National Certificate: Information Technology (Systems Development)
Software Design & Development
SAQA ID: 48872
NQF level: 5

Programme Title: SDD (AD, GD, SD)
Module Title: Programming with C#
Module Code: PRG521
Summative Exam: Semester 1

2024

CTU training solutions
Creative & Technology Universitas

Table of Contents

EXAMINATION RULES AND REGULATIONS

The Summative Assessment will now be conducted in this room on the following qualification:

National Certificate: Information Technology (Systems Development) presented at NQF level 5.

Students who have not registered for this programme and or module should now leave the venue.

- This paper consists of a 3-hour duration.
- ID cards and student cards must be placed on the top right-hand corner of the desk.
- You may not disregard the instructions of the invigilator.
- If you do not obey these instructions, you render yourself liable to suspension from future assessments, and you may be refused credits for your assessments.
- No smoking, eating, or drinking is allowed in the venue.
- You are not allowed to assist another student or try to assist him/her or communicate with anybody other than the Invigilators.
- Any questions should be directed to an Invigilator.
- If you require attention, please raise your hand; an invigilator will attend to you.
- No explanations of questions may be requested or provided.
- You may not create a disturbance in the venue or behave in an improper or unseemly manner.
- Please ensure that **ONLY the Campus Online browser** is open on your computer. Please note that Campus Online has a logging system that tracks your actions and that we will be able to trace logins in and out of Campus Online during your exam session.
- You may not make use of any paraphrasing/copying, or access any **AI Apps, example ChatGPT, Co-pilot or any other App** during your exam.
- This is a **closed-book** test/ examination.
- No cell phones or smart watches are allowed in the venue.
- **Please hand in all switched-off cell phones and smart watches at reception. Cell phones and smart watches may not be stored in any bag in the venue.**
- **No handbags, laptop bags, or any other bags will be allowed in the exam venue.**
- All calculator cases, wallets, etc. must be placed on the floor.
- You will not be able leave the venue during the **first hour and the last 15 minutes** of the session. In an emergency, a student will be allowed to leave the venue under supervision.
- You must show the Invigilator evidence that you **submitted your assessment on CampusOnline, that you are logged out of Campus Online**, and that your computer is switched off before you leave the exam venue.
- As soon as you have submitted your exam you must leave the venue quietly.

Programming with C# (NQF 5)
Summative Assessment
Examiner: Thabang Mashile
Internal Moderator: Dr. Cath Opperman
Date: 20/09/2024
Duration: 180 minutes (3 hours)
Total marks: 100

Instructions:

1. This is a closed-book test/ examination.
2. Independent work is required.
3. Ensure that your project is named and submitted as per specifications.
4. Read your brief carefully.
5. Complete all questions.

All the required evidence must be submitted to CampusOnline in a single merged pdf file saved as: ***yourname_studentnumber_PRG521_SA.zip***.

Please note: The submission rules for summative assessments as described in your Program guide state that students do not have an extended three (3) days to submit their final exam project as with formative assessment submissions.

No exam projects will be accepted after the due date.

Please note: Any student found guilty of submitting plagiarised written and/or visual content will face disciplinary action.

Exit Level Outcomes

A learner will be able to:

- Communicate effectively with fellow IT staff & users of information systems.
- Understand the role of technology in the business context.
- Demonstrate an understanding of problem-solving techniques, and how to apply them in a systems development environment.
- Demonstrate an understanding of Systems Development, with all its implications.
- Relate business problems and information technology solutions.
- Apply the principles of creating computer software.

Assessment Outcomes

Students should be able to:

- Understanding recursion and its implementation in C#.
- Knowledge of method overloading and its practical use.
- Comprehension of optional parameters in method definitions.
- Familiarity with de-constructors and their use.
- Understanding the use of static members in a class.
- Ability to create and use extension methods.
- Ability to handle multiple exception types in C#.
- Understanding the concept and implementation of rethrowing exceptions.

Summative Assessment Submissions

Assessment	Submission date
PRG521 Summative Assessment	20 Sept 2024 12:00 PM

Please note – There are two steps that need to be completed before an Exam is deemed to be submitted successfully:

- **Step 1:** Evidence required in the specified formats is submitted to CampusOnline by the deadline date specified.
- **Step 2:** Evidence required in the specified formats to be uploaded in the specified location.
- Exams not submitted in this manner will not be assessed.

Please complete the declaration of authenticity below for all assignments:

DECLARATION OF AUTHENTICITY

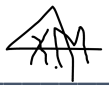
I Xola Magatya

(FULL NAME)

*hereby declare that the contents of this assignment*_____

is entirely my own work with the exception of the following documents (List the documents and page numbers of work in this portfolio that were generated in a group):

Activity	Date

Signature: 

Date: 20/09/2024

SECTION A - Theory

Instructions: Use the answer space provided to answer the following questions.

Question 1

1.1 Explain the concept of recursion in C#. (3)

Recursion it is whereby a method calls itself repeatedly until specified condition has been met.

example:

```
return n * Factorial( n - 1 );
```

1.2 What is method overloading in C#? (3)

Method overloading it when you are able to define multiple methods in the same class , with the same name but different parameters.

1.3 Describe the use of optional parameters in C#. (2)

Optional parameters lets you create methods where some parameters are not required , if you dont provide value for optional parameters the default value will be used.

