Answer to the Question No. 1

Code:

```
class Course {

    String courseName;
    String courseCode;
    int creditHours;

    Course(String courseName, String courseCode, int creditHours) {
        this.courseName = courseName;
        this.courseCode = courseCode;
        this.creditHours = creditHours;
    }

    String calculateDifficulty() {
        return "Standard difficulty";
    }
}

class UndergraduateCourse extends Course {

    boolean generalEducationRequirement;

    UndergraduateCourse(String courseName, String courseCode, int creditHours, boolean
generalEducationRequirement) {
        super(courseName, courseCode, creditHours);
        this.generalEducationRequirement = generalEducationRequirement;
    }
```

```
    @Override
    String calculateDifficulty() {
        if (generalEducationRequirement == true) {
            return "Difficulty level of " + courseCode + ": Standard";
        } else {
            return "Difficulty level of " + courseCode + ": Hard";
        }
    }
}

class PostgraduateCourse extends Course {

    boolean researchComponent;

    PostgraduateCourse(String courseName, String courseCode, int creditHours, boolean
researchComponent) {
        super(courseName, courseCode, creditHours);
        this.researchComponent = researchComponent;
    }

    @Override
    String calculateDifficulty() {
        if (researchComponent == true) {
            return "Difficulty level of " + courseCode + ": Hard";
        } else {
            return "Difficulty level of " + courseCode + ": Standard";
        }
    }
```

}

Explanation:

Answer to the Question No. 2

Code:

```
class Student {

    String studentID;
    String name;
    ArrayList<Course> enrolledCourses;

    Student(String studentID, String name) {
        this.studentID = studentID;
        this.name = name;
        this.enrolledCourses = new ArrayList<>();
    }

    void enrollInCourse(Course course) {
        enrolledCourses.add(course);
    }

    void dropCourse(Course course) {
        enrolledCourses.remove(course);
    }

    void showCourses() {
        System.out.println(name + " is currently enrolled in: ");
        for (int i = 0; i < enrolledCourses.size(); i++) {
            System.out.println(enrolledCourses.get(i).courseName);
        }
    }
```

```
}

class UndergraduateStudent extends Student {

    String advisorName;

    UndergraduateStudent(String studentID, String name, String advisorName) {
        super(studentID, name);
        this.advisorName = advisorName;
    }

    @Override
    void enrollInCourse(Course course) {
        if (!(course instanceof UndergraduateCourse)) {
            System.out.println("Undergraduate student " + name + " cannot enroll in postgraduate
course " + course.courseName);
        } else {
            if (enrolledCourses.size() < 6) {
                enrolledCourses.add(course);
            } else {
                System.out.println("Undergrad student " + name + " has reached maximum course
limit");
            }
        }
    }
}

class PostgraduateStudent extends Student {
```

```
    String thesisTopic;

    PostgraduateStudent(String studentID, String name, String thesisTopic) {
        super(studentID, name);
        this.thesisTopic = thesisTopic;
    }

    @Override
    void enrollInCourse(Course course) {
        if (enrolledCourses.size() < 4) {
            enrolledCourses.add(course);
        } else {
            System.out.println("Postgrad student " + name + " has reached maximum course limit");
        }
    }
}
```

Explanation:

Answer to the Question No. 3

The implementation of ‘enrollInCourse(Course course)’ method in Student, UndergraduateStudent and PostgraduateStudent is shown below:

Student:

```
void enrollInCourse(Course course) {
        enrolledCourses.add(course);
    }
```

UndergraduateStudent:

```
@Override
    void enrollInCourse(Course course) {
        if (!(course instanceof UndergraduateCourse)) {
            System.out.println("Undergraduate student " + name + " cannot enroll in postgraduate course " + course.courseName);
        } else {
            if (enrolledCourses.size() < 6) {
                enrolledCourses.add(course);
            } else {
                System.out.println("Undergrad student " + name + " has reached maximum course limit");
            }
        }
    }
```

PostgraduateStudent:

```
@Override
    void enrollInCourse(Course course) {
        if (enrolledCourses.size() < 4) {
```

```
            enrolledCourses.add(course);
        } else {
            System.out.println("Postgrad student " + name + " has reached maximum course limit");
        }
    }
```

Explanation: