**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:**

**Main Class:**

public class UniversityManagementSystem {

public static void main(String[] args) {

UndergraduateStudent u1 = new UndergraduateStudent("2023100000457", "Aayat", "Mahbub Hasan");

UndergraduateStudent u2 = new UndergraduateStudent("2023100000458", "Tasin", "Md. Omar");

PostgraduateStudent p1 = new PostgraduateStudent("2023100000459", "Kayes", "Md. Rakin");

PostgraduateStudent p2 = new PostgraduateStudent("2023100000456", "Aoyon", "Md. Arnob");

UndergraduateCourse uc1 = new UndergraduateCourse("UCourse-1", "000", 45, true);

UndergraduateCourse uc2 = new UndergraduateCourse("UCourse-2", "001", 45, false);

UndergraduateCourse uc3 = new UndergraduateCourse("UCourse-3", "002", 45, true);

UndergraduateCourse uc4 = new UndergraduateCourse("UCourse-4", "003", 45, true);

UndergraduateCourse uc5 = new UndergraduateCourse("UCourse-5", "004", 45, false);

UndergraduateCourse uc6 = new UndergraduateCourse("UCourse-6", "005", 45, true);

UndergraduateCourse uc7 = new UndergraduateCourse("UCourse-7", "006", 45, false);

PostgraduateCourse pc1 = new PostgraduateCourse("PCourse-1", "000", 30, true);

PostgraduateCourse pc2 = new PostgraduateCourse("PCourse-2", "001", 30, false);

PostgraduateCourse pc3 = new PostgraduateCourse("PCourse-3", "002", 30, false);

PostgraduateCourse pc4 = new PostgraduateCourse("PCourse-4", "003", 30, true);

PostgraduateCourse pc5 = new PostgraduateCourse("PCourse-5", "004", 30, true);

u1.enrollInCourse(uc1);

u1.enrollInCourse(uc2);

u1.enrollInCourse(pc3);

u1.enrollInCourse(uc3);

u1.enrollInCourse(uc4);

u1.enrollInCourse(uc5);

u1.enrollInCourse(uc6);

u1.enrollInCourse(uc7);

u1.showCourses();

u1.dropCourse(uc6);

u1.showCourses();

u1.enrollInCourse(uc7);

u1.showCourses();

u2.enrollInCourse(uc2);

u2.enrollInCourse(uc3);

u2.showCourses();

p1.enrollInCourse(pc1);

p1.enrollInCourse(pc2);

p1.enrollInCourse(pc3);

p1.enrollInCourse(pc4);

p1.enrollInCourse(pc5);

p1.showCourses();

}

}