



4. Identify and apply appropriate data types within a Java solution.
5. Describe and identify key object-oriented programming concepts.
6. Differentiate between the memory allocation approach for primitive and reference data types in Java.
7. Examine the code available in the Java standard class libraries, and incorporate relevant Java standard classes into object-oriented design and program construction.
8. Create and document program design solutions for simple Java programs.
9. Given a solution design, create programmer-defined classes and incorporate these classes into Java program solutions.
10. Distinguish among the options for input and output using Java, and select appropriate approaches for a given Java solution.
11. Describe scope and persistence of objects and variables in object-oriented programming.
12. Identify and correctly apply sequence, selection, and iteration/repetition patterns in object-oriented Java solutions and program designs.
13. Identify and apply advanced class and object features, including: overloading methods and constructors, argument passing, object return from methods, and organizing classes into packages.
14. Manipulate collections of data using arrays and objects to solve a given problem using Java.
15. Describe the different sorting options available and select the best basic sort for use in a Java solution.
16. Apply test-first development to the construction of an object-oriented computer program.
17. Read and interpret UML 2.0 diagrams that document a problem, and implement the proposed solution using Java.
18. Implement professional standards and guidelines for designing and coding Java computer programs.
19. Present and justify, to a group of peers, the design and implementation of a problem solution.
20. Plan for and schedule adequate time to complete labs and projects no later than the required due date.
21. Consult various online and independent resources to independently attempt to resolve problems BEFORE requesting assistance from co-workers/co-learners or supervisor/instructor.
22. Determine when it is appropriate to seek assistance, from co-workers/co-learners or supervisor/instructor to resolve problems that could not be resolved independently.

**Course Grading Policy:** Your final grade for this course will be based on four components, namely exams, programming assignments, labs, and unannounced quizzes. Within each

component (for example, the *labs* component), all items (for example, all labs) will have equal weight. In other words, each component grade will be computed as the average of all items in the component (however, see important note below). Your overall numerical grade for the course will be computed as the weighted sum of the component grades using the following weights:

Component	Weight
Exams (3)	60%
Assignments (4 to 6)	25%
Weekly labs	10%
Quizzes	5%

**Important note:** If you receive a zero on more than 20% of your lab grades, your **overall lab grade** will automatically be reduced to 0. The same rule applies to your **overall quiz grade**.

Finally, your letter grade for the course will be computed as follows:

Numerical Score	Grade	Numerical Score	Grade
>=92	A	72-78	C
90-92	A-	70-72	C-
88-90	B+	68-70	D+
82-88	B	62-68	D
80-82	B-	60-62	D-
78-80	C+	<60	F

While this overall grading scheme is fixed, I will be happy to discuss any issue you may have with individual grades. If you notice a mistake or have a question regarding a specific grade, please come and talk to me *as soon as possible*. Do not wait until the end of the semester to bring up grading issues.

**Attendance and Participation:** You are expected to not only attend **every** class meeting but also to come **prepared** for and **participate** actively in it. Necessary preparation requires you to have studied and assimilated the material covered in previous sessions, to have met with me or the tutors outside of class to discuss any questions you may have, to have done the assigned reading (when applicable), and to have completed the programming assignments on time. I have high expectations for my students and will always support and encourage you. I **strongly encourage you to ask any question** or raise any issue you have with the course either during or at the end of class, or during my office hours. I will also gladly meet with you by appointment.

**Late Submissions:** The submission procedure for your assignments and labs will be described later. However, please note that each assignment and lab will come with a deadline (day and time) after which any submission is considered late. The late-submission policy works as follows:

Turned in	Penalty
on due date but after deadline	10%
One day late	25%
Two days late	50%
Three days late	75%
Four days late	100%

Weekend days and holidays count as "regular days" when computing late penalties. Extensions on assignments may be granted at the discretion of the instructor if you provide a valid justification (in the form of a written excuse from a medical doctor or the Dean of Students Office) **before** the due date.

If you miss a scheduled exam, you **may** be able to take a make-up exam provided you give the instructor a valid justification (see above) ahead of time if possible. Only one make-up exam will be given. It will be a comprehensive exam scheduled at the end of the semester. It will take place at the testing center. Similarly, there will be no make-up quizzes unless the instructor is provided with a valid justification (see above) for your absence on the day of the quiz.

**Collaborating versus Cheating:** Unless otherwise stated in the assignment or lab, all submissions must be the work of a single student (the one whose name appears on the submission, that is). While it is acceptable to discuss the assignments at a high level (for example, at the design level) with others, you must submit your own work. You may not "borrow" any piece of code or design of any length from someone else, unless you can live with a zero and the other potential academic sanctions of cheating (see [UWO Student Discipline Code 2007](#), Chapter UWS 14).

In conclusion, be aware that computer science classes require a lot of work in addition to active participation in class. It takes considerable practice to develop the technical and analytical skills targeted by this course. You will need to spend **at least (and typically more than) three hours of effort outside of class for each in-class hour**. Having said this, I expect every hardworking student to do well in this course.