



Syllabus

SCHOOL OF TECHNOLOGY & COMPUTING

IS 375: C# - Intermediate

5 Credits

Effective: Spring 2020

Access to the Internet is required.

All written assignments must be in Microsoft-Word-compatible formats.

See the library's APA Style Guide tutorial for a list of resources that can help you use APA style.

FACULTY

Faculty Name: FACULTY NAME

Contact Information: CONTACT INFORMATION

[INSTRUCTOR MAY INSERT PERSONAL MESSAGE IF DESIRED]

COURSE DESCRIPTION

In this course students develop their understanding of the C# programming language, applying it to applications, databases and Windows programming. In the first part of the course students use C# / C++ to create code. The course then covers designing, implementing and accessing databases to store large data sets. Students will learn the concepts of event-driven programming, message processing, and Windows program structure.

COURSE RESOURCES

Doyle, B. (2015). *C# Programming: From Problem Analysis to Program Design (5th Edition)*. Cengage Learning. (ISBN: 1305724674, 9781305724679)

CITYU LEARNING GOALS

This course supports the following City University learning goals:

- Professional competency and professional identity

COURSE OUTCOMES

In this course, learners:

- Access data from databases in the program context.
- Design and implement databases.
- Implement applications effectively in C#.
- Apply object-oriented concepts.
- Follow quality coding practices to produce documented, legible, maintainable, reusable and efficient code.
- Implement Windows applications.

CORE CONCEPTS, KNOWLEDGE, AND SKILLS

Topics include:

- Add a status bar to an application.
- Add controls to a dialog.
- Add handlers to update menu properties.
- Add toolbar buttons and associate them with existing menu items.
- Alter the sequence of program execution based on a comparison.
- Apply logical operators and expressions to obtain a value.
- Become familiar with Visual C# professional integrated development environment (IDE).
- Compare and contrast methods of storing data.
- Compare C# programming to C++ programming.
- Compile, link and execute C# programs.

- Create a basic Windows Forms application.
- Create a basic Windows Presentation Foundation application.
- Create a dialog class to manage a dialog.
- Create a function to service the message generated when a menu item is selected.
- Create and execute basic Windows programs.
- Create and modify menu properties.
- Create and modify menu resources.
- Create and use constructors for classes.
- Create complex queries which join data from multiple tables.
- Create dialog resources.
- Define a new class in terms of an existing one.
- Define and implement classes.
- Define and use init only fields.
- Define and use literal fields.
- Define and use properties in a C# class.
- Define custom shapes.
- Define handlers for mouse messages.
- Define what is accessible in a DLL.
- Define, initialize and use a Struct.
- Demonstrate how entities and attributes are identified.
- Demonstrate how views can create alternate perspectives of the data which can be useful for use in programs.
- Describe and implement a function for handling Windows messages.
- Describe how boxing and unboxing are used to allow fundamental types to behave as objects.
- Describe how Single Document Interface (SDI) and Multiple Document Interface (MDI) applications differ.
- Describe how SQL statements can be used to manage a database or schema.
- Describe modes of interacting with databases.
- Describe namespaces.
- Describe the visibility determined by Access Specifiers for class and interface members.
- Determine how to print from an application.
- Diagram the .NET Framework and discuss advantages it offers.
- Discover basic input and output in C#.
- Discover the mechanism for accepting a variable number of arguments in a C# program.
- Discuss how C# classes support the overloading of operators.
- Discuss the concept and purpose of normalizing tables in a database.
- Discuss the rules for deriving classes in C#
- Distinguish the types of relationships between two entity sets, including 1 to 1, 1 to many, many to 1, and many to many.
- Draw a graphical representation of an entity relationship model.
- Examine how relationships are used to associate entities.
- Examine the strengths and weaknesses of different database models.
- Explain and provide examples of the first normal form.
- Explain and provide examples of the second normal form.
- Explain and provide examples of the third normal form.
- Explain the coordinate system Windows uses for drawing in a window.
- Explain the type and scope of a variable.
- Explain tracking handles and tracking references and why they are needed in a CLR program.

