

## Project -Universit\_System.py

```
1 # --- 1. Person Class (Base Class) ---
2 class Person:
3     def __init__(self, person_id: str, name: str):
4         if not person_id:
5             print("Warning: Person ID should not be empty.")
6         if not name:
7             print("Warning: Person name should not be empty.")
8         self._id = person_id
9         self._name = name
10
11    def get_id(self):
12        return self._id
13
14    def get_name(self):
15        return self._name
16
17    def __str__(self):
18        return f"ID: {self._id}, Name: {self._name}"
19
20    def to_dict(self):
21        return {
22            'id': self._id,
23            'name': self._name
24        }
25
26 # --- 2. Student Class ---
27 class Student(Person):
28     def __init__(self, student_id: str, name: str, major: str):
29         super().__init__(student_id, name)
30         if not major:
31             print("Warning: Student major should not be empty.")
32         self._major = major
33         self._enrolled_course_codes = []
34
35     def get_major(self):
36         return self._major
37
38     def set_major(self, new_major: str) :
39         if not new_major:
40             print("Warning: Major should not be empty.")
41         self._major = new_major
42
43     def get_enrolled_course_codes(self) :
44         return list(self._enrolled_course_codes)
45
46     def enroll_course(self, course_code: str):
47         if not course_code:
48             print("Error: Course code cannot be empty.")
```

```

49         return
50     if course_code not in self._enrolled_course_codes:
51         self._enrolled_course_codes.append(course_code)
52
53 def drop_course(self, course_code: str):
54     if not course_code:
55         print("Error: Course code cannot be empty.")
56         return
57     if course_code in self._enrolled_course_codes:
58         self._enrolled_course_codes.remove(course_code)
59
60 def display_details(self):
61     return (f"{super().__str__()}, Type: Student, Major: {self._major}, "
62             f"Enrolled Courses: {len(self._enrolled_course_codes)})")
63
64 def to_dict(self) -> dict:
65     data = super().to_dict()
66     data.update({
67         'type': 'student',
68         'major': self._major,
69         'enrolled_course_codes': self._enrolled_course_codes
70     })
71     return data
72
73 # --- 3. Faculty Class ---
74 class Faculty(Person):
75     def __init__(self, faculty_id: str, name: str, department: str):
76         super().__init__(faculty_id, name)
77         if not department:
78             print("Warning: Faculty department should not be empty.")
79         self._department = department
80         self._assigned_course_codes = []
81     def get_department(self):
82         return self._department
83
84     def set_department(self, new_department: str):
85         if not new_department:
86             print("Warning: Department should not be empty.")
87         self._department = new_department
88
89     def get_assigned_course_codes(self):
90         return list(self._assigned_course_codes)
91
92     def assign_course(self, course_code: str) :
93         if not course_code:
94             print("Error: Course code cannot be empty.")
95             return
96         if course_code not in self._assigned_course_codes:
97             self._assigned_course_codes.append(course_code)
98

```

```
99     def unassign_course(self, course_code: str):
100         if not course_code:
101             print("Error: Course code cannot be empty.")
102             return
103         if course_code in self._assigned_course_codes:
104             self._assigned_course_codes.remove(course_code)
105
106     def display_details(self):
107         return (f"{super().__str__()}, Type: Faculty, Department: {self._department}, "
108                 f"Assigned Courses: {len(self._assigned_course_codes)}")
109
110     def to_dict(self):
111         data = super().to_dict()
112         data.update({
113             'type': 'faculty',
114             'department': self._department,
115             'assigned_course_codes': self._assigned_course_codes
116         })
117         return data
118
119 # --- 4. Course Class ---
120 class Course:
121     def __init__(self, course_code: str, title: str, credits: int, prerequisites: list = None):
122         if not course_code:
123             print("Warning: Course code should not be empty.")
124         if not title:
125             print("Warning: Course title should not be empty.")
126         if not credits or credits <= 0:
127             print("Warning: Credits should be a positive number.")
128         if prerequisites is None:
129             prerequisites = []
130         self._course_code = course_code
131         self._title = title
132         self._credits = credits
133         self._prerequisite_codes = list(prerequisites)
134         self._enrolled_student_ids = []
135         self._assigned_faculty_id = None
136
137     def get_course_code(self):
138         return self._course_code
139
140     def get_title(self) :
141         return self._title
142
143     def get_credits(self) :
144         return self._credits
145
146     def get_prerequisite_codes(self):
```

```
148     return list(self._prerequisite_codes)
149
150     def get_enrolled_student_ids(self) :
151         return list(self._enrolled_student_ids)
152
153     def get_assigned_faculty_id(self):
154         return self._assigned_faculty_id
155
156     def set_assigned_faculty_id(self, faculty_id) :
157         if faculty_id is not None and not faculty_id:
158             print("Warning: Faculty ID should not be empty if provided.")
