Queen's University
Faculty of Engineering and Applied Science

CMPE 212 Notes

Object-Oriented Programming

Patrick Taylor

Winter 2026

Contents

1 Introduction to Java and OOP

1.1 Course Administration

1.1.1 Instructor and Staff

- **Instructor:** Dr. Burton Ma
- **Email:** burton.ma@queensu.ca (Please include "CMPE 212" in the subject line).
- **TAs:** Two teaching assistants (details to be posted on OnQ).
- **Office Hours:** To be announced.

1.1.2 Resources

- **Textbook:** None required.
- **Recommended:** *Head First Java* (available via Library) or *Effective Java* by Joshua Bloch.
- **Jupyter Notebooks:** A comprehensive set of interactive notes is available (originally for CISC 124, adapted for this course). These cover more depth than lectures but only lecture material is tested.

1.1.3 Assessment

Component	Weight	Notes
Labs (8 assessed)	40%	5% each. Lab 1 is for setup (not marked).
Quizzes (3)	45%	15% each. Held during lab slots.
Exam	15%	Not comprehensive; effectively a final test.

Table 1.1. Course Grading Scheme

1.2 Introduction to Object-Oriented Programming (OOP)

1.2.1 Procedural vs. Object-Oriented

In procedural programming (like C), programs are structured around **functions** and **logic** (loops, conditionals). Data is often separated from the functions that manipulate it.

In **Object-Oriented Programming (OOP)**:

- The program consists of interacting **objects**.
- Objects contain both **State** (data/information) and **Behavior** (methods/functions).
- Interaction happens via **Message Passing** (one object calling a method of another).

1.3 Java Basics

1.3.1 Overview

Java is a general-purpose, cross-platform, object-oriented language created in the early 1990s.

- **Write Once, Run Anywhere:** Java runs on a Virtual Machine (JVM), not directly on the hardware. This allows the same compiled bytecode to run on any machine with a JVM.
- **Memory Management:** unlike C, Java handles memory allocation and deallocation automatically via **Garbage Collection**. There are no pointers and no `malloc`/`free`.

1.3.2 Comparison with C

Feature	**C**	**Java**
Paradigm	Procedural	Object-Oriented
Pointers	Yes (manual mem. access)	No (references only)
Memory	`malloc` / `free`	Automatic (Garbage Collection)
Strings	Array of `char` (`char*`)	`String` class (immutable)
Booleans	`int` (0 is false)	`boolean` (`true`/`false`)
Int Size	Architecture dependent	Fixed (e.g., `int` is always 32-bit)

1.3.3 Hello World Example

```
public class HelloWorld {
    // Main method: Entry point of the application
    public static void main(String[] args) {
```

```
        // System.out.println is used instead of printf for basic output
        System.out.println("Hello, World!");
    }
}
```

Listing 1.1. Hello World in Java

Note. In Java, every method must belong to a class. Class names typically use **UpperCamelCase**, while variable and method names use **lowerCamelCase**.

1.3.4 Primitive Types

Java defines fixed sizes for its primitive types to ensure portability:

- `int`: 32-bit signed integer.
- `long`: 64-bit signed integer.
- `double`: 64-bit floating point.
- `boolean`: true or false (cannot be cast to int).
- `char`: 16-bit Unicode character (unsigned).

1.3.5 Strings

Strings in Java are objects of the `String` class. They are **immutable**, meaning once created, their value cannot be changed. String concatenation is performed using the `+` operator.

```
String s = "Hello";
String t = "World";
String u = s + " " + t; // "Hello World"
```