2015/2016 EXAM GUIDE

CONTENTS

OBJECTIVE AND COURSE CONTENTS

OBJECTIVES

- Being familiar with some fundamental concepts in computer programming
- Introduction into basic concepts in software engineering
- Design, software architecture, software implementation and software maintenance
- Familiarization with some of the software tools used in large-scale application development
- Basic knowledge of Python 3.x programming language
- Using the Python language in software development, testing, running and debugging applications
- Learning/honing your own programming style ☺

COURSE CONTENT

PROGRAMMING IN THE LARGE

1. Introduction
2. Procedural programming. Compound Types
3. Test Driven Development
4. Modular Programming
5. Design guidelines in large scale programming
6. Exceptions
7. User Defined Types
8. UML. Design Principles.
9. Layered architecture. Inheritance.
10. Program Testing. Refactoring.

PROGRAMMING IN THE SMALL

11. Recursion. Computational complexity
12. Searching. Sorting
13. Problem solving methods
14. Preparing for the exam

EVALUATION

DURING THE SEMESTER

[L] 30% Laboratory – one grade on your semester activity

- 70% - Arithmetic average of 11 lab assignments, with 1 for the labs you were not graded for.
- 30% - Simulation grade

NB! The lab grade must be >5.00 (***2 decimals, no rounding***) in order to enter the written/practical examination

[S] 0 – 0.5 Seminar bonus – optional bonus for your activity during the seminars. This is added to the final grade.

DURING THE EXAM SESSION

[W] 40% Written exam – on your examination date.

NB! The written exam grade must be >5.00 (***2 decimals, no rounding***) in order to pass the class.

[P] 30% Practical exam – on your examination date, after the written exam (with a short break in between)

NB! You are encouraged to take this exam on your own laptop.

NB! The practical exam grade must be >5.00 (***2 decimals, no rounding***) in order to pass the class.

Final grade: 0.3*L + 0.4*W + 0.3*P + S.

THE RETAKE SESSION

- During the retake session you can hand in laboratory work, but are limited to a maximum laboratory grade [L] of 5.00.
- You can choose to retake the written, practical, or both examinations in case you have failed/not attended during the regular examination session.
- If you want to increase the grade you obtained during the regular session, you may partake during the retake session. Your final grade will be the largest one between those obtained.

EXAMINATION DATES

22.01.2016 – groups 917, 918 (backup date for groups 915, 916)

- Written exam at 11:00, C335
- Practical exam at 14:00, L307

27.01.2016 – groups 911, 914 (backup date for groups 912, 913)

- Written exam at 8:00, C335
- Practical exam at 8:00, L307

29.01.2016 – groups 915, 916 (backup date for groups 917, 918)

- Written exam at 11:00, C310
- Practical exam at 14:00, L307

06.02.2016 – groups 912, 913 (backup date for groups 911, 914)

- Written exam at 8:00, C335
- Practical exam at 11:00, L307

Important observations!

- **Make sure you've fulfilled your financial obligations towards the University, otherwise we are not allowed to grade you.**
- **To take the exam on the backup date, send me an email (arthur at cs.ubbcluj.ro) at least 48h beforehand!**
- **Re-check the date/time of the exam beforehand**
- **Arrive on time and bring a photo ID**
- **Bring your laptop**

WHAT THE EXAMINATION WILL COVER

BASIC ELEMENTS OF THE PYTHON LANGUAGE

- Instructions: ***=,==, if, while, for.***
- Predefined data types: ***integer, real, string, list, dictionary, tuple.***
- Functions: ***defining, parameter transmission, specification.***
- User defined types – ***classes, objects, (static) methods, attributes, inheritance.***
- Exceptions – defining ***exception types, raising, catching***.

1. What is the result of running the following code in Python:

```
def f(l):
    print("A")
    if l == []:
        raise ValueError()
    print("B")

def start():
    l = []
    try:
        print("A")
        f(l)
        print("D")
    except ValueError:
        print("C")

start()
```

Options:

a) AAD
b) AAC
c) A followed by Exception, program crash
d) AAB

2. What is the result of running the following code in Python:

```
class A:
    def f(self, l,nr):
        l.append(nr)

class B:
    def g(self, l, nr):
        nr=nr-1
        l = l+[-2]

a = A()
b = B()
l = [1,2]

c = -1
a.f(l,6)
b.g(l,c)
print(l,c)
```

ALGORITHMS – SPECIFICATION/TESTS/IMPLEMENTATION

Possibilities:

- You are given the specification – implement and test
- You are given the implementation – specify and test it
- You are given a test function – implement and specify the function it tests