159         self._assigned_faculty_id = faculty_id
160
161     def add_prerequisite(self, prerequisite_code: str):
162         if not prerequisite_code:
163             print("Error: Prerequisite code cannot be empty.")
164             return
165         if prerequisite_code not in self._prerequisite_codes:
166             self._prerequisite_codes.append(prerequisite_code)
167
168     def add_student_id(self, student_id: str):
169         if not student_id:
170             print("Error: Student ID cannot be empty.")
171             return
172         if student_id not in self._enrolled_student_ids:
173             self._enrolled_student_ids.append(student_id)
174
175     def remove_student_id(self, student_id: str) :
176         if not student_id:
177             print("Error: Student ID cannot be empty.")
178             return
179         if student_id in self._enrolled_student_ids:
180             self._enrolled_student_ids.remove(student_id)
181
182     def display_details(self):
183         faculty_info = f"Assigned Faculty ID: {self._assigned_faculty_id}" if
184         self._assigned_faculty_id else "No faculty assigned"
185         prereqs = ", ".join(self._prerequisite_codes) if self._prerequisite_codes else "None"
186         return (f"Course Code: {self._course_code}, Title: {self._title}, Credits:
187 {self._credits}\n"
188             f"    Prerequisites: {prereqs}\n"
189             f"    Enrolled Students: {len(self._enrolled_student_ids)}\n"
190             f"    {faculty_info}")
191
192     def to_dict(self):
193         return {
194             'course_code': self._course_code,
195             'title': self._title,
196             'credits': self._credits,
197             'prerequisite_codes': self._prerequisite_codes,
```

```
196     'enrolled_student_ids': self._enrolled_student_ids,
197     'assigned_faculty_id': self._assigned_faculty_id
198 }
199
200 # --- 5. University Class ---
201 class University:
202     def __init__(self):
203         self._students: dict = {}
204         self._faculty: dict = {}
205         self._courses: dict = {}
206
207
208     def add_student(self, student) :
209         if student.get_id() in self._students:
210             print(f"Error: Student with ID '{student.get_id()}' already exists.")
211             return False
212         self._students[student.get_id()] = student
213         print(f"Student '{student.get_name()}' added successfully.")
214         return True
215
216     def remove_student(self, student_id: str) :
217         if student_id not in self._students:
218             print(f"Error: Student with ID '{student_id}' not found.")
219             return False
220
221         student = self._students[student_id]
222         if student.get_enrolled_course_codes():
223             print(f"Error: Student '{student.get_name()}' is enrolled in courses. Drop them first.")
224             return False
225
226         for course_code in list(student._enrolled_course_codes()):
227             if course_code in self._courses:
228                 self._courses[course_code].remove_student_id(student_id)
229
230         del self._students[student_id]
231         print(f"Student '{student_id}' removed successfully.")
232         return True
233
234     def add_faculty(self, faculty) :
235         if faculty.get_id() in self._faculty:
236             print(f"Error: Faculty with ID '{faculty.get_id()}' already exists.")
237             return False
238         self._faculty[faculty.get_id()] = faculty
239         print(f"Faculty '{faculty.get_name()}' added successfully.")
240         return True
241
242     def remove_faculty(self, faculty_id: str) :
243         if faculty_id not in self._faculty:
244             print(f"Error: Faculty with ID '{faculty_id}' not found.")
```

```
245     return False
246
247     faculty = self._faculty[faculty_id]
248     if faculty.get_assigned_course_codes():
249         print(f"Error: Faculty '{faculty.get_name()}' is assigned to courses. Unassign them first.")
250     return False
251
252     for course_code in list(faculty._assigned_course_codes):
253         if course_code in self._courses:
254             if self._courses[course_code].get_assigned_faculty_id() == faculty_id:
255                 self._courses[course_code].unassign_faculty_id()
256
257     del self._faculty[faculty_id]
258     print(f"Faculty '{faculty_id}' removed successfully.")
259     return True
260
261 def add_course(self, course):
262     if course.get_course_code() in self._courses:
263         print(f"Error: Course with code '{course.get_course_code()}' already exists.")
264     return False
265     self._courses[course.get_course_code()] = course
266     print(f"Course '{course.get_title()}' added successfully.")
267     return True
268
269 def remove_course(self, course_code: str):
270     if course_code not in self._courses:
271         print(f"Error: Course with code '{course_code}' not found.")
