Programme Code: TU856, TU858
Module Code: CMPU2016
CRN: 23094, 22399

TECHNOLOGICAL UNIVERSITY DUBLIN

CITY CAMPUS - GRANGEGORMAN

TU856 – BSc. (Honours) in Computer Science
TU858 – BSc. (Honours) in Computer Science (International)

Year 2

SEMESTER 2
EXAMINATIONS 2024/25

Object Oriented Programming

Internal Examiner:
Dr. Lucas Rizzo
Dr. Jelena Vasić
Dr. Paul Doyle

External Examiner:
Dr. Colm O'Riordan

Exam Duration: 3 hours

Instructions:
There are 2 sections on the paper: section A and section B. Candidates must answer two questions out of each section. Answer four questions in total. All questions carry equal marks.

SECTION A
OO Programming through Python

Answer any TWO questions of the THREE Questions in this Section

1. (a) Consider the following Python class definition:

```
class Rectangle:
    def __init__(self, width, height):
        self.width = width
        self.height = height

    def __mul__(self, factor):
        return Rectangle(self.width * factor, self.height * factor)

    def __str__(self):
        return "Rectangle({}, {})".format(self.width, self.height)
```

(i) Identify the attributes and methods in the Rectangle class **(2 marks)**

(ii) Write a short statement defining what operator overloading is in Python. **(3 marks)**

(iii) Describe the purpose of the implemented methods in the class. **(3 marks)**

(iv) Describe the result of each line of the following code snippet and state the expected output: **(5 marks)**

```
r1 = Rectangle(2, 4)
r2 = r1 * 3
print(r2)
```

(v) Describe the reason why the following code results in an error. Explain how it could be fixed (no need to write the code). **(5 marks)**

```
r3 = Rectangle(5, 10)
r4 = (2, 3) * r3
print(r4)
```

(b) A company keeps track of employee salaries using a dictionary where keys represent employee names and values represent their salaries.

```
def process_salaries(salaries):
    high_earners = {}

    for salary in salaries.values():
        if salary > 50000:
            high_earners[salary] = True
        else:
            high_earners[salary] = False

    return high_earners

emp_salaries = {"John":30000, "Mike":55000,
"Sarah": 55000}

high_emp = process_salaries(emp_salaries)
```

(i) Identify and describe the issue in how employees are stored in the `high_earners` dictionary. **(4 marks)**

(ii) Modify the function so that it correctly identifies employees earning above 50,000. **(3 marks)**

2. (a) A company is developing a basic inventory system using Python lists.

```
products = ["apple", "banana", "carrot"]
quantities = [5, 10, 3]

quantities[-1] = 8  # Updating stock count

print(products + quantities)  # Line 1
print([products[1], quantities[-1]])  # Line 2
print(products.append(quantities))  # Line 3
products.sort()
print(products)  # Line 4
```

(i) What is the output of Line 1 (as indicated by comments)? How was it achieved? **(2 marks)**

(ii) What is the output of Line 2 (as indicated by comments)? Explain how indexing works in this case. **(2 marks)**

(iii) What is the output of Line 3 (as indicated by comments)? Explain how it was achieved. **(3 marks)**

(iv) Line 4 (as indicated by comments) results in an error. Why? **(3 marks)**

(b) (i) Design a class named `ScienceFairProject` that tracks the following information for each project: the project title, the student's name, their school, the category (e.g., Biology, **(5 marks)**

Chemistry, Physics), and the project's score (out of 100). Include a `__str__` method to enable easy printing of the project details.

(ii) Implement the `__lt__` method in your `ScienceFairProject` class to overload the < operator. A project with a lower score should be considered worse than a project with a higher score. If two projects have the same score, the project whose title comes first alphabetically should be considered worse. **(5 marks)**

(iii) Implement a new `CSProject` class to represent computer science projects that inherits from `ScienceFairProject`. The `CSProject` should have the same attributes as `ScienceFairProject` plus an additional attribute called `programming_language` (e.g., Python, Java, C++) to specify the programming language used in the project. Implement the class with a constructor and `__str__` methods. **(5 marks)**

3. (a) Analyse the Python code below, which aims to read student grades from a dictionary and perform arithmetic operations:

```
grades = {'Alice': '85', 'Bob': 92, 'Charlie': 'absent'}

try:
    student_name = input("Enter student name:")
    grade = grades[student_name]
    final_score = int(grade) + 5
except KeyError:
    final_score = 'Student not found'
except ValueError:
    final_score = 'Invalid grade format'
except TypeError:
    final_score = 'Grade type error'

print(final_score)  # Line 1
```

(i) Explain what an exception is in Python and describe the purpose of two built-in exceptions (3 marks) **(3 marks)**