1.4 Explain the concept of de-constructors in C#. (3)

It is the type of method used in classes to break down an object into its individual compenents and it is useful when you need to access multiple properties of an object at the same time

1.5 Discuss the use of static members in a C# class. (3)

Static members are used to share data and the functionality of the class without needing to create the instance object of the class.

1.6 What are extension methods in C#? (3)

It is a way to add functionality to existing types without modifying their source code or creating new derived types

1.7 Explain the concept of class derivation and method overriding in C#. (4)

Class derivation allows you to create a new class based on a existing class.

Method overriding allows derived class to provide specific implementation of a method that is already defined in its base class.

1.8 Describe the purpose of abstract classes in C#. (3)

Purpose of abstract classes provides common base for derived classes, And it also enforce implementation also making it easier for you to reuse the code again

1.9 Explain how to handle multiple exception types in C#. (3)

Create multiple catch blocks for different errors , one catch with related errors , catching base exception to handle similar errors together & finally block for clean up task.

1.10 What does it mean to rethrow an exception in C#? (3)

It means to catch an exception and then throw it again.

1.11 Explain the concept of encapsulation in C#. How does it contribute to the security and robustness of an application? (6)

Encapsulation is a way to bundle data and methods that work on that data into a single unit called a class , hiding some of the internal details of that class from the outside world.

& it contribute to the security and robustness of an application by protecting data, preventing errors , simplifying errors & maintanance making code easier to understand and test.

1.12 Describe the role of interfaces in C#. How do they differ from abstract classes? (6)

Interfaces are set of methods , properties, events or indexers that implementing classes must provide.

And the difference between Interfaces and abstract classes is that Interfaces contains only the signatures of the methods, properties, events & no implementation is provided

while the abstract classes contain fully both implemented methods and abstract methods

1.13 Define polymorphism in the context of C#. Differentiate between compile-time and run-time polymorphism. (6)

Polymorphism - Is a feature allowing objects to be treated as instances of their parent class , even when they are instances of the derived classes.

The difference between compile-time and run-time polymorphism is that the run-time polymorphism occurs when the method to be executed is determined at run-time usually involves method overriding.

& the compile-time polymorphism occurs when the method to be executed is determined at compile-time usually involves method overloading.

1.14 Define the concept of 'LINQ'. (2)

Language Intergrated Query is a feature that allows you to query and manipulate data in a short & clear readable way.

(50 Marks)

SECTION B – Practical

Instructions: Use the relevant software application to answer the following questions. Please note that making use of GUI is optional.

Question 2

Write a C# program that simulates a simple car dealership inventory management application. The program should include the following:

Methods and Parameters

- Implement a method to add a new car `'AddCar'` to the dealership's inventory. Use method overloading to allow adding a car with different sets of parameters (e.g., with or without optional features).
- Implement a recursive method to search for a car by its VIN (Vehicle Identification Number).

Classes

- Create a Car class with properties such as `'Make', 'Model', 'Year', 'VIN'`, and `'Price'`. Implement a de-constructor to clean up any resources if necessary.
- Create a `'Dealership'` class that uses static members to keep track of the total number of cars. Implement an extension method to display car details.

Exception Handling

- Implement exception handling in the `'AddCar'` method to handle cases such as duplicate VINs or invalid data.
- Demonstrate catching multiple exception types, such as `'ArgumentNullException'` and `'FormatException'`, when adding a new car.
- Implement a general catch block to handle any other exceptions and rethrow the caught exception for logging purposes.

(50 Marks)

[Total = 100 Marks]

ASSESSMENT RUBRIC

Methods and parameters

Poor (0 – 1 Mark)	Needs Improvement (2 – 3 Marks)	Satisfactory (4-5 Marks)	Excellent (6-8 Marks)
Method not implemented or incorrect	Method implemented with major issues	Method implemented with method overloading containing minor issues	Implement a method to add a new car to the dealership's inventory
Method not implemented or incorrect	Method implemented with minor issues	Method implemented and functioning with minor issues	Implement a recursive method to search for a car by its VIN

Classes

Poor (0 – 1 Mark)	Needs Improvement (2 – 3 Marks)	Satisfactory (4-6 Marks)	Excellent (7-9 Marks)
Class not implemented or incorrect	Class implemented with major issues	Class implemented with minor issues	Class correctly implemented with all properties and a de-constructor. (& properties such as Make, Model, Year, VIN, and Price)
Class not implemented or incorrect	Class implemented with minor issues	Class implemented with major issues	Class correctly implemented with static members and an extension method (& keep track of the total number of cars)

Exception handling

Poor (0 – 1 Mark)	Needs Improvement (2 – 3 Marks)	Satisfactory (4 Marks)	Excellent (5 Marks)
Exception handling not implemented or incorrect	Exception handling implemented with minor issues	Exception handling correctly implemented for all specified cases	Implemented exception handling in the AddCar method to handle cases such as duplicate VINs or invalid data
Not demonstrated or incorrectly demonstrated	Partially demonstrated with minor issues	Correctly demonstrated and functioning correctly	Demonstrated catching multiple exception types, such as ArgumentNullException and FormatException, when adding a new car
Not implemented or incorrectly implemented	Partially implemented with minor issues	Correctly implemented and explained	Implemented a general catch block to handle any other exceptions and rethrow the caught exception for logging purposes