272     return False
273
274     course = self._courses[course_code]
275     if course.get_enrolled_student_ids():
276         print(f"Error: Course '{course.get_title()}' has enrolled students. Drop them first.")
277     return False
278     if course.get_assigned_faculty_id():
279         print(f"Error: Course '{course.get_title()}' has an assigned faculty. Unassign them first.")
280     return False
281
282     for student_id in list(self._students.keys()):
283         student = self._students[student_id]
284         if course_code in student.get_enrolled_course_codes():
285             student.drop_course(course_code)
286
287     for faculty_id in list(self._faculty.keys()):
288         faculty = self._faculty[faculty_id]
289         if course_code in faculty.get_assigned_course_codes():
290             faculty.unassign_course(course_code)
291
292     del self._courses[course_code]
```

```
293     print(f"Course '{course_code}' removed successfully.")
294     return True
295
296 def enroll_student_in_course(self, student_id: str, course_code: str) :
297     student = self._students.get(student_id)
298     course = self._courses.get(course_code)
299
300     if not student:
301         print(f"Error: Student with ID '{student_id}' not found.")
302         return False
303
304     if not course:
305         print(f"Error: Course with code '{course_code}' not found.")
306         return False
307
308     if course_code in student.get_enrolled_course_codes():
309         print(f"Info: Student '{student.get_name()}' is already enrolled in
'{course.get_title()}'.")
310         return False
311
312
313     for prereq_code in course.get_prerequisite_codes():
314         if prereq_code not in student.get_enrolled_course_codes():
315             print(f"Error: Student '{student.get_name()}' has not met prerequisite
'{prereq_code}' for '{course.get_title()}'.")
316             return False
317         student.enroll_course(course_code)
318         course.add_student_id(student_id)
319         print(f"Student '{student.get_name()}' enrolled in '{course.get_title()}'"
320             successfully.")
321         return True
322
323 def drop_student_from_course(self, student_id: str, course_code: str):
324     student = self._students.get(student_id)
325     course = self._courses.get(course_code)
326
327     if not student:
328         print(f"Error: Student with ID '{student_id}' not found.")
329         return False
330
331     if not course:
332         print(f"Error: Course with code '{course_code}' not found.")
333         return False
334
335     if course_code not in student.get_enrolled_course_codes():
336         print(f"Info: Student '{student.get_name()}' is not enrolled in
'{course.get_title()}'.")
337         return False
338     student.drop_course(course_code)
339     course.remove_student_id(student_id)
```

```
339         print(f"Student '{student.get_name()}' dropped from '{course.get_title()}'"
340     successfully.")
341     return True
342
343     def assign_faculty_to_course(self, faculty_id: str, course_code: str):
344         faculty = self._faculty.get(faculty_id)
345         course = self._courses.get(course_code)
346
347         if not faculty:
348             print(f"Error: Faculty with ID '{faculty_id}' not found.")
349             return False
350         if not course:
351             print(f"Error: Course with code '{course_code}' not found.")
352             return False
353
354         if course.get_assigned_faculty_id() == faculty_id:
355             print(f"Info: Faculty '{faculty.get_name()}' is already assigned to
356         '{course.get_title()}'.")
357             return False
358
359         if course.get_assigned_faculty_id() is not None:
360             current_assigned_faculty = self._faculty.get(course.get_assigned_faculty_id())
361             if current_assigned_faculty:
362                 current_assigned_faculty.unassign_course(course_code)
363             print(f"Info: Unassigned previous faculty '{course.get_assigned_faculty_id()}'
364         from '{course.get_title()}'.")
365             course.set_assigned_faculty_id(faculty_id)
366             faculty.assign_course(course_code)
367             print(f"Faculty '{faculty.get_name()}' assigned to '{course.get_title()}'"
368         successfully.")
369             return True
370
371     def unassign_faculty_from_course(self, faculty_id: str, course_code: str):
372         faculty = self._faculty.get(faculty_id)
373         course = self._courses.get(course_code)
374
375         if not faculty:
376             print(f"Error: Faculty with ID '{faculty_id}' not found.")
377             return False
378         if not course:
379             print(f"Error: Course with code '{course_code}' not found.")
380             return False
381
382         if course.get_assigned_faculty_id() != faculty_id:
383             print(f"Info: Faculty '{faculty.get_name()}' is not assigned to
384         '{course.get_title()}'.")
385             return False
386             course.set_assigned_faculty_id(None)
387             faculty.unassign_course(course_code)
388             print(f"Faculty '{faculty.get_name()}' unassigned from '{course.get_title()}'"
389         successfully.)