(ii) What output does Line 1 (as indicated in comments) produce if the input is 'Bob'? Justify your answer clearly. **(2 marks)**

(iii) What output does Line 1 (as indicated in comments) produce if the input is 'Alice'? Clearly explain why this happens. **(2 marks)**

(iv) What output does Line 1 (as indicated in comments) produce if the input is 'Charlie'? Clearly explain why this happens. **(3 marks)**

(b) Imagine you own a popular pizza shop called "Slice of Joy," and you want to automate your pizza ordering system. However, you've accidentally hired a developer who lost the `pizza` module and only gave you the main scope of the system.

```
from pizza import Pizza, Pizzeria

# Main scope
pizzeria = Pizzeria("Slice of Heaven")

crazy_pineapple_pizza = Pizza(name="Crazy
Pineapple", toppings=["pineapple", "cheese",
"ham"], slices=8)

pizzeria.add_pizza_to_menu(crazy_pineapple_pizza)
```

(i) What is the relationship between the Pizzeria and Pizza class and how can you see that in the code? **(2 marks)**

(ii) Implement the Pizza class constructor with attributes for name, toppings, and slices. **(2 marks)**

(iii) Implement methods to your Pizza class to add toppings, remove toppings and check if the pizza has a certain topping. **(5 marks)**

(iv) Implement the Pizzeria class constructor. Include attributes for name (string) and menu (a collection of pizzas). **(2 marks)**

(v) Implement the `add_pizza_to_menu` method to add a pizza to the menu and facilitate customer orders based on pizza name. Make sure to not add the pizza if it is already in the menu. You can use the pizza name attribute to identify a pizza. **(4 mark)**

SECTION B
OO Programming through Java

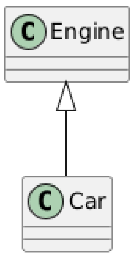
Answer any TWO questions of the THREE Questions in this Section

4. **(a)** Explain the difference between abstraction and generalisation. **(4 marks)**

(b) Use two Java constructs to illustrate generalisation. One must be a construct available in object-oriented languages only and another a construct that has nothing to do with object-orientation. For each construct you must explain how it embodies generalisation, providing a brief example in code or pseudocode. **(10 marks)**

(c) Name two software quality attributes that benefit from generalisation. For each explain how it benefits. **(6 marks)**

(d) Give an example in Java code or pseudocode, with a brief explanation, of how abstraction helps with achieving the object-oriented principle of high cohesion – low coupling. **(5 marks)**

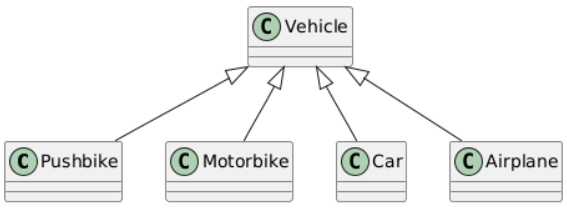
5. **(a)** 

Explain what is wrong with the design in the given UML class diagram, if the classes are meant to represent the things most commonly understood as ‘engine’ and ‘car’. **(4 marks)**

(b) Redraw the diagram to reflect how things really are in the real world. **(4 marks)**

(c) Extend the diagram to add another class that models the real world and has a relationship (of any kind) with at least one other class in the diagram. Explain your thinking. **(4 marks)**

(d) The UML class diagram below is a snippet of the design for a program that will be used by a versatile neighbourhood mechanic. Assuming that the specific vehicle types have to be represented as distinct classes, which common coding principle is probably being violated in this portion of the design and why? **(4 marks)**



(e) Write a minimal implementation in Java of the class Vehicle and two of the subclasses from part (d). The implementation must include: **(9 marks)**

- method `notifyOwner(String message)`, which uses a name and email address to send the passed-in message via a made-up API
- method `orderParts()`, which places an order, via another made-up API

When deciding how to implement the methods, assume that ordering parts for different vehicles involves different processes and that notifying owners is the same regardless of vehicle type. Include any required fields in your implementation and make fields and methods public only if necessary. Assume that the class `Part` is already defined and represents any part for any vehicle.

6. **(a)** What are the three kinds of exceptions in Java, with respect to how they arise and whether they must be part of method signatures? Explain the reasoning behind this classification. And give an example scenario for each of the three kinds (no code needed). **(9 marks)**

(b) Define a Java class to implement a suitable exception for the following scenario: a method you defined needs a positive integer passed as an argument but it receives the value -3. **(4 marks)**

(c) Write some Java or pseudocode that implements a scenario where it is appropriate for an exception to be 'let through' by a method, rather than caught by it. Explain the scenario and the code. **(8 marks)**

(d) What is the point of the `finally` clause? Why not just write the same code below a `try-catch` statement? Explain. **(4 marks)**