3. Implement and test the function having the following specification

```
"""
    Compute the sum of even elements in the given list
    input:
        l - the list of numbers

    output:
        The sum of the even elements in the list

    Raises TypeError if parameter l is not a Python list
    Raises ValueError if the list does not contain even numbers
"""
```

4. Specify and test the following function

```
def function(n):
    d = 2
    while (d < n - 1) and n % d > 0:
        d += 1
    return d >= n - 1
```

ALGORITHM COMPLEXITY

You are given a function – analyze its complexity (best case, average case, worst case) as well as the extra-space complexity

5. Analyze the runtime complexity for the following function:

```
def complexity_1(x):
    m = len(x)
    found = False
    while m >= 1:
        c = m - m / 3 * 3
        if c == 1:
            found = True
        m = m / 3
```

6. Analyze the runtime complexity for the following function:

```
def complexity_2(x):
    found = False
    n = len(x) - 1
    while n != 0 and not found:
        if x[n] == 7:
            found = True
        else:
            n = n - 1
    return found
```

7. Analyze the runtime complexity for the following function:

```
def complexity_3(n, i):
    if n > 1:
        i *= 2
        m = n // 2
        complexity_3(m, i - 2)
        complexity_3(m, i - 1)
        complexity_3(m, i + 2)
        complexity_3(m, i + 1)
    else:
        print(i)
```

SEARCHING AND SORTING

Implement one of the studied search or sort functions:

- Searching:
  - Sequential search
  - Binary search
- Sorting
  - Bubble sort (with optimization)
  - Insertion sort
  - Merge sort
  - Quick sort

8. Write a function that sorts a list of numbers and has worst-case complexity of T(n) ∈ $O(n^2)$

9. You are given a shopping list that contains several ***Products***.

- **Product** has ***name***, ***type*** and ***price***

Write a function for sorting the shopping list:

- Alphabetically, by product name
- By price, decreasing

PROGRAMMING TECHNIQUES

Studied during this course:

- Divide and conquer
- Backtracking
- Greedy
- Dynamic programming

Possibilities:

- You are given a problem statement with a solution within one of the given techniques.
- Select the adequate technique for solving a given problem statement
- What we will ask:
  - Schematically indicate the solution
  - ***Backtracking:***
    - ***consistent***() and ***solution***() functions
  - ***Divide and conquer:***
    - ***Divide step*** – description, explain why you choose to do it that way
    - ***Conquer –*** describe how it works
    - ***Combine –*** describe how it works
  - ***Greedy method:***
    - Describe the ***set of candidates***
    - Describe how you make each ***selection***
    - Describe how you ***update*** the set of candidates
  - ***Dynamic programming:***
    - How is the principal of optimality observed
    - The recurrence describing the algorithm

10. Determine the longest subsequence of even numbers in a list using dynamic programming

11. Select the most appropriate technique and describe the solution for calculating the sum of the even numbers in a given list.

PRACTICAL EXAMINATION

Below you will find a problem statement similar to what you can expect to receive during the practical exam. The problem statement will in general follow the requirements set out between Lab 5 and Lab 10, will require writing specification, tests and the implementation of layered architecture.

Observations:

1. Solving the following problem statement completely should be possible for you in a timespan between 3 and 4 hours, as it has a longer list of requirements.
2. You are encouraged to bring your own laptop to the exam. You are free to use your preferred IDE as well as python implementation (2.x or 3.x). **Make sure your IDE is set up correctly and it works!**
3. The problem must be started from an empty workspace. The only allowed documentation is the one in your **Python/Doc** folder on the hard drive.
4. In order to pass the practical exam, you must ***implement at least 1 functionality*** end-to-end.

Things not required for the practical exam:

- Unlimited undo and redo operations.

PROBLEM STATEMENT

Write an application to help with the management of laboratory activities for a faculty course such as FP. Students enrolled in the class can be assigned one of the **20 problem statements** (numbered from **1 to 20**) from each **laboratory**, and when they turn it in they are graded. The application will be used by the teacher and will provide the following functionalities:

- Add a student to the course. Each student has an ***id***, a ***name*** and a ***group***. You cannot have more than one student having the same ***id***, as well as students without a name or group.
- Remove a student from the course. A student can only be removed while they have not received any grades.
- Assign a laboratory problem statement to a student. You cannot assign more than one problem statements from the same laboratory to a student. If the student was already assigned a problem, the program must report the error.
- Assign a laboratory to a group. Each student in the group will be assigned a problem statement. Implement an algorithm to assign students with problem statements (e.g. each subsequent student is assigned the next problem in the list of problem statements). In case a student was already assigned a problem statement, this must not be changed!
- Grade a student for a laboratory, with an integer grade 1 to 10. Validate that the grade is valid. Grades cannot be changed!
- Best/worst students in a group. Given a group, list its students in decreasing order of their average grade.
- Students failing the class. Provide the list of all the students (regardless of group), for whom the average grade is less than 5.
- Undo/redo the last performed operation that changes the list of students or grade assignments.