```

```
384     return True
385
386     def get_course_roster(self, course_code: str):
387         course = self._courses.get(course_code)
388         if not course:
389             print(f"Error: Course with code '{course_code}' not found.")
390             return []
391
392         roster = []
393         for student_id in course.get_enrolled_student_ids():
394             student = self._students.get(student_id)
395             if student:
396                 roster.append(student)
397             else:
398                 print(f"Warning: Student ID '{student_id}' found in course '{course_code}'"
399                      "roster but student object does not exist in the system.")
400
401     def display_all_students(self):
402         if not self._students:
403             print("\nNo students registered.")
404             return
405         print("\n--- All Students ---")
406         for student in self._students.values():
407             print(student.display_details())
408         print("-----")
409
410     def display_all_faculty(self) :
411         if not self._faculty:
412             print("\nNo faculty registered.")
413             return
414         print("\n--- All Faculty ---")
415         for faculty in self._faculty.values():
416             print(faculty.display_details())
417         print("-----")
418
419     def display_all_courses(self):
420         if not self._courses:
421             print("\nNo courses registered.")
422             return
423         print("\n--- All Courses ---")
424         for course in self._courses.values():
425             print(course.display_details())
426         print("-----")
427
428 # --- 6. Console Application ---
429     def get_valid_input(prompt: str, input_type=str, min_value=None, max_value=None):
430         while True:
431             try:
432                 value = input(prompt).strip()
```

```
433     if not value and input_type == str:
434         print("Input cannot be empty. Please try again.")
435         continue
436     if input_type == int:
437         value = int(value)
438         if min_value is not None and value < min_value:
439             print(f"Input must be at least {min_value}. Please try again.")
440             continue
441         if max_value is not None and value > max_value:
442             print(f"Input must be at most {max_value}. Please try again.")
443             continue
444     return value
445 except ValueError:
446     print(f"Invalid input. Please enter a valid {input_type.__name__}.")
447 except Exception as e:
448     print(f"An unexpected error occurred: {e}. Please try again.")
449
450
451 def main():
452     university = University() # Initialize university; data is empty on each run
453
454     while True:
455         print("\n===== University Management System =====")
456         print("1. Manage Students")
457         print("2. Manage Faculty")
458         print("3. Manage Courses")
459         print("4. Enroll/Drop Student in Course")
460         print("5. Assign/Unassign Faculty to Course")
461         print("6. View Course Roster")
462         print("7. Display All Students")
463         print("8. Display All Faculty")
464         print("9. Display All Courses")
465         print("0. Exit")
466         print("=====")
467
468     choice = get_valid_input("Enter your choice: ", int, 0, 9)
469
470     if choice == 1: # Manage Students
471         while True:
472             print("\n--- Manage Students ---")
473             print("1. Add Student")
474             print("2. Remove Student")
475             print("3. Back to Main Menu")
476             student_choice = get_valid_input("Enter your choice: ", int, 1, 3)
477
478             if student_choice == 1:
479                 student_id = get_valid_input("Enter student ID: ")
480                 name = get_valid_input("Enter student name: ")
481                 major = get_valid_input("Enter student major: ")
482                 new_student = Student(student_id, name, major)
```

```

483         university.add_student(new_student)
484         input("Press Enter to continue...")
485     elif student_choice == 2:
486         student_id = get_valid_input("Enter student ID to remove: ")
487         university.remove_student(student_id)
488         input("Press Enter to continue...")
489     elif student_choice == 3:
490         break
491
492 elif choice == 2: # Manage Faculty
493     while True:
494         print("\n--- Manage Faculty ---")
495         print("1. Add Faculty")
496         print("2. Remove Faculty")
497         print("3. Back to Main Menu")
498         faculty_choice = get_valid_input("Enter your choice: ", int, 1, 3)
499
500     if faculty_choice == 1:
501         faculty_id = get_valid_input("Enter faculty ID: ")
502         name = get_valid_input("Enter faculty name: ")
503         department = get_valid_input("Enter faculty department: ")
504         new_faculty = Faculty(faculty_id, name, department)
505         university.add_faculty(new_faculty)
506         input("Press Enter to continue...")
507     elif faculty_choice == 2:
508         faculty_id = get_valid_input("Enter faculty ID to remove: ")
509         university.remove_faculty(faculty_id)
510         input("Press Enter to continue...")