- Explain what a class destructor is and when and why it is necessary.
- Explain when normalizing data may not be appropriate.
- Explore the three basic ways of creating an interactive Windows application using Visual C++.
- Handle multiple choice situations.
- Have a program capture the mouse.
- Identify key items for data base design including data to be stored, operations to be performed, frequency of operations and constraints on the data.
- Illustrate the organization of a Windows program.
- Implement loops in a C# program using for, for-each, while and do-while.
- Implement scrolling in a view.
- Implement serialization.
- Implement SQL statements used to fetch, store and manipulate data.
- Implement variables in C#.
- Install and browse a relational database.
- Make objects of a class serializable.
- Outline how the Entity-Relationship Model is used in designing databases.
- Outline the basic structure of a Window.
- Outline the structure of a C# program.
- Pass data to and from functions.
- Program the creation of a dialog box and get information back from the controls in it.
- Program the mouse to draw shapes in a window.
- Review example relational databases.
- Understand how inheritance fits into object-oriented programming.
- Understand how serialization works.
- Understand the Windows API and how it is used.
- Understand what an exception is and how to write exception handlers.
- Use generic functions in C#.
- Use recursive functions.
- Use statements to compare data values.
- Use the capabilities provided by a device context to draw shapes.
- Use virtual functions.
- Write and implement partial functions.
- Use, declare and initialize arrays of different types.
- Work with strings and arrays in C# programs.
- Write and declare C# functions.
- Write multiple functions with a single name to handle different kinds of data automatically.

OVERVIEW OF COURSE GRADING

The grades earned for the course will be derived using City University of Seattle’s decimal grading system, based on the following:

OVERVIEW OF REQUIRED ASSIGNMENTS	% OF FINAL GRADE	POINTS
The Muddiest Point (MP)	5%	50 = 5 points * 10 modules
Concept Test (CP)		
Discussion Board (DB)	10%	150 = 15 points * 10 modules

Virtual Lab (VL)		
Programming Exercise (PE)	40%	400= 40 points * 10 modules
Hands-On Practice (HOP)	25%	250= 25 points * 10 modules
Research Paper (RP)		
Knowledge Check (KC)		
Team Project (TP)	20%	Proposal: 30 points Progress: 70 points Final Report: 100 points Subtotal: 200 points
TOTAL	100%	1,000 points

The following approaches are used for developing this course content:

Assessment

- Summative Assessment. https://en.wikipedia.org/wiki/Summative_assessment
- Formative Assessment. https://en.wikipedia.org/wiki/Formative_assessment

Classroom Assessment Techniques

- The Muddiest Point. https://en.wikipedia.org/wiki/Classroom_Assessment_Techniques

Active Learning. https://en.wikipedia.org/wiki/Active_learning

- Flipped Classroom. https://en.wikipedia.org/wiki/Flipped_classroom
- Just-in-time Teaching (JiTT). https://en.wikipedia.org/wiki/Just-in-time_teaching
- Peer Instruction. https://en.wikipedia.org/wiki/Peer_instruction

Learning Theory

- Learning-by-doing. <https://en.wikipedia.org/wiki/Learning-by-doing>
- Project-Based Learning (PBL). https://en.wikipedia.org/wiki/Project-based_learning
- Social Learning. [https://en.wikipedia.org/wiki/Social_learning_\(social_pedagogy\)](https://en.wikipedia.org/wiki/Social_learning_(social_pedagogy))

Evidence-Based Practice (EBP). https://en.wikipedia.org/wiki/Evidence-based_practice

- Pair Programming. https://en.wikipedia.org/wiki/Pair_programming
- Stand-up Meeting. https://en.wikipedia.org/wiki/Stand-up_meeting
- Agile Software Development. https://en.wikipedia.org/wiki/Agile_software_development

SPECIFICS OF COURSE ASSIGNMENTS

The instructor will provide grading rubrics that will provide more detail as to how this assignment will be graded.

Muddiest Points

Each week, students are required to finish the muddiest point activity before starting their class. The purpose of this activity is not to evaluate your knowledge but to encourage your engagement in class. This activity consists of an essay question called the muddiest point and one or more multiple choice questions.

The Muddiest Point is a general Classroom Assessment Technique (CAT), in which we can assess how well students are comprehending key points during a lesson or a course. This technique asks students to briefly state what part of the assignment was most confusing for them.

This activity must be done before the next class. Students that do not complete this activity before the beginning of the next class will lose a maximum of 5 points from the weekly class activities.

<i>Components</i>	<i>% of Grade</i>
Quality Participation: Meets requirements in a timely manner	60%
Writing: Is clear, concise, and grammatically correct.	20%
Accuracy: Answers quizzes correctly	20%
TOTAL	100%

Concept Tests

Each week, students in on-ground classes are required to complete the ConcepTest, which is a set of multiple-choice questions. The purpose of this activity is to evaluate how well the materials provided in class are retained by students. The ConcepTest is a general CAT.

