled(offering.getCourse()))
        {
            throw new AlreadyEnrolledException("You are already enrolled in "
                    + "this course");
        }

        // Check if the student meets the prerequisites
        if(!meetsPrerequisites(offering.getCourse()))
        {
            throw new IncompletePrerequisitesException("Prerequisites not met "
                    + "for this course");
        }

        CourseEnrolment newEnrolment = new CourseEnrolment(offering);
        currentEnrolments.add(newEnrolment);

        return newEnrolment;
    }

    public boolean alreadyEnrolled(Course course)
    {
        for(CourseEnrolment enrolment : currentEnrolments)
        {
            if(enrolment.getCourseOffering().getCourse().equals(course))
            {
                return true;
            }
        }
        return false;
    }

    public boolean meetsPrerequisites(Course course)
    {
        for(Course prereq : course.getPrereqs())
        {
            if(!(exemptCourses.contains(prereq) || completedCourses
                    .contains(prereq)))
            {
                return false;
            }
        }
        return true;
    }

    public void registerTutorial(CourseEnrolment courseEnrolment,
            Tutorial tutorial) throws OverCapacityException,
            TutorialAlreadyEnrolledException
    {
        if(tutorial.getNumberEnrolled() >= tutorial.getVenue().getCapacity())
        {
            throw new OverCapacityException("Venue at max capacity");
        }

        if(courseEnrolment.getTutorial() != null)
        {
            throw new TutorialAlreadyEnrolledException("You already have a "
                    + "tutorial registered for this course");
        }

        courseEnrolment.setTutorial(tutorial);

        tutorial.setNumberEnrolled(tutorial.getNumberEnrolled() + 1);
    }

    public void deregisterTutorial(CourseEnrolment courseEnrolment)
            throws TutorialNotEnrolledException
    {
        if(courseEnrolment.getTutorial() == null)
        {
            throw new TutorialNotEnrolledException("You are not enrolled in a "
                    + "tutorial for this course");
        }

        // Reduce number enrolled in tutorial by 1
        Tutorial tutorial = courseEnrolment.getTutorial();
        tutorial.setNumberEnrolled(tutorial.getNumberEnrolled() - 1);

        courseEnrolment.setTutorial(null);
    }

    public void printTimetable()
    {
        System.out.println("Name: " + name);

        // Get maximum length for each field
        int courseIDMaxLen = "Course".length();
        int activityMaxLen = "Activity".length(); // "Tutorial" or "Lecture"
        int locationMaxLen = "Location".length();
        int dayMaxLen = "Day".length();
        int startTimeMaxLen = "Start Time".length(); // Always longer/equal to
        int endTimeMaxLen = "End Time".length();     // any time
        int staffMaxLen = "Staff".length();
        for(CourseEnrolment enrolment : currentEnrolments)
        {
            CourseOffering offering = enrolment.getCourseOffering();
            Lecture lecture = offering.getLecture();
            Tutorial tutorial = enrolment.getTutorial();

            // Check if the length of this 'Course' field is largest so far
            if(offering.getCourse().getID().length() > courseIDMaxLen)
            {
                courseIDMaxLen = offering.getCourse().getID().length();
            }

            // Check if the length of this 'Location' field is largest so far
            if(lecture.getVenue().getLocation().length() > locationMaxLen)
            {
                locationMaxLen = lecture.getVenue().getLocation().length();
            }
            if(tutorial.getVenue().getLocation().length() > locationMaxLen)
            {
                locationMaxLen = tutorial.getVenue().getLocation().length();
            }

            // Check if the length of this 'Day' field is largest so far
            if(lecture.getDay().getDisplayName(TextStyle.FULL,
                    Locale.getDefault()).length() > dayMaxLen)
            {
                dayMaxLen = lecture.getDay().getDisplayName(TextStyle.FULL,
                        Locale.getDefault()).length();
            }
            if(tutorial.getDay().getDisplayName(TextStyle.FULL,
                    Locale.getDefault()).length() > dayMaxLen)
            {
                dayMaxLen = tutorial.getDay().getDisplayName(TextStyle.FULL,
                        Locale.getDefault()).length();
            }

            // Check if the length of this 'Staff' field is largest so far
            if(lecture.getStaff().getName().length() > staffMaxLen)
            {
                staffMaxLen = lecture.getStaff().getName().length();
            }
            if(tutorial.getStaff().getName().length() > staffMaxLen)
            {
                staffMaxLen = tutorial.getStaff().getName().length();
            }
        }
        int totalLength = courseIDMaxLen + activityMaxLen + locationMaxLen
                + dayMaxLen + startTimeMaxLen + endTimeMaxLen + staffMaxLen
                + 20;