511     elif faculty_choice == 3:
512         break
513
514 elif choice == 3: # Manage Courses
515     while True:
516         print("\n--- Manage Courses ---")
517         print("1. Add Course")
518         print("2. Remove Course")
519         print("3. Add Prerequisite to Course")
520         print("4. Back to Main Menu")
521         course_choice = get_valid_input("Enter your choice: ", int, 1, 4)
522
523     if course_choice == 1:
524         course_code = get_valid_input("Enter course code (e.g., CS101): ")
525         title = get_valid_input("Enter course title: ")
526         credits = get_valid_input("Enter credits: ", int, 1)
527         prereq_input = get_valid_input("Enter prerequisite course codes (comma-separated, leave blank if none): ")
528         prerequisites = [p.strip() for p in prereq_input.split(',') if p.strip()]
529         new_course = Course(course_code, title, credits, prerequisites)
530         university.add_course(new_course)
531         input("Press Enter to continue...")

```

```

532         elif course_choice == 2:
533             course_code = get_valid_input("Enter course code to remove: ")
534             university.remove_course(course_code)
535             input("Press Enter to continue...")
536         elif course_choice == 3:
537             course_code = get_valid_input("Enter course code to add prerequisite to:
")
538             prereq_code = get_valid_input("Enter prerequisite course code to add: ")
539             course = university._courses.get(course_code)
540             if course:
541                 if prereq_code not in university._courses:
542                     print(f"Warning: Prerequisite course '{prereq_code}' does not
exist in the system. Adding prerequisite anyway.")
543                     course.add_prerequisite(prereq_code)
544                     print(f"Prerequisite '{prereq_code}' added to '{course_code}'.")
545                 else:
546                     print(f"Error: Course '{course_code}' not found.")
547                     input("Press Enter to continue...")
548             elif course_choice == 4:
549                 break
550
551     elif choice == 4: # Enroll/Drop Student in Course
552         while True:
553             print("\n--- Enroll/Drop Student ---")
554             print("1. Enroll Student in Course")
555             print("2. Drop Student from Course")
556             print("3. Back to Main Menu")
557             enroll_drop_choice = get_valid_input("Enter your choice: ", int, 1, 3)
558
559             if enroll_drop_choice == 1:
560                 student_id = get_valid_input("Enter student ID: ")
561                 course_code = get_valid_input("Enter course code to enroll in: ")
562                 university.enroll_student_in_course(student_id, course_code)
563                 input("Press Enter to continue...")
564             elif enroll_drop_choice == 2:
565                 student_id = get_valid_input("Enter student ID: ")
566                 course_code = get_valid_input("Enter course code to drop from: ")
567                 university.drop_student_from_course(student_id, course_code)
568                 input("Press Enter to continue...")
569             elif enroll_drop_choice == 3:
570                 break
571
572     elif choice == 5: # Assign/Unassign Faculty to Course
573         while True:
574             print("\n--- Assign/Unassign Faculty ---")
575             print("1. Assign Faculty to Course")
576             print("2. Unassign Faculty from Course")
577             print("3. Back to Main Menu")
578             assign_unassign_choice = get_valid_input("Enter your choice: ", int, 1, 3)
579

```

```
580     if assign_unassign_choice == 1:
581         faculty_id = get_valid_input("Enter faculty ID: ")
582         course_code = get_valid_input("Enter course code to assign to: ")
583         university.assign_faculty_to_course(faculty_id, course_code)
584         input("Press Enter to continue...")
585     elif assign_unassign_choice == 2:
586         faculty_id = get_valid_input("Enter faculty ID: ")
587         course_code = get_valid_input("Enter course code to unassign from: ")
588         university.unassign_faculty_from_course(faculty_id, course_code)
589         input("Press Enter to continue...")
590     elif assign_unassign_choice == 3:
591         break
592
593 elif choice == 6: # View Course Roster
594     course_code = get_valid_input("Enter course code to view roster: ")
595     roster = university.get_course_roster(course_code)
596     if roster:
597         print(f"\n--- Roster for Course '{course_code}' ---")
598         for student_item in roster:
599             print(student_item.display_details())
600             print("-----")
601     else:
602         if course_code in university._courses:
603             print(f"\nNo students enrolled in course '{course_code}'.")
604         input("Press Enter to continue...")
605
606 elif choice == 7: # Display All Students
607     university.display_all_students()
608     input("Press Enter to continue...")
609
610 elif choice == 8: # Display All Faculty
611     university.display_all_faculty()
612     input("Press Enter to continue...")
613
614 elif choice == 9: # Display All Courses
615     university.display_all_courses()
616     input("Press Enter to continue...")
617
618 elif choice == 0:
619     print("Exiting University Management System. Goodbye!")
620     break
621
622 if __name__ == "__main__":
623     main()
624
```