This activity must be completed before the next class. Students that do not finish this activity due the due date / time will lose a maximum of 5 points from the weekly class activities.

<i>Components</i>	<i>% of Grade</i>
Quality Participation: Meets requirements in a timely manner	100%
TOTAL	100%

Hands-on-Practice (HoP)

If necessary, the instructor may assign hands-on practices to individual (usually in an online course) or a pair of students (usually in an on-campus course).

If students do not finish this activity, they will lose the maximum 5 points from the weekly Class Activities.

<i>Components</i>	<i>% of Grade</i>
Quality Participation: Meets requirements in a timely manner	80%
Accuracy: Answers questions correctly	20%
TOTAL	100%

Programming Exercise (PE)

The students must individually perform the programming exercise which is based on the topics and Hands-on Practice. No code sharing or copying from other sources are allowed. Non-executable programs will not be graded. The programs in poor coding styles will be asked to be resubmitted.

<i>Components</i>	<i>% of Grade</i>
Accuracy: Answers questions correctly	80%
Writing: Is clear, concise, and grammatically correct.	20%
TOTAL	100%

- **PE1 : Multi-dimensional Array Program**

Write an application that creates a two-dimensional array. Allow the user to input the size of the array (number of rows and number of columns). Fill the array with random numbers between 0 and 100. Search the array for the largest value. Display the array values, numbers aligned, and the indexes where the largest value is stored.

The instructor may provide further specific details.

<i>Components</i>	<i>% of Grade</i>
Implement C# applications	30%
Quality coding practices	20%
Access data	50%
TOTAL	100%

- **PE 2 : Windows Application Program (I/O)**

Create a Windows application that can be used to input a user’s name. Include an appropriate label indicator for the name and a textbox for the input entry. A button labeled Submit should retrieve and display the value entered on another label positioned near the bottom of the form. The font color for the text for the label object should be yellow. Change the background color of the form to an appropriate one to use with your yellow text. Change the Font property to a font of your choice. The size of the font for all objects except the Button should be at least 14 points. The Button font should be 16 points. Add a title caption of “Name Retrieval App” to the form. Initially clear the text from the label that will display your final answer. When the Submit button is pressed, retrieve the name and concatenate that value with a Congratulatory message. For example, you might display, “Congratulations, Brenda Lewis, you retrieved the data!” if your name was Brenda Lewis. Position the button so it is in the center of the form. Align the other controls so they are aesthetically pleasing. Be sure to change the default names of all controls involved in program statement.

The instructor may provide further specific details.

<i>Components</i>	<i>% of Grade</i>
Implement C# applications	20%
Quality coding practices	10%
Proper form development	50%
Access data	20%
TOTAL	100%

- **PE 3 : Event Handling (Forms)**

Create a Windows application that can be used to change the form color. Your form background color should initially be blue. Provide at least two buttons with two different color choices and a Reset option. Change the font style and size on the buttons. Align the buttons so that they are in the center of the form. The color choice buttons should be the same size. Add event handlers for the buttons so that when the user clicks the button, the form changes color, and a message box is displayed alerting the user as to what color the

form is. Be sure to name any controls used in program statements prior to registering your event. Change the default title bar text.

Hint: This exercise may require you to do some research. You may want to review the code placed in the .Designer.cs file after you set the form's initial color.

The instructor may provide further specific details.

<i>Components</i>	<i>% of Grade</i>
Implement C# applications	30%
Object-oriented programming	20%
Quality coding practices	20%
Manage events	30%
TOTAL	100%

- **PE 4 : “Equipment Needs” Form Program**

Create a Windows application that can be used as a sign-up sheet for ski equipment for the Flyers Sports Club. The club has ski equipment that it makes available to members at a minimal charge. In an attempt to determine what type of equipment members might need for an upcoming trip, they have asked you to design and implement an equipment-needs form. Include CheckBox objects that allow users to select the type of gear they will need to purchase for the trip. Include selections of snow gloves, skis, goggles, earmuffs, and other items you feel are appropriate. Include at least one picture image on your application. After all selections are made, display a message indicating what items have been selected. You will probably want to include menu options to display and clear the order for the next user. Also include an option that enables the user to exit the application.

The instructor may provide further specific details.