        // Print top of table
        System.out.println(String.format("+%" + totalLength + "s+", "")
                .replace(' ', '-'));
        // Print table headers
        System.out.println(String.format("| %-" + courseIDMaxLen + "s | %-"
                + activityMaxLen + "s | %-" + locationMaxLen + "s | %-"
                + dayMaxLen + "s | %-" + startTimeMaxLen + "s | %-"
                + endTimeMaxLen + "s | %-" + staffMaxLen + "s |", "Course",
                "Activity", "Location", "Day", "Start Time", "End Time",
                "Staff"));
        // Print header separator
        System.out.println(String.format("| %-" + courseIDMaxLen + "s | %-"
                + activityMaxLen + "s | %-" + locationMaxLen + "s | %-"
                + dayMaxLen + "s | %-" + startTimeMaxLen + "s | %-"
                + endTimeMaxLen + "s | %-" + staffMaxLen + "s |", "", "", "",
                "", "", "", "").replace(' ', '-'));

        // Print each table row
        for(CourseEnrolment enrolment : currentEnrolments)
        {
            CourseOffering offering = enrolment.getCourseOffering();
            Lecture lecture = offering.getLecture();
            Tutorial tutorial = enrolment.getTutorial();

            // Print lecture information
            String s = "| ";
            s += String.format("%-" + courseIDMaxLen + "s",
                    offering.getCourse().getID());
            s += String.format(" | %-" + activityMaxLen + "s | ", "Lecture");
            s += String.format("%-" + locationMaxLen + "s",
                    lecture.getVenue().getLocation());
            s += " | ";
            s += String.format("%-" + dayMaxLen + "s", lecture.getDay()
                    .getDisplayName(TextStyle.FULL, Locale.getDefault()));
            s += " | ";
            s += String.format("%-" + startTimeMaxLen + "s",
                    String.format("%8s",
                            timeFormat.format(lecture.getStart())));
            s += " | ";
            s += String.format("%-" + endTimeMaxLen + "s",
                    String.format("%8s", timeFormat.format(lecture.getEnd())));
            s += " | ";
            s += String.format("%-" + staffMaxLen + "s",
                    lecture.getStaff().getName());
            s += " |";
            System.out.println(s);

            // Print tutorial information
            s = "| ";
            s += String.format("%-" + courseIDMaxLen + "s",
                    offering.getCourse().getID());
            s += String.format(" | %-" + activityMaxLen + "s | ", "Tutorial");
            s += String.format("%-" + locationMaxLen + "s",
                    tutorial.getVenue().getLocation());
            s += " | ";
            s += String.format("%-" + dayMaxLen + "s", tutorial.getDay()
                    .getDisplayName(TextStyle.FULL, Locale.getDefault()));
            s += " | ";
            s += String.format("%-" + startTimeMaxLen + "s",
                    String.format("%8s",
                            timeFormat.format(tutorial.getStart())));
            s += " | ";
            s += String.format("%-" + endTimeMaxLen + "s",
                    String.format("%8s", timeFormat.format(tutorial.getEnd())));
            s += " | ";
            s += String.format("%-" + staffMaxLen + "s",
                    tutorial.getStaff().getName());
            s += " |";
            System.out.println(s);
        }

        // Print bottom of table
        System.out.println(String.format("+%" + totalLength + "s+", "")
                .replace(' ', '-'));
    }

    public String getPassword()
    {
        return password;
    }

    public void setPassword(String password)
    {
        this.password = password;
    }

    public String getSNum()
    {
        return sNum;
    }

    public String getName()
    {
        return name;
    }

```

```java
    public String getPhone()
    {
        return phone;
    }

    public ArrayList<CourseEnrolment> getCurrentEnrolments()
    {
        return currentEnrolments;
    }

    public String[] getCurrentEnrolmentNames()
    {
        ArrayList<String> enrolmentNames = new ArrayList<String>();
        for(CourseEnrolment enrolment : currentEnrolments)
        {
            enrolmentNames.add("[" + enrolment.getCourseOffering().getCourse()
                    .getID() + "] " + enrolment.getCourseOffering().getCourse()
                    .getName());
        }
        return enrolmentNames.toArray(new String[currentEnrolments.size()]);
    }

    public ArrayList<Course> getCompletedCourses()
    {
        return completedCourses;
    }

    public ArrayList<Course> getExemptCourses()
    {
        return exemptCourses;
    }

    public String toString()
    {
        String s = "Student ID: " + sNum;
        s += "\nName" + name;
        s += "\nPhone Number: " + phone;
        s += "\nAddress: " + address;
        s += "\nDOB: " + dateFormat.format(dateOfBirth);
        s += "\nCurrent enrolments: ";

        String delim = "";
        for(CourseEnrolment offering : currentEnrolments)
        {
            s += delim + offering.getCourseOffering().getCourse().getID();
            s += " - " + offering.getCourseOffering().getCourse().getName();
            delim = ", ";
        }

        return s;
    }
}

```