Project Work

Project work covers 4 weekly programming exercises (= 40 points). The workload (number of hours spent on the project work) is estimated to be four times the hours spent on weekly exercises.

Preferably project work is done in groups of 2-4 people.

Team up with Like-Minded People:

1. **Minimal Effort:** I just want to pass this course. A **grade of 1** from project work is ok for me. I'll be happy if I can put a bit more effort and maybe get a **grade 2**.
2. **Moderate Effort:** I can and want to put a bit more effort into project work and I'm happy with **grades 2-3**.
3. **High Effort:** I want to learn as much as possible and I'm willing to put a lot of time and effort into the project work to achieve **grades 4 or even 5**.
4. **Maximum Effort:** I want to learn as much as possible and I'm willing to put in as much time and effort as needed to achieve a **grade of 5**. I will choose the topic that pushes my skills. I'm willing to put in extra effort to achieve an excellent grade.
5. **Grade Focused:** I just want a high grade no matter what. To be honest I'm not willing to put in much effort in learning and I don't care much about the learning process, only the high grades.

Prepare for the Project Work:

1. **Go through** all your returned weekly programming exercises (exercises 1 – 7).
    a. List every task from each exercise that you have not finished.
    b. If you aim for a higher grade (4 – 5), add as many additional features to your project work as you have unfinished tasks in exercises 1 – 7. The topics of these features should cover the topics of the tasks.
2. **Estimate**, how many hours you have spent on this course:
    a. Doing the weekly exercises
    b. Keep track of how many hours you spend on this project work.

General Instructions:

3. **Think of** a topic for your project work. The topic cannot be exactly the same as those already done in exercises or presented by teachers in class (on this or any other course). Think of a "coherent" project work topic.
4. The following areas shall be covered in the work:
    a. Multiple classes
    b. Multiple modules
    c. Multiple instances of a class
    d. At least one part of the work modeled using UML sequence diagram
    e. At least one part of the work modeled using flow chart
    f. At least one part of the work modeled using UML class diagram
    g. Some interaction/relationship between classes (association)
    h. Inheritance
    i. Objects passed as function arguments
    j. Some data structure used (list, tuple, dictionary, …)
    k. Polymorphism

Additionally, these following guidelines must be adhered to:

a. Python Style Guide followed
b. Git version control used
c. Code should be well commented
d. Documentation (see bullet 6)
e. Testing (and evidence of testing visible in return document (test report))

5. **Assessment criteria:**
    a. pass (20 points): **project work should demonstrate students' ability to code classes in modules, create instances of class and pass instances as function arguments.** Most of the requirements listed in bullet 4 are visible in the project work. **The project work should demonstrate that students can confidently use the most important techniques of object-oriented programming.** Work is rather well documented.
    b. Challenging (31-40 points): **All topics listed in bullet 4 are covered.** The project work should demonstrate that students master all the topics covered in OOP course and exercise topic is not trivial. The work is very well documented and tested, and all instructions are followed. The more you see effort the higher the points. A simple program fulfilling all the requirements will give you 31 points.
6. Ideal Project Work Timeline

Week 1: Initial Planning and Setup

- Define the preliminary topic for your project.
- Create a schedule outlining what to do and when.
- Set your project goal (refer to bullet 5).
- Plan how to achieve the goal.
- List any missing tasks and topics (refer to bullet 1).
- Track the percentage of exercise tasks completed.
- Record the time spent on weekly exercises (refer to bullet 2a).
- Record the time spent on this part of the project work (refer to bullet 2b).
- Describe the topic and start coding (aim for 25-50% completion).

Week 2-3: Development and Mid-Project Review

- Continue developing your project based on the initial plan.
- Update the schedule if necessary.
- Ensure that most of the coding is completed (aim for 75-90% completion).
- Prepare a mid-project review document covering progress, challenges, and any adjustments made.

Week 3-4: Finalization and Submission

- Finalize the project, ensure all requirements are met (refer to bullet 3).
- Update the schedule if there are any changes.
- Track weekly hours spent on the project (refer to bullet 2b).
- Complete/update all diagrams and ensure the program is finished and well-commented.
- Conduct a self-assessment: what went well, challenges and successes, what you would still like to learn, and whether you are happy with the outcome.
- Prepare the final return document covering all the mentioned aspects (including the self-assessment). Make sure that all the code is in a remote repository.

Demonstration:

- **Deadline: before the the demonstration, if you want to demonstrate on 15th submit your work before the event and if you want to demonstrate on 23rd submit your work before that event.**
- Prepare to demonstrate your work in class for the whole class.
- Each group will have approximately 5 minutes for their demonstration.
- Ensure all materials and environments are ready for the demo.
- Demo days: **15.4.** and **23.4.**