<i>Components</i>	<i>% of Grade</i>
Forms development	70%
Quality coding practices	30%
TOTAL	100%

- **PE 5 : Banking Account**

Create a base class for a banking account. Decide what characteristics are common for checking and saving accounts and include these characteristics in the base class. Define subclasses for checking and savings. In your design, do not allow the banking base account to be instantiated—only the checking and saving subclasses. Include a presentation class to test your design.

The instructor may provide further specific details.

<i>Components</i>	<i>% of Grade</i>
Implement C# applications	50%
Object-oriented programming	30%
Quality coding practices	20%
TOTAL	100%

- **PE 6 : File (Input / Output) Program**

Write an application that retrieves both string data and numbers from a text file. Test your solution by retrieving names of students and three scores per line from a text file. Process the values by calculating the average of the scores per student. Write the name and average to a different text file. Display on the console screen what is being written to the new file. Test your application with a minimum of eight records in the original file. Hint: You might consider adding delimiters between the data values in the original text file to simplify retrieving and processing the data. Include appropriate exception-handling techniques in your solution. When the application closes, locate the text file and verify its contents.

The instructor may provide further specific details.

<i>Components</i>	<i>% of Grade</i>
Windows Applications development	50%
Object-oriented programming	30%
Quality coding practices	20%
TOTAL	100%

- **PE 7: Family Database**

Create a small Family database with one table to include data about members of your family. Include data fields such as first name, last name, type of relationship, hometown, and age. Include one field that uniquely identifies each record, such as a family member number. You can be creative with the family member number or use the auto-generated number from the database. Populate the database with members of your family. Be sure to include data about yourself in the table. Place at least 10 records in your database table, even if it is necessary to make up information. Write a C# program to display all of the data that appears in the database table on a data grid.

The instructor may provide further specific details.

<i>Components</i>	<i>% of Grade</i>
Windows Applications development	30%
Object-oriented programming	10%
Quality coding practices	20%
Design and implement databases	20%
Access data	20%
TOTAL	100%

- **PE 8 : Sports Database**

Create a small Sports database with two tables: Team and Athlete. The Team table should include fields for the type of team (e.g., basketball), coach's name (both last and first), and the season the sport is most active (S for spring, F for Fall, or B for both). The Athlete table should include fields for student number, student first and last names, and type of sport. Use the same identifier for type of sport in both tables to enable the tables to be related and linked. Populate the tables with sporting teams from your school. Write a C# program that displays information about each team, including the names of the athletes.

The instructor may provide further specific details.

<i>Components</i>	<i>% of Grade</i>
Windows Application development	30%
Object-oriented programming	10%
Quality coding practices	20%
Design & implement databases	20%
Access data	20%
TOTAL	100%

- **PE 9: Web Forms Controls Application**

Using Web Forms controls, create a Web application to store a user's To Do list. Include two TextBox objects, a Button object and a ListBox object. Allow users to input their name in one TextBox and To Do tasks into the other TextBox. Use those values to populate the ListBox object. Allow users to make a selection for which item to tackle next from the list. Display their name and the selection on a Label object and then remove that item from the ListBox.

The instructor may provide further specific guidance.

<i>Components</i>	<i>% of Grade</i>
Windows Applications development	30%
Object-oriented programming	20%
Quality coding practices	20%
Access data	30%
TOTAL	100%

- **PE 10 : Web Application (File I/O)**

Create a Web application that enables the user to enter first name, last name, and e-mail address. Accept those values and store them in a text file. Allow the user to input the path where the file should be stored. After retrieving the values from the user, display on the web page both the full file path (including the name of the file) and all values stored in the file. Confirm that the values are written to the file.

The instructor may provide further specific details.

<i>Components</i>	<i>% of Grade</i>
Implement C# applications	50%
Object-oriented programming	30%
Quality coding practices	20%
TOTAL	100%

Research Paper (RP)

In the research paper, each student will use an instructor approved topic relevant to the course. The paper must be no less than 3-4 pages, excluding the title and reference pages, using APA format, with at least 5 recent, scholarly, peer-reviewed references. As in any scholarly writing, students should not merely copy information from another author, but use evidence to support the contentions they have drawn from their findings and critically analyze related literature - each paper needs to be an analytical paper, not a summary of readings.

Students must cite the sources of all ideas, facts, tables, images, figures, formulas and information used

that are not their own, even if they have put the information into their own words. Failure to do so is plagiarism, even if the oversight is unintentional. Papers must be in compliance with the University's academic integrity policy, as described in the university catalog.