NB! Data is **loaded** and **saved** to two text files: "***students.txt***", "***grades.txt***". When starting the program, make sure to have at least 10 students in the students file.

NB! Solutions without file-based repositories are acceptable, but the maximum grade for such solutions is ***7***.

NB! Your solution must adhere to the principles of layered architecture studied during the course. You will have domain, repository, controller, and UI packages/modules. The diagram below provides a guideline regarding the implementation, but it is not meant to be exhaustive.

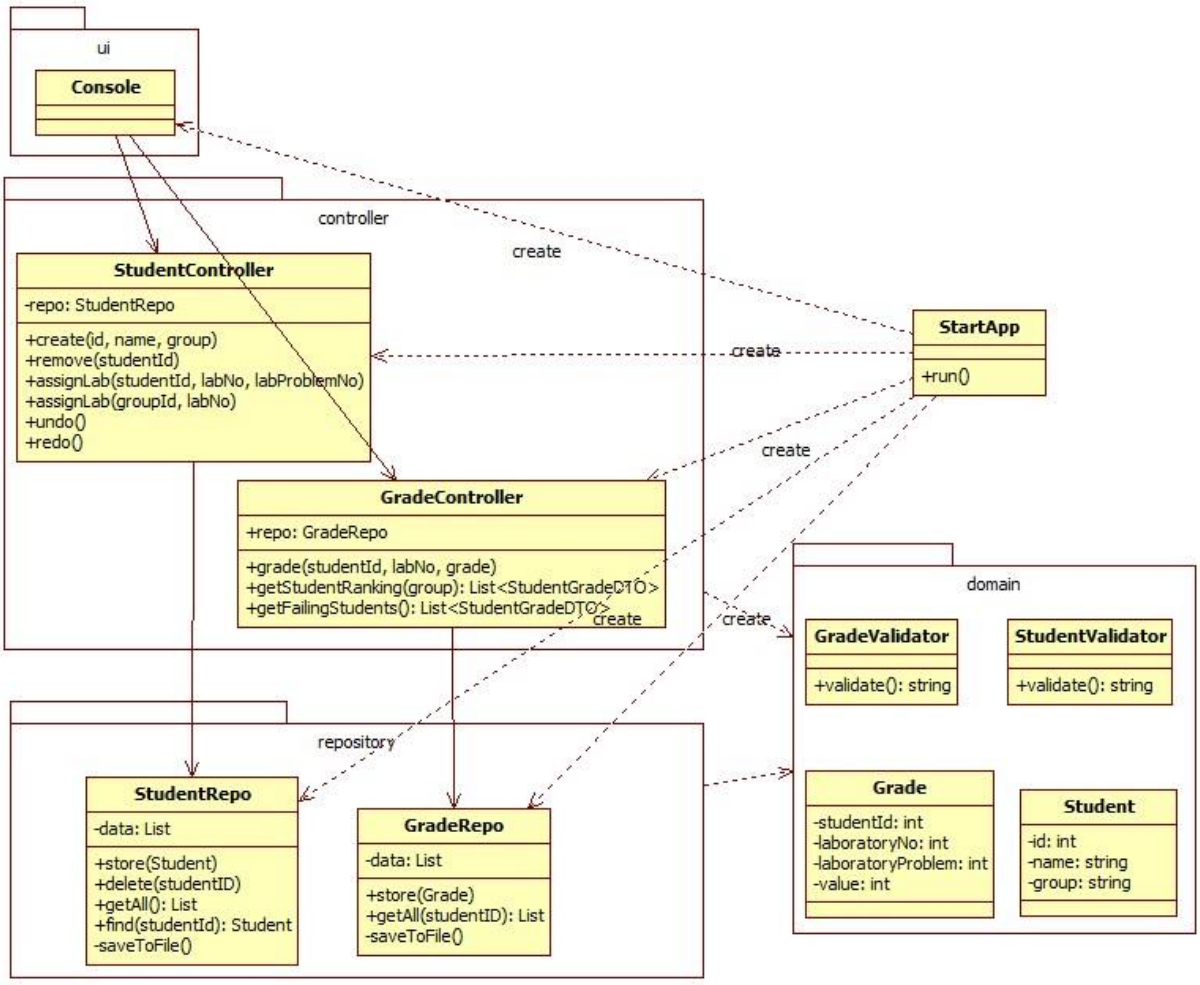

FIGURE 1 - CLASS DIAGRAM