<i>Components</i>	<i>% of Grade</i>
Structure: Consists of the required report elements.	20%
Content: Demonstrates critical analysis and synthesis of concepts.	60%
Reference: Is pertinent to the topic and cited properly.	10%
Writing: Is clear, concise, and grammatically correct.	10%
TOTAL	100%

Team Project

The team project is a report on a contemporary subject related to software development, preferably related to programming. Student teams, in communication with the instructor for the course, will select a subject on which they wish to report. This report should evaluate how the subject affects or will affect development and delivery of software solutions.

The report should comprise 10 - 12 pages of content. References and citations need be included. Student teams are expected to employ APA formatting of citations, footnotes, and bibliography.

Three milestones: at least three submissions

- Proposal (1 page; 30 points)
- Progress (5-7 pages; 70 points; graded only after the proposal has been submitted)
- Final (8-10 pages; 100 points; graded only after the progress has been submitted)

Student teams are expected and encouraged to use the assigned readings, videos, and other materials used throughout the quarter on this project. Student teams will need to utilize additional sources that were not assigned. Student teams must cite the sources of all ideas, facts, and information used that are not their own, even if they have put the information into their own words. Failure to do so is plagiarism, even if the oversight is unintentional.

<i>Components</i>	<i>% of Grade</i>
Introduction of Subject	20%
Analysis of Environment / Technology	35%
Effect of Technology	35%
APA Style (Citations and References)	10%
TOTAL	100%

Discussions and Instructor Determined Assignments

The instructor will lead course discussions and provide additional assignments as appropriate. Rubrics may be revised appropriately by the instructor.

<i>Components</i>	<i>% of Grade</i>
Quality of responses	30%
Timeliness of responses	20%
Implement C# applications	20%
Object-oriented programming	10%
Windows Applications	20%
TOTAL	100%

Knowledge Check (KC)

Students will complete weekly quizzes that are from the course content to reflect on what they have learned in the course. Completing all KCs will help ensure that you successfully master the concepts in this course. The best way for you to gain a thorough understanding of the underlying concepts is to apply those concepts to solve the quizzes. You should focus on the underlying principles, rather than just memorizing information.

<i>Components</i>	<i>% of Grade</i>
Accuracy: Answers questions correctly.	100%
TOTAL	100%

COURSE POLICIES

Late Assignments

LATE ASSIGNMENT

Participation

PARTICIPATION

Professional Writing

Assignments require error-free writing that uses standard English conventions and logical flow of organization to address topics clearly, completely, and concisely. CityU requires the use of APA style.

UNIVERSITY POLICIES

You are responsible for understanding and adhering to all of City University of Seattle's academic policies. The most current versions of these policies can be found in the [University Catalog](#) that is linked from the CityU Web site.

Scholastic Honesty

Scholastic honesty in students requires the pursuit of scholarly activity that is free from fraud, deception and unauthorized collaboration with other individuals. You are responsible for understanding CityU's policy on scholastic honesty and adhering to its standards in meeting all course requirements. A complete copy of this policy can be found in the [University Catalog](#) in the section titled *Scholastic Honesty* under *Student Rights & Responsibilities*.

Attendance

Students taking courses in any format at the University are expected to be diligent in their studies and to attend class regularly.

Regular class attendance is important in achieving learning outcomes in the course and may be a valid consideration in determining the final grade. For classes where a physical presence is required, a student has attended if s/he is present at any time during the class session. For online classes, a student has attended if s/he has posted or submitted an assignment. A complete copy of this policy can be found in the [University Catalog](#) in the section titled *Attendance Policy for Mixed Mode, Online and Correspondence Courses*.

SUPPORT SERVICES

Disability Resources

If you are a student with a disability and you require an accommodation, please contact the Disability Resource Office as soon as possible. For additional information, please see the section in the [University Catalog](#) titled *Students with Special Needs* under *Student Rights & Responsibilities*.

Library Services

CityU librarians are available to help you find the resources and information you need to succeed in this course. Contact a CityU librarian through the [Ask a Librarian](#) service, or access [library resources and services online](#), 24 hours a day, seven days a week.

Smarthinking

As a CityU student, you have access to 10 free hours of online tutoring offered through Smarthinking, including writing support, from certified tutors 24 hours a day, seven days a week. Contact CityU's Student Support Center at help@cityu.edu to request your